Déverrouillage Automatique Partition Principale

> Pour réaliser ce tuto il est nécessaire d'avoir au préalable créé la partition principale chiffrée avec LUKS.
>
> Voir Partitionnement des disques + Chiffrement (avancé)

Le but de ce tuto est de pouvoir déverrouiller la partition principale d'un serveur avec une clé USB. La clé USB contiendra un fichier qui correspondra à une clé accepté par LUKS.

Pour ce faire nous allons modifier l'`initramfs`. Petite explication : L'initramfs est un système de fichiers temporaire utilisé pendant le processus de démarrage du système. L'initramfs contient les modules du noyau et les scripts nécessaires pour monter le système de fichiers racine réel.

L'automatisation du déverrouillage d'une partition chiffrée au démarrage peut-être mis en place en suivant les étapes suivantes :

- 1/ Création d'un fichier de clé random
- 2/ Ajouter le fichier de clé à la partition LUKS
- 3/ Créer le script de déverrouillage
- 4/ Activer les modules attendu
- 5/ Recompiler l'init

1/ Création d'un fichier de clé

la commande suivante créera un fichier au contenu aléatoire d'une taille de 4096 bits (mieux qu'un mot de passe de 20/30 caractères....). Vous pouvez utiliser n'importe quel fichier comme fichier clé, mais je pense qu'un fichier de 4kb avec un contenu aléatoire convient bien.

```
sudo dd if=/dev/urandom of=/root/keyfile bs=1024 count=4
```

Comme vous pouvez le voir avec l'arrtibut **`of`** , le fichier sera généré dans le dossier **`/root`** avec le nom **`keyfile`**.

Avant d'utiliser la nouvelle clé, rendre le fichier clé accessible en lecture seule à root

```
sudo chmod 0400 /root/keyfile
```

> Cela rendra le fichier clé lisible uniquement par root. Si quelqu'un accède à ce fichier clé, vous avez de toute façon un problème plus important sur votre serveur.

> Une autre solution consiste à attribuer à root:root le droit d'accès au fichier clé souhaité et à le placer dans le dossier /root.

2/ Ajouter le fichier à LUKS

Les dispositifs LUKS/dm_crypt peuvent contenir jusqu'à 10 fichiers clés/mots de passe différents. Ainsi, en plus du mot de passe déjà configuré, nous allons ajouter ce fichier clé comme méthode d'autorisation supplémentaire.

```
sudo cryptsetup luksAddKey /dev/sdX /root/keyfile
```

Il vous sera d'abord demandé d'entrer un mot de passe (existant) pour déverrouiller le lecteur.

3/ Préparer le support USB

Tout d'abord nous allons identifier notre support USB pour le formater dans un format compatible avec un noyau linux .

Pour lister les disques nous allons utiliser la commande `fdisk -l` ce qui donnera :

```
...
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1    *           32 61439999 61439968  29,3G  b W95 FAT32
```

```
...
```

> Si le périphérique n'est pas considéré comme "Amorçage", ce n'est pas grave.

Ou avec la commande `lsblk -f` :

```
...
sdb
└─sdb1      vfat     FAT32 PHILIPS UFD 1872-8C67                              29,3G     0%
/media/usbkey/PHILIPS UFD
...
```

Notre clé USB doit être au format `ext4` qui est nativement compatible avec un noyau Linux contrairement au format `FAT32`.

