# Digital Forensics lab set-up

# Location and physical security (I)

- **secure facility**
  - geographical location
    - remote or easily securable
    - free from vulnerabilities to natural disasters or interference
  - implement defense in-depth multi-layered security
    - biometric access (fingerprints, iris scans)
    - keycard entry systems
    - video surveillance (CCTV) and 24/7 security personnel
    - alarm systems integrated with local law enforcement

# Location and physical security (II)

- **air-gapped environment**
  - physically isolated system (including the internet)
  - prevent any external connection to workstations or network sections
- **faraday cage rooms**
  - prevent external wireless signals from interfering or compromising data-processing area
- **environmental control**
  - maintain appropriate temperature and humidity for electronic devices (servers, workstations)
  - fire suppression systems to protect from fire
    - ...with no harm to electronic devices
      - e.g. based on Halon or FM-200

# Data Protection (I)

- **segregated networks**
  - network segmentation (analysis, data storage, sensitive communication)
    - through VLAN and/or VPN
  - encrypted communication
    - adopting high-grade encryption standard (AES-256, RSA-4096)
- **least-privileged access control**
  - role or attribute based
    - e.g. *admins* for system management
    - *analysts* for digital forensics analysis
    - *supervisors* for auditing and review

# Data Protection (II)

- **disaster data recovery readiness**
  - (verified) back-up systems
  - (verified) uninterruptible power supplies (UPS)
- **logging**
  - enable logging for all critical components
    - servers, workstations, databases
      - operating system logs (e.g. *login attemps, file access, administrative actions)*
    - network devices (e.g. *routers, firewalls, switches*)
      - logs on network activity, configurations changes, access attemps
    - end-user devices
      - app-specific logs, user activity, critical transactions

# Data Protection (III)

- **auditing readiness**
  - availability of audit trails
    - user specific
      - e.g. login-logout times, executed commands, accessed files, configuration changes
    - application specific
      - e.g. database audit logs to track queries on sensitive information and administrative actions
  - immutable trails storage
    - WORM (write-once- read many)
    - hashing and/or digital signatures
    - Policy-based access-restriction
      - e.g. based on separation-of-duty principle (different roles to r/w)

# Hardware requirements (I)

- **high-performance workstations**
  - fast multi-core processors
  - consistent RAM (128GB or more)
  - high-speed SSDs (many TBs)
- **storage forensic servers**
  - large-scale storage for forensic images (tens of Terabytes organised in RAID HD and backup servers)
- **HW acceleration for specific tasks**
  - e.g. GPU for password cracking

# Hardware requirements (II)

- **write-blockers**
- **imaging devices**
  - e.g. tableau comprehensive imaging
- **mobile device acquisition tools**
  - e.g. Cellebrite UFED


Source: *https://siliconforensics.com/products/forensic-hardware/tableau-instant-imaging-response-kits.html*


Source: *https://en.wikipedia.org/wiki/Cellebrite_UFED#/media/File:Ufed_mobile_phone_imaging_device_(8661348282).jpg*

# Hardware requirements (III) - UFED

- **Universal Forensic Extraction Device**
  - complete physical device storage image extraction
  - bit-by-bit copy of flash memory
  - access to hidden files and directory
  - access to specific content (contacts, SMS, call logs, multimedia, ...)
  - (with credential knowledge) can extract cloud data associated with the mobile
    - (e.g. iCloud, Google drive, social media like instagram and Facebook, …)
  - can extract data from many apps (e.g. *WhatsApp, FB messenger, Snapchat)*
  - can support encryption/locking bypass
    - exploiting brute-forcing and mobile vulnerabilities

# Software

- **Forensics lab must provide**
  - flexibility and scalability
    - due to different scenarios requirements
  - provide safe environments
    - where evidences can be analysed with no system integrity compromise
  - easy recovery after a "failure scenario"
    - because some tests might be pretty "invasive"…
  - when possible, should have a "roaming soul"...
    - ...to go, when possible, to the crime scene

# Portable OS

- **create a trusted environment**
  - not related to the one under analysis
- **many solutions based on linux distros “live CDs” (often Debian or Ubuntu based)**
  - security general-purpose
    - GRML, Kali, ...
  - forensics special-purpose
    - CAINE, Helix, SIFT, DEFT, Tsurugi, …
- **also in a virtual machine**
  - copy the evidences in the VM
  - create virtual disks
    - and attach them to the virtual environment

# Mounting (I)

- **the process of making a file system accessible**
  - suitable to interact with
    - local file systems (e.g. ext4, NTFS, HFS, …)
    - external physical devices (like usb drives, DVDs, SDs, ...)
    - network file systems (like NFS, SMB, …)
- **at a specific "point" in the directory tree**
  - This is the root of the "new" file system
  - i.e. you can navigate it from that "point"
  - known as "*mount point*"
    - e.g. */mnt/usb*

