Digital Forensics

Introduction To Forensic Science

Civil law = dispute between individuals (companies also) (rectify a wrong, honour agreements, settle disputes) * Tort (breach of a rule, moral or other)/Contract/Trust (trustee etc) * Preponderance of the evidence (most compelling) * can b asked to incriminate yourself

Criminal law = Gov punish undesirable acts 2 provide deterrence/retribution/incapacitation/restriction/rehabilitation * Common law judicial system (UK, developed by law courts w no specific basis in statutes)/ Penal code [criminal law] judicial system (crim law compiled in a document) * Beyond a reasonable doubt * Can't be forced to incriminate yourself

Note: Can be both * crimes prosecuted at discretion of state (might not if too costly) * most computer crimes are 2 expensive so mostly individuals have 2 file law suit

<u>Investigators</u>

Investigate (anybody, person in auth, police officer)
Caution (person in auth, police officer)
Arrest (anybody – in act of - , person in auth – in act of - , police officer – in suspicion -)

*Although citizen arrest is likely 2 → false imprisonment law suit * and remember caution b4 asking questions else in-admissable*

Person In Authority = charged w investigating (security officers, gamekeeper, security consultation, info sec proff at a company)

Investigators, subject 2 Human Rights and Investigation powers, may: question any person * search premises and property incl. Company comps but messier 4 byod * record internal transmissions * seize material in possession of the company

<u>Laws</u>

Computer Misuse Act 1990

Investigatory powers act 2016 (extended 2000): unlawful by high court (contravened GDPR + other EU law), European Court Of Justice ruled 'indiscriminate retention of all comm data was disproportionate and illegal' Empowers w warrant to: conduct interception, equipment inteference (bugging end devices – wht about integrity of evidence?), bulk comm and data acquisition.

Parallel construction – can't be made 2 say they interfered w device so can create false history of investigation.

<u>Evidence Types:</u>

Witness = traditional or expert witness 4 interpreting evidence (transcripts of interview under caution in UK)
Documentary = written docs, photos, tape, ilm (*dig evidence also counts but you'd need expert witness oft to authenticate*)
Real = physical (material) evidence like murder weapon, or device.

Crown Prosecution Service 2013: types of evidence obtained by computers:
1) Evidence processed by a computer when functioning as a calculator – Real evidence * assumes computer inherent trustworthy
2) Evidence that computer is programmed 2 record – Real evidence * recording devices assumed cant be tampered with
3) Evidence processed by comp but entered by a person – Heresay * not admissable in criminal normally

What Makes Evidence Admissible

U.S Dauber Test:

1. Scientific expert testimony must proceed from scientific knowledge (Not an opinion)
2. Testimony is relevant 2 task at hand and expert has relevent qualifications
3. Testimony must use scientific methodology ID'd as having:
    1. Empirical testing - reliable/falsifiable/testable
    2. Subject 2 peer review and publication (DF unlikely 2 b publd bc so volatile and research protected by trade secrets)
    3. Known or potential error rate (hard to define in DF when u deal w active adversaries)
    4. maintenance of standards and controls in operation
    5. degree of acceptability by relevant scientific community (doable but difficult bc fast paced community)

Australian says expert evidence rather than technical if: 1) does the subjt matter requre special understanding → 2) enough body of knowledge for expert opinion 2 be reliable → 3) does expert witness have enough experience or study 2 be of use 2 the court

English courts = relaxed abut admissability of evidence used by expert witnesses * what qualifies is vague "...study, training, experience, of any other appropriate means" * suggested 2 be codified

Evidence needs to be relevant (= logically goes to proving or disproving some fact at issue) and admissible (= relates to facts at issue or cirumstances that made said facts probable/improbable, it has been properly obtained incl. Chain of evidence)

Expert Witness

In England and Wales – duty to the court * duty 2 inform all parties if original statement is changed bc error or opinion.

Issue joint statement if both parties hire expert wit (even if statement is a disagreement)

Expert Witness Report (CPR's):

1. Detail expert qualification
2. Detail literature
3. A statmnt setting out substance of all facts
4. Makes clear whch stated facts are within experts own knowledge
5. States all people involvd (in any experiment/test/examination)

Difficulties With Digital Evidence:

- Evidence rarely located only on one device, by cloud & synching etc, who has access and who could manipulate?
- Distributed actors (bit nets etc), may reside in countries w relaxed laws or dont extradite
- Witnesses 2 comp crimes oft other computers, which can be more objective, but must be full understood and can be manipulated
- Digital evidence is fragile 2 accidental/deliberate loss/change/obliteration/relocation → null and void (IT support job often necessitates obliterating evidence)

So digital crimes rarely prosecuted

Electronic Evidence Principles

1 – Integrity Of Evidence Must Be Properly Maintained (lol but IPA2016 contravenes?)
2 – Examiners should be appropriately qualified 2 examine the data & b able to explain/document their actions and findings at later data
3 – Examiners shud maintain notes as they go (detailed enuff 4 independent, qualified, 3rd person to recreate)
4 – Case Officer has overall responsibility 4 ensuring law and principles adhered 2

Note: all digital evidence subject 2 same rules/laws for documentary evidence.
Note: so many laws (these incl. ISO 17025 2005, ISO 17020: 2012) make it difficult 4 smaller labs to stay in business

Stages Of Investigation

Freeze the scene

Photograph 2 place evidence * collect standard forensics like cctv, fingerprints etc * locate all physcal devices (& all devices w memory) and put in sealed bag if feasible * id witnesses * DONT SWITCH COMPS ON OR OFF

Collection
Manner will depend on device (volatile, or network data need to be captured?) * ownership of device hard 2 pin down, could be multiple owners ↔ multiple devices, so who had access etc * careful of remote wiping/modification * some services will already have collected data but legal barriers

Mirror Image (if no networking or volatile 2 consider)
necessary not to fuck with integrity * use write-blocking-device so dont write on original * use crypto hashes to check integrity (but then key mngmnt is an issue) * need special devices which gets expensive since storage is always changing (hardware not software which would need to be verified up the wole stack) * but then some encrypt data making a mirror image useless. * commericla tools can help

Common Mistakes:

- Losing power when RAM memory could reveal passwords, access 2 encyped volumes, etc
- Deleting evidential files (e.g IT support) since its not alwys clear what will b needed for forensic investigation
- Writing 2 evidential disk
- Installing diagnostic tool on disc
- Changes to data/time stamp
- Relocating evidential files/directories and creating them
- Recovering deleted files onto disk
- Executing native software or apps on evidential disk

Collecting Info On Live Systems
Approaches: VM snapshots * hardware debuggers and in circuit emulators * exploiting hibernation mode (powered down but maintaining previous state) * explot crash dumps (tho could be encrypted)
Challenges: counter-forensics * size of memory → 2 much to analyse * changes in subsystems and architecture

Challenges 2 digital evidence: Data easily changed * steganography * encryption * data destruction * easy 2 misinterprate * time/date stamps shud b used w caution

Chain of Evidence = passing of data provable, observed necessary rules, and all access proven to have not effected content else evidence is lost.

