TP - Hardening Linux

L'objectif de ce TP était de construire un serveur ArchLinux proposant des mesures de sécurisation de base au niveau de sa mise en place.

ArchLinux : Arch Linux est une distribution flexible de linux modelable pour en faire tout ce que nous voulons. Elle est rapide, légère, complète et la plupart des pièces sous le capot sont facilement compréhensibles et adaptables, ce qui la rend particulièrement adaptée à l'apprentissage des mécanismes de Linux.

Avantages

- Rolling release
- Principe KISS
- Pacman
- Arch Build System
- Arch User Repository
- Documentation

Inconvénients

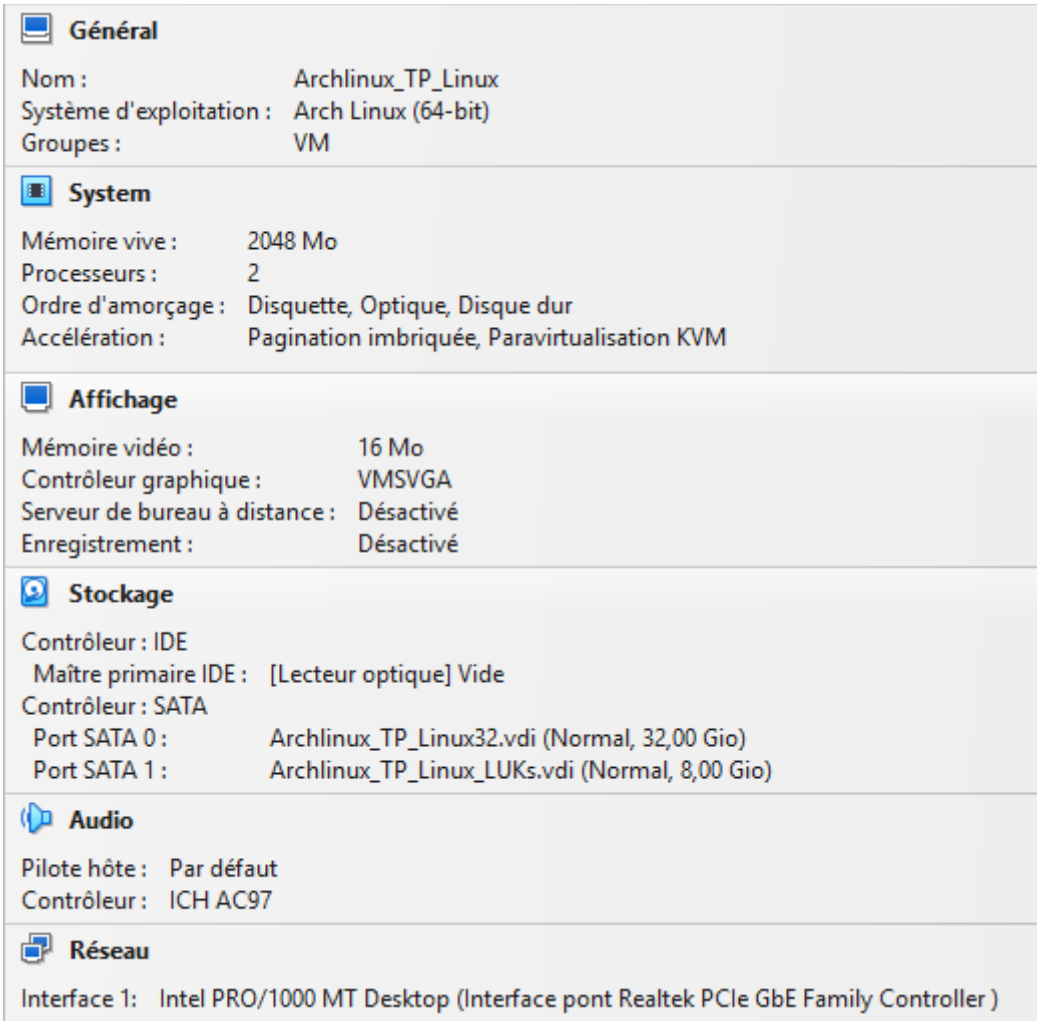
- Difficulté d’utilisation
- Installation
- Potentielle instabilité

SOMMAIRE :

1. TP - Hardening Linux lvl 1

1. Gestion de la racine

Avant toute chose, il nous faut télécharger l'ISO de ArchLinux et configurer VirtualBox pour un démarrage sans accrochage.

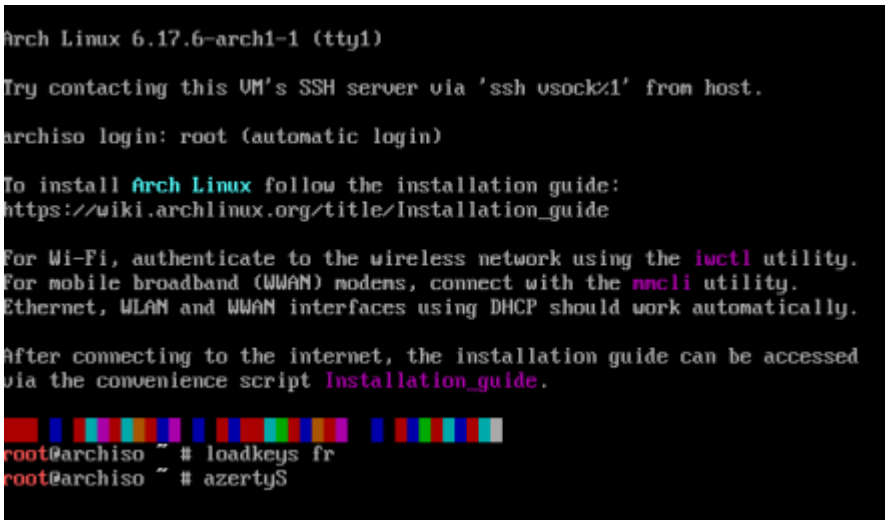


Pour ce qui est des contrôleur SATA, je n'ai pas pu rajouter un 3ème pour simuler ma clé USB donc j'ai coupé en 2ème le disque SATA 1 avec donc la partition chiffré LUKS et la clé USB. Gardez ça en tête pour après car vous aller penser que mon scripte de la clé va chercher dans mes disques alors que c'est bien le scripte de la clé qu'il va chercher.

