Echec OP – WriteUp – 0xE0L

Part0/3 – Chall d'introduction :

Rien de bien sorcier, on affiche les partitions de ce volume via fdisk : `fdisk -l fcsc.raw`

On reçoit :

```
Disk fcsc.raw: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 60DA4A85-6F6F-4043-8A38-0AB83853E6DC

Device       Start      End  Sectors  Size Type
fcsc.raw1     2048     4095     2048    1M BIOS boot
fcsc.raw2     4096  1861631  1857536  907M Linux filesystem
fcsc.raw3  1861632 20969471 19107840  9.1G Linux filesystem
```

Le flag est donc : **FCSC{60DA4A85-6F6F-4043-8A38-0AB83853E6DC}**

Part1/3 – Déchiffrement et montage du disque :

Dans cette partie, on va devoir déchiffrer et monter le disque sur notre machine Linux d'investigation.

Quelques commandes de repérage :

```
> fdisk -l fcsc.raw
Disk fcsc.raw: 10 GiB, 10737418240 bytes, 20971520 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 60DA4A85-6F6F-4043-8A38-0AB83853E6DC

Device       Start      End  Sectors  Size Type
fcsc.raw1     2048     4095     2048    1M BIOS boot
fcsc.raw2     4096  1861631  1857536  907M Linux filesystem
fcsc.raw3  1861632 20969471 19107840  9.1G Linux filesystem
```

```
> mmls ./fcsc.raw
GUID Partition Table (EFI)
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Safety Table
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  Meta      0000000001   0000000001   0000000001   GPT Header
003:  Meta      0000000002   0000000033   0000000032   Partition Table
```

```
004:  000       0000002048   0000004095   0000002048
005:  001       0000004096   0001861631   0001857536
006:  002       0001861632   0020969471   0019107840
007:  -------   0020969472   0020971519   0000002048   Unallocated
```

```
> hexdump -C -s $((512*1861632)) ./fcsc.raw | head -n 1
38d00000  4c 55 4b 53 ba be 00 02  00 00 00 00 00 00 40 00  |LUKS..........@.|
```

Nous faisons donc face à du LUKS ! On sait que le mot de passe du chiffrement de disque est `fcsc2022`, ça va nous servir pour déchiffrer la partition par la suite.

Comme on le voit, la partition intéressante est la 002, c'est d'ailleurs la plus massive.

Je commence naïvement par saisir la commande suivante, pensant qu'on doit déchiffrer **tout le volume** LUKS : `sudo cryptsetup luksOpen fcsc.raw dmpData`

Malheureusement, cela renvoie une erreur (sans surprise quand on sait pourquoi) : `fcsc.raw n'est pas un périphérique LUKS valide.`

(C'est là que je me dis qu'il s'agit peut-être de LVM / LVM2 ? Un faisant un `strings` + `grep LVM2`, il y a effectivement des occurrences ! Ça sera bon à savoir pour la suite, mais ça n'est pas la bonne piste dans l'immédiat)

En fait, si ma commande d'avant ne marchait pas, c'est tout simplement parce-que j'essaie de déchiffrer TOUT LE VOLUME en LUKS, alors qu'en fait seule la grosse partition 006 est concernée. On va donc devoir l'extraire ! Pour cela, on utilise la commande : `mmcat fcsc.raw 6 > dump6.luks`

Une fois la partition extraite après plusieurs minutes, en faisant un `file dump6.luks`, on semble obtenir ce que l'on cherchait : `dump6.luks: LUKS encrypted file, ver 2 [, , sha256] UUID: 45e2f0c4-6640-453d-8b7a-8a60bd61c63d`

Ceci étant fait, on peut enfin essayer de déchiffrer la partition via : `sudo cryptsetup luksOpen dump6.luks dmpData`

On rentre le mot de passe (`fcsc2022`) et la partition est bien montée ! On peut la retrouver dans `/dev/mapper/dmpData`.

