Auditing and Forensics

University of Illinois

ECE 422/CS 461

Announcements

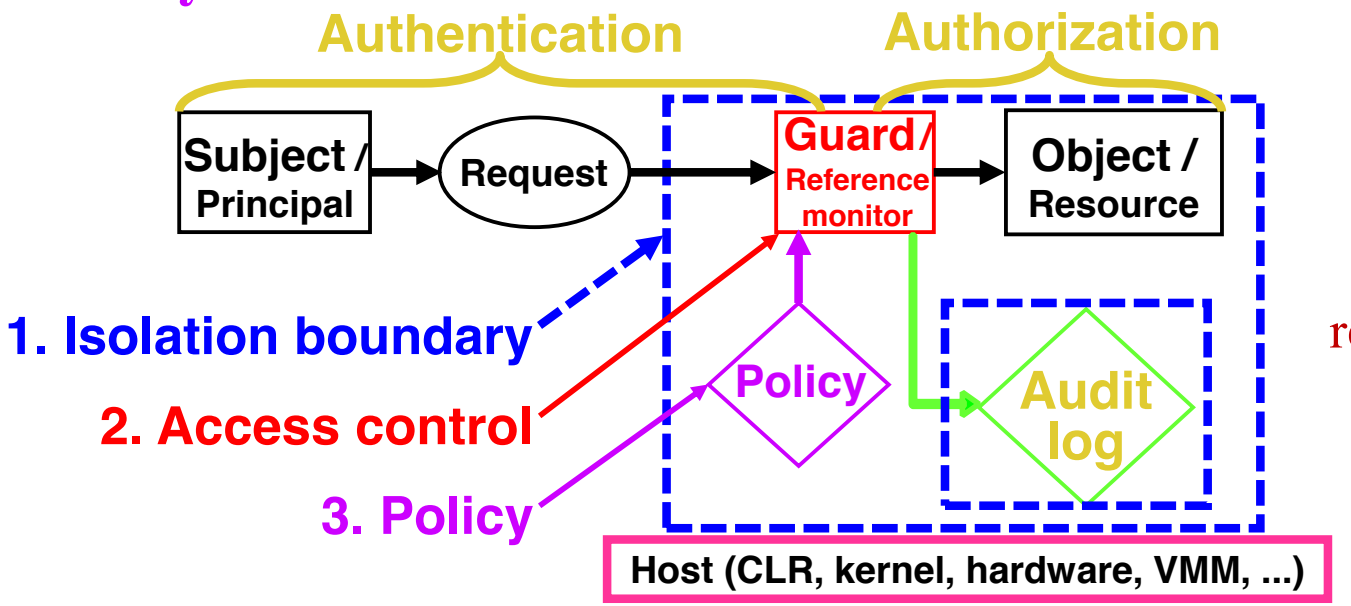
- Practice Exam available on Canvas!
  - Quizzes->Practice Quizzes->Midterm Exam (Fall 21)
- Questions about Midterm?
- MP2 due on Thursday.
- Extra Office Hours tomorrow during Discussion Sections (1-5pm) in Siebel 2406.

Goals

- By the end of this chapter you should:
  - Understand basic concepts in system auditing and how they relate to authorization
  - Identify the goals and procedures of digital forensics
  - Explore overview of methods for file systems forensics
  - Consider a variety of different auditing and forensic investigation frameworks

Access Control

1. **Isolation boundary** limits attacks to channels (no bugs)
2. **Access Control** for channel traffic
3. **Policy** sets the rules

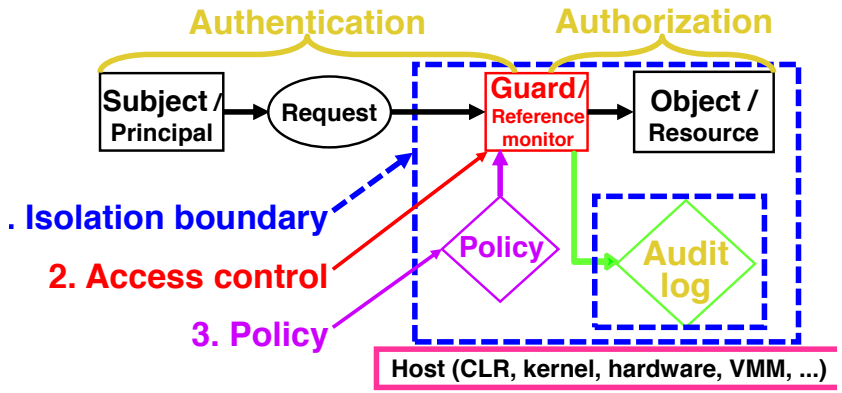


Anderson report 1972

[Lampson, SOSP'15]

The **Gold** Standard…

- **Au**thenticate principles: Who made a request?
  - People, but also channels/servers/programs
- **Au**thorize access: Who is trusted with a resource?
  - *Group* principles and resources to simplify management
- **Au**dit requests: Who did what when?