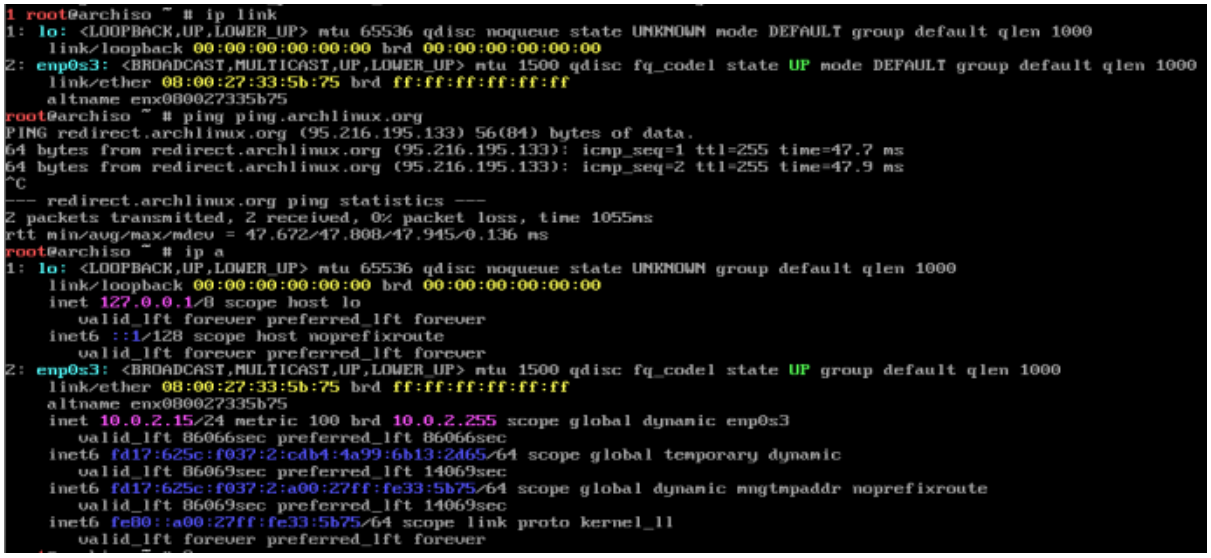
Je me suis mis en Accès par pont aussi car pour le SSH on à tester et cela fonctionnais mieux.

On lance donc la VM ArchLinux

Première chose à faire lors de l'initialisation c'est de set le keyboard en azerty avec loadkeys fr pour avoir un azerty comme j'utilise pour mon pc hôte. (Plus facile pour la suite)



Ensuite on va check la configuration internet, à ce moment-là j'étais encore en NAT donc c'est pourquoi j'ai une adresse IP en 10.0.2.15 :



Enfin on set le localtime à Europe/Paris pour que notre vm soit bien synchronisé au niveau de l'heure et de la date :

```
127 root@archiso ~ # timedatectl
               Local time: Wed 2025-11-19 11:05:49 UTC
           Universal time: Wed 2025-11-19 11:05:49 UTC
                 RTC time: Wed 2025-11-19 10:58:42
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

```
1 root@archiso ~ # timedatectl set-timezone Europe/Paris
root@archiso ~ # timedatectl
               Local time: Wed 2025-11-19 12:07:34 CET
           Universal time: Wed 2025-11-19 11:07:34 UTC
                 RTC time: Wed 2025-11-19 10:58:48
                Time zone: Europe/Paris (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

A. Installation

Initialisation des disques selon un schéma demandé que vous pouvez retrouver ci-dessous :

ID	MountPoint	Type de Partition	Taille	Rôle
1	/	Primaire (p)	1G	Système de fichiers racine
2	/boot	Primaire (p)	512M	Contient les noyaux/fichiers de démarrage
3	SWAP	Primaire (p)	2G	Espace d'échange
4	**Conteneur**	**Étendue (e)**	Reste	Conteneur pour les partitions logiques 5 et 6
5	/var	Logique (l)	4G	Données variables du système
6	/usr	Logique (l)	4G	Programmes et bibliothèques
7	/home	Logique (l)	Reste	Répertoires utilisateurs

```
Device     Boot    Start      End  Sectors Size Id Type
/dev/sda1           2048  4196351  4194304   2G 83 Linux
/dev/sda2        4196352  6293503  2097152   1G 83 Linux
/dev/sda3        6293504 14682111  8388608   4G 82 Linux swap / Solaris
/dev/sda4       14682112 67108863 52426752  25G  5 Extended
/dev/sda5       14684160 31461375 16777216   8G 83 Linux
/dev/sda6       31463424 48240639 16777216   8G 83 Linux
/dev/sda7       48242688 67108863 18866176   9G 83 Linux
```

Pour créer ces partitions il faut entrer dans fdisk de sda pour ensuite créer les partitions avec la commande n pour nouvelle partition puis suivre les étapes de construction de ces partitions.

Une fois créer, il faut formater les partitions avec les systèmes de fichiers appropriés selon leurs utilités :

```
root@archiso ~ # mkfs.ext4 /dev/sda1
mke2fs 1.47.3 (8-Jul-2025)
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 007b8787-b9e8-4476-a6ce-8006ed18f0b4
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@archiso ~ # mkfs.fat -F 32 /dev/sda2
mkfs.fat 4.2 (2021-01-31)
```

```
root@archiso ~ # mkswap /dev/sda3
Setting up swapspace version 1, size = 4 GiB (4294963200 bytes)
no label, UUID=98c554b8-db56-4265-addd-47399e539d1c
```

```
root@archiso ~ # mkfs.ext4 /dev/sda5
mke2fs 1.47.3 (8-Jul-2025)
Creating filesystem with 2097152 4k blocks and 524288 inodes
Filesystem UUID: 4dcdaff5-5feb-4e42-8bbf-161633674454d
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@archiso ~ # mkfs.ext4 /dev/sda6
mke2fs 1.47.3 (8-Jul-2025)
Creating filesystem with 2097152 4k blocks and 524288 inodes
Filesystem UUID: d92342b4-c254-4e9f-8a2f-8f1bed40e835
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
root@archiso ~ # mkfs.ext4 /dev/sda7
mke2fs 1.47.3 (8-Jul-2025)
Creating filesystem with 2358272 4k blocks and 589824 inodes
Filesystem UUID: 003764ee-c512-4ba6-a8ab-2279a045bd6c
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

Montage de la racine, activation du swap et montage des autres partitions avec les options de sécurités demandé :

