PROYECTO EN ARCH LINUX

A. Fundamentos del Cifrado de Disco y Almacenamiento

A.1. Cifrado de Disco Completo (FDE) con LUKS

LUKS (*Linux Unified Key Setup*) es el estándar de cifrado a nivel de bloque para Linux. En lugar de cifrar solo archivos, cifra la partición completa (excepto la partición /boot). Esto asegura que incluso los metadatos del sistema de archivos y el *swap* están protegidos, haciendo imposible la recuperación de datos sin la llave correcta.

Concepto: El Encabezado LUKS almacena los metadatos de cifrado y los Key Slots (ranuras de clave) que contienen las claves cifradas que, a su vez, pueden descifrar la clave maestra del volumen.

A.2. Esquema de Particionado para FDE

Objetivo: Crear una estructura de disco donde el sistema operativo y los datos están protegidos, pero los archivos necesarios para el *pre-arranque* (el kernel) no lo están.

Partición	Uso	Cifrado	Razón
/dev/sda1	/boot	**No**	Debe ser visible para el *firmware* (BIOS/UEFI) y el gestor de arranque (GRUB) antes de que el sistema pueda cargar el módulo de descifrado.
/dev/sda2	Raíz (/)	**Sí**	Contiene el sistema operativo, archivos de usuario y datos sensibles.

Comando Clave (Ejemplo con fdisk):

Bash

```
fdisk /dev/sda

# 1. Crear /dev/sda1 (ej: 512MB) - Tipo Linux

# 2. Crear /dev/sda2 (el resto del espacio) - Tipo Linux
```

A.3. Gestión de Key Slots y Llaves de Acceso

Un volumen LUKS tiene 8 *slots* (ranuras) de clave (del 0 al 7). Cada slot puede almacenar una clave distinta (una contraseña, un *Keyfile* o una frase de paso).

- Al cifrar inicialmente, se ocupa el Slot 0 (con la contraseña principal).

- Posteriormente, usaremos otro slot (ej: Slot 1) para el *Keyfile* USB.
- Si se pierde la USB, se puede usar la contraseña del Slot 0. Si se compromete el Slot 0, se puede eliminar sin afectar las demás.

Comando para Cifrar (Iniciando el FDE):

Bash

Cifra la partición /dev/sda2 y pide la contraseña principal (Slot 0)

cryptsetup luksFormat /dev/sda2

A.4. Cifrado de Ficheros vs. Cifrado de Bloque

Este proyecto utilizara cifrado de bloque (LUKS), que es superior al cifrado de ficheros (como eCryptfs o GPG) porque:

1. **Metadatos:** Protege el tamaño de los archivos, los nombres y la estructura de directorios (que son visibles en el cifrado de ficheros).
2. **Rendimiento:** Opera a nivel de disco, con un *overhead* mínimo.

A.5. Configuración de LVM sobre LUKS (Opcional Avanzado)

Recomendación: Para usuarios que desean mayor flexibilidad, se recomienda configurar LVM (*Logical Volume Manager*) después de abrir el volumen LUKS. Esto permite crear múltiples volúmenes lógicos (e.g., root, home, swap) dentro del contenedor cifrado único, facilitando el cambio de tamaño sin afectar el cifrado.

Secuencia de Comandos (Después de abrir el volumen):

Bash

Abrir el volumen LUKS (si no está abierto)

cryptsetup open /dev/sda2 cryptvol

Crear un Grupo de Volúmenes (Volume Group)

pvcreate /dev/mapper/cryptvol

vgcreate vg0 /dev/mapper/cryptvol

Crear Volúmenes Lógicos (ej: root y home)

lvcreate -L 20G vg0 -n root

lvcreate -l 100%FREE vg0 -n home

B. Implementación de la Llave USB y Configuración de Arranque

B.1. Generación y Protección del Keyfile

La seguridad del *Keyfile* radica en su aleatoriedad y en la restricción estricta de sus permisos.

1. **Montar la USB:** Asumiendo que la USB se monta en /mnt/usb.
2. **Generar datos aleatorios:** Usamos /dev/urandom para garantizar un alto nivel de entropía.

