# 2022.05.18 Sesión Forense

**Objetivo Sesión**:

- Ver particiones, sistemas de archivos, cómo montar, cómo formatear.
- Necesario porque se trabaja muy seguido con imágenes forenses (copia fiel a un dispositivo de almacenamiento)
- Formatos de imágenes forenses:
  - dd - Más famoso en ctf porque es el formato en crudo de un dispositivo de almacenamiento. En el campo laboral, si bien se ocupa, no es el más utilizado
  - ewf.- Puede almacenar datos del caso, hashes, nombres de los analistas, etc
- Los formatos mas utilizados son de una tecnología particular como ewf que es de ENCASE

```
foo@bar:~$ fdisk -l
```

- Se pueden ver los dispositivos conectados.
- /dev/sda
- Muestra las particiones de los dispositivos conectados, donde inicia y termina cada partición y la cantidad de sectores que la conforman y si es booteable o no

```
foo@bar:~$ lsblk
```

- blk hace referencia a block. Un dispositivo de bloques es un dispositivo de almacenamiento porque el almacenamiento se lleva a cabo en conjunto de bloques, conjunto de bytes. List block.
- "brw-rw----" La b cuando se hace un ls -l /dev/sda hace referencia a que tratamos con un dispositivo de bloque.
- Se pueden ver los dispositivos de bloques conectados (disco duro, usb), sus particiones.
- Útil porque nos dice dónde están montados estos dispositivos.
  - Montado: Que se puede acceder a sus archivos porque así como lo tenemos sólo está conectado. Que todo su sistema de archivos se mapea al nuestro.

```
foo@bar:~$ ls /mnt
```

- Por defecto tiene varias carpetas para montar windows, ewf, e01 pero sólo son carpetas así que se puede utilizar la que queramos.

**Para montar**

```
foo@bar:~$ mount /dev/sdb1 /mnt/usb
```

- Indicar la carpeta en la que queremos montar y qué dispositivo queremos montar.

**Para desmontar**

```
foo@bar:~$ umount /mnt/usb
```

- Indicar el punto de montaje.
- No se puede si se está dentro de la carpeta, se está siendo utilizado. Si hay un proceso que está escribiendo en este punto de montaje no se puede desmontar. A esto se refiere desconectar de manera segura.
- Antes de poderlo desconectar necesitamos desmontarlo para asegurarnos que ya termino cualquier operación de escritura/lectura de este dispositivo y que no va a fallar el proceso.

```
foo@bar:~$ ls /dev/sd*
```

- Se enlista el dispositivo principal como archivo así como las particiones.

```
foo@bar:~$ fsstat /dev/sdb
```

- Se puede ver el sistema de archivos que está asociado a una partición.
- Si se realiza directamente en una USB, nos marcará que no encuentra cuál es el sistema de archivos.
    - Esto se debe a que busca desde el inicio de la usb (su primer sector), porque generalmente en el inicio de las particiones se indica que tipo de sistema de archivos va a estar utilizando.

```
foo@bar:~$ fsstat /dev/sdb1
```

- Nos indica el sistema de archivos, p.e. EXT3, NTFS.

Cada sistema de archivos implementa sus propias reglas de cómo almacena los archivos, los datos, metadatos, cómo se distribuye, cómo accede a los datos. Esto es importante por las marcas de tiempo porque indican cómo un archivo se modifico, cuando se accedió a él. La marca de tiempo de creación no existe para todos los archivos

```
foo@bar:~$ fsstat /dev/sdb -o 2048 | head
```

- Se le indica el offset para que no empiece desde el inicio, por ejemplo 2048 (indicado por fdisk -l).

Lo primero que está en el dispositivo es una tabla de particiones, la cual indica cómo se van a distribuir esas particiones, si son primarias, extendidas, tamaños. Toda esa información se almacena en el MBR (Master Boot Record) y por eso se salta esos primeros sectores.

Diferencia entre GPT y MBR

- MBR tiene un máximo de 4 particiones primarias, de las cuales la 4ta puede no ser primaria sino extendida y así obtener más particiones. También sólo funciona con dispositivos de hasta 2TB, así que es común verlo en computadoras de trabajo. GPT es más nuevo y puede tener hasta 128 particiones. Éste se suele ver en servidores donde el almacenamiento es mucho mayor.

**Creación de nuestra primera imagen forense**

Copia bit a bit: Es totalmente fiel a la original, pues si uno calcula el hash de todo su contenido va a ser exactamente igual el mismo el de la copia que el de la original. La copia se realiza por bloques.

```
foo@bar:~$ dd if=/dev/sdb of=usb.dd
```

- En un CTF es común ver archivos img y dd que hacen referencia a una imagen en crudo, copia tal cual
- Nos permitirá copiar un archivo en otro archivo y lo hace de tal manera que sea confiable para hacer imágenes forenses.
- Recibe el archivo de entrada "input file" y el nombre del archivo de salida "output file"
- No da una especie de feedback, no sabemos cuánto tarda. Esto depende de varios factores: capacidad del dispositivo de almacenamiento, velocidad de la conexión USB, del dispositivo en el que se está escribiendo. Es por eso que no es tan útil.

```
foo@bar:~$ dcfldd if=/dev/sdb of=usb.dd hash=sha1
```

- Versión de dd con esteroides.
- Podemos pedirle que nos calcule el hash y cuál es el algoritmo de hashing.

dd sí es un comando confiable. Pero en el campo laboral casi no se ocupa porque exigen que se usen herramientas con ciertas certificaciones, por lo que se usan herramientas de paga muy grandes como ENCASE.
Se puede modificar el tamaño de bloque y así mejorar la velocidad.
En la vida laboral siempre hay que revisar la hash porque hay que ver que no se haya modificado la evidencia.

```
foo@bar:~$ fdisk -l usb.dd
```

- Nos da la misma información como si fuera un dispositivo de bloque.

```
foo@bar:~$ ls -l
total 15155204
-rw-r--r-- 1 root root 15518924800 May 17 18:34 usb.dd
```

- Aparece como un archivo regular, no de bloque. De esta manera no lo podríamos montar tan directamente en una carpeta.