J'ai donc naïvement essayé de monter ce périphérique dans un dossier créé au préalable (`mkdir /mnt/fcsc2022`) via la commande : `mount /dev/mapper/dmpData /mnt/fcsc2022/`
Mais cela ne marche pas : `mount: /mnt/fcsc2022: unknown filesystem type 'LVM2_member'.`

Eh oui, comme on l'a évoqué vite fait avant, il s'agit d'une partition LVM2 chiffrée par LUKS ! Il faut donc la monter un peu plus spécifiquement. Pour cela rien de bien compliqué :

- On fait un `sudo lvscan` pour monter la partition spécifique à LVM2, qui nous répond "`ACTIVE '/dev/ubuntu-vg/ubuntu-lv' [9,09 GiB] inherit`" → maintenant on va réellement pouvoir monter ce dernier périphérique !
- On peut maintenant faire notre mount sur ce second périphérique créé : `mount /dev/ubuntu-vg/ubuntu-lv /mnt/fcsc2022/`

Miracle ! On constate que tout est bien monté dans /mnt/fsc2022/ !

Maintenant, reste à répondre à la question de cette étape : quand a été créé ce système de fichier (au format ISO-8601, en UTC+0) ?

Une recherche rapide sur internet nous apprend que pour cela, on peut utiliser l'outil tune2fs en lui passant en argument le périphérique de la partition (classiquement `sudo tune2fs -l /dev/sda1`). En l'occurrence, le périphérique est **/dev/ubuntu-vg/ubuntu-lv**, donc on fait un : `sudo tune2fs -l /dev/ubuntu-vg/ubuntu-lv`

Ça fonctionne ! La ligne qui nous intéresse est la suivante : `Filesystem created:       Sun Mar 27 05:44:49 2022`

Mais attention, si on lit le manuel de tune2fs, on voit qu'il nous retourne une date au format UTC+décalage de la machine qui effectue la commande ! Or, à ce moment-là ma machine de forensics est en UTC+2 !

On devine donc qu'en UTC+0 (le décalage demandé) l'heure réelle est 03:44:49.
On en déduit donc le flag : **FCSC{2022-03-27T03:44:49Z}**

Part2/3 – Récupération du mot de passe de l'utilisateur :

Juste avant de commencer, pour mettre un peu de contexte, notons que la machine est un Ubuntu20.

On nous demande le mot de passe de l'utilisateur, donc on se rend compte qu'il s'agit de obob si on fait un `ls /mnt/fcsc2022/home/.`

Naïvement, on pourrait essayer d'aller voir son hash qui se trouve dans le /etc/shadow (`obob:$6$cvD51kQkFtMohr9Q$vE2L5CUX3jDZgVUZGOFNUFsSHGomH/EP5yYQA3dcKMm9U00mvA9pLzo7Z.Ki6exchu29jEENxtBdGUXCISNxL0:19078:0:99999:7:::`) pour essayer de le bruteforcer, mais l'énoncé nous indique assez franchement que ça n'est sans doute pas la bonne approche : « *La force ne résout pas tout…* ». Sous-entendu, *inutile de recourir à la force brute*.

Intuitivement, au vu du nom du challenge (échec OP comme "*OPerational security*"/ opsec ?), je me dis que l'utilisateur a peut-être fait une bourde et mis son mot de passe quelque-part dans des fichiers de logs… Ou alors, il l'a tapé dans une commande.

Si on retient cette dernière hypothèse, on va donc voir dans le `.bash_history` de obob. Il ne contient pas de mot de passe, mais des petits hints comme quoi notre approche n'est pas déconnante :

```
exit
w
ls
perfect opsec
sudo -su -
sudo su -
exit
cd pkpas/
ls
cd big-list-of-naughty-strings/
git pull
git status
git log
cd
ls
w
exit
shutdown
```

Un peu plus tard, je finis par aller voir dans le `.bash_history` de root et là, j'obtiens ceci :

```
exit
passwd obob
CZSITvQm2MBT+n1nxgghCJ
exit
```

Effectivement, en voulant set le mot de passe pour obob, l'admin a fait une erreur et l'a saisi en ligne de commande ! Il a oublié d'en supprimer l'historique, donc on peut le récupérer.

