Protection des données

1. Préambule

L'objectif de ce TP est de découvrir différents mécanismes de protection des données reposant sur les critères de confidentialité et d'intégrité

2. Prérequis

Deux machines virtuelles au format VirtualBox sont utilisées pour ce TP :

- SES_TP2_Debian : système Debian 9 utilisé lors du TP2 (pour mémoire, le compte utilisateur est *tp* et les mots de passes sont également *tp*), en fonction de ce qui a été fait au TP2, il peut être préférable de repartir d'un état zéro (snapshot ou redéploiement de la machine virtuelle) ;
- SES_TP1_Win10 : dérivée de la machine virtuelle Windows 10[1] mises à disposition par Microsoft pour tester le navigateur Edge, le clavier a été passé en français pour éviter les désagréments de saisie avec un clavier QWERTY, l'antivirus Windows Defender a été désactivé et plusieurs outils ont été ajoutés (Mimikatz, John-The-Ripper et la suite Sysinternals).

3. Chiffrement de partition

Exercice 3-1

Cet exercice est principalement destiné à faciliter la prise en main du couple cryptsetup/dm-crypt en créant un conteur chiffré sur la machine virtuelle SES_TP2_Debian. Pour les étudiants maîtrisant

Avant de commencer, il faut vérifier que le noyau contient bien l'extension nécessaire (intégrée au noyau => CONFIG_DM_CRYPT=y ou sous forme de module => CONFIG_DM_CRYPT=m).

```
grep DM_CRYPT /boot/config-`uname -r`
```

Pour réaliser le conteneur, un fichier de 100 Mo va être utilisé

```
sudo dd if=/dev/zero of=/var/conteneur-chiffre.part bs=512 count=200k
sudo losetup /dev/loop0 /var/conteneur-chiffre.part
```

Si le périphérique /dev/loop0 est déjà utilisé, il taper la commande suivante pour identifier un périphérique boucle disponible

```
sudo losetup -f
```

1 https://developer.microsoft.com/en-us/microsoft-edge/tools/vms/

Avant de créer le système de fichier, il est nécessaire de mettre en place une structure de données adaptée

```
sudo cryptsetup luksFormat /dev/loop0
```

→ taper YES (en majuscule) puis un mot de passe

La commande suivante permet au système d'accéder aux blocs de données en clair via l'interface « Device Mapper ».

```
sudo cryptsetup luksOpen /dev/loop0 conteneurClair
```

Maintenant que la partie déchiffrée (en « clair ») de notre conteneur est accessible, on peut l'utiliser comme une partition classique et installer un système de fichiers.

```
sudo mkfs -t ext4 /dev/mapper/conteneurClair
sudo mount /dev/mapper/conteneurClair /mnt
```

Pour vérifier (de façon très très basique) que le chiffrement est réel, un fichier texte est crée.

```
sudo nano /mnt/fic1.txt
```

avec le texte suivant :

```
Ce texte est-il bien protégé ?
```

Démonter le système de fichiers et la couche de chiffrement.

```
sudo umount /mnt
sudo cryptsetup luksClose conteneurClair
```

Exécuter la commande suivante :

```
strings /var/conteneur-chiffre.part | grep texte
```

Cette commande affiche-t-elle le texte saisi ? Comment pourrait-on s'assurer de la qualité du chiffrement réalisé ?

_____

_____

Afin de mieux comprendre comment fonctionne dm-crypt en mode LUKS, il est possible d'obtenir des informations avec la commande suivante :

```
/sbin/cryptsetup luksDump /var/conteneur-chiffre.part
```

Parmi les informations affichées, il est intéressant de regarder les champs « ciper name » « mode », « hash spec », ceux comportant la mention MK (pour « Master Key ») et les « Key Slots ».

Par défaut, la clé maître n'est pas affichée pour des raisons de sécurité. Il est cependant possible de l'afficher avec la commande suivante :

```
sudo cryptsetup --dump-master-key luksDump /var/conteneur-chiffre.part
```

Par défaut, un seul mot de passe a été défini, il est possible d'en ajouter d'autres

```
sudo cryptsetup luksAddKey /var/conteneur-chiffre.part
```

Pour identifier le changement, refaire

```
cryptsetup luksDump /var/conteneur-chiffre.part
```

Que constatez-vous ?