# Mounting (II)

- **mounting process involves**
  - checking of the file system on the storage medium for errors
  - loading the file system's metadata
    - how files and directories are organized
  - associating the "new" file system with a mount point in the existing directory hierarchy
- **once mounted, the operating system can read from and write to the mounted file system with usual operations**
- **linux example (shell commands):**
  - *mount /dev/sdb1 /mnt/usb*
  - *ls –la /mnt/usb*

# Mounting (III)

- **unmounting is the reverse operation**
  - detaches the file system from the directory structure
  - crucial for safely removing storage device due to caching and buffering techniques for performance improvement
    - ongoing I/O operation
    - write-behind cache
      - delayed writing changes to disk
    - open files in active processes
      - internal file system structures (e.g. file pointers) may become inconsistent if the file system is brutally removed
    - Inconsistent integrity information
      - metadata that track the state of the file system might not be updated

# Mounting (IV)

- **unmounting steps are thus**
  - data buffers flushing
    - OS writes cached data from memory to drives
  - closing of open file handles
    - OS closes any active file handles, ensuring no further file access
  - **updating file system metadata**
    - OS update metadata and journal on the file system (e.g. changes on the directory structure,  file permissions, ownership)
  - **file system detaching**
    - OS detaches file system from the directory hierarchy, ensuring no more r/w possible operations on the drive

# Command line tools for devices interactions

- **informations on connected (even not mounted) devices**
  - e.g. *dmesg, lsblk, fdisk, eventvwr, Get-WinEvent, device manager, Get-Disk, Get-Partition, wmic diskdrive, DiskPart, ...*
- **mounting in an integrity preserving way**
  - *mount -o ro /dev/sdb1 /mnt/usb*
- **mounting verification**
  - *mount | grep sdb1*
    - */dev/sdb1 on /mnt/usb type vfat (ro, noexec, nosuid, nodev)*
      - *ro:* read-only
      - *noexec*: prevents execution of binaries on the mounted file system
      - *nosuid:* disable the set-user ID bits on files on the mounted systems (cannot run with elevated privileges)
      - *nodev:* No special "device" files on the mounted system

# Possible device types (I)

- **block device**
  - handle data in fixed size blocks
  - allow for random access
    - *find /dev -type b*
- **character device**
  - handle data as continuous stream
    - e.g. terminals and consoles
    - *ls –l /dev |grep '^c'*

# Possible device types (II)

- **network device**
  - handle packet-based data transfer
  - use sockets and syscall to communicate
    - e.g. */dev/wlan0*
- **pseudo-device**
  - **interface to software feature**
  - **not directly related to actual HW**
    - but can emulate HW features
    - e.g. */dev/null, /dev/zero, /dev/random, /dev/loop*
- **special-purpose device**
  - very specific purpose for system function (managing system memory, interacting with the kernel, …)
  - e.g. */dev/mem /dev/kmem, /dev/console*

# access to data at rest (Linux)

- **through the filesystem (after mounting process)**
  - access to device data from the mount point
  - adopt usual file system metaphors
    - files, directories, metadata, …
      – OS manage and update metadata
    - usual file operation (read, write, delete)
    - access control through file system rules (e.g. rwx) and usual operation (e.g. *ls, cp, mv, chmod*)
- **through direct access**
  - to the block device (e.g. */dev/sda1)*
  - allow access to raw (unstructured) blocks of data
    - risk of data corruption (no file system validation, access thourgh byte level commands like *dd* or *hexdump*)
    - foundamental for bit-by-bit copy

# Linux distro interaction with devices

- **mount block devices as read-only**
  - many times by default
  - through command-line tool
    - *mount –o ro /dev/sdb1 /mnt/evidence1*
- **set the block device to read-only**
  - e.g. through command line tool like *blockdev* ioctls
    - *blockdev –setro /dev/sdb1*
- **or through GUI**
  - e.g. CAINE (11.x) provides *UnBlock* graphic tool
    - list of devices and write permission status
    - allow to change from/to  Read-Only and Read-Write

# disk image mounting

- **disk image = file containing bit-per-bit copy of an HD**
  - can be obtained through command-line tools like *dd*
- **manipulated as a normal physical device**
- **there are both GUIs and command line based tools**
  - *e.g. disk Image Mounter (CAINE 11.x)*
    - click and choose the file image
  - *imount* command line utility to mount volumes in *Encase* and *dd* format
    - *imount –v image.dd*

# Write-protected set-up

- **attach the Device**
  - physically connect the storage device to your forensic workstation
  - preferably use a hardware write blocker to ensure no writes occur physically.
- **set block device to Read-Only**
- **mount Filesystem in Read-Only Mode**
- **check the mount status**
  - e.g. *mount | grep /mnt/<yourforensicdir>*
- **verify Read-Only Status of the Block Device**
  - **e.g.** `blockdev --getro /dev/sdX`
- **Note: device level protection override filesystem requests**
  - ...but may cause errors and inconsistences (journal/metadata)