```
root@archiso ~ # mount /dev/sda1 /mnt
root@archiso ~ # swapon /dev/sda3
root@archiso ~ # mkdir -p /mnt/boot /mnt/var /mnt/usr /mnt/home
root@archiso ~ # mount -o nosuid,nodev,noexec /dev/sda2 /mnt/boot
root@archiso ~ # mount -o nosuid,nodev,noexec /dev/sda5 /mnt/var
root@archiso ~ # mount -o nodev /dev/sda6 /mnt/usr
root@archiso ~ # mount -o nosuid,nodev,noexec /dev/sda7 /mnt/home
root@archiso ~ #
```

Utilisation de l'outil reflector pour récupérer les 20 miroirs de repo les plus rapides pour les écrire dans le fichier mirrorlist :

```
root@archiso ~ # reflector --country France --age 24 --protocol https --sort rate --save /etc/pacman.d/mirrorlist
```

Ce qui nous donne :

```
  GNU nano 8.6                                                                  /etc/pacman.d/mirrorlist
################################################################################
################## Arch Linux mirrorlist generated by Reflector ##################
################################################################################

# With:       reflector --country France --age 24 --protocol https --sort rate --save /etc/pacman.d/mirrorlist
# When:       2025-11-19 13:11:38 UTC
# From:       https://archlinux.org/mirrors/status/json/
# Retrieved:  2025-11-19 13:10:31 UTC
# Last Check: 2025-11-19 13:08:36 UTC

Server = https://mirror.thekinrar.fr/archlinux/$repo/os/$arch
Server = https://mirror.cyberbits.eu/archlinux/$repo/os/$arch
Server = https://fr.mirrors.cicku.me/archlinux/$repo/os/$arch
Server = https://mirrors.gandi.net/archlinux/$repo/os/$arch
Server = https://mirror.bakertelekom.fr/Arch/$repo/os/$arch
Server = https://archmirror.hogwarts.fr/$repo/os/$arch
Server = https://mirrors.jtremesay.org/archlinux/$repo/os/$arch
Server = https://mirror.peeres-telecom.fr/archlinux/$repo/os/$arch
Server = https://mirrors.eric.ovh/arch/$repo/os/$arch
Server = https://mirror.rznet.fr/archlinux/$repo/os/$arch
Server = https://archlinux.mailtunnel.eu/$repo/os/$arch
Server = https://mirror.trap.moe/archlinux/$repo/os/$arch
Server = https://mirror.theo546.fr/archlinux/$repo/os/$arch
Server = https://mirror.smayzy.ovh/archlinux/$repo/os/$arch
Server = https://arch.yourlabs.org/$repo/os/$arch
Server = https://mirror.its-tps.fr/archlinux/$repo/os/$arch
Server = https://f.matthieul.dev/mirror/archlinux/$repo/os/$arch
Server = https://mirror.wormhole.eu/archlinux/$repo/os/$arch
Server = https://mirror.oldsql.cc/archlinux/$repo/os/$arch
Server = https://archlinux.mirrors.ovh.net/archlinux/$repo/os/$arch
```

Installation de base fonctionnelle avec le noyau et les utilitaires de firmware ainsi que le microcode :

```
root@archiso ~ # pacstrap /mnt base linux linux-firmware amd-ucode nano man-db man-pages_
```

Validation du fichier fstab et rajout des 2 entrées demandé : /tmp et /proc avec les paramètres de sécurité demandé

```
  GNU nano 8.6                                                  /mnt/etc/fstab
# Static information about the filesystems.
# See fstab(5) for details.

# <file system> <dir> <type> <options> <dump> <pass>
# /dev/sda1
UUID=007b8787-b9e8-4476-a6ce-8006ed18f0b4       /               ext4            rw,relatime     0 1

# /dev/sda2
UUID=FD13-6511          /boot           vfat            rw,nosuid,nodev,noexec,relatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=mixed,utf8

# /dev/sda5
UUID=4dcdaff5-5feb-4e42-8bbf-16163367445d       /var            ext4            rw,nosuid,nodev,noexec,relatime 0 2

# /dev/sda6
UUID=d92342b4-c254-4e9f-8a2f-8f1bcd40e835       /usr            ext4            rw,nodev,relatime       0 2

# /dev/sda7
UUID=003764ee-c512-4ba6-a8ab-2279a045bd6c       /home           ext4            rw,nosuid,nodev,noexec,relatime 0 2

# /dev/sda3
UUID=98c554b8-db56-4265-addd-47399e539d1c       none            swap            defaults        0 0

tmpfs                                           /tmp            tmpfs           defaults,nosuid,nodev,noexec    0 0

proc                                            /proc           proc            defaults,hidepid=2      0 0
```

Comme les fichiers sont monté, on va pouvoir entrer dans le nouveau system chroot grâce au point d'entrer :

```
root@archiso ~ # arch-chroot /mnt
[root@archiso /]# _
```

Génération des fichiers de locales puis définition de la locale par défaut :

```
[root@archiso /]# locale-gen
Generating locales...
  en_US.UTF-8... done
  fr_FR.UTF-8... done
Generation complete.
```

```
[root@archiso /]# echo 'LANG="fr_FR.UTF-8"'>/etc/locale.conf
```

Configuration du clavier par défaut :

```
[root@archiso /]# echo 'KEYMAP=fr'>/etc/vconsole.conf
```

Configuration du hostname de la machine puis mise à jour de la résolution locale :