Comandos Clave:

Bash

Genera un Keyfile de 4096 bytes con datos aleatorios

dd if=/dev/urandom of=/mnt/usb/usb_keyfile.bin bs=4096 count=1

Restringir permisos: Nadie (excepto root) puede leer, escribir o ejecutar.

Esto previene que una aplicación maliciosa lo copie si la USB está conectada.

chmod 000 /mnt/usb/usb_keyfile.bin

B.2. Identificación Persistente de Dispositivos (UUID)

Los nombres de dispositivo como /dev/sdb o /dev/sdc son dinámicos y cambian. Usaremos el UUID del sistema de archivos de la USB para referirnos a ella, asegurando que el sistema siempre encuentre la llave correcta, independientemente del puerto en el que se conecte.

Comando para Obtener el UUID de la USB:

Bash

lsblk -f

Buscar la línea correspondiente a tu partición USB (ej: /dev/sdb1)

Ejemplo de salida: /dev/sdb1 vfat 1A2B-C3D4...

Copiar el UUID (e.g., 1A2B-C3D4)

B.3. Configuración del Mapeo de Cifrado (/etc/crypttab)

Propósito: Este archivo le dice al sistema de *pre-arranque* (el initramfs) cómo abrir el volumen cifrado.

Estructura del archivo (/etc/crypttab):

<nombre_mapeo> <dispositivo_cifrado> <llave_de_acceso> <opciones>

Ejemplo Específico:

Bash

Usando el UUID para la partición USB

1. Nombre para el volumen abierto | 2. Partición LUKS | 3. RUTA al keyfile en la USB | 4. Opciones

cryptvol /dev/sda2 /dev/disk/by-uuid/1A2B-C3D4:/usb_keyfile.bin luks,keyscript=/usr/bin/cat,retry=5

Nota: El keyscript asegura que el contenido binario del Keyfile se pase correctamente como clave.

B.4. Modificación y Reconstrucción del initramfs

El initramfs (ramdisk de inicio) es el entorno más crucial. Si el kernel no tiene los módulos para leer USB y LUKS, el arranque fallará.

1. **Editar /etc/mkinitcpio.conf:** Añadir los *hooks* necesarios en la secuencia correcta.

Línea HOOKS modificada (Crucial la secuencia):

Bash

HOOKS=(base udev autodetect modconf block **usb keymap encrypt** filesystems keyboard fsck)

- **usb:** Módulo necesario para la detección del hardware USB.
- **keymap:** Para asegurar que el teclado sea funcional (si es necesario).
- **encrypt:** El módulo que gestiona el desbloqueo de LUKS.

2. **Reconstruir la imagen de arranque (¡Obligatorio!):**

Bash

mkinitcpio -P

B.5. Configuración del Gestor de Arranque (GRUB)

Objetivo: Asegurarse de que el gestor de arranque (GRUB) pase los parámetros correctos al kernel para que sepa qué hacer con el volumen cifrado.

1. **Editar /etc/default/grub:**

Línea GRUB_CMDLINE_LINUX (Añadir parámetros):

Bash

GRUB_CMDLINE_LINUX="cryptdevice=/dev/sda2:cryptvol root=/dev/mapper/vg0-root resume=/dev/mapper/vg0-swap"

- cryptdevice=/dev/sda2:cryptvol: Le dice al kernel que /dev/sda2 es un dispositivo cifrado que debe abrirse y mapearse como cryptvol.

2. **Actualizar GRUB (¡Obligatorio!):**

Bash

grub-mkconfig -o /boot/grub/grub.cfg

C. Control de Acceso y Autenticación de Usuario (PAM)

C.1. Sistema PAM (Pluggable Authentication Modules)

PAM actúa como un intermediario entre una aplicación (e.g., login, gdm, sudo) y los mecanismos de autenticación. Permite agregar módulos de seguridad (como *pam_usb*) sin modificar el código de las aplicaciones.