Formater la clé USB

La commande pour formater la clé USB est la suivante : `fidsk /dev/sdb`
Cette commande permet dans l'ordre d'exécution de :

- `d` Supprimer la partition 1
- `n` Créer une nouvelle partition de type primaire (Tout laisser par défaut)
- `w` Enregistrer et appliquer les modifications

```
root@server: fdisk /dev/sdb

Bienvenue dans fdisk (util-linux 2.37.2).
Les modifications resteront en mémoire jusqu'à écriture.
Soyez prudent avant d'utiliser la commande d'écriture.


Commande (m pour l'aide) : d
Partition 1 sélectionnée
La partition 1 a été supprimée.

Commande (m pour l'aide) : n
Type de partition
   p   primaire (0 primary, 0 extended, 4 free)
   e   étendue (conteneur pour partitions logiques)
Sélectionnez (p par défaut) :
```

```
Utilisation de la réponse p par défaut.
Numéro de partition (1-4, 1 par défaut) :
Premier secteur (2048-61439999, 2048 par défaut) :
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-61439999, 61439999 par défaut) :

Une nouvelle partition 1 de type « Linux » et de taille 29,3 GiB a été créée.

Commande (m pour l'aide) : w
La table de partitions a été altérée.
Appel d'ioctl() pour relire la table de partitions.
Synchronisation des disques
```

Maintenant que la nouvelle partition est créé on va lui mettre l'étiquette ext4, avec la commande `mkfs.ext4 /dev/sdb1` ce qui donne :

```
mke2fs 1.46.5 (30-Dec-2021)
/dev/sdb1 contient un système de fichiers vfat étiqueté « PHILIPS UFD »
Procéder malgré tout ? (o,N) o
En train de créer un système de fichiers avec 7679996 4k blocs et 1921360 i-noeuds.
UUID de système de fichiers=6eb7057f-b73a-4586-9112-06c4fe7bb191
Superblocs de secours stockés sur les blocs :
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208,
4096000

Allocation des tables de groupe : complété
Écriture des tables d'i-noeuds : complété
Création du journal (32768 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété
```

Voila maintenant nous avons bien notre clé USB prête à l'emploi. Pour confirmer que la clé est au bon format on peut refaire la commande `lsblk -f` ainsi que la commande `fidisk -l` :

```
...
sdb
└─sdb1      ext4     1.0         6eb7057f-b73a-4586-9112-06c4fe7bb191
...
```

```
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
```

```
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1              2048 61439999 61437952  29,3G 83 Linux
```

Transférer la clé de déchiffrage sur la clé usb

Avant de pouvoir transférer le fichier il faut monter la clé USB pour qu'elle soit accessible :

```
# Création du dossier de montage
mkdir /mnt/usbkey

# Montage de la clé USB
mount /dev/sdb1 / mnt/usbkey
```

Déplacer ou copier le fichier généré *(à l'étape 1)* vers la clé USB :

```
cp /root/keyfile /mnt/usbkey/.
```

Créer le script à exécuter au démarrage

Pour ce faire nous allons ajouter un script qui sera utilisé par initramfs-tools pour construire l'image initramfs. Tous les scripts se trouvent dans l'emplacement `/etc/initramfs-tools/scripts/` . Dans ce répertoire, se trouvent les différentes étapes du processus de démarrage pour configurer et monter le système.

Nous allons nous intéresser au dossier `local-top` qui contient des scripts exécutés après les scripts dans `init-top`, mais avant les scripts dans `init-premount`. Ces scripts sont utilisés pour des tâches locales spécifiques avant le montage du système de fichiers racine.

Créer le fichier `unlock-luks-from-usb` dans le répertoire vu précédemment : `/etc/initramfs-tools/scripts/local-top/` ce qui donne :

```
nano /etc/initramfs-tools/scripts/local-top/unlock-luks-from-usb
```