Par défaut, cryptsetup demande peu d'informations à l'utilisateur lors de la mise en place du chiffrement. La plupart des paramètres peuvent être modifiés, par exemple pour changer l'algorithme de chiffrement utilisé, la source d'aléa pour la génération de la clé maître ou la taille de clé. Il est également possible d'utiliser un fichier de clé à la place d'un mot de passe.

```
sudo dd if=/dev/zero of=/var/conteneur-chiffre-bis.part bs=512 count=200k
sudo losetup /dev/loop1 /var/conteneur-chiffre-bis.part
dd if=/dev/random of=secretKey1 bs=1c count=256
sudo cryptsetup luksFormat /dev/loop1 --hash SHA512 --cipher serpent-xts-plain64 --key-size 512 --use-random --key-file secretKey1
```

L'utilisation de /dev/random (cas du paramètre --use-random) peut rendre l'opération plus longue, notamment sur une machine virtuelle, mais elle est censée garantir une meilleure qualité d'aléa que /dev/urandom.

Le choix de l'algorithme de chiffrement symétrique SERPENT est donné à titre d'exemple, la discussion autour de la robustesse de cet algorithme sort du champ de ce TP.

L'ajout d'une clé supplémentaire se fait de façon similaire à l'ajout de mot de passe.

```
dd if=/dev/urandom of=secretKey2 bs=1c count=256
sudo cryptsetup luksAddKey /var/conteneur-chiffre-bis.part secretKey2 -d secretKey1
```

Pour une question de temps, /dev/urandom est utilisé ici car, sur une machine virtuelle, la réserve d'entropie du générateur d'aléa du noyau Linux peut être basse compte-tenu des commandes précédentes.

Si vous analyser (i.e. luksDump) les informations relatives à ce nouveau conteneur, voyez-vous une différence entre les fichiers de clés et les mots de passe utilisés pour déverrouiller la clé maître ?

Quel mécanisme privilégieriez-vous ?

_____

_____

Exercice 3-2

Avant de réaliser cet exercice, il est conseillé de faire un *snapshot* (instantané dans la traduction française de Virtual Box) pour faciliter le retour en arrière en cas de problème et tester certaines hypothèses de l'exercice 5-1.

En vous inspirant des billets http://www.pavelkogan.com/2014/05/23/luks-full-disk-encryption/ et https://outflux.net/blog/archives/2017/08/30/grub-and-luks/, chiffrez le contenu du répertoire /boot (mais pas /boot/grub) de la machine SES_TP2_Debian. L'utilisation d'un live-CD peut s'avérer pratique.

L'approche suivie avec le fichier crypto_keyfile vous parait-elle sûre ? Quelles alternatives peut-on envisager ?