Computer Security Technology Planning Study, 1972

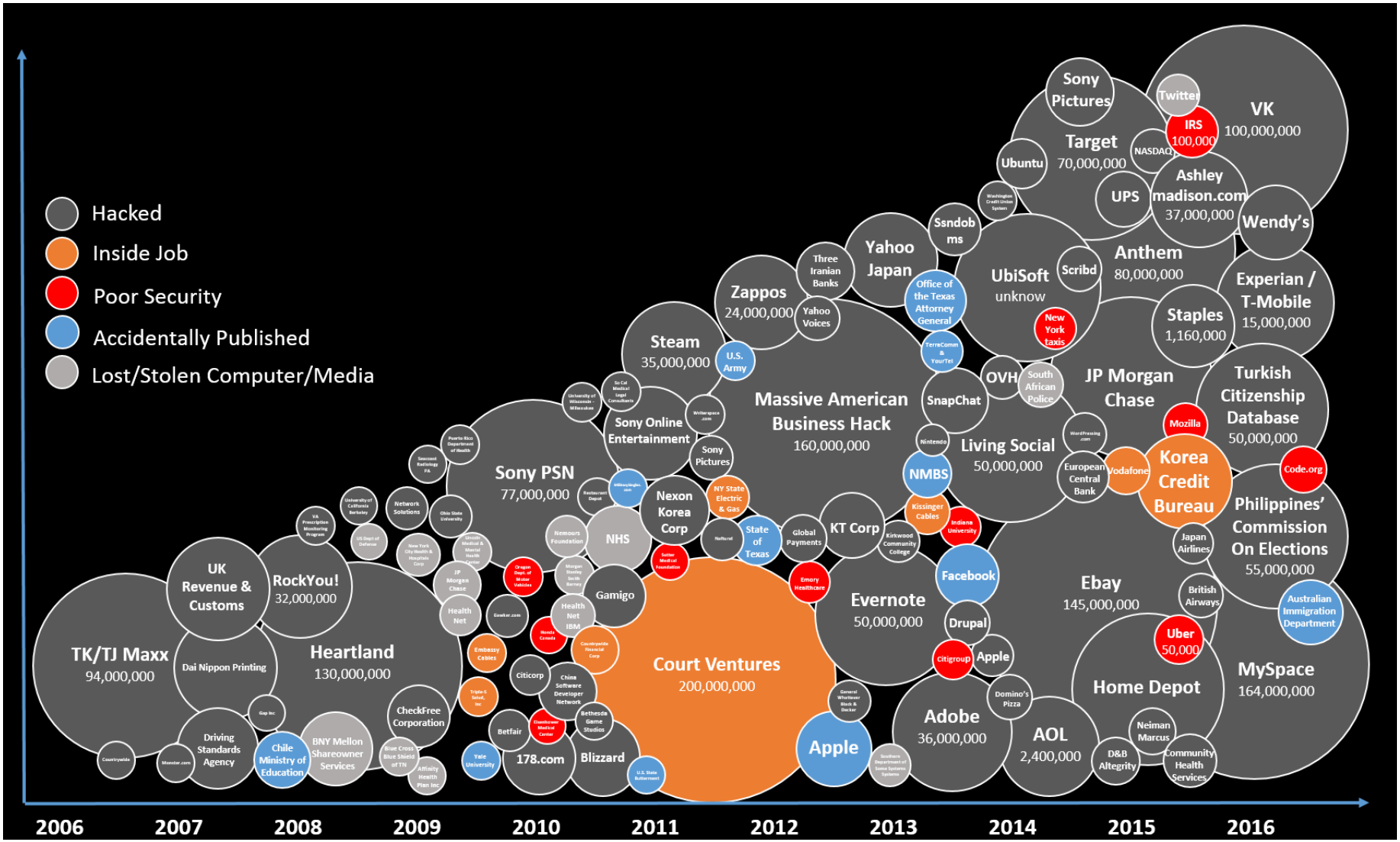
"The emphasis on an audit capability is a reflection of the desire to conduct security surveillance operations in a resource sharing system in order to detect breaches of security or penetration attempts... To date the emphasis on instrumentation has been for system performance measurement. While it can be seen that a security audit capability requires many of the same points of measurement, the security audit differs in what is recorded, and more importantly how it relates the measurement to the real world of users, terminals, communications lines, etc. Further, from a security audit viewpoint, while all possible measurements are not of interest all of the time, all possible measurement; will be of interest (not all at once) at some time."

- James Anderson

Auditing Motivation

- Dating back to earliest days of access control, violations of security policy were expected and anticipated.
- When violations occur, we need a way to detect, investigate, and respond to such incidents.
- "Perfect Security" would not require auditing, but even at the height of secure system design it was acknowledged that this was unattainable.

We are far from perfect security



Source: World's Biggest Data Breaches, Information is Beautiful

Advanced Persistent Threats

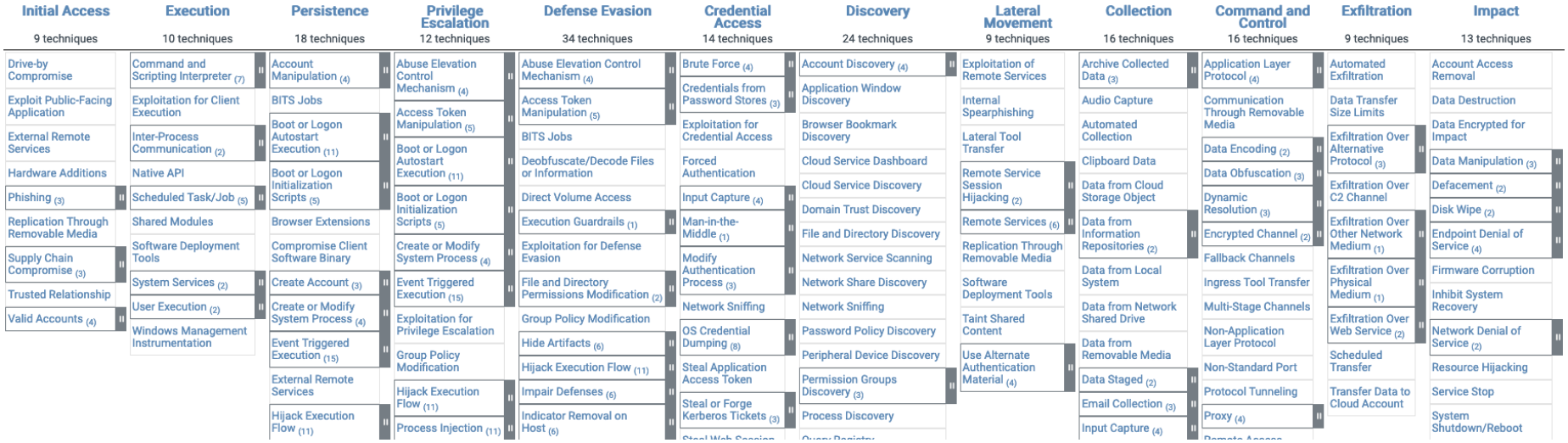
- These intrusions are driven by Advanced Persistent Threats (APTs).
- In contrast to simple malware, APTs are sophisticated and stealthy threat actors, often sponsored by nation states.
- Known for their ”Low and Slow” attack pattern:
  - Low: *Stealthy*, using existing system resources to “Life off the Land” and avoid detection.
  - Slow: Take their time to spread their presence in the system; “dwell times” are often many months.

ATT&CK Enterprise Matrix

In a nutshell, the MITRE ATT&CK matrix is designed to build a taxonomy APT behaviors.

Tactics

Techniques



APT Tactics

1. Reconnaissance: gather information for planning future operations.
2. Resource Development: establish resources to support operations.
3. Initial Access: get into the target environment.
4. Execution: run malicious code.
5. Persistence: maintain foothold.
6. Privilege Escalation: Gain higher-level permissions.
7. Defense Evasion: Avoid being detected.