Técnica para montar: Mapear un archivo regular a un dispositivo loop (de bucle)

- Este dispositivo mapea el archivo como si se trata de un archivo de bloques.

```
foo@bar:~$ losetup -a
```

- Nos indica que dispositivos de bucle están siendo utilizados.
  losetup -a (que dispositivos de bluce estan siendo utilizados)

- Nos indica el dispositivo de bucle (/dev/loop0 p.e.) , a qué archivo regular está asociado, cual es su inicio y cual su final.

```
foo@bar:~$ losetup -f usb.dd -o $((2048*512)) --sizelimit=$((12582912*512))
```

- Le especificamos qué archivo queremos que lea, el offset en el que se especifica por donde inicia (no sabe por defecto el tamaño de los sectores, asi que se especifica el tamaño del sector y en que sector empieza) y el límite de tamaño, en el que se especifica en donde termina (se especifica el tamaño del sector y en que sector termina).

```
foo@bar:~$ ls /dev/loop*
```

- Se puede observar que hay varios dispositivos de bucle. Por defecto existen 7-8.

El siguiente paso es montar la imagen de la usb.

```
foo@bar:~$ mount /dev/loop0 /mnt/usb
```

Resumen de los pasos:

1. Asociar la partición a un dispositivo de bucle
2. Montar el dispositivo de bucle como si fuera un dispositivo normal en alguna carpeta que nosotros queramos.

Se pueden crear archivos nuevos dentro de la imagen porque lo montamos en una carpeta de nuestro sistema. Cuando se realiza un análisis forense, se debe montar con ciertas características para proteger la evidencia:

```
foo@bar:~$ mount /dev/loop0 /mnt/usb -o ro,noexec
```

- ro significa readonly, solo lectura. Por el otro lado, noexec no nos permitirá ejecutar nada que esté en ese punto de montaje (útil porque ahí puede haber malware que se pueda ejecutar. )

Se recomienda hacer copias del dispositivo original, pues éste ya no se toca y se resguarda. Se recomiendan dos copias del dispositivo: copia de trabajo y copia de respaldo

```
foo@bar:~$ losetup -d /dev/loop1
```

- Permite borrar un mapeado específico.

```
foo@bar:~$ losetup -D
```

- Borra todos los archivos de bucle.

```
foo@bar:~$ losetup -Pf usb.dd
```

- Con -P losetup se encarga de buscar dónde inicia y dónde termina y automáticamente asociar las diferentes particiones a un dispositivo de bucle.

Se crea un archivo lleno de puros cero y se comienza a particionar para ver las tablas de particiones de mbr.

```
foo@bar:~$ dd if=/dev/zero of=img.dd bs/1M count=100
```

- Con bs se especifica el tamaño del bloque y que nos copie un número determinado de bloques.
    - Linux tiene archivos especiales de utilidad para diferentes escenarios
    - /dev/null: Es como la nada, cuando se tiene errores que no queremos que se impriman y se manda a la basura.
    - /dev/random: Nos entrega tantos números aleatorios como le solicitemos. Similar a /dev/urandom
    - /dev/zero: Nos entrega tantos ceros como nosotros le solicitemos.

```
foo@bar:~$ hexdump -C img.dd | less
```

- Si queremos ver el contenido del archivo creado, podemos usar hexdump.

```
foo@bar:~$ fdisk img.dd
```

- Se puede particionar un archivo regular.
- Opciones útiles:
    - m para ver la ayuda.
    - p para que nos muestre cómo está actualmente la tabla de particiones.
    - n para crear una nueva partición.
        - Crea partición primaria o extendida
        - Número de partición.
        - Sector donde inicia (default 2048)
        - Sector donde termina, ya sea el número del sector o el tamaño que se quiera, p.e. +10M.
    - w Confirma las acciones (crear tabla de particiones p.e.) y se sale de fdisk.

```
foo@bar:~$ losetup -f img.dd -o $((2048*512)) --sizelimit=$((22527*512))
```

- Para asignarles sistemas de archivos a cada sistema se utiliza nuevamente losetup, pues primero se asocia con un dispositivo loop para que se maneje como un sistema de bloques.

Si se intenta montar con "mount /dev/loop0 /mnt/usb" dará error porque no hay un sistema de archivos asociado.

```
foo@bar:~$ mkfs.ext4 /dev/loop0
```

- Le asigna el sistema de archivos ext4 al dispositivo loop. Después ya se puede montar con "mount /dev/loop0 /mnt/usb"

Para revisar el mbr de la imagen se crea primero una copia de nuestra imagen:

```
foo@bar:~$ dd if=img.dd of=mbr bs=512 count 1
```

- Sólo se copia uno porque el mbr tiene un tamaño de 512.

Para revisarlo solo basta un "hexdump -C mbr | less"

- python3 -c "print(hex(2048))" nos permite saber el hexadecimal de nuestros sectores.
- Estos números se pueden ver en el hexdump representado con little endian. Hay un espacio para describir la primera partición, el siguiente para la siguiente partición y así con un máximo de cuatro particiones.
- Empieza con 83, diciendonos que es un tipo linux. Indica en que sector comienza (800 p.e.) y su total de sectores.
- Fdisk revisa los primeros 512 bytes del dispositivo y de estos sacar toda esa información.
- El final de mbr siempre termina con 55aa