_____

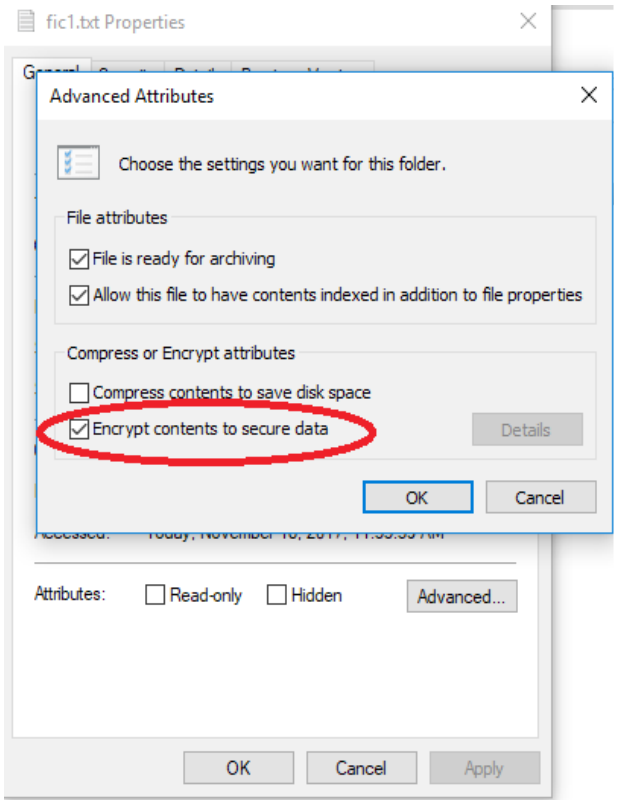
_____

4. Chiffrement de fichiers

Exercice 4-1

Le chiffrement de fichiers au moyen d'EFS peut se faire de 2 manières sous Windows : au travers de l'interface graphique de l'explorateur ou en ligne de commande. Si la première option peut paraître plus conviviale, elle s'avère fonctionnellement plus limitée que la seconde.

L'interface graphique est accessible via l'interface attributs avancées (fichier ou répertoire). Le chiffrement se fait en cochant « Encrypt contents to secure data ». L'ajout d'autres utilisateurs dans la liste d'accès (bouton « Details ») se fait dans un deuxième temps après avoir chiffré l'objet ciblé.



Créer un fichier fic1.txt sur le volume D: et activer le chiffrement via l'interface graphique. Analyser le fichier avec la commande cipher (EFSDump de Sysinternals n'apporte pas réellement d'informations complémentaires) et commenter les informations retournées.

```
cipher /C D:\fic1.txt
```

Un certificat EFS est créé automatiquement et une clé privée lui est associée (la sauvegarde de l'ensemble n'est pas nécessaire dans le cadre d'un TP). Pour consulter le certificat, utiliser la commande certmgr.msc et regarder de la contenu du magasin « Personal ». Examiner le certificat généré, que constatez-vous ?

L'empreinte courante du certificat utilisée pour EFS se retrouve dans le registre. Avec Powershell, il est possible de l'afficher :

```
cipher /Y
(Get-ItemProperty 'HKCU:\Software\Microsoft\Windows NT\CurrentVersion\EFS\CurrentKeys').CertificateHash | Format-Hex
```

De même, on peut consulter les magasins de certificats que l'on retrouve avec certmgr.msc :

```
ls cert:\CurrentUser\My
```

`ls cert:\CurrentUser\AddressBook` (correspond à Other People dans certmgr.msc, sa présence est assez aléatoire)

```
ls cert:\CurrentUser\TrustedPeople
```

Générer une nouvelle paire de clé en ligne de commande, noter l'empreinte du certificat et vérifier que l'empreinte courante a été modifiée

```
cipher /K /ECC:256
```

> **Important** : l'implémentation d'EFS n'étant pas dénuée de bugs, les certificats basés sur des clés ECDSA ne sont pas publiés automatiquement dans le magasin « Trusted People » de la machine (le magasin de l'utilisateur courant hérite du magasin de la machine) mais le magasin « Other People » de ma machine, ce qui se vérifie à l'aide des 3 commandes Powershell suivantes :
>
> ```
> ls cert:\CurrentUser\TrustedPeople
> ls cert:\LocalMachine\TrustedPeople
> ls cert:\LocalMachine\AddressBook
> ```
>
> Bien que ceux-ci puissent apparaître dans le magasin « Other People », pour la suite du TP, 3 options sont possibles :
>
> - utiliser des clés RSA plutôt que des clés ECDSA ;
> - exporter les certificats générés (bien évidemment, sans la clé privée) et de les importer au besoin dans les magasins des autres utilisateurs ;
> - déplacer les certificats générés du magasin « Other People » de ma machine vers le magasin « Trusted People » de la machine (nécessite un compte administrateur comme IEUser).

Remplacer la clé utilisée pour le fichier fic1.txt avec la nouvelle clé générée et analyser les détails du fichier avec la commande cipher, que remarquez-vous ?

```
cipher /rekey D:\fic1.txt
```

_____

_____

_____

Créer 2 nouveaux utilisateurs, util_3 et security_officer, sur le système. Si aucune politique de mot de passe spécifique n'est activée (ajouter un mot de passe sinon), les commandes suivantes tapées dans une invite de commande élevée devrait suffire :

```
net user util_3 /add
net user security_officer /add
```

Se connecter en util_3 et lui générer une paire de clé ECDSA 256 bits pour EFS.