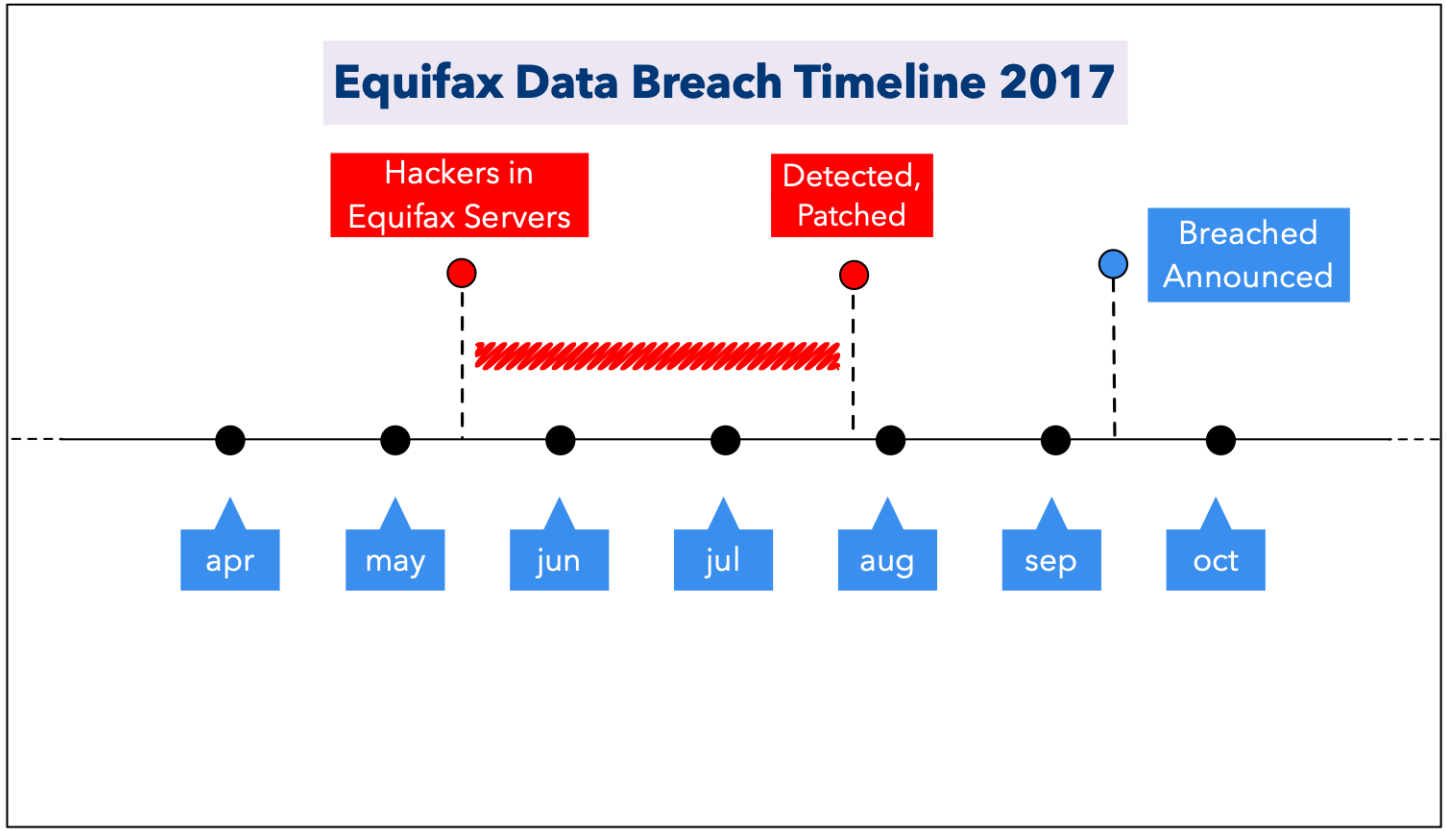
APT Tactics

8. Credential Access: Steal account names and passwords
9. Discovery: Figure out the target's environment.
10. Lateral Movement: Move through the environment.
11. Collection: Gather data of interest to their goal.
12. C&C: Communicate with compromised systems to control them.
13. Exfiltration: Steal data
14. Impact: Manipulate, interrupt, or destroy systems and data.

Advanced Persistent Threats

Insight: Many data breaches take time to execute…

…. creating an opportunity for defenders to repel the attack.



System Auditing

- Provides record of events to enable attack investigation and reconstruction
- Audit logs describe data's life cycle:
  - Modification
  - Deletions
  - Creations
- Also describes relationships between processes
- We can analyze audit logs to identify relationships and dependencies between different system events!

Linux Audit Framework

- Linux Audit creates audit records inside the kernel
- Available on vanilla Linux kernels > version 2.6
- It collects information regarding:
  - Kernel event (System calls)
  - User events (Audit-enable programs)
- Does not provide additional security in and of itself — e.g., it does not protect your system from unauthorized data accesses.