Pour y mettre le contenu suivant :

```
#!/bin/sh

PREREQ=""

prereqs() {
    echo "$PREREQ"
}

case $1 in
prereqs)
    prereqs
    exit 0
    ;;
esac

# Variables
CRYPT_NAME="sda5_crypt"
USB_UUID="XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX" # Remplace par l'UUID exact de la partition de la clé USB
KEYFILE="/mnt/usbkey/keyfile"   # Emplacement (laisser /mnt/usbkey) + Nom du fichier sur la clé USB
HDD_UUID="XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX" # UUID exact de la partition LUKS a déchiffrer

# On attend que les périphériques USB soient prêts
sleep 3

# Création du point de montage
mkdir -p /mnt/usbkey

# En cas d'erreur pour aider au débuggage décommenter cette ligne
# ls -al /dev/disk/by-uuid

# Cherche le slot de la clé USB via l'UUID
for dev in /dev/disk/by-uuid/$USB_UUID; do
    echo "[unlock-luks-from-usb] Tentative de montage de $dev..."
    mount -t ext4 $dev /mnt/usbkey && break
done

# Si le fichier clé est présent, on tente le déchiffrement
if [ -f "$KEYFILE" ]; then
```

```
    echo "[unlock-luks-from-usb] Clé trouvée. Déverrouillage..."
    # Cherche le slot de la partition LUKS à partir de l'UUID
    for dev in /dev/disk/by-uuid/$HDD_UUID; do
        echo "[unlock-luks-from-usb] Récupération de la partition LUKS $dev..."
        echo "[cryptsetup] tentative de déverrouillage"
        cryptsetup luksOpen $dev "$CRYPT_NAME" --key-file "$KEYFILE" && exit 0
        echo "Échec du déverrouillage "
    done
else
    echo "[unlock-luks-from-usb] Fichier clé non trouvé sur la clé USB."
fi

# Si on arrive ici, on ne déverrouille pas, donc cryptsetup demandera le mot de passe
```

Maintenant il faut prendre soin de remplacer les `XXXXXXX-XXXX-XXXX-XXXX-XXXXXXXX` par les vrais UUID des variables.
Commençons par l'UUID de la clé USB, c'est très simple nous pouvons faire la commande blkid pour lister les UUIDs des périphériques de stockage présent sur le serveur :

```
/dev/sdb1: UUID="8fedfdda-5293-43ec-96ce-0902440234c6" BLOCK_SIZE="4096" TYPE="ext4"
PARTUUID="128faffe-01"
```

Il faut récupérer tout ce qui est entre guillemet de la variable `UUID` pour le mettre dans le script pour la variable `USB_UUID`.

Il nous manque plus que l'UUID de la partition LUKS à déchiffrer, pour ce faire, il est possible d'utiliser la commande précédente `blkid`. Pour reconnaître le bon UUID, c'est celui qui a le type `crypto_LUKS`, comme ci-deesous :

```
/dev/sda5: UUID="502cdf24-1935-4f4d-8262-9434ba606114" TYPE="crypto_LUKS" PARTUUID="485azer1-
05"
```

Pour résumer la partie variable de notre script ressemble à ceci :

```
# Variables
CRYPT_NAME="sda5_crypt"
USB_UUID="8fedfdda-5293-43ec-96ce-0902440234c6" # Remplace par l'UUID exact de la partition de
la clé USB
KEYFILE="/mnt/usbkey/keyfile"   # Emplacement (laisser /mnt/usbkey) + Nom du fichier sur la
clé USB
HDD_UUID="502cdf24-1935-4f4d-8262-9434ba606114" # UUID exact de la partition LUKS a déchiffrer
```

Rendre la clé USB visible au démarrage

Même si on a formaté notre clé USB dans un format nativement géré par Linux. Il se peut que la clé USB ne soit visible au démarrage car des modules permettant d'identifier la clé USB seront manquant de l'initramfs.

> Pour qu'elle soit visible au démarrage sans rajouter la configuration ci-dessous, la clé USB doit être identifié comme "Amorçable" lorsque vous faîtes la commande `fdisk -l` .