Procedure to using storage media as evidence:
Discharge static energy → host system switched off usually → comp and config photographed * serial numbers etc noted → remove cover and photograph internal config → remove storage device → device itself photo → serial number, manu, of device recorded → put in anti-static bag, then envelope, then sealed w tape → sign and date envelope → store in suitable environment (that wont damage the device ; solid state pretty robust)

Examination
Goal = reduce data 2 get key evidence * can u still process media/devices after all that time? * tech review under reproducible lab conditions * examin incl. System timestamps, registry files, input/ouput, acccess files, swap file, slackspace, etc

Analysis
Analyse and organise extracted evidence meaningfully * in criminal cases u want Actus Reus (the guilty act) and Mens Rea (the guilty mind – was it deliberate) * everything documented (time of investigation, software/tools used, expert re-construction of evnts if used, continuity of evidence) * tools can help w documentation but must themselves b considered trustworthy/not subject to manipulation

Note: The rigidity mostly criminal law
Note: civil inverstigations can still trip-wire counter forensics so be careful out there ;)

Storage Forensics I

Hard Disk Operation – Traditional
HDDs = electromechanical * controllers maintained map of phy disk layout (cylinders/heads/sectors -CHS) – now Linear Block Address w single numerical addresses

Hard Disk Operation – Modern

Internal OS, interface vitrual LBA or CHS (just gets remapped) * controller logic on disk (now has signal processing for read/write etc) * most now have self-encrypting dives and so cryptographic disk erasure.

Volume Management

Software and firmware need 2 talk * The OS and boot loader talk through BIOS interface

Master Boot Record

Sector 0 (512 bytes) contain: partition table, bootstrap code, timestamps and signatures. MBR holds 4 primary entries * each entry ID's file system used * primary partion can be extended * booting must occur from primary * implementations vary * used to rely on CHS specification but now supports LBA * max disk size of 2.2 TB

GUID Partition Scheme

Intro of globally unique identifier partition table (GPT) and Unified Extensible Firmware Interface (UEIF) removes size restrictions * uses LBA * dummy MBR so old hosts dont think is empty * GPT header has pointer to partition array * partition entry contains type, unique id, beginning and end numbers, optional partition name

Unified Extensible Firmware Interface

Replaces BIOS * can be thought of like an OS * boot/run-time services * can be extended by vendors * device drivers and graphical interfaces * supports network booting * can be booted from abitrary partitions * offers secure boot but so complex easy to find vulns and load malicious code into UEFI

Hard Disk Encryption and Forensics

Key disclosure laws

veers close to self-incrimination * criminals weigh up worse punishment

UK Regulation of Investigatory Powers Act 2000 (RIPA): by court order * max penalty of 2 years imprisonment

US FBI uses national security letters: subsequent gag orders were rescinded on challenge * key disclosure oft defeated in court due to 5th amendment against self incrimination

South Africa: 10 year imprisonment 4 not disclosing

Other: rubber-hosing (torture)

Hardware-based Full Disk Encryption/Self encrypting drives

ATA drive lock used to be populr * require password to read data * not actually encrpyting * defeatable by overwriting flag or changing password in drive service area.

TCG Opal Storage Specification 2.01

Current standard for self encrypting

Key mgmt handled on device (key never leaves disk) * data encrypted at rest * can be used 4 cryptographic erasure * specifies seperate authentication mechanism (and key escrow mech for recovery) 4 accessing media encryption key * can have pre-boot auth (firmware auth layer) – multi-factor or external key mgnmt software

SEDs show a vitual view of the disk (a lie): a 128 Mbyte MBR shador, mapped to LBA 0, and w/o a key all rest is zeroed out

File System Concepts

Directories, files, attributes * Mostly byte orientated (cam be record orientated) → mapped to block * sectors (smallest, usually 512 bytes, low level ops occur on these) * clusters (groups of sectors, makes allocation units manageable)

File Allocation Tables

FAT Organisation

Volume has: reserved area (boot sector and housekeeping info) * Fat area * Data area * Fat 12/16 has a special root directory region

Reserved area : jump instruct to boot code * # of FATS (usually 2, for backup) * total # sectors, * etc

In FAT32 – boot sector usually followed by 'FS Information Sector' to speed up storage mngmt (has number of free clusters and next free cluster) * FSINFO also has 2 signature strings that myb cud be used to reverse engineer the location of FAT meta data if FAT table is corrupted

FAT Table Structure

Entires defined in file allocation chains * entries contain pointer 2 next entry (or note last entry/bad cluster/ unused cluster)

FAT Directory files:

Attributes for directory entires: read only, long file name, directory

First byte of an entry: 0x0 if unallocated * 0xE5 if deleted * first byte of file name

FAT Fragmentation:

Allocation linear (look for first free) * in simple algs, deleted files left in situ * over time becomes fragmented * also multi-tasking interleaves ops and causes fragmentation

Some strats 2 alleviate: exFat (addiitonal structs 4 finding contiguous free space) * windows NT attempts 2 allocate larger segments in advance

ExFat:

proprietary by microsoft * works better for small number of large files (flash drives, memory cards, or device where u cant have more complex overhead) * bitmap allocation * bad blocks marked on per cluster * time stamp has 10ms resolution (better than before and now easier to determine order files were written) * pre-allocates clusters (minimised fragmentation) – free up any not actually used

Reconstructing Fat Structures

Duplication:

Forensic Duplication = ability 2 produce an identical byte stream from the duplicate as from the original

Forensic Duplicate = 1 to 1 bit information (need same amount of available space)

Qualified Duplicate = same info but in altered form (lossless compression, meta data etc)