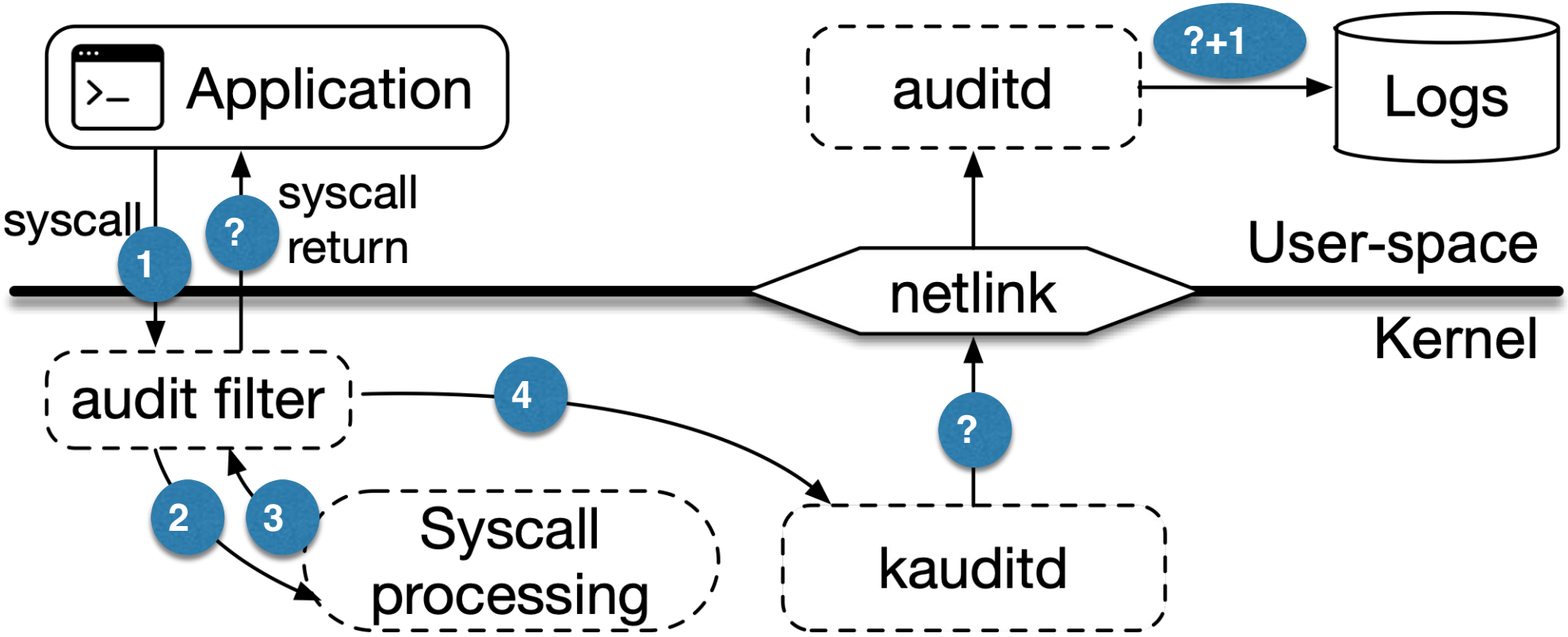
Linux Audit Use Cases

- **Watching File Accesses**: Audit can track whether a directory or file has been accessed, modified, exec'd.
- **Monitor System Calls**: Generate a log entry every time a particular system cal is used.
- **Monitor Network Access**: `iptables` and `ebtables` can be configured to trigger audit events.
- **Record commands run by user terminals**

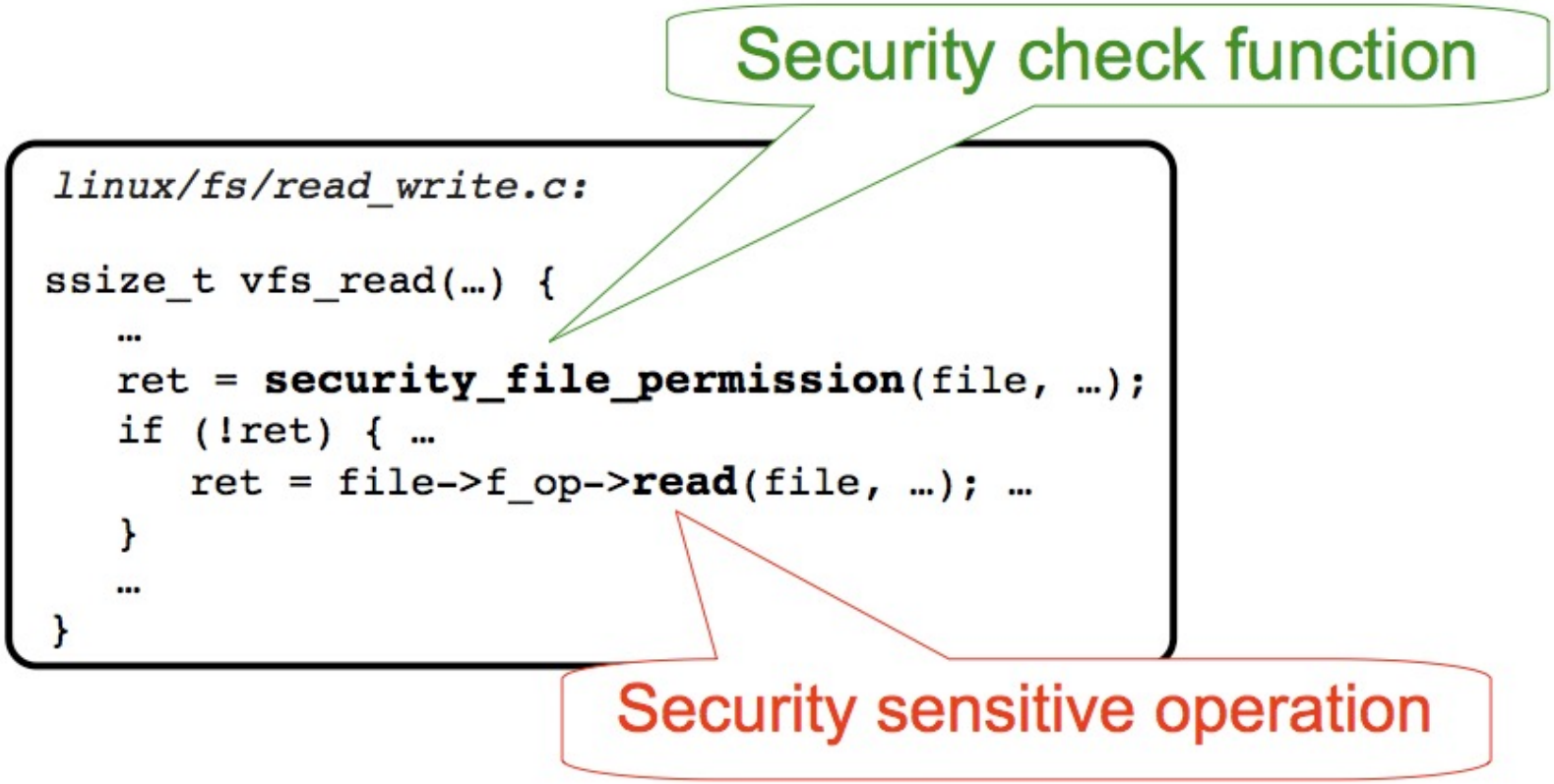
How Linux Audit Works

- Auditing hooks around the kernel intercept system calls and records the relevant context
  - Where are audit hooks placed relative to security hooks?
- The auditd daemon ingests kernel events via a netlink socket and writes the audit reports to disk/network.
- Various command line utilities take care of displaying, querying, and archiving the audit trail.