C.2. Uso del Módulo pam_usb

Objetivo: Extender la verificación USB más allá del arranque, forzando la presencia de la llave para el inicio de sesión del usuario.

1. **Instalar:**

Bash

Podría estar en AUR, usar yay o similar

pacman -S pam_usb

2. **Configurar Dispositivo y Usuario:**

Bash

1. Registrar la llave USB (usando su serial único)

pamusb-conf --add-device mi_llave_principal

2. Vincular el usuario a la llave registrada

pamusb-conf --add-user tu_usuario --device mi_llave_principal

3. Habilitar en PAM (Modificar /etc/pam.d/system-auth):

Añadir la línea de pam_usb al principio de la sección auth:

... otras configuraciones

auth sufficient pam_usb.so # <--- AÑADIR ESTA LÍNEA

auth required pam_unix.so try_first_pass

...

El parámetro sufficient significa que si la autenticación por USB tiene éxito, no necesita la contraseña (pam_unix.so), cumpliendo el requisito de acceso sin contraseña.

C.3. Reglas de udev para Detección de USB

Se pueden crear reglas udev para ejecutar scripts de manera instantánea cuando la llave USB se retira.

Regla de Ejemplo (en /etc/udev/rules.d/99-usb-lock.rules):

SUBSYSTEM=="usb", ATTR{serial}=="TU_NUMERO_SERIAL", ACTION=="remove", RUN+="/usr/local/bin/bloquear_sesion.sh"

Donde /usr/local/bin/bloquear_sesion.sh es un script que contiene el comando para bloquear la pantalla (e.g., loginctl lock-session).

C.4. Gestión de la Sesión (Bloqueo/Desbloqueo)

El script activado por udev debe usar la herramienta adecuada para forzar el bloqueo, dependiendo del entorno de escritorio:

- **GNOME:** gdbus call --session --dest org.gnome.ScreenSaver --object-path /org/gnome/ScreenSaver --method org.gnome.ScreenSaver.Lock
- **General (systemd):** loginctl lock-session

D. Seguridad, Mantenimiento y Recuperación

D.1. Backup del Encabezado LUKS (¡Crítico!)

El encabezado LUKS contiene la clave maestra cifrada y la información de los *Key Slots*. Si se corrompe (por un error de disco o un error de *software*), todos los datos se vuelven irrecuperables.

Comando Clave (Ejecutar inmediatamente después del cifrado):

Bash

Guardar el encabezado en un archivo seguro, fuera del disco cifrado

cryptsetup luksHeaderBackup /dev/sda2 --header-backup-file /ruta/segura/header_backup.bin

D.2. Revocación de Llaves (Eliminación de Slots)

Si la llave USB se pierde o es robada, debes invalidarla inmediatamente. Esto se hace eliminando la clave del *slot* asociado.

1. **Verificar el slot del Keyfile:**

Bash

cryptsetup luksDump /dev/sda2 | grep "Key Slot"

2. **Eliminar la llave (ej: Slot 1):**

Bash

Se requerirá una clave válida existente (ej: la contraseña principal) para confirmar

cryptsetup luksKillKey /dev/sda2 1

D.3. Proceso de Actualización de Arch Linux

Recordatorio: Las actualizaciones importantes del kernel de Arch pueden requerir una reconstrucción de la imagen initramfs.

- **Después de cada actualización del paquete linux:** siempre ejecuta mkinitcpio -P y, si actualizaste GRUB, grub-mkconfig -o /boot/grub/grub.cfg.

D.4. Modificación de la Contraseña/Keyfile Principal

Se puede cambiar o reemplazar cualquier clave sin afectar los datos.

1. **Cambiar una contraseña existente (Slot 0):**

Bash

cryptsetup luksChangeKey /dev/sda2

2. **Generar una nueva llave de recuperación (USB distinta):**

- Genera un nuevo *Keyfile* (Sección B.1).
- Añádelo como una clave nueva al siguiente slot disponible (ej: Slot 2):

Bash

```
cryptsetup luksAddKey /dev/sda2 /mnt/usb_nueva/nueva_ke
```