Restored Image = forensic or qualified restored 2 another storage medium (really us put duplicate on server and work on virtual drive)

Mirror Image = created from hardware with bit by bit copy from one hard drive 2 another (not really done anymore)

Any device providing imaging must: ensure no write occurs on evidential disk (incl. Re-mapping that can happen automatically on read-only) * sector by sector copying (id error conditions, integrity or data traceable thru hashes etc) * record additional info (timestamp of duplication session, diagnostic info from device)

And provide assurance of all above (expert witness when challenged , maybe trst results) * hardware devices are primitive precisely bc it makes it easier to guarantee above and validate (software uve got to validate stack)

Difficulties:

Volumes can cross drives (you may be only able to ID through snapshot of volatile configs in situ) * disks/partitions may have unallocated slcack space (which can hide whole file systems) * strategic overwriting counter-forensics (overwiriting meta-data unlikely 2 be detected, harder 2 detect in well used systems bc u expect nonsense data, avaids anti-virus bc they tend to look at file access not raw blocks)

Deteletion

1. Directory entry first byte set to 0xE5
2. Entries in FAT table zeroed

Recovery:

1. Scan file system file directory by file directory
2. Compile list of all beginning with 0xE5
3. Change first character back
4. Restore FAT table entries (possible if contiguous more difficult if not)

Caveats:

Orphaned files when directory entry re-used (more common than file location being overwritten)* even if file is reallocated and contents overwritten, some data might remain (on average only half the cluster will be used) * careful about crossing file generations.

Storage Forensics II

Microsoft Storage Architectures

Communcation btwn storage and OS is Complex and layered * storage comms thru miniports (vendor supplied, and the only thing vendors write)

Can also comm thru: iSCSI (storage area network protocol, over tcp/ip) * multipath I/O (high availability envs, consistency must be enforced and synched w/o read writes arriving asynchronosly)

Everything in Win is an object w Object Manager retaining global unique name space (we just see drive letters)

Partition manager = function (main) driver for disk and maintains the Master Boot records and GUID Partition Tables on disk (all partitions have unique ID so no multipath errors, objects created 4 each primary and extended partition)

Windows Volume Handling

When Partition added vol managers asked if they manage it (if they do, delegate to em)

Partition 0 reps whole disk (it will ignore all logical partitions and reutnr whole lot)

Some apps bypass OS file system layers and operate on raw partitions (e.g Oracle DBMS)

Dynamic Disk Concepts

Virtual Disk Service subsystem's Logical Disk Manager maintains single db of all disks that have been attatched to that device (incl. Dynamic disk guid and name of the host, whch can be used to tell if u've collected all storage devices)

RAID

Raid 0 – Striping: Take data & share between disks * enhanced performance * increased failure rate

Raid 1 – Mirroring: duplicate data * enhance reliability * slowest drive dets. Overall speed

Raid 5 – Block-level striping – store across n disks w 1 used for parity * data recoverable if n – 1 is working (or if one drive happens 2 contain enough info) * 1 drive failing probably indicated others soon to fail bc bought at same time * recovrering data and replacing is high intensity and might trigger further failure * preventative maintenance is import.

Volume Shadow Copy Service and File History

VSS allows snapshot of file sys

Clone Shadow Copies = split mirror made by duplicating vol → splitting from the live

Copy-on-write = differential copy made * overlay changes on earlier live data * supported by default

If files always in use, difficult 2 copy: can freeze and thaw messaging to allow snapshotting

Windows VSS

orig 4 checkpointing in update * vol shadow copying in win 8/8.1 but needed external drive (it wud hold in cache til attached) * now just used for restore points

Windows File History

Like VSS but ops on files rather than blocks * we can recover files * requires NTFS (for its change journal)

Windows NTFS File System

Features: Data Streams (mult data streams where stream IDs a new data attr. on the file, handle can be opened on each w seperate file locks and sizes but common permissions) * remapping for recovering from bad clusters (shouldn't need, bad clusters r suspicious)

Change Logs – track file sys events * has to be activated and actually used by apps

NTFS structure – everything a file (incl. Own meta-data) * Master File Table (MFT) holds array of file records * when mounted metadata read and in mem data structures made (Vol boot sector gives address of MFT, MFT is first entry and second record points to a MFT mirror located elsewhere on disk) * MTF mirror doesnt contain all records but metadata to help if original corrupted

File reference numbers and records

All files have File Reference (unique ID) w file number + seq number: helps us det. if file current generation of previous.

Files are structured entries w attribute/value parts * payloads are unnamed data attribute * each attribute stored as sep stream of bytes in file

NTFS tried to put all atts in MFT record * atts that fit are called resident attributes (vs non-resident) * attr headers are always resident where values may not be * data runs or extents are allocated for data that doesnt fit (tries to make them contiguous) * small file payloads may be resident.

File Creation:

1. Check file sys has space, allocate clstrs
2. Locate MTF entry currently available, allocate
3. Set MFT as occupied, initialise it
4. Save file to clusters (if needed, else save in entry)

File Deletion:

At each step, record action in log file:

1. Go to root dir, read index attr of MTF, find in B-tree 2 locate index of file, read indexed entry 2 retrieve number of MTF entry that represents the file, delete the index entry. B-tree may readjust
2. Set MTF entry for deleted file as deleted and make avail 2 others
3. If data attr is non-resident, read the data runs and obtain addressed of clusters allocated and set corresponding bitmap bits to 0 (making cluster available to others)

File Recovery;

For NTFS, makes little diff if contiguous or fragmented

1. Scan MTF entry by entry, make note of each w deletion market
2. Extract MFT entires data attributes (if resident, this'll contain whole file, otherwise it'll give addresses in data runs), check bit map for corresponding addresses. If 1, overwritten. Otherwise read, save, verify cluster contents

<u>To recover file names</u>

1. Scan MFT looking for entries for directories (deleted and existing)
2. Extract indec attributes of these, which contain the file_name struct.
3. Match recovered file contents from data attributes in MFT entries to name found in deleted index entries by correlating info (like time stamps or file types).