How Linux Audit Works



Comparing Audit Filters to LSM's Security Hooks



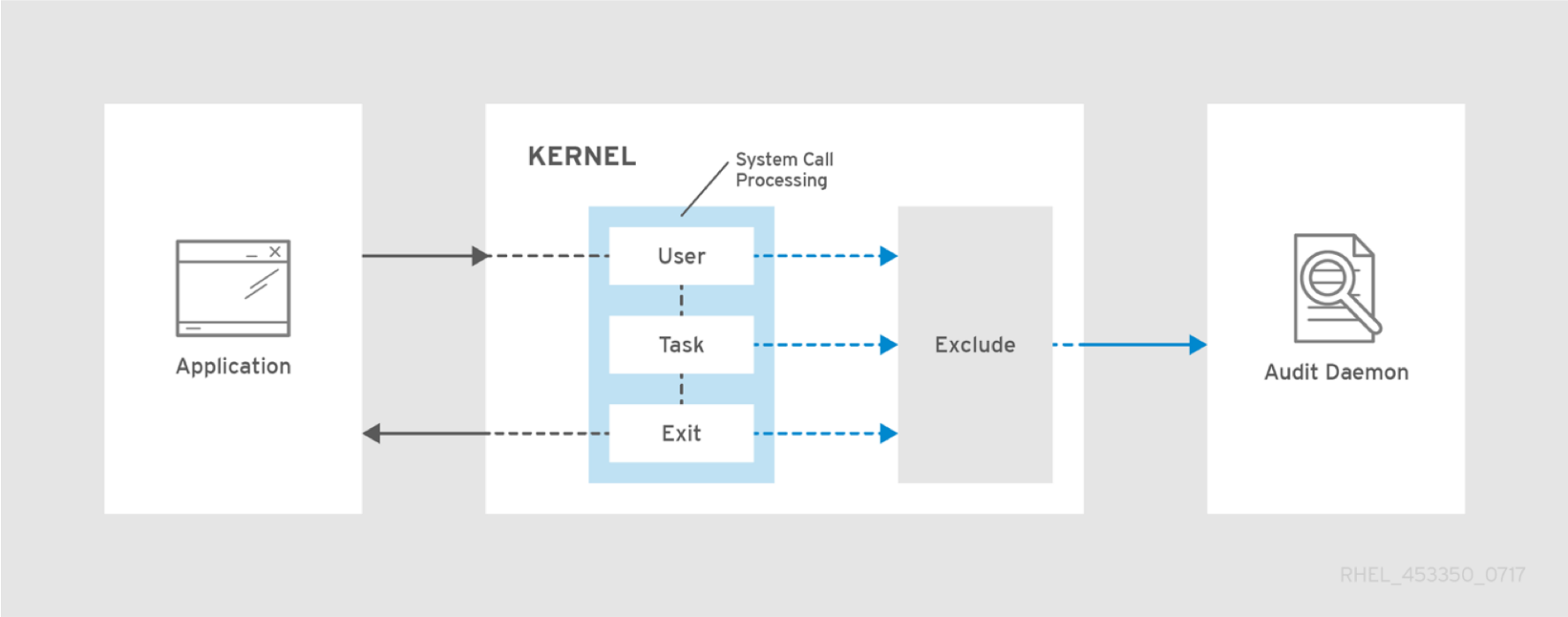
Comparing Audit Filters to LSM's Security Hooks

```
static inline void syscall_enter_audit(struct pt_regs *regs, long syscall)
{
        if (unlikely(audit_context())) {
                unsigned long args[6];

                syscall_get_arguments(current, regs, args);
                audit_syscall_entry(syscall, args[0], args[1], args[2], args[3]);
        }
}
```

```
static void syscall_exit_work(struct pt_regs *regs, unsigned long ti_work)
{
        bool step;

        audit_syscall_exit(regs);

        if (ti_work & _TIF_SYSCALL_TRACEPOINT)
                trace_sys_exit(regs, syscall_get_return_value(current, regs));

        step = report_single_step(ti_work);
        if (step || ti_work & _TIF_SYSCALL_TRACE)
                arch_syscall_exit_tracehook(regs, step);
}
```

Linux Audit Filtering

All hooks are defined, but may not be triggered based on active audit configuration….



Linux Audit Utilities

- `auditctl` — utility for managing the auditd daemon; returns information on the audit subsystem's current status and can be used to add and delete rules
- `ausearch` — utility for searching for events in log files
- `aureport` — utility for generating reports on the audit system
- `autrace` — utility for tracing a specific process with custom rules (think `strace`)
- `audisp` — 'multiplexor' that sends events to other programs that want to analyze events in realtime

Creating Rules

- auditctl is command line utility to :
  - Control behaviour of audit daemon (auditd)
  - Add and remove audit rules
- There are two main types of rules:
  - File system audit rules
  - System call audit rules

File System Rules

- File System rules are sometimes called watches.
- Used to audit access to particular files or directories that you may be interested in.
- The syntax of these rules generally follow this format:

```
-w path-to-file -p permissions -k keyname
```

- permission are any of the following:

  r - read of the file

  w - write to the file

  x - execute the file

  a - change in the file's attribute

System Call Rules

- Loaded into a matching engine that intercepts each syscall that programs make.
- Very important to only use syscall rules when you have to since these affect performance.
- The syntax of these rules generally follow this format:

```
-a action,list -S syscall -F field=value -k keyname
```

- To see files opened by a specific user:

```
-a exit,always -S open -F auid=1337
```

- To see unsuccessful open calls:

```
-a exit,always -S open -F success=0
```

Linux Audit Example

- To track a file by inode number:

```
# auditctl -a exit,always -S open -F inode=`ls -i /etc/auditd.conf | gawk '{print $1}'`

# auditctl -l AUDIT_LIST: exit,always inode=1637178 (0x18fb3a) syscall=open
```