```
[root@archiso /]# echo 'monarchB3cyber' > /etc/hostname
  GNU nano 8.7
# Static table lookup for hostnames.
# See hosts(5) for details.
127.0.0.1        localhost
::1              localhost
127.0.1.1        monarchB3cyber
```

Ajout du HOOKS “usr” en tout dernier dans le fichier de conf /etc/mkinitcpio.conf :

```
HOOKS=(base systemd autodetect microcode modconf kms keyboard keymap sd-vconsole block filesystems fsck usr)
```

Puis reload de mkinitcpio :

```
[root@archiso /]# mkinitcpio -P
==> Building image from preset: /etc/mkinitcpio.d/linux.preset: 'default'
==> Using default configuration file: '/etc/mkinitcpio.conf'
  -> -k /boot/vmlinuz-linux -g /boot/initramfs-linux.img
==> Starting build: '6.17.8-arch1-1'
  -> Running build hook: [base]
  -> Running build hook: [systemd]
  -> Running build hook: [autodetect]
  -> Running build hook: [microcode]
  -> Running build hook: [modconf]
  -> Running build hook: [kms]
  -> Running build hook: [keyboard]
  -> Running build hook: [keymap]
  -> Running build hook: [sd-vconsole]
  -> Running build hook: [block]
  -> Running build hook: [filesystems]
  -> Running build hook: [fsck]
  -> Running build hook: [usr]
==> Generating module dependencies
==> Creating zstd-compressed initcpio image: '/boot/initramfs-linux.img'
  -> Early uncompressed CPIO image generation successful
==> Initcpio image generation successful
```

B. Configuration des données chiffrées sous /data

Installation de encrypt et lvm2 & ajout des HOOKS “encrypt” et “lvm2” (il faut bien respecter un ordre dans cette liste) :

```
HOOKS=(base systemd autodetect microcode modconf kms keyboard keymap sd-vconsole block encrypt lvm2 filesystems fsck usr)
```

Puis régénération de l’Initramfs encore une fois.

Partitionnement du disk 2 pour LUKS et la key-file (le moment où j’ai un seul disque mais que je coupe en deux pour faire croire que j’en ai 2 car je n’ai pas pu en rajouter un autre sur mon VirtualBox) :

```
Device     Boot  Start      End  Sectors  Size Id Type
/dev/sdb1         2048   206847   204800  100M 83 Linux
/dev/sdb2       206848 16777215 16570368  7.9G 8e Linux LVM
```

Key file formaté puis copie du fichier key file créé précédemment :

```
[root@archiso /]# mkfs.vfat -F 32 /dev/sdb1
mkfs.fat 4.2 (2021-01-31)
[root@archiso /]# mkdir /mnt/usb_key
mkdir: cannot create directory '/mnt/usb_key': File exists
[root@archiso /]# mount /dev/sdb1 /mnt/usb_key
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
[root@archiso /]# cp /keyfile /mnt/usb_key/keyfile
[root@archiso /]# umount /mnt/usb_key
[root@archiso /]# rmdir /mnt/usb_key
[root@archiso /]#
```

Puis chiffrage de la partition grâce au system Luks :

```
[root@archiso /]# cryptsetup luksFormat /dev/sdb2

WARNING!
========
This will overwrite data on /dev/sdb2 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sdb2:
Verify passphrase:
```

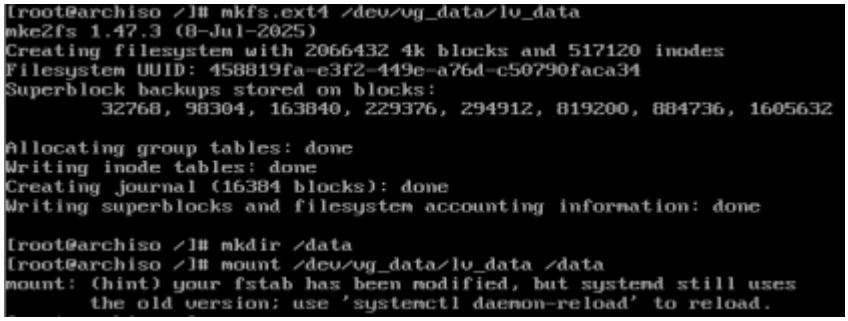
Ajout la key file au conteneur :

```
[root@archiso /]# cryptsetup luksAddKey /dev/sdb2 /keyfile
Enter any existing passphrase:
```

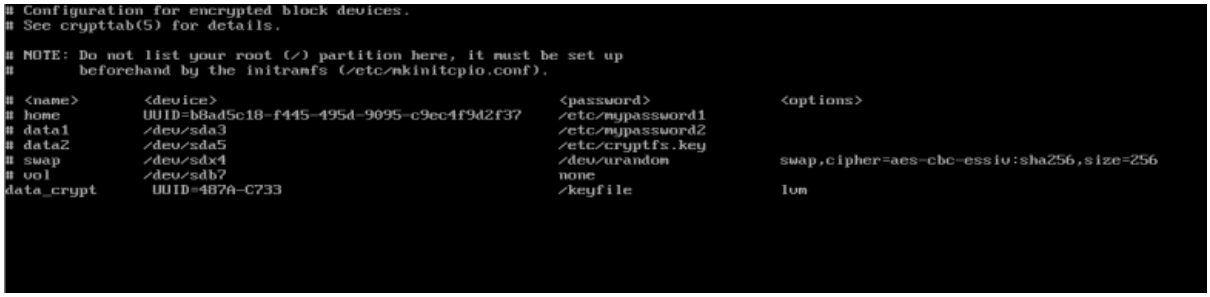
Création des volumes logiques :