<u>File and Volume Encryption</u>

Windows supports 2 kinds:

Encrypting file System – per-file enc (not incl. Boot-up/registry/page files) * symmetric cipher

<u>Limitation</u>: key mngmnt (sec dependent on mgmnt of pub keys incl. Recovery keys) * file system transactions (if copy file out of NTFS as auth user, data unencrypted) * unexpected propogation (protects data at rest, anything using in live FS or sent out to external will be unencrypted)

BitLocker – volume encryption * strict hardware reqs (incl, Trusted Platform Module – default TPM w/o release key could allow coldboot attack) * automatic encryt variant Device Encryption avail in latest windows (uses microsoft accounts, admin/authed in AD may gen recovery keys) * not like SED must have pre-boot env (small limited OS vuln to bootkits) * key material can persist after reboot (but easier 2 go for recovery keys)

<u>Physical Storage Architecture in Magnetic Medium</u>

interrogate firmware 4 lower lvl access (2 get to bad blocks or other internal structs unavail to OS)

<u>If firmware not accessible/damaged recover:</u>

Transplant = sub components from another drive * calibration difficult and more expensive as structs get smaller * phys destrcution may have lost data

Detection = platters on spin stand → scanning electron microscopes (2 detect magnetism)/ scanning w optical detection (detect electromechanical deflection to detect negligible magnetisation) * very expensive (techniques tricky, precise environemnts etc) * might not b admissable in criminal court

Counter Forensics:

Destroying magnetism – X - need field stength of obscene lvls

Shredding Disks – X – Partial recovery

Burning optical disks - √

Burning hard disks – X – glass/ceramic paltters need obscene temp

Chemical burn hard disk - √

Any self encrypting device destory TPM - √

Host Forensics

Windows Kernel Architecture

OS prvide abstraction layer btwn users & physical cmpnts (simplification, emulation of cmpnts, multiplexing, protective mechanisms) * ring based priv mngmt

Universal Windows platform = mchnism 4 app development 2 limited interface 2 run across platforms. * not good for FA (direct api calls leave clearer artifacts 4 analysis) * most apps still run natively

Executive Cmpntns:

- Object Manager = create/destroy/control/ protects objcts * creatues Unique Identifiers tht we can use 2 trace relationships and add temporal info to snapshots
- Message Passing Interfaces: Advanced Local Procedure Calls, I/O manager, cache manager, process manager (can hold admin even after termination of process, mbe can grab it?) * memory objs persist for a while so u can see left over messaging
- Plug and Play = handles installation of device drivers * artefacts left after components been removed * can see if seized all devices
- Kernel Debugger = debugger symbols not always available and some malware wont even run if they are, so oft end up with raw machine code
- Security Reference Monitor = enforces security policy & audits * use 2 find who did what at what priv level

I/O Architecture

API calls → dozens of I/O requests which create artefacts in memory

Windows Management Instrumentation services can be useful 4 forensics 2 investigate resources

Windows modularises device drivers into class, port, and mini-port drivers → vendors only write miniport drivers → easier for analysts bc 2/3 of stack is stable/well known

Modern Graphics Processing Units run own OS 4 offloaded processing * interact w main OS like its across a network w gpu not having direct access to main memory * highly vulnerable (w sprawling and constantly updating code) * no forensic support for GPU subsystem

Configuration Database

Registry:

A typed DB w keys & values

HKEY_CURRENT_USER: data allocated w current user stored in sep file, for roaming profiles (roaming slow due to sync at login/out * forensic analyst can collect login behaviour etc)

HKEY/PERFOMANCE_DATA/CURRENT_CONFIG: if machine performace well known by analyst, artifacts left here might give indication of malware

Active Directory:

Used in networked windows environments for user auth, data, and access management * can do analysis of a damaged/missing machines * harder to attribute actions perfectly as AD could be manipulated by anyone w authority

Live Forensics on Windows

Live forensics = snapshot of memory OR live debugging * needed if cant take machine offline * wary of counter forensics that will suicide the machine (if critical)

Adv: assists w static analysis * obtain passwords, key mateiral, unencrpted data * reconstruct session and app info * ID network traffic and app

Challenges: volatility (even if largely frozen during snapshot, memory will be changing somewhere) * Artifacts (software taking snapshot will be part of snapshot) * Manipulation (mst windows tools go through regular APIs using standard calls which could be manipulated.

A lot of apps don't minimise amount of sensitive info in volatile memory, and even released memory is rarely overwritten bc performance hits * this info could be found in crash dump.

Crash Dumps can be useful * harder to force as windows improves * windows auto-reboot loses dump data * can still have consistency issues, that might have even caused the crash, (other memory still moving in background)

Windows Security Architecture

Security Arch: can add extra auth thru authentication packages * logonUI = interface launch in seperate desktop other apps cant access (protect against fake login screens, ctrl+alt+delete to reach)

Access Control Mechanisms

All ents ID'd thru Security Identifiers (SIDs): Machine gets SID at install → accounts entites derived * domain accounts have Relative SID based on domain controllers * Well-known SID for special roles and groups * cloned machines need 2 adjust SID 4 proper resource management (like file locks) * ephemeral SID for login called logon SID (so access tokens in file sys image and memory image wont match, you'll need to map using Security Reference Monitor; easy to id bc prefix is fixed [S-1-5-5-0])

SID specify integrity levels (e.g untrusted, low, medium, high, system) * initial access tokens created at logon, inherited, and can be modified or filtered * impersonation tokens are temporary 4 service to act on behalf of client (like remote access apps) * objects have security descriptors w owner and group SIDs, discretionary access control lists, system access control list

Auditing Mechanisms

Audit events are as trustworthy as the machines that supply them

Can be generated by: Object Manager (from access checks, but rarely generates bc performance hits) * Kernel-level code * User-level code (but must have priv level set in system access control list)

Used to be useless (audit logs were saved to a ring buffer so attacker could overwrite if flushed enough events)

Remote logging means you can now review across number of machines

Virtualisation

Apps should run on VM as if on host * hardware seperation only as trusted as virtual monitor

Approaches to simulation:

fetch/decode/execute – simple and slow

trap and emulate – priv instructions cause priv trap state * VM monitor requires use of priv level * not all architectures support w their instruction set * cost can be high

Binary translation – substitute priv code out for 'safe' equivs that cause desired effect in mapped virtual space * complex architectures have to maintain an internal state * can only handle benign code (self-modifiying code, eg, is too much)

Forensic Challenges:

- Layered Virtual Memory: map virtual to host, host to physical * can just take disk image (but have to prove forensically sound)
- Counter forensics: easily detect when running in sandbox bc ur interacting with a standard interface vs physical
- Cloud services – might not even have access 2 platform/hardware (have 2 rely on watever services you are given)
- Moving Targets – virtualisation architectures keep being changed which makes it more difficult to analyse.