  - When someone opens the file, this message is logged

```
type=PATH msg=audit(1251123553.303:206): item=0 name="/etc/audit/audit.rules" inode=77546

dev=fd:01 mode=0100640 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:auditd_etc_t:s0
```

aureport example

Produce a summary report for a given timeframe:

```
[root@tecmint ~]# aureport -ts yesterday -te now --summary -i

Summary Report
======================
Range of time in logs: 09/18/2017 11:48:11.093 - 09/19/2017 06:01:01.589
Selected time for report: 09/18/2017 00:00:00 - 09/19/2017 06:51:24
Number of changes in configuration: 641
Number of changes to accounts, groups, or roles: 0
Number of logins: 11
Number of failed logins: 9
Number of authentications: 31
Number of failed authentications: 10
Number of users: 4
Number of terminals: 11
Number of host names: 2
Number of executables: 8
Number of commands: 7
Number of files: 1
Number of AVC's: 0
Number of MAC events: 11
Number of failed syscalls: 0
Number of anomaly events: 1
Number of responses to anomaly events: 0
Number of crypto events: 234
Number of integrity events: 0
Number of virt events: 0
Number of keys: 4
Number of process IDs: 455
Number of events: 1236

[root@tecmint ~]#
```

aureport example

Produce a report describing all logins:

```
[root@tecmint ~]# aureport -l

Login Report
============================================
# date time auid host term exe success event
============================================
1. 08/09/2017 12:49:55 -1 ? tty1 /usr/bin/login yes 340
2. 08/09/2017 13:02:47 -1 ? tty1 /usr/bin/login yes 366
3. 08/09/2017 13:07:38 -1 ? tty1 /usr/bin/login yes 357
4. 08/09/2017 13:18:47 (unknown) 192.168.56.1 ssh /usr/sbin/sshd no 408
5. 08/09/2017 13:19:38 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 426
6. 08/09/2017 15:59:56 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 370
7. 08/10/2017 01:38:53 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 374
8. 08/10/2017 11:34:12 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 379
9. 08/12/2017 12:06:31 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 373
10. 08/17/2017 12:04:00 -1 192.168.56.1 /dev/pts/0 /usr/sbin/sshd yes 377
11. 08/28/2017 13:27:04 -1 ? tty1 /usr/bin/login yes 367
12. 08/28/2017 17:16:22 -1 ? tty1 /usr/bin/login yes 342
13. 08/29/2017 23:31:17 -1 ? tty1 /usr/bin/login yes 359
```

Threat Investigation

- aureport lets you sample logs activity, but what if you need to drill down into specific chains of events?
- Example: Your AntiVirus detects malware. How did it come to exist on your machine, and what did it do?

```
chromium.exe reads from ip 10.0.0.2
chromium.exe reads from ip 165.10.0.1
chromium.exe reads from ip 91.0.0.2

  …

chromium.exe downloads a.ppt
chromium.exe downloads b.doc
chromium.exe downloads malware.exe

  …

malware.exe reads /etc/passwd
malware.exe sends /etc/passwd to ip
X.X.X.X
```

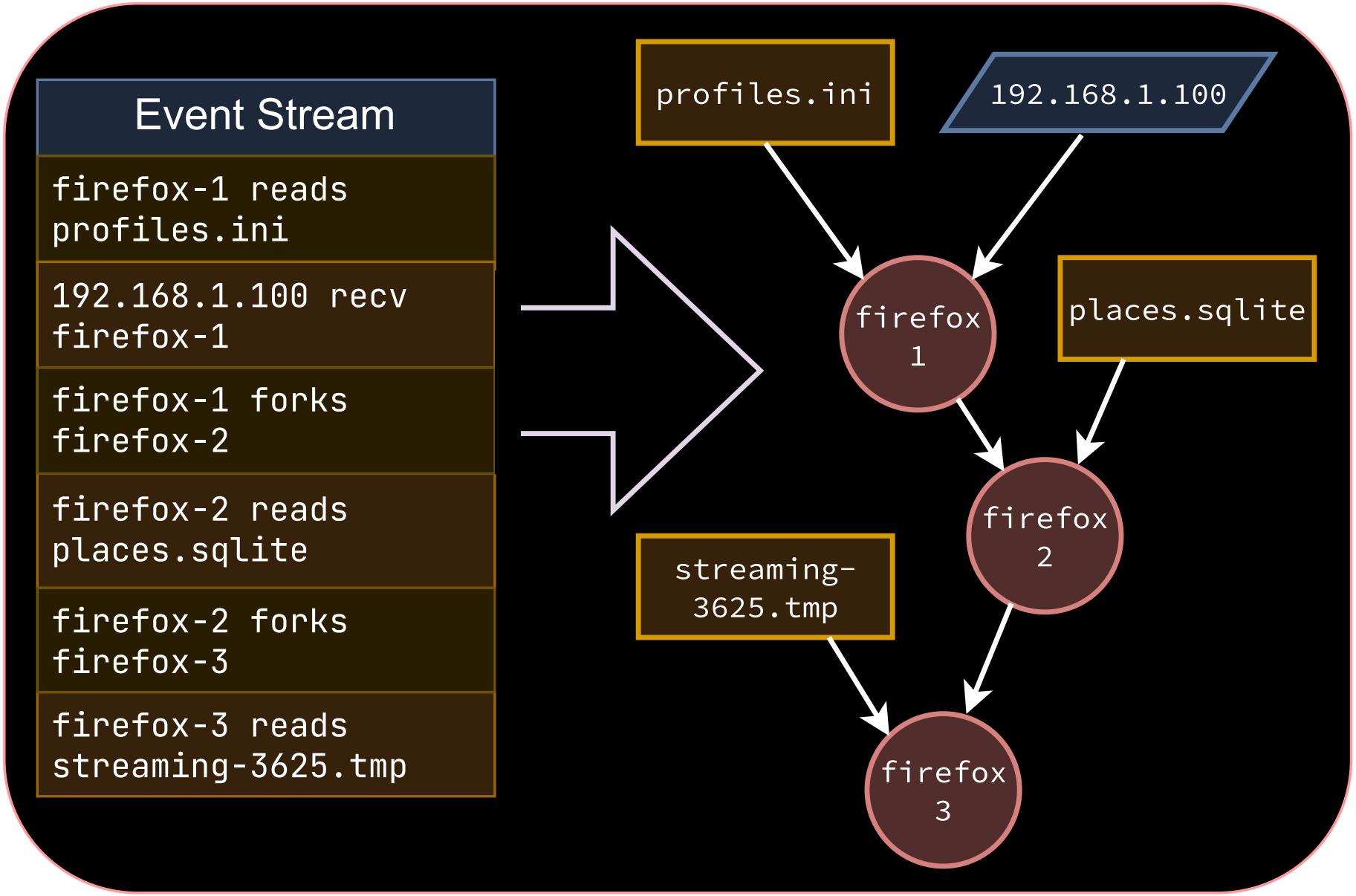
Threat Investigation

- One option would be to store logs in a relational database, then use SQL to query them…
- But queries would get big very fast, featuring lot of tedious join statements.
- Is there a better option?