Créer un fichier fic2.txt et ajouter IEUser dans les accès (pour la commande cipher, l'option /user:username n'est pas utilisable sans un Active Directory).

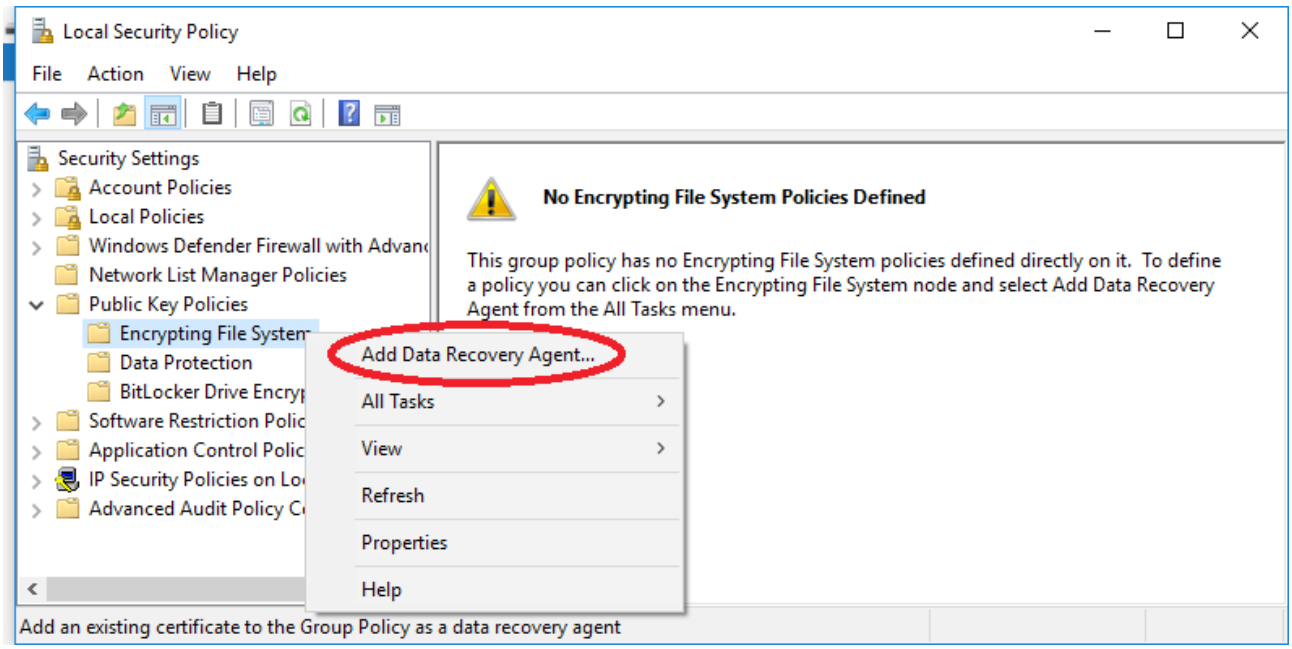
Se connecter en IEUser, vérifier que l'accès à fic2.txt est effectif puis ajouter du texte et enregistrer le fichier. Puis, ajouter util_3 dans les accès EFS du fichier fic1.txt.

Se connecter en util_3, ajouter du contenu dans le fichier fic.txt et enregistrer les modifications apportées. Essayer de retirer l'accès EFS à IEUser pour le fichier fic1.txt. Que pensez-vous du résultat obtenu ? Peut-on utiliser des ACL NTFS pour améliorer la situation ?

Se connecter en security_officer et vérifier que l'accès aux fichiers fic1.txt et fic2.txt est bloqué. Créer un certificat de recouvrement EFS :

```
cipher /R:D:\recoveryEFS /ECC:521
```

Avec le compte IEUser, lancer secpol.msc et ajouter le certificat D:\recoveryEFS.cer comme agent de recouvrement.



A l'issue, vérifier si le certificat de recouvrement a été ajouté aux fichier fic1.txt et fic2.txt avec la

commande cipher /C. Est-ce le cas ?

Élaborer une méthode pour propager le certificat de recouvrement à l'ensemble des fichiers.