Memory And Live Forensics

External acquisition = hardware-based * target not altered

In-sys acquisiton = software based * presence alters mem image * myb triggers counter-forensics * not atomic (Causality mght b inaccurate)

Direct Kernel Object Manipulation used by malware leaves artefacts (find inconsitent data structures – e.g process nt in process list)

Hardware Based Acquisition Techniques
Direct Memory Access apprch: Dornseif and Eckstein (over firewire, cross platform, one way uses plug-and-play bt this will alter target system) * Tribble (over PCI, avoids plug-and-play, PCI card doesn't announce presence on bus → CPU halted → physical memory read via DMA)
Note; Hardware approach will depend heavily on architectures

Countering Memory Forensics
Forms: Denial of service (halt or crash when detect acq) * Covering (prevent tools from obtaining parts of memory) * Full-replacement (sophisticated covering, replace memory read by acq w plausible benign data)
Techng: syscall proxying (interposition between sys calls e.g hooking, filter drivers) * in-memory library acq (dynamically replace code in sys libraries since hard 2 validate)
Note: Both can be detected e.g by concurrent monitoring of critical data structures

Targeting the tools: since variety used limited* look @ RE techqs used and trip analysis * target consistency checks tools used to subvert memory corruption (allowing memory corruption)

Translation Lookaside Buffers = used 2 split into data access and code access * Shadow Walker desynched the buffers so you'd get diff memory for the same virtual page depending on buffer (so u could conceal processes) * now TLB is shared, but virtualisation support (extended page table) can be used to force seperate TLB entries

Subvert Hardware Access = modify chip bridge set devices connect through over PCI

Persistence
Some RAM types retain after power off: magnetic core (years) * SRAM (battery backup) * DRAMs (depend on temp of comps, cold can yield usable 4 <= 1 min)
Issues: As compnts shrink, grace period shrinks * for badly desiged DRAM comps reading memory in unussual patterns can cause adjacent bits 2 flip (not likely problem if u read linearly)

Non-volatile storage forensics
Storage devices use interface 2 give logical view of physcial * Even if host issues overwrite, data may still be on disk → access low-level interface (or phys inspection if damaged)

Flash Memory
NAND flash memory = overwrites/erases faster * uses store and download access * must be read/written in blocks
NOR = execute in place access

Flash Memory and File Systems
Naive implementatins: uneven use of storage space (file allocation table and root dir in FAT) * blocks huge 128kBytes vs 512 byte blocks

Flash Translation Layer = used in device or in system * translates storage area into smaller virtual sectors/chunks * appearance of writing in place (modify in same physical location)* actually wear levelling (writing in same phy area damages block so write everywhere w approx same freq)

Garbage collection is aggressive and happens in the bg * causes self-corrosion of court evidence * can deactivate in SSDs and software controlled FTLs * otherwise bypass mngmt layer

Erasing and wiping SSDs
SED and delete key material
Secure erase : doesn't necessarily overwrite all spare capactiy * overwriting slow (especially bc thermal mngmt) * wud req 20 passes before certain.

Trim cmnds can be sent to signal block isnt needed * SSD may lie, not erase data, and just return zeroes when read (you'd have 2 go below FTL to get data)

Defect Management: single page on block producing error → whole block marked as invalid (then becoming skippable under Skip Bad Block Strategy) or replaced logically from a Reserve Block Area * can be used 2 hide info from the OS

Recovering Flash Memory

2 bypass FTL: use JAG boundary scan features (2 test electronic circuits) * chip off technique (attach chip 2 test equip, can be expensive – especially w complex layered ciruits like iphones)

If chip damaged: small pieces could retain data

NOTE: Large SSDs are a bitch 2 analyse w phs access since FTL parallelises and wrties across large num of chips

Firmware and Forensics

Code for start up in firmware * usually non-volatile flash memory

Pre-boot process historically

CPU reset → CPU and chipset init 0> init of cache as RAM → init D(RAM), create address map → Enumerate PCI(e) devices → Execute option ROMs on attached devices → load and execute MBR

UEFI boot process

Phase 1: SEC = sec phase tht prepares/may verify sys from cold boot onwards * init cache, memory translation registers, cache-as-RAM, and the Trusted Platform Module
Phase 2: PEI = pre EFI configures the platforms and inits DXE (can verify it)
Phase 3: DXE = driver execution env * loads drivers for configs devices and hands over to UEFI shell/ boot loader * services loaded here can remain active even when OS loaded

SEC and PEI rely on Static Core Root of Trust for Measuremet * must be trusted implicity * contains hashes act as root of a hierachy of signatures * not supposed to be modifiable (not always true in practice)

Attack Surface:

SEC/PEI firmware protected with dig signatures * anchor (for verification) provided on shipping hardware * some steps poorly protected by CPU vendors (e.g on microcode updates) * SPI Flash protection must be enabled to prevent persistent BIOS infection
UEFI has own OS that can load custom aps (e.g network pooling) * attack targets

Compromising Firmware

Why: non-volatile * not imaged by main memory mapping * analysts have 2 know specific details of firmware to investigate so difficult * no interface 4 reading so ud have to take the flash memory out * capable of lying to the OS
Vuln because: not easy 4 enterprises 2 keep updated (required knowledge of firmware each device has) * not all BIOS check signatures/check them well * can fake signature check if bad configuration

Bootkit attacks against UEFI: Replace windows boot manager * replace fallback boot loader * modify or replace DXE drivers * patch UEFI 'option ROMs' * modify SEC/PEI/DEX code in Windows Secure Boot facility to invalidate chain of assumptions

Selected Aspects Of Network Forensics

Host based info collection

Some info only at end points/intermediate nodes * trustworthy? * desirable info is volatile, not necessarily logged/audited

Scan hosts for open ports/active net connections * e.g netstat/sysinternals * scanning could tip off malware (open scan tools → further dns lookups for more context) – tho u could turn it off.