Flag : **FCSC{CZSITvQm2MBT+n1nxgghCJ}**

Part3/3 – Récupération de l'IP de l'administrateur :

Je m'appuie pas mal sur grep et la regex suivante pour rechercher des IPs dans les fichiers de logs, entre autres : `grep -r -E -o "(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)" /var/log/`

Je constate assez rapidement les choses suivantes :

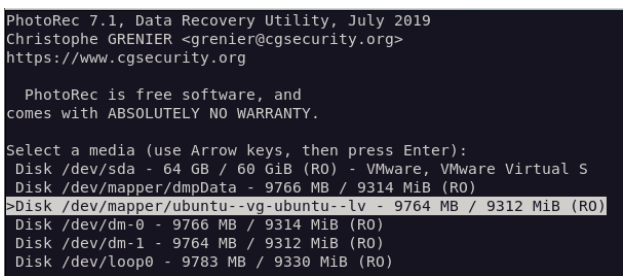
- Il semble que l'IP de l'administrateur soit 172.16.123.130 (je le vois notamment dans les logs Apache, dans syslog un peu aussi), ou bien 172.16.123.1. Malheureusement, en tentant de valider ces 2 IPs, ce n'est pas ça.
- A côté, on constate que l'IP de la passerelle à laquelle est connectée le serveur est 172.16.123.2, et l'IP du serveur lui-même 172.16.123.129.

Je poirote un moment, puis je me fais la remarque suivante : si, comme l'indique l'énoncé, « *l'administrateur a tenté de dissimuler son IP* », c'est qu'il a supprimé quelque-chose comme des logs peut-être ? Si tel est le cas, peut-être faudrait-il faire du **carving** !

Effectivement, bizarrement il n'y a pas de fichier `/var/log/auth.log` qui est normalement présent sous Debian/Ubuntu (après ça ne veut pas dire qu'il a été supprimé, peut-être juste pas paramétré sur cette distrib)… De plus, dans les logs Nginx il n'y a pas grand-chose non plus donc on peut se demander si quelque-chose ne cloche pas, cependant cela ne signifie pas automatiquement que les logs ont été modifiés. Un seul moyen de s'en assurer : le carving.

Je vais spoil un peu : **la solution est effectivement d'utiliser le carving. J'ai perdu trop de temps à ne pas en faire au début, alors que l'énoncé donne un indice, c'est donc une des choses que j'aurais dû faire assez rapidement** ! Ça m'aurait permis de gagner 1h.

Pour cela nous allons utiliser l'outil **PhotoRec** : tout d'abord commencer par créer un dossier où seront stockés les logs extraits (`mkdir /mnt/fcsc2022-carved/`). Ensuite, lancer l'outil via `sudo photorec`. Il nous demande quel périphérique choisir, on choisit évidemment le second que nous avions monté (`/dev/ubuntu-vg/ubuntu-lv`) :



Ensuite, il nous demande si on souhaite effectuer le carving sur la partition principale (choix n°2 sur le screen) ou sur tout le disque (choix n°1 sur le screen). J'ai d'abord commencé avec juste la partition principale (que j'ai personnellement extraite par la suite dans `/mnt/fsc2022-carved/`), puis j'ai refait une seconde extraction ensuite avec tout le disque (dans `/mnt/fcsc2022-carved-mega/` pour ma part). Voici :

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

Disk /dev/mapper/ubuntu--vg-ubuntu--lv - 9764 MB / 9312 MiB (RO)