Se connecter avec le compte security_officer et tester l'accès aux fichiers qui ont intégré le certificat de recouvrement. Pour que l'accès soit effectif, il faut que le certificat et la clé privée associée soit dans le magasin de l'utilisateur qui doit utiliser l'accès de recouvrement. Pour cela, il suffit de double cliquer sur le fichier D:\recoveryEFS.pfx et suivre l'assistant ou lancer la commande suivante :

```
certuil -user -ImportPFX D:\recoveryEFS.pfx
```

Tester l'accès aux fichiers, celui-ci doit être effectif. L'accès en recouvrement permet-il de modifier le contenu d'un fichier ? Qu'en pensez-vous ?

Lors de l'import, une option de protection de la privée peut être activée (en cochant la case ou en ajoutant le mot-clé ProtectHigh à la fin de la commande certutil). Supprimer le certificat précédemment importé du magasin et refaite l'import en activant la protection. Que constatez-vous ?

Depuis la version 1709 de Windows 10, il est possible d'utiliser EFS sur des partitions FAT, cela peut-il combler l'absence de contrôle d'accès de ce système de fichiers (par rapport à NTFS) ?

Au final, le système de chiffrement EFS vous parait-il abouti ?

5. Intégrité d'un système de fichiers

Exercice 5-1

La machine virtuelle SES_TP2_Debian contient un deuxième disque identifié par /dev/sdb. Celui va être utilisé pour illustrer le fonctionnement du module noyau dm-verity. L'utilitaire veritysetup sera utilisé pour cette exercice, il peut être utile d'ajouter le paramètre --debug pour disposer d'informations complémentaires.

Installer un système de fichiers EXT4 sur le périphérique /dev/sdb1 puis monter la première partition sous /mnt, créer un fichier avec un contenu quelconque et démonter cette partition.

```
sudo mkfs -t ext4 /dev/sdb1
sudo mount /dev/sdb1 /mnt
sudo touch /mnt/fic
sudo umount /mnt
```

Générer l'arbre lié à /dev/sdb1 sur /dev/sdb2 et noter le hash racine renvoyé par la commande pour l'utiliser dans la commande suivante. Vérifier le résultat avec dmsetup et examiner la structure des périphériques de type bloc.

```
sudo veritysetup format /dev/sdb1 /dev/sdb2
sudo veritysetup create verityVol /dev/sdb1 /dev/sdb2 hashracineinitial
sudo dmsetup table
lsblk
```

Monter le pseudo-périphérique créé sous /mnt. Que remarquez-vous ?

```
sudo mount -t ext4 /dev/mapper/verityVol /mnt
```

Démonter le et retirer le du système.

```
sudo umount /mnt
sudo dmsetup remove /dev/mapper/verityVol
ou
sudo veritysetup remove verityVol
```

Remonter /dev/sdb1 sous /mnt, modifier le contenu du système de fichiers (modification du fichier existant, ajout ou suppression de fichier, …), puis démonter le.

Vérifier l'intégrité de /dev/sdb1 avec la commande suivante.

```
sudo veritysetup verify /dev/sdb1 /dev/sdb2 hashracineinitial
```

Recréer le pseudo-périphérique /dev/mapper/verityVol. Pouvez-vous le monter sous /mnt ?

Regénérer l'arbre et noter le nouveau hash racine et vérifier qu'il est possible de monter le pseudo-périphérique (à recréer) sous /mnt.

Dans un contexte où un attaquant peut modifier le contenu du disque dur de la machine mais pas le microprogramme de démarrage (i.e. BIOS ou UEFI) ni les options de démarrage, comment peut-on procéder pour s'assurer que le hash racine qui sera transmis n'est pas modifié par l'attaquant :

- dans le cas où le cas où /boot n'est pas chiffré (situation de l'exercice 3-1) ?

- dans le cas où /boot est chiffré (situation de l'exercice 3-2) ?

Testez votre approche et comparez-la avec celles d'autres étudiants (voisin, binôme, ...). Le cas de la corruption des partition contenant les données et/ou l'arbre est exclu du contexte car il va conduire à une attaque en disponibilité. Ici, on s'intéressera uniquement au contrôle d'intégrité des données.