TCP = stateful * kernel state keep track * state info may persist after connection closed (implicit disconnections where wait)
UDP = stateless * may need 2 analyse app state to understand conn status

Transient network connections = after host state machine purged, collect info from: audit records (local firewall/connection filters) * cached dns resolution entries * LDAP * Net BIOS name tables * routing tables [if entires complex, could indicate malware]

Other sources: Anti-virus * IDS/IDP * auditing information
NOTE: Limited by trustworthiness

Network-Component Forensic Info Sources

Scanning network = limited info and may involve observation errors

Info gathered (thru tools like nmap): availability of hosts * query available services (not straightforward) * fingerprinting OS (baseline future behaviour by analysing response behaviour, can be delib masquarade, firewall can invalidate, timing not a reliable technique)

Printers are a good target.

Targeting IP packets

Unusual fields in packets → covert channel? * manip of offset and packet reassembly

Scan techniques:

SYN Scanning = half connections * SYN, SYN-ACK, RST * faster * unobtrusive * no response/ ICMP unreachable might indicate filtered port

TCP Connection = full connection * SYN, SYN-ACK, ACK, DATA, RST * might avoid detection from firewall

UDP Scan = apps might ignore half-formed requests or not matching current app state

TCP Scan = send TCP segments * behaviour to rule breaking packets can ID active ports

SCTP Init Scan = TCP-SYN scan 4 SCTP protocol

You can alter flags in TCP header → diff implementations may react differently (may be detected by firewall or IDS)

Observing Network Traffic

bc hosts untrustworthy or not looking for info on particular endpoint

Observe suing: active network components (restrictions on traffic volume, performance hit might b detected) * passive observation (possible 2 detect but avoids possible contamination)

Network Taps

Can't just intercept since end-to-end: Take signal, splice it, regenerate, feed it back. Preconfigured and usually limited to single port (can get multi-port devices but num doesn't compare to normal switch & expensive)

Protocol Analysis

Adversaries can obfuscate network traffic: non-canononical ports * nesting protocols (done over HTTP port 30 e.g)

Analsysis 2 deobfuscate (e.g wireshark)

Custom protocol → analysis difficult (RE it but not possible if disfiguring gaps in netowkr traffic or not enough volume captured)

Wireless Trafic

Analyse access points * Insert own access point sharing same SSID as expected (devices may autoconnect) - run in monitor mode so association not visible * IDing origins of traffic difficult (limited auth, MACs spoofable), so fingerprinting circumstantial (Extensible Auth Protocol offers some link to auth data)

Issues: Too much data * capture limited by permitted obtrusiveness * source trustworthy? Quality of capture * detail of capture (full, header only?)

With only header info/network flows ID patterns of communication * Correlating events across network has synch issues (misordering, desynched clocks etc) – might be synched at higher layer like TCP

Email Forensics

Protocols

Services: transmission (standard protocol) * storage * access

NOTE: Storage and access may transform message arbitrarily

SMTP = store and forward * doesnt assume end-to-end/ recipient mailbox online * TCP * no validation steps for server/email addresses *

Process: Mail User Agent initialises header (can be altered by any Mail Transport Agent or Mail delivery agent) → MUA transmits to MTA → which contacts O or more further MTA depending on mailbox (MTAs locate recipient host using Domain Name systems Mail Exchanger)

Fields: MessageID = unique 2 email, specific 2 version, different from history → spoofing * In-reply-to = link message id * received = tracking info

Consistency Validation: secure SSL/TLS doesn't guarantee receipt * dig sigs and enc keys don't guarantee intent (since signing gateways sign on behalf of/ compromised key material) * But consisten forgery is challenging and can provide cirumstantial evidence (from using known path, format headers, etc)

Mail Delivery Agents
Standard: POP (retrieve-and-delete protocol) * IMAP (local email cached copy, served can know whether email read but cant say who read it obv)

Proprietary: exchange, domino etc * tools to analyse but cant guarantee completeness of email extraction

Web access: e.g gmail * feature rich apps leave data in local browser cache

Mobile access: e.g activesync * used by IOT

Crypto Mechanisms
Not compelling (since signing gateways sign on behalf of / compromised key material u would need continiuity of evidence for key material) * anti-span mechanisms (e.g Domain-keys Identified Mail) could hlp establish plausible message trace w/o encyrpytion end-to-end

Forensic Investigation of Email Traffic
Freedom of info laws → possible 2 retrieve masses of data (need data mining technqs) * not just content but comm networks (incl. Consolidating one user across accounts)

Microsoft Exchange Server
To extract: Tools (Exchange Management Shell) * back up copies of Exchange Server database (lots of data, meta-data cn be valuable) * Message tracking logs (for intra-org emails meta-data, such as path taken by message, 30 day tracing by default, 1GB circular buffer) * Admin has discovery tools (In-place E-discovery: search >=1 mailbox using filters, recover deleted/modified emails, single item recovery and litigation hold/in-place must be enabled and usually only is when reg requires it. Auditing Mailbox Access: can track mailbox operations e.g delete).

Microsoft User Agent
If password recoverable, access outlook files. Operation modes;
Cached Exchange mode = local copy of exchange server mailbox cached in OST file, periodically synced
Online Mode = mssgs retrieved frm exchange server and cached in mem not OST file * records submit, delivery, creation, last modification time * straightforward exporting of mailbox will reset last modification time to time of export * PR_CONVERSATION_INDEX allows ID of mssg chains by linking mssg GUIDS (track insetions, deletions, and oth consistency checks within conversations)

Microsft can store email data locally (in EML files)

Unified Communications = built in feature so messgs across apps (facebook, google, linkedin, etc) will be stored by default in local cache

Malware Forensics

Malware = softwr violates sec policy * 1s that dont need 2 propogate can delete all traces of existence
Trapdoor = code permit priv access if specific undoc'd action is performed * inserted by malware or during dev

Mobile Malware = targeted theft of credentials & banking (as not much else of use)

Virus
Virus' = Piggy-back off other code * not nec host-resident, might sit in virtual env like browsers (so analyse diff abstraction levels and system image → not enough)

Propagation technique: Boot-Sector (sub boot-code w infectec code, resurgence since 2010) * Executable Infection (itsert into executable code – in mem, storage, or both – more effectv if code freq used – e.g DLL, drivers, shared libraries)

Macro-virus = target feature rich embedded langs * AJAX, media formats, in office productivity tools, REST * JIT compilation → not visible in sys image * exploit underlying OS interface * write interpreter in diff lang to code, like JS interpreter for malicious C code

Malware Detection Issues:

- Signature matching (even when polymorphic generated) = timing of updates * only ever have subset of possible (even known)
- General halting problem <=> generally detecting whether program is infected (undecidable)
- Baseline app behaviour and control flow → ID virus' based on control flow fingerprints = difficult (programs update/ patch changing behaviour)
- False Positives (don't want to disrupt work)
- Timing (oft want more time than u have to analyse)

Note: detection facilitated by use of shared exploit kits * e.g Angler In 2015

Malware Infiltration Strategies

Targeted = known vulns of services etc * need to scan * scanning cn reveal malicious actors (and could be traceable) * sys usually has firewalls filtering responses so scanning =/= straightforward.