```
[root@archiso /]# cryptsetup open /dev/sdb2 data_crypt
Enter passphrase for /dev/sdb2:
[root@archiso /]# pvcreate /dev/mapper/data_crypt
  Physical volume "/dev/mapper/data_crypt" successfully created.
[root@archiso /]# vgcreate vg_data /dev/mapper/data_crypt
  Volume group "vg_data" successfully created
[root@archiso /]# lvcreate -l 100%FREE vg_data -n lv_data
  Logical volume "lv_data" created.
[root@archiso /]#
```

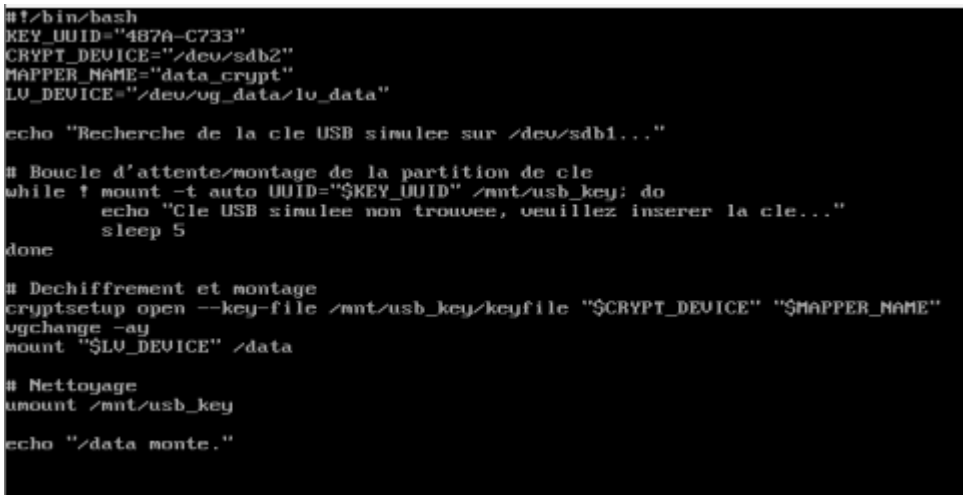
Formatage et création du point de montage /data :



Ajout de UUID dans /dev/sdb1qui est donc ma clé :



Script get_data.sh qui existe si le montage n'a pas été possible et qui recherche la clé USB (/dev/sdb1) jusqu'à ce qu'il la trouve et qu'il procède successivement au déchiffrement et au montage du dossier /data, de plus on doit le rendre exécutable avec chmod +x <fichier> :



2. Authentification

Définition du mot de passe du superutilisateur root en sachant qu'il sera désactivé plus tard :

```
[root@archiso /]# pssswd
bash: pssswd: command not found
[root@archiso /]# passwd
New password:
Retype new password:
passwd: password updated successfully
```

Ensuite on va créer les utilisateurs localadm et rescue en leurs définissant un mot de passe :

```
[root@archiso /]# useradd -m -g users -G wheel localadm
[root@archiso /]# passwd localadm
New password:
Retype new password:
passwd: password updated successfully
```

```
[root@archiso /]# useradd -m -g users -G wheel rescue
[root@archiso /]# passwd rescue
New password:
Retype new password:
passwd: password updated successfully
```

Le moyen le plus facile de désactiver le superutilisateur root est de désactiver la connexion directe en changeant son shell en un shell non-interactif :

```
[root@archiso /]# usermod -s /usr/bin/nologin root
```

Pour répondre à la problématique de :
Deux comptes doivent être créés :

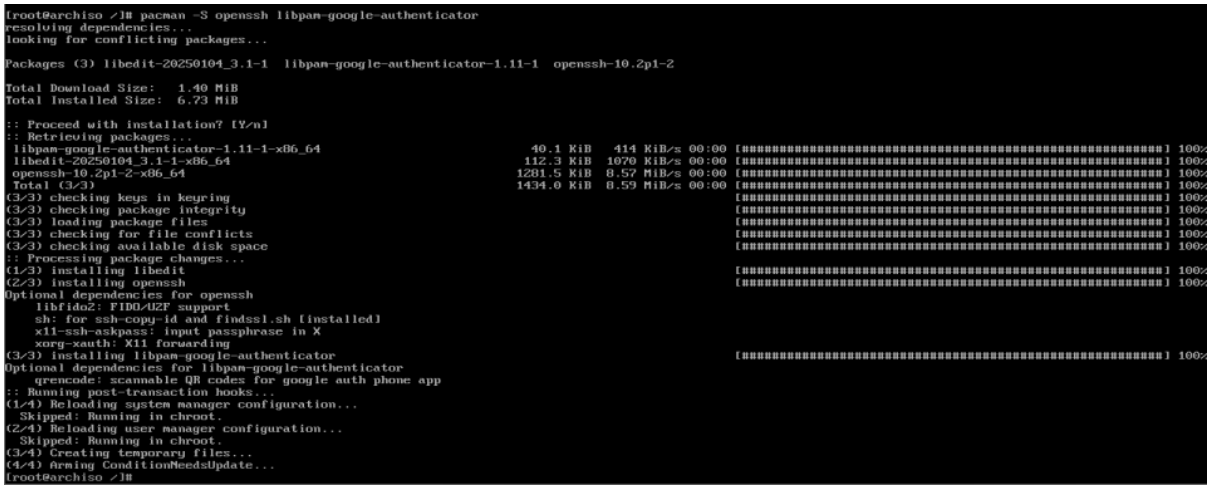
- localadm : Ce compte fait partie des sudoers (en NOPASSWD).
- rescue : Ce compte fait également partie des sudoers (en ALL).

On les rajoute les 2 au groupes wheel, on décommente la ligne qui dit que les users du groupes wheel (réservé au sudoer) doivent mettre leur mot de passe
De là les 2 utilisateurs font partis des sudoers avec obligation du mot de passe.
Pour faire en sorte que localadm n'ai pas cette obligation, on va rajouter une ligne :

localadm ALL=(ALL:ALL) NOPASWD: ALL

Grâce à ces changements, on répond donc à la problématique donnée.

Installation des paquets openssh et google-authenticator :



Activation du 2FA dans le fichier de conf sshd_config & bloquage du compte rescue :

```
# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the KbdInteractiveAuthentication and
# PasswordAuthentication.  Depending on your PAM configuration,
# PAM authentication via KbdInteractiveAuthentication may bypass
# the setting of "PermitRootLogin prohibit-password".
# If you just want the PAM account and session checks to run without
# PAM authentication, then enable this but set PasswordAuthentication
# and KbdInteractiveAuthentication to 'no'.
#UsePAM yes_
```

```
DenyUsers rescue
```

Configuration du pam pour le 2FA + suppression de la ligne :
auth include system-remote-login
Car on a eu un souci, lors des tests, la connexion SSH ne fonctionnait pas alors que si on enlevait cette ligne, on pouvait se connecter uniquement avec le code TOPT et plus avec le mot de passe en revanche.