Run query + New Save

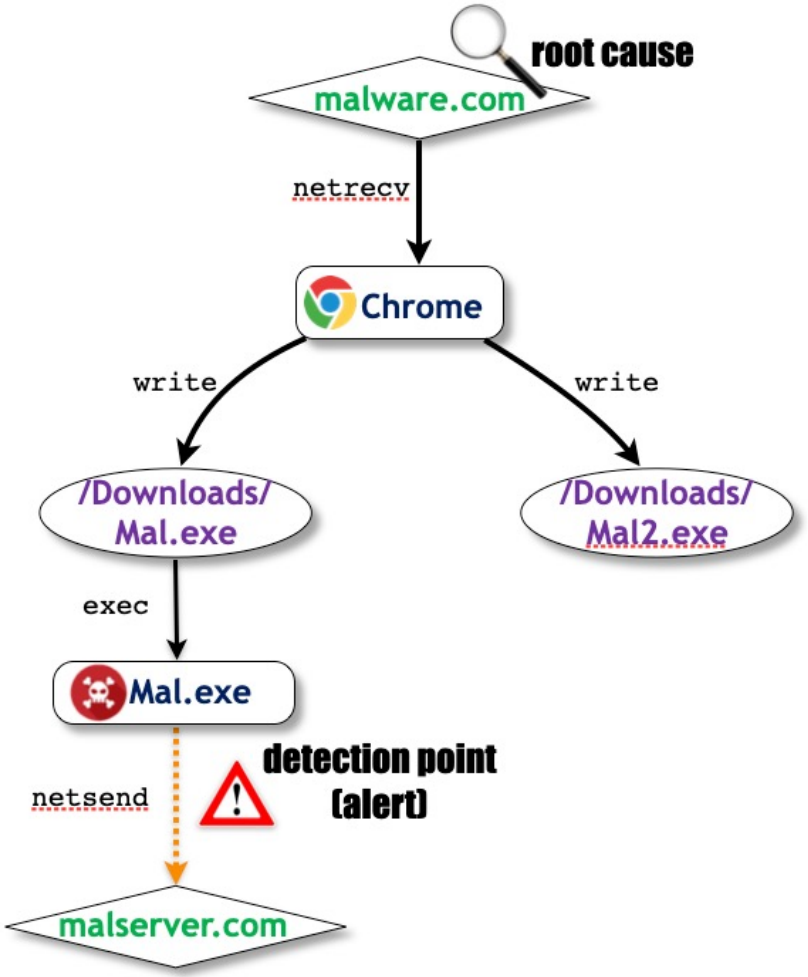
```
ProcessCreationEvents
| where ProcessCommandLine has "update.exe"
| where (ProcessCommandLine contains "http") and (ProcessCommandLine contains "--update")// a
| extend exeURL = case(ProcessCommandLine has "=",split(ProcessCommandLine, "=", 1),
                       ProcessCommandLine !has "=", split(ProcessCommandLine, "--update ",1),
                       "Default")
| where exeURL != "Default"
| sort by EventTime desc
|project EventTime,
          ComputerName,
          exeURL,
          FolderPath,
          ProcessCommandLine,
          AccountName,
          InitiatingProcessCommandLine,
          ReportId,
          ProcessId,
          InitiatingProcessId
```

Audit Logs as a Causal (Provenance) Graph



Data Provenance

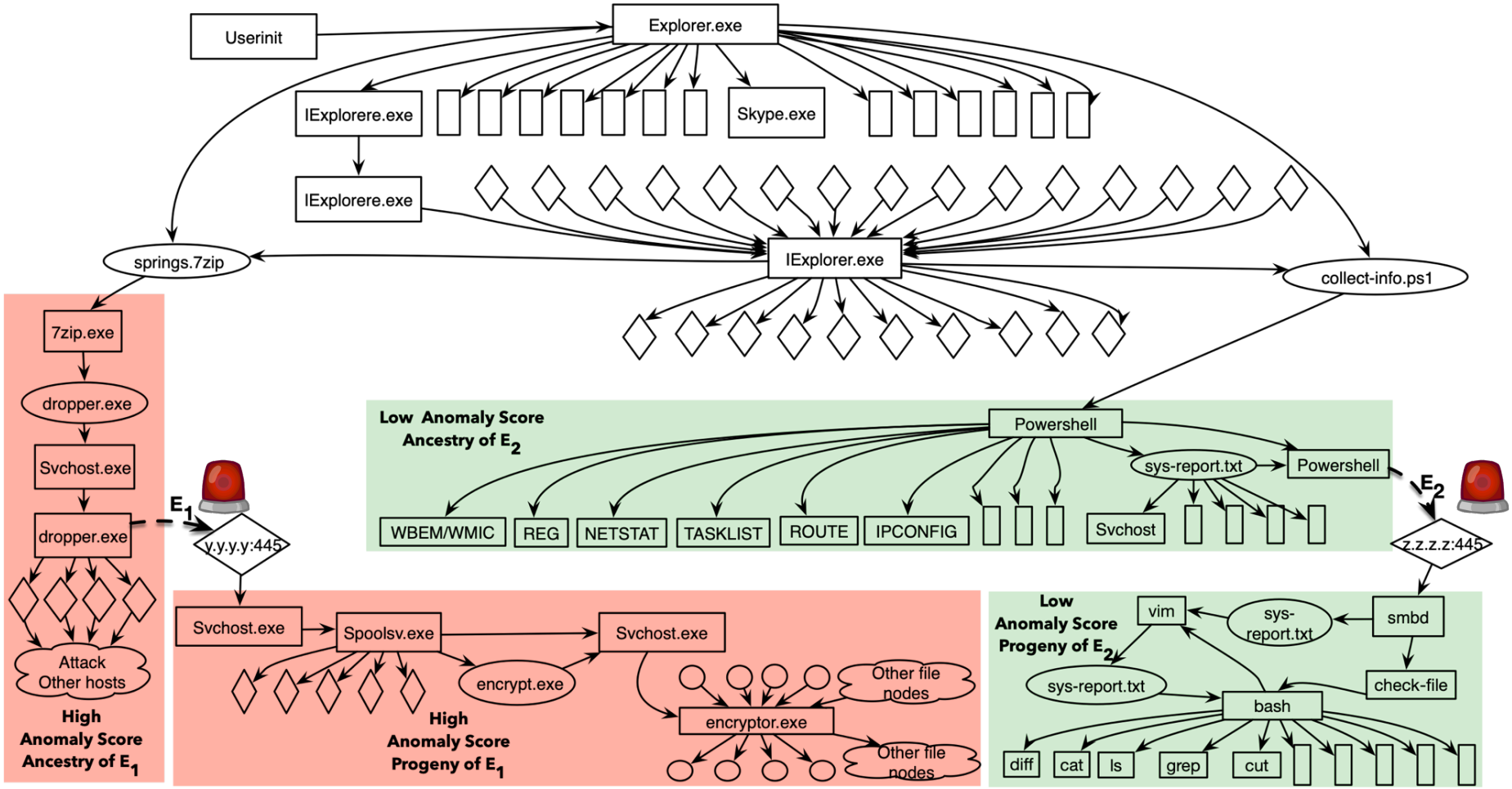
- Idea: Model related log events as a causal relationship graph.
  - Vertices: Files, Processes, etc.
  - Edges: System Accesses (e.g., `read`, `write`, `fork`)
- Backtrace queries identify root cause of a detection point
- Forwardtrace queries identify full attack footprint starting from a root cause.