Passive Network Based Infiltrations

(Partially) Trusted comm mechanisms = shared file system * IM for active content (end-to-end not observable 2 network → hosts cant be trusted either after compromise) * Mobile phone networks * file transfer networks * etc

Document based malware = oft support intermediate macro-code

Microsoft & Open office = in 2015 font displaying mech was vulnerable * such complex code base

PDF = embed hyperlinks for drive-by-download * support encryption (and format encoding) tht can be used by polymorphic/metamorphic malware * supports JS (& other scripting langs) * added sandbox model but it leaks

Drive-by-download = site has web content that exploits a web browser vuln * could be done in memory only so no evidence remains in storage

Video-data = dangerous * HTML5 container and codec permutations → potential vulnerabilities * youtube doesn't scan for malware * Flash video for multi-stage malware attack against syrian gov (multi-stage challenge 4 memory imaging bc 2 early and payload not downloaded, 2 late and compromised → untrustworthy)

Man-in-the-middle = intercept and modify traffic * where authentication is weak * not always possible in dynamic switched network

Man-on-the-side = read traffc and add more * relies on timing advantage w traffic reaching faster than original request satisfied * needs significant resources (I.e like law enforcement)

Note: Hoster of mal code might not be compromised – e.g exploit might be redirection thru malvertising. * Sites cud look legit thru registering using puny code.

Active Network Based Infiltrations

Blind infiltration = no feedback * just deploy a payload

Semi-blind = # vulns u probe for will be small * avoids detection

General Worm Propagation Strats:

- Scanning = probe based on address generator * random scanning (effective 4 ipv4) * subnet scanning (seqeuntial scanning) * hit-list (payload contains chosen targets) * topological scanning (network or layer protocols – e.g peer-to-peer, IM, email address books) 2 provide info on hosts 2 infect) * Routing worms (Type of topological, use routing protocol info 2 id address blocks 4 target scanning)
- Co-ordinated scan = diff worm instance scan addresses using family of generated functions
- Flash = prepare vuln host structure before-hand, just propagate along it.
- Meta-Server = server asked for host names and addresses

- Topological = inform address gen using knowledge from infected hosts (e.g config files, logs)
- Contagion = propagate across existing comm channels

Kermack-McKendrick model 4 patient infection and has limited usefulness 4 computer worms * demos usefulness of vaccination strats (but timing is limited after intial infection)

Detection of scanning worms
Signature: not effective * worms move fast so sigs might not be available * polymorphic worms
Detect patterns in network traffic: use network telescopes (monitoring mechanisms 4 larger address spaces by watching dark unused address-space for suspicious activity, less useful for ipv6, less useful in non-random scanning strats) * requires further analysis when collected (filter out the noise, worms use stealth mechs – dyanmic address allocation/ network or port address translation 2 disguise behaviour, pattern ID or anomaly detection, reduce dimensionality of features/normalise data and eliminate noise/ filter/ machine learning/ kalmon filters – signal processing tool for filtering noisy signals - * effective 4 only simple propagation techniques * rely on distributed sensors w limited reliatbility and accuracy and can also be compromised * need a baseline (not trivial in dynamic environments)

Evading forensics: move fast so detection 2 slow to prevent infection * detect sandbox * obfuscate code (But Romberick worm made detection easier by adding 97% padding code – was intended 2 protect against competing authors, it was set to self destruct if detected analysis)

Example worms: SQL slammer (very fact, required no handshake, propagated over UDP in single packet) * Conficker worm (mutl variant w increasing functionality, re-enabled patched vulns, required expertise in mult domains to protect against, example of code re-use – common mistakes helps w attribution -)

Malware Ex-filtration Mechanisms
After infection, data commed to outside world * sneaky particularly import for C&C

Backdoor

- Simple = open port 2 be contracted over * network & port address translation tables can make difficult * firewall configs will oft not allow direct connections from external
- Port knocking = multi-stage access where connection opened on firewall after sequence of pre-specified connection attempts * used by legit services 2
- Reverse backdoor = contact out * avoids NAT and firewall issues * use 'cutout' or rendez-vous intermediate nodes to min traceability

Covert channel
Covert channel = used by adversaries to overlay communications

- Covert storage channels = use attribute of shared resources
- Covert timing channels = use temporal or ordering rel among access 2 shared resources * forcing paging * yielding line quantim * holding onto shared res for measured time
- Noiseless covert channel = use resource only avail to sender/reciever
- Noise covert channel = use resource also available to other subjects

Problem 4 adversaries establishing end points: scaling up ruins temporal comms * trying 2 prevent trace back * how do u learn of compromised host?

Removal and Mitigations: can't perfectly isolate in interactive environments * id covert channel computationally hard * reduce bandwidth of channels * ensure uniform resource availability * randomise resource access (and noise)

Fast-Flux networks = use DNS network obfuscation 2 reduce traceablity of C&C networks * uses fast IP changes & short TTL intervals * single flux = frequent change of host address * name server flux = addresses of name servers change * double flux = combo of both above
Note: these techniques also used by legit apps like Content Delivery Networks (amazon cloudfront, cloudflare etc)

Malware concealment
Binary Modification techniques: Substitution (overwrite target exe/memory block, may be trivial to identify)* prepending (malware executed 1st, obfuscate file-size/sig/relative addresses) * appending (req well known entry point or modified start, may reside in unclaimed areas) * cavity (inject code at arbitrary location and dynamically specify entry point, dynamic relocation → more code injected which can be protected by ofuscation techq) * multi-cavity (like cavity but payload split into multiple segment and inserted into diff areas)

Packers = obfuscation tech that compresses and encrypts malware payloads * may themselves employ polymorphism