```
#%PAM-1.0

auth       required   pam_google_authenticator.so
auth       include    system-remote-login
account    include    system-remote-login
password   include    system-remote-login
session    include    system-remote-login
```

Configuration de Google Authenticator pour l'user localadm :

```
[localadm@archiso ~]$ /usr/bin/google-authenticator

Do you want authentication tokens to be time-based (y/n)
Do you want authentication tokens to be time-based (y/n) y
Failed to use libqrencode to show QR code visually for scanning.
Consider typing the OTP secret into your app manually.
Your new secret key is: 4XTWUT5F2CREUKP4NWRQ3WQWK5YZSDGS
Enter code from app (-1 to skip): 896682
Code confirmed
Your emergency scratch codes are:
  88170904
  73089163
  53376582
  69365664
  52858481

Do you want me to update your "/home/localadm/.google_authenticator" file? (y/n)
```

On journalise ensuite dans un fichier de logs pour améliorer la lisibilité de l'utilisation des privilèges sudo :

```
[root@archiso /]# touch /var/log/sudo.log
[root@archiso /]# chmod 600 /var/log/sudo.log
```

On rajoute évidemment dans le fichier sudoer le chemin pour enregistrer les logs.

3. Bootloader

Installation de grub :

```
[root@archiso /]# pacman -S grub
resolving dependencies...
looking for conflicting packages...

Packages (1) grub-2:2.14rc1-2

Total Download Size:    7.54 MiB
Total Installed Size:  41.10 MiB

:: Proceed with installation? [Y/n]
:: Retrieving packages...
 grub-2:2.14rc1-2-x86_64                    7.5 MiB  25.8 MiB/s 00:00 [##########################################################] 100%
(1/1) checking keys in keyring                                       [##########################################################] 100%
(1/1) checking package integrity                                     [##########################################################] 100%
(1/1) loading package files                                          [##########################################################] 100%
(1/1) checking for file conflicts                                    [##########################################################] 100%
(1/1) checking available disk space                                  [##########################################################] 100%
:: Processing package changes...
(1/1) installing grub                                               [##########################################################] 100%
:: Install your bootloader and generate configuration with:
     # grub-install ...
     # grub-mkconfig -o /boot/grub/grub.cfg
Optional dependencies for grub
    dosfstools: For grub-mkrescue FAT FS and EFI support [installed]
    efibootmgr: For grub-install EFI support
    freetype2: For grub-mkfont usage
    fuse3: For grub-mount usage
    libisoburn: Provides xorriso for generating grub rescue iso using grub-mkrescue
    libusb: For grub-emu USB support [installed]
    lzop: For grub-mkrescue LZO support
    mtools: For grub-mkrescue FAT FS support
    os-prober: To detect other OSes when generating grub.cfg in BIOS systems
    sdl: For grub-emu SDL support
:: Running post-transaction hooks...
(1/1) Arming ConditionNeedsUpdate...
[root@archiso /]#
```

Sur le MBR :

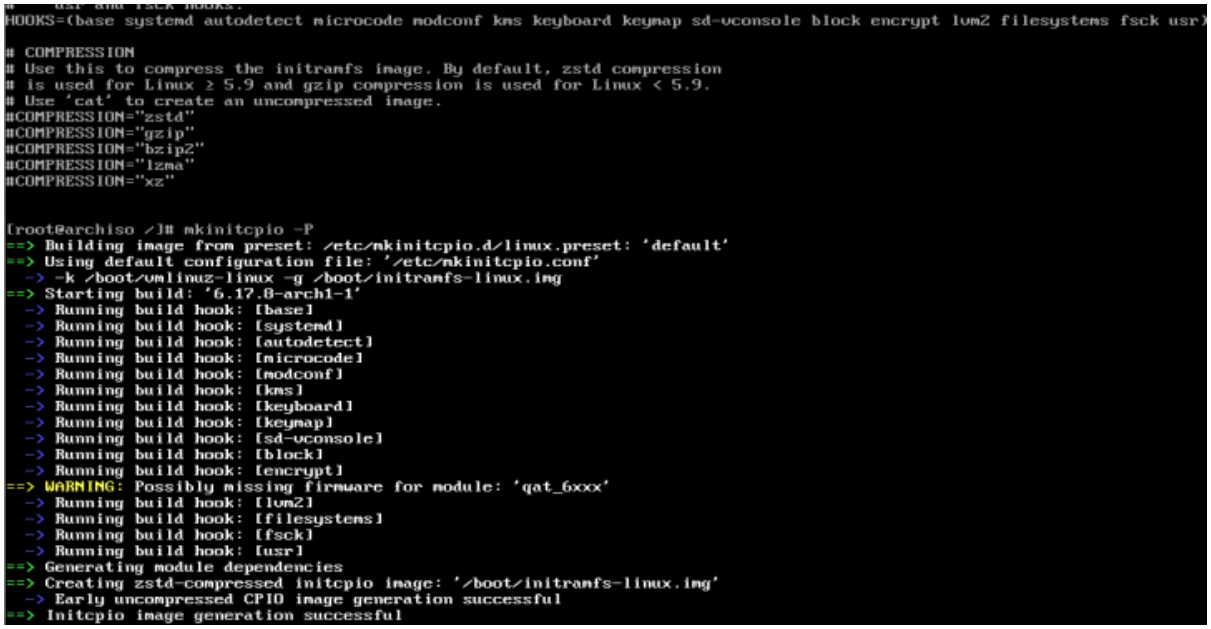
```
[root@archiso /]# grub-install /dev/sda
Installing for i386-pc platform.
Installation finished. No error reported.
```

Génération du MDP PBKDF2 pour ensuite copier le hash dans /etc/default/grub
(Ce fut une étape très longue car je n'avais pas le copier/coller, c'est fou comme je peux être à la fois très productif alors que je me complique beaucoup la tâche…)

Génération du fichier de configuration :