     Partition                  Start        End    Size in sectors
      Unknown                  0   0  1 19070975   0  1   19070976 [Whole disk]
>   P ext4                     0   0  1 19070975   0  1   19070976
```

Je pensais que faire sur la partition principale suffirait, mais en faisant sur tout le disque j'ai eu l'impression d'extraire plus de fichiers intéressants ! Le mieux est de tester mais de mon expérience sur ce challenge en particulier, l'extraction sur TOUT le disque a donné plus de résultats intéressants (suis passé de 1.6 à 3.3G de données – peut-être juste que tout le disque les extraie en double ?).

Bref, on lui indique ensuite qu'il s'agit bien d'un filesystem ext :

```
PhotoRec 7.1, Data Recovery Utility, July 2019
Christophe GRENIER <grenier@cgsecurity.org>
https://www.cgsecurity.org

   P ext4                     0   0  1 19070975   0  1   19070976

To recover lost files, PhotoRec needs to know the filesystem type where the
file were stored:
>[ ext2/ext3 ] ext2/ext3/ext4 filesystem
 [ Other     ] FAT/NTFS/HFS+/ReiserFS/...
```

Si on a choisi d'extraire TOUT le disque à l'étape d'avant, Photorec nous demande directement où on souhaite extraire nos fichiers. Si on a choisi d'extraire seulement la partition ext4, Photorec nous demande maintenant si on souhaite effectuer le carving sur toute la partition (`whole`) ou juste sur l'espace libre au sein de cette partition (`free`). Là aussi, à voir selon chacun quitte à tester les 2 au pire : `whole` sera plus complet mais plus lourd, `free` plus léger mais potentiellement plus propice à nous faire louper des choses.

En l'occurrence, pour « juste la partition », j'ai choisi de faire juste sur le `free` space. Pour tout le disque, cette option ne nous est pas demandée car par défaut absolument tout sera extrait, c'est l'option la plus complète.

Bref, cela fait on choisit le répertoire d'extraction qu'on avait créé en amont puis on la lance ! Au bout d'une ou deux minutes, tous les fichiers sont extraits.

Sur les répertoires extraits, je reprends donc l'utilisation de notre bon vieux `grep -r` pour chercher des choses récursivement dans tous les fichiers. Assez vite, je ne retrouve pas tout de suite d'IP mais

je constate qu'effectivement l'utilisateur a tenté de masquer ses traces (via "`grep -ir "access" fcsc2022-carved/ |grep obob`") :

Il a supprimé `access.log` ! Comme quoi mon intuition n'était pas mauvaise ! J'en déduis que ce `recup_dir.5/f13226096.txt` est peut-être une sorte d'audit.log. Bref, qu'importe, il va falloir retrouver les fichiers supprimés.

D'ailleurs pour auth.log, on fait la même constatation (via "`grep -ir "auth.log" fcsc2022-carved/ |grep obob`"), il a été supprimé par l'administrateur :

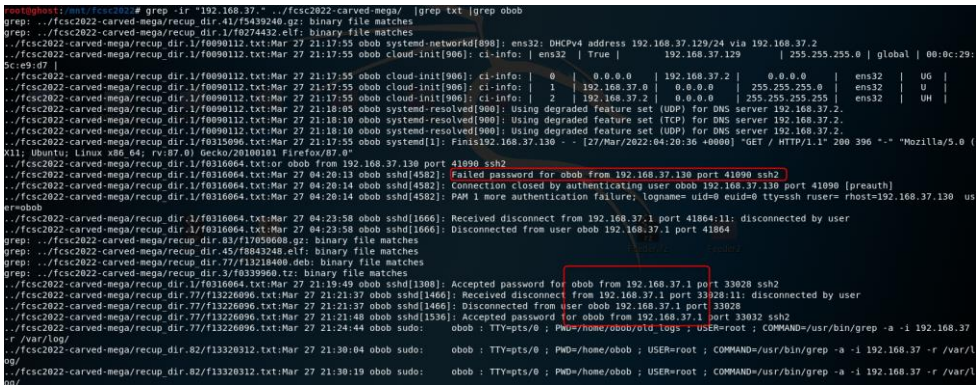
```
root@ghost:/mnt/fcsc2022# grep -ir "auth.log" ../fcsc2022-carved/ |grep obob
grep: ../fcsc2022-carved/recup_dir.1/f1445760.elf: binary file matches
../fcsc2022-carved/recup_dir.1/f0278496.txt:Mar 27 21:22:42 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob/old_logs ; USER=root ; COMMAND=/usr/bin/cp /var/log/auth.log .
grep: ../fcsc2022-carved/recup_dir.5/f13218400.deb: binary file matches
../fcsc2022-carved/recup_dir.6/f13320312.txt:Mar 27 21:30:18 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob ; USER=root ; COMMAND=/usr/bin/rm /var/log/auth.log
grep: ../fcsc2022-carved/recup_dir.2/f4857232.deb: binary file matches
../fcsc2022-carved/recup_dir.5/f13226096.txt:Mar 27 21:22:42 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob/old_logs ; USER=root ; COMMAND=/usr/bin/cp /var/log/auth.log .
../fcsc2022-carved/recup_dir.5/f13226096.txt:Mar 27 21:25:31 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob/old_logs ; USER=root ; COMMAND=/usr/bin/rm -r /var/log/auth.log
../fcsc2022-carved/recup_dir.5/f13226192.txt:Mar 27 04:07:59 obob sudo:     obob : TTY=tty1 ; PWD=/var ; USER=root ; COMMAND=/usr/bin/rm -r log/auth.log
grep: ../fcsc2022-carved/recup_dir.3/f8839144.elf: binary file matches
```

On tombe également sur une autre tentative de masquage des traces, très intéressante :