Rootkits = conceal actions and permit take over of compromised sys w elevated privs * binary ‘user-land’ rootikits (sub or mod binary executables to avoid detection by tools, tools have 2 be known by author, law enforcement could use new tools or bypass interfact and go to raw, size become issue as rootkits grow sophisticated) * Kernel root kids (mod kernel components thru DKOM or loadable kernel modules etc, hooking In obv sys calls/driver interfaces = easy to detect, can hook in non-obv, analysis of kernel structures for anomalises/signatures = slow, analysis of structures assumes knowledge of live behaviour of kernel) * persistence (modify registy keys, start up files, boot records, firmware, addons to apps etc)

Malware Analysis Techniques
Detection: no perfect mechanism (halting prob)
Signature: Limited usuage * sometimes code re-used so may be detectable in new strains * most malware obfuscate

Static: snapshot of code might not be completely self contained (need knowledge of dependencies, some dependencies only exist in context IE registry keys) * requires disassembly (debugging symbols are unlikely 2 be included) * malware obfuscated delib against pop debugging tools

Dynamic: observing code run is less resource intensive * must run on machine/env that resembled target 4 actual behavior * malwr oft designed 2 detact sandbox env/obs tools and might self-destruct/act differently.

Mobile Forensics

General Mobile Device Forensics
Find data: Non-volatile memory ([u]sim, on-board mem, memory cards, SRAM) * Captured by net operator (geolocation traces) * Layered Services (e.g cloud)

Investigations
With Access

- Manual exam and screen capture: actions taken need 2 be recorded * mst likely 2 preserve integrity
- Interrogate Memory Locations: commercial tools * need to be certified against OS * rapid OS makes it hard 4 comm tools to keep up * might miss deleted/obfuscated Data

W/O Access
Reset the pin w personal unlocking key (from provider or device reset procedure) * smudges on glass 4 pin * Brute force pin (beware rate limiting, forced erasure of device)

Low Level Access

- Clone the (u)SIM: if damaged, inaccessible etc * might rely on non-universal Trusted Module in handset * isolate from network (else mite trigger remote wiping) * if locked sim not all data can be read/recovered
- Circuit-JTAG: circuit board might support test circuitry Joint Test Action Group * debug interface might support break points/line stepping * if need real time operation use an in-circuit emulator * interfaces oft proprietary/under NDA * multiple circuits → linked in Boundary Scan Chain (4 testing inerconnection) → can be obfuscated * JTAG may be unavailable (protective measures or bc damaged)
- Circuit-direct Memory Access: physical access not used (features 2 small, slight vibes throw off detecting) * Flash mem can survice damage * Access memory circuits thru reballing or chipoff tech * takes so much time only suitable 4 non-volatile * some circuits tamper resistant (self erasing/encrypted bus lines and memory/ devices hardened against side channel attacks)
- Flasher Boxes and JailBreaks: cant give exact effect of these on device * flasherboxes uses (usually) undocumented maintanence codes 2 access device * Jailbreaks attack vuln in OS

Simple Counter Forensics

Use stolen phones/disposable: actually easy to correlate (new phone activated near old, transferred info, similar calling patterns)

Pre-paid Sim cards: actually only avail in UK w/o registration

Android Mobile Forensics

Android Storage Access

Get to the linux sys below the android subsystem

Sections: Bootloader * Boot * Splash (power on scrn) * user data (internal storage 4 app data in /data) * system (contain binaries, libraries, pre-sinstalled apps, in /system) * cache (4 when restrt applications incl. Recovery logs and davik VM cache)

Data Storage: Find app data in shared preferences, internal storage, external storage, SQLite DB, network (e.g cloud service, altho amount of recoverable data depends on provider)

File System: Run time sys doesn't care what type * YAFFS2 is more useful 4 forensic analysis

YAFFS2: log structured * optimised for NAND flash * use chunks and blocks * data deletion (mark chunks as obsolete → garbage collection clears obsolete chunks (by default aggressive in low storage space conditions) → block refreshing for use levelling (although rarely performed))

Storage Encryption: full disk enc optional (bc it takes battery so manu didn't like) * encryption strong * key encryption key stored in trusted execution env to limit offline attacks * key is derived from password so only as strong as

Android Memory Access

Accessible over Android Debug Bridge (not accessble by default, but tools can bypass) * NANDroid backup (designed 4 rooting, allow full flash memory image, many tools w certifcation) * layering between linux host and dalvik app space → seperate analysis * mem layout of data structures not straightforward bc interpreted/compiled code

IOS Mobile Forensics

Note: Security thru obscurity

Choices for encryption: No protection (encrypted w UID key 2 prevent chip-off forensic and fast cryptographic wiping, unprotected when device on) * protected until first user auth * complete protection (require UID/passcode key, plaintext retained 10seconds after locking) * Protected unless open (if app open when locked, data kept in plaintext until file is closed)

Cloud backup = data available 2 law enforcement (thru a mutual assitance request)

Note: Encryption is prevalent, - iTunes and iCloud are more promising targets thru apple acccounts

< IOS 7, get device access thru pairing (Juice Jacking): user auths → device trusted → access 2 all data in plaintext * now File Relay no longer activated BUT pairing record still exploitable (use trusted device 2 get encrypted data then brute force offline)

> IOS 8, app encryption required: b4 they were unencrypted @ all times since remain authed when locked out

>IOS 8.3 lockdown cert for syncing device with previously paired iOS devices w/o re-entering credentials, these are gone now.

Effaceable Storage = 4 quick erasure * remote wiping commands * trigger on multiple PIN lockout

Biometric

- biometric data can obviously be forensically useful as well, although spoof protection is weak
- Biometric template data is stored a Secure Enclave within the A7 (iPhone 5s) or A8 (iPhone 6), A10 (iPhone 7) and A11/12 (iPhone 8/X/Xs/Xr), A13 (11) SoC, but calculations on sensor data and templates is performed outside this enclave — this may allow recovery of most recent sensor data capture
- SIM PIN and sometimes handset PIN (graphical passwords, etc.) can be re-set by PUK from provider or device re-set procedure
    - Biometrics can partially replace device PIN, but usually (e.g. iOS) not immediately after cold boot

‘The act of releasing an object does not mean the memory for the object is immediately overwritten. Thus, you can find traces of prior objects in freed or de-allocated memory long after the sockets have been used. This is discussed in more detail later in the chapter. ‘