```
[root@archiso /]# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-linux
Found initrd image: /boot/amd-ucode.img /boot/initramfs-linux.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

Verification du fichier /etc/mkinitcpio.conf puis regénération de l'image

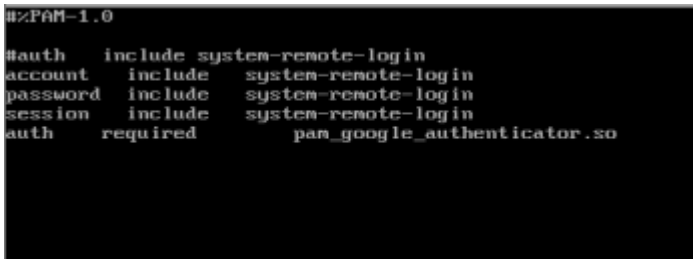


4. Test

- **Localadm**

La connexion en direct avec le compte localadm fonctionne très bien :

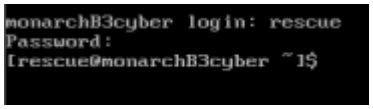
Cependant le test de SSH ne fut pas très concluant comme vous avez pu le remarquer. Je ne peux me connecter avec mon mot de passe ou le TOPT car il doit y avoir un conflit ou un souci de version.
En revanche lorsque je change le fichier de conf /etc/pam.d/sshd en mettant ceci :



La connexion est possible seulement je dois mettre uniquement mon TOPT et non mon mot de passe

- **Rescue**

La connexion avec le compte rescue fonctionne très bien en direct :



Lors de l'essai en SSH la connexion nous est bien refusé comme voulut.

- **Root**

Essaie de la connexion en direct avec le superutilisateur root qui échoue comme voulut & de même pour SSH :

2. TP - Hardening Linux lvl 2

L'objectif de ce TP 2 est de construire une version améliorée de la machine issue du TP "Hardening Linux lvl 1" en proposant les mesures de sécurité complémentaires.

1. Paramétrage du noyau

Premièrement, on va s'occuper de paramétrer le noyau pour suivre les bonnes pratiques de configuration recommandée par l'ANSSI. On peut donc les trouver sur internet et les suivre, les voici d'ailleurs ci-dessous :

List of recommended kernel configuration options

The detailed kernel configuration options are presented as seen in the `sysctl.conf` configuration file:

```
# Restrict access to the dmesg buffer (equivalent to
# CONFIG_SECURITY_DMESG_RESTRICT=y)
kernel.dmesg_restrict=1
# Hide kernel addresses in /proc and various other interfaces,
# including from privileged users
kernel.kptr_restrict=2
# Explicitly specify the process id space supported by the kernel,
# 65536 being an example value
kernel.pid_max=65536
# Restrict the use of the perf subsystem
kernel.perf_cpu_time_max_percent=1
kernel.perf_event_max_sample_rate=1
# Prohibit unprivileged access to the perf_event_open () system call.
# With a value greater than 2, we impose the possession of
# CAP_SYS_ADMIN, in order to collect the perf events.
kernel.perf_event_paranoid=2
# Activate ASLR
kernel.randomize_va_space=2
# Disable Magic System Request Key combinations
kernel.sysrq=0
# Restrict kernel BPF usage to privileged users
kernel.unprivileged_bpf_disabled=1
# Completely shut down the system if the Linux kernel behaves
# unexpectedly
kernel.panic_on_oops=1
```

Ci-dessous le résultat de mes commandes une à une :

Restrict access to the dmesg buffer

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.dmesg_restrict=1
kernel.dmesg_restrict = 1
```

Restrict use of ptrace to child processes

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.yama.ptrace_scope=1
kernel.yama.ptrace_scope = 1
```

Prevent kernel logging from leaking to users

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.kptr_restrict=2
kernel.kptr_restrict = 2
```

Limit the CPU usage for each user

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.pid_max=65536
kernel.pid_max = 65536
```

Limit the CPU time for the perf subsystem

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.perf_cpu_time_max_percent=1
kernel.perf_cpu_time_max_percent = 1
```

Limit the sample rate for perf events

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.perf_event_max_sample_rate=1
kernel.perf_event_max_sample_rate = 1
```

Enable the perf events paranoid setting

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.perf_event_paranoid=2
kernel.perf_event_paranoid = 2
```

Disable Magic SysRq key combinations

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.sysrq=0
kernel.sysrq = 0
```

Disable BPF for unprivileged users

```
[localadm@monarchB3cyber kernel]$ sudo sysctl -w kernel.unprivileged_bpf_disabled=1
kernel.unprivileged_bpf_disabled = 1
```

On ajoute tout cela dans le fichier /etc/sysctl.conf pour que les règles soient appliquées au démarrage :

```
  GNU nano 8.7                                                  sysctl.conf
kernel.dmesg_restrict=1
kernel.yama.ptrace_scope=1
kernel.kptr_restrict=2
kernel.pid_max=65536
kernel.perf_cpu_time_max_percent=1
kernel.perf_event_max_sample_rate=1
kernel.perf_event_paranoid=2
kernel.sysrq=0
kernel.unprivileged_bpf_disabled=1
```

Puis rechargement des paramètres du noyau :

```
[localadm@monarchB3cyber etc]$ sudo sysctl -p
kernel.dmesg_restrict = 1
kernel.yama.ptrace_scope = 1
kernel.kptr_restrict = 2
kernel.pid_max = 65536
kernel.perf_cpu_time_max_percent = 1
kernel.perf_event_max_sample_rate = 1
kernel.perf_event_paranoid = 2
kernel.sysrq = 0
kernel.unprivileged_bpf_disabled = 1
```

On produit donc ensuite un script de validation de conformité de la configuration :

```
  GNU nano 8.7                                    check_kernel_params.sh
#!/bin/bash

declare -A params=(
        ["kernel.dmesg_restrict"]="1"
        ["kernel.yama.ptrace_scope"]="1"
        ["kernel.kptr_restrict"]="2"
        ["kernel.pid_max"]="65536"
        ["kernel.perf_cpu_time_max_percent"]="1"
        ["kernel.perf_event_max_sample_rate"]="1"
        ["kernel.perf_event_paranoid"]="2"
        ["kernel.sysrq"]="0"
        ["kernel.unprivileged_bpf_disabled"]="1"
)

for param in "${!params[@]}"; do
        current_value=$(sysctl -n $param 2>/dev/null)
        expected_value=${params[$param]}

        if [ "$current_value" != "$exptected_value" ]; then
                echo "Non-conformite detectee pour $param. Valeur actuelle: $current_value, valeur attendue: $expected_value"
        else
                echo "$param est conforme avec la valeur $current_value"
        fi
done
```