Par exemple une clé USB identifié comme "Amorçable" ressemble à ceci :

```
Disque /dev/sdb : 29,3 GiB, 31457280000 octets, 61440000 secteurs
Disk model: PHILIPS USB
Unités : secteur de 1 × 512 = 512 octets
Taille de secteur (logique / physique) : 512 octets / 512 octets
taille d'E/S (minimale / optimale) : 512 octets / 512 octets
Type d'étiquette de disque : dos
Identifiant de disque : 0x128faffe

Périphérique Amorçage Début      Fin Secteurs Taille Id Type
/dev/sdb1           *  2048 61439999 61437952  29,3G 83 Linux


 # Il y a une petite étoile * en dessous d'Amorçage
```

Ce n'est pas grave si une clé USB n'est pas Amorçable il suffit de rajouter la configuration suivante dans le fichier `/etc/initramfs-tools/modules` :

```
# List of modules that you want to include in your initramfs.
# They will be loaded at boot time in the order below.
#
# Syntax:  module_name [args ...]
#
# You must run update-initramfs(8) to effect this change.
#
# Examples:
#
# raid1
# sd_mod
```

```
usb_storage
ehci_pci
ehci_hcd
ohci_hcd
uhci_hcd
xHCI_hcd
usbhid
sd_mod
ext4
```

Explication des modules

Module	Description	Utilisation
`usb_storage`	Module pour la prise en charge des périphériques de stockage USB.	Permet de monter et d'accéder aux périphériques de stockage USB, comme les clés USB et les disques durs externes.
`ehci_pci`	Module pour le contrôleur hôte EHCI (Enhanced Host Controller Interface) PCI.	Prend en charge les contrôleurs USB 2.0 sur les bus PCI.
`ehci_hcd`	Module pour le contrôleur hôte EHCI (Enhanced Host Controller Interface).	Prend en charge les contrôleurs USB 2.0.
`ohci_hcd`	Module pour le contrôleur hôte OHCI (Open Host Controller Interface).	Prend en charge les contrôleurs USB 1.1.
`uhci_hcd`	Module pour le contrôleur hôte UHCI (Universal Host Controller Interface).	Prend en charge les contrôleurs USB 1.1.
`xhci_hcd`	Module pour le contrôleur hôte xHCI (eXtensible Host Controller Interface).	Prend en charge les contrôleurs USB 3.0.
`usbhid`	Module pour la prise en charge des périphériques HID (Human Interface Device) USB.	Permet la prise en charge des périphériques d'interface humaine comme les claviers, les souris et les manettes de jeu USB.
`sd_mod`	Module pour la prise en charge des périphériques de stockage SD (Secure Digital).	Permet de monter et d'accéder aux cartes SD et aux périphériques de stockage SD.
`ext4`	Module pour le système de fichiers ext4.	Permet de monter et d'accéder aux systèmes de fichiers ext4, qui est le système de fichiers par défaut pour de nombreuses distributions Linux.

Mise à jours de l'initramfs

Dans le contexte de la configuration du chiffrement du disque, nous utilisons cette commande après avoir modifié les fichiers de configuration comme `/etc/initramfs-tools/modules` et `/etc/initramfs-tools/scripts/local-top/`. Cela garantit que les changements sont inclus dans l'initramfs et sont disponibles pendant le processus de démarrage.

Pour mettre à jours l'initramfs il faut faire la commande :

```
update-initramfs -u -k all
```

- `-u` : Cette option signifie "update" (mettre à jour). Elle indique à la commande de mettre à jour l'initramfs existant plutôt que de créer un nouveau.
- `-k all` : Cette option spécifie pour quels noyaux l'initramfs doit être mis à jour. `-k all` signifie que l'initramfs doit être mis à jour pour tous les noyaux installés sur le système.

> Le système de démarrage est maintenant prêt, plus qu'a tester en redémarrant le serveur avec `reboot`.
> Si tout se passe bien le serveur doit démarrer normalement.
> Si le fichier n'est pas trouvé sur la clé, le mot de passe pour déverrouiller le disque dur sera demandé.

> Il est possible que quelque chose se passe mal dû a une mauvaise configuration. Dans ce cas nous allons nous retrouver dans le terminal d'initramfs. Si c'est le cas, il est possible de débloquer le serveur en suivant le tuto : Démarrer via le terminal initramfs

---

Revision #10
Created 30 June 2025 11:46:29 by gpatruno
Updated 3 July 2025 10:12:10 by gpatruno