```
Mar 27 21:29:40 obob systemd: pam_unix(systemd-user:session): session opened for user obob by (uid=0)
Mar 27 21:30:04 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob ; USER=root ; COMMAND=/usr/bin/grep -a -i 192.168.37 -r /var/log/
Mar 27 21:30:04 obob sudo: pam_unix(sudo:session): session opened for user root by obob(uid=0)
Mar 27 21:30:04 obob sudo: pam_unix(sudo:session): session closed for user root
Mar 27 21:30:18 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob ; USER=root ; COMMAND=/usr/bin/rm /var/log/auth.log
Mar 27 21:30:18 obob sudo: pam_unix(sudo:session): session opened for user root by obob(uid=0)
Mar 27 21:30:18 obob sudo: pam_unix(sudo:session): session closed for user root
Mar 27 21:30:19 obob sudo:     obob : TTY=pts/0 ; PWD=/home/obob ; USER=root ; COMMAND=/usr/bin/grep -a -i 192.168.37 -r /var/log/
Mar 27 21:30:19 obob sudo: pam_unix(sudo:session): session opened for user root by obob(uid=0)
Mar 27 21:30:19 obob sudo: pam_unix(sudo:session): session closed for user root
Mar 27 21:30:43 obob sshd[1356]: Received disconnect from 172.16.123.1 port 55180:11: disconnected by user
```

L'administrateur vérifie que son IP n'apparaisse plus dans les logs (suite aux suppressions qu'il a faites) via `grep`, sans la mettre complètement cependant. Mais on voit qu'elle commence par 192.168.37, ce qui est déjà un très bon indice !

On n'a plus qu'à creuser un petit peu avec grep (genre `grep -ir "192.168.37." fcsc2022-carved-mega/ |grep txt |grep obob`), et on tombe assez rapidement sur des choses comme ça :



Pour info, on peut aussi tomber dessus en utilisant la regex de parsing IP évoquée en tout début de cette partie 3 !

J'en arrive aux conclusions suivantes, proches de celles en 172.16.123 évoquées au début :

- L'IP de l'administrateur est soi 192.168.37.130, soit 192.168.37.1.
- 192.168.37.2 est la passerelle, est l'IP du serveur est 192.168.37.129.

Je teste d'abord avec `FCSC{192.168.37.130}`, c'est un échec. Je teste ensuite avec `FCSC{192.168.37.1}` : victoire ! Ouf, il ne me restait plus qu'un essai ! Flag : **FCSC{192.168.37.1}**