Et toutes les modifications des paramètres du noyau seront détectées et journalisées dans un fichiers log :
/var/log/kernel_modif.log

```
  GNU nano 8.7                                      log_kernel_modif.sh
#!/bin/bash

sysctl -a | while read line; do
        echo "$(date) - $line" >> /var/log/kernel_modif.log
done
```

2. Authentification utilisateur

Installation de libpwdquality :

```
[localadm@monarchB3cyber /]$ sudo pacman -S libpwquality
[sudo] Mot de passe de localadm :
résolution des dépendances...
recherche des conflits entre paquets...

Paquets (2) cracklib-2.10.3-1  libpwquality-1.4.5-6

Taille totale du téléchargement :  0,36 MiB
Taille totale installée :         1,34 MiB

:: Procéder à l'installation ? [O/n]
:: Récupération des paquets...
 libpwquality-1.4.5-6-x86_64                    93,0 KiB   911 KiB/s 00:00 [##########################################################] 100%
 cracklib-2.10.3-1-x86_64                      277,2 KiB  2009 KiB/s 00:00 [##########################################################] 100%
 Total (2/2)                                   371,0 KiB  2,38 MiB/s 00:00 [##########################################################] 100%
(2/2) vérification des clés dans le trousseau                              [##########################################################] 100%
(2/2) vérification de l'intégrité des paquets                              [##########################################################] 100%
(2/2) chargement des fichiers des paquets                                  [##########################################################] 100%
(2/2) analyse des conflits entre fichiers                                  [##########################################################] 100%
(2/2) vérification de l'espace disque disponible                           [##########################################################] 100%
:: Traitement des changements du paquet...
(1/2) installation de cracklib                                             [##########################################################] 100%
(2/2) installation de libpwquality                                         [##########################################################] 100%
Dépendances optionnelles pour libpwquality
    python: Python bindings
:: Exécution des crochets (« hooks ») de post-transaction...
(1/1) Arming ConditionNeedsUpdate...
```

Parametrage du fichier /etc/security/pwquality.conf selon les consignes :

```
minlen = 16
minclass = 4
dcredit = 2
lcredit = 2
ucredit = 2
ocredit = 2
maxrepeat = 2
maxsequence = 1
usercheck = 1
enforcing = 1_
```

Ajout de la ligne pour que le compte se bloque pendant 15 minutes après 3 échecs :

```
auth     required       pam_tally2.so deny=3 unlock_time=900
```

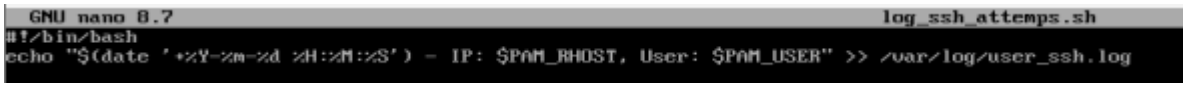
Ajout de la règle que l'utilisateur ne peut se connecter uniquement entre 8h et 20h :

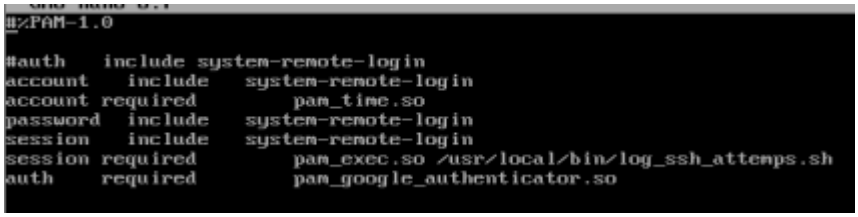
```
  GNU nano 8.7                                          /etc/security/time.conf
#
# users
#       is a logic list of users or a netgroup of users to whom this
#       rule applies.
#
# NB. For these items the simple wildcard '*' may be used only once.
#
# times
#       the format here is a logic list of day/time-range
#       entries the days are specified by a sequence of two character
#       entries, MoTuSa for example is Monday Tuesday and Saturday. Note
#       that repeated days are unset MoMo = no day, and MoWk = all weekdays
#       bar Monday. The two character combinations accepted are
#
#               Mo Tu We Th Fr Sa Su Wk Wd Al
#
#       the last two being week-end days and all 7 days of the week
#       respectively. As a final example, AlFr means all days except Friday.
#
#       each day/time-range can be prefixed with a '!' to indicate "anything
#       but"
#
#       The time-range part is two 24-hour times HHMM separated by a hyphen
#       indicating the start and finish time (if the finish time is smaller
#       than the start time it is deemed to apply on the following day).
#
# for a rule to be active, ALL of service+ttys+users must be satisfied
# by the applying process.
#
#
# Here is a simple example: running blank on tty* (any ttyXXX device),
# the users 'you' and 'me' are denied service all of the time
#
#blank;tty* & !ttyp*;you|me;!Al0000-2400

# Another silly example, user 'root' is denied xsh access
# from pseudo terminals at the weekend and on mondays.

#xsh;ttyp*;root;!WdMo0000-2400

#
# End of example file.
#
sshd;*;$lt;localadm$gt;;Al0800-2000_
                                                        [ 66 lignes écrites ]
```

Configuration du fichier /etc/ssh/sshd_config pour mettre en place une restriction d'authentification SSH spécifique

Il faut bien penser que ces arguments doivent exister et ne pas être commenté :
PasswordAuthentication no
ChallengeResponseAuthentication yes
UsePAM yes
AuthenticationMethods publickey,keyboard-interactive

Création d'un script de journalisation exécuté au travers du module pam_exec :





3. Audit utilisateur

Activation de auditd :

FIN