- We can perform causal analysis of attack behaviors using provenance graphs.

Investigating Security Alerts

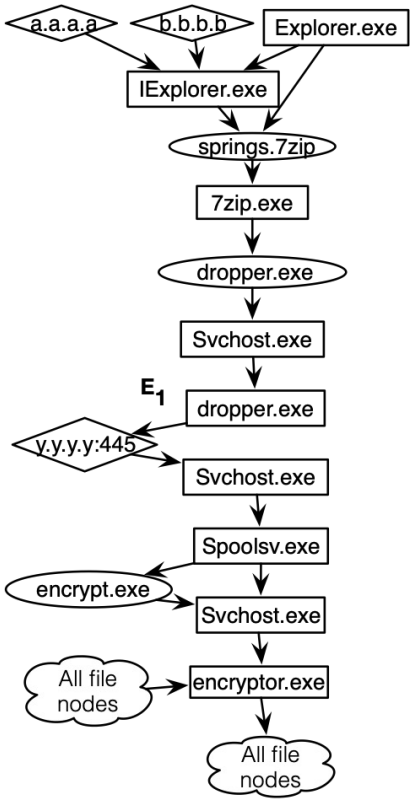
We can use provenance analysis to investigate security alerts fired by other monitoring products:



WannaCry attack scenario

Challenges in Auditing

- Dependency Explosion: From log's perspective, each new process output is dependent on all prior process inputs!
  - A "Semantic Gap" issue
- Log Storage and Management – huge amounts of data generated in a very small amount of timte.



The same WannaCry attack scenario, pruned to remove Dependency Explosion

Digital Forensics

- Forensics – *the use of science or technology to discover evidence for a court of law*
  - Used in both criminal and civil law
- Digital forensics – forensics relating to digital devices
- Usually trying to recreate a chain of events
- Audit logs are great for digital forensics, but they're just one of many tools used by forensic investigators!

Daubert Standard

- Rules for the admissibility of scientific evidence in a court of law
- Evidence must be relevant and reliable
  - Is the method subject to testing?
  - Are there established error rates?
  - Is it generally accepted by the community?
  - Has the technique been peer reviewed?
- Judge is the ultimate “gatekeeper”

Investigation Procedure

1. Preparation – get authorization, equipment, and personnel
2. Survey – consider the scene and make a plan
3. Preservation – seize physical devices and data
4. Examination – extract evidence (artifacts)
5. Analysis – combine evidence and draw conclusions
6. Reporting – summarize findings

Digital Artifacts

Operating system: event logs, registry data

File system: access times, modification times

Disk: deleted files, hidden partitions

Internet: browser history, email

Media: photos, videos, audio

Documents: Office, PDFs, RTF, XML

Databases: MySQL, Oracle

Application data: instant messaging

Preservation Decision

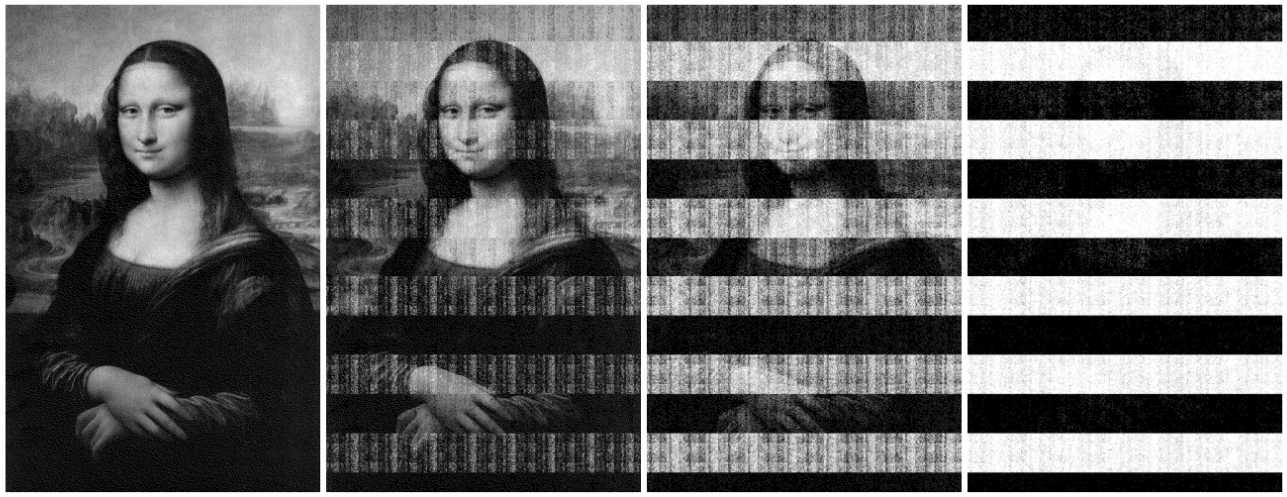
- Live analysis – look at evidence on live computer
  - prevents file encryption/passwords
  - risks damaging evidence
- Dead analysis – image drives
  - bit-for-bit forensic duplicate using a write blocker
  - analysis can be conducted later
  - identify hidden or deleted files

Preservation Decision - Live Analysis

- When system is running, additional information can be collected
  - active connections
  - active processes
  - loaded kernel modules
  - open files
- Problems
  - analysis modifies state of system –
  - rootkits (or other modifications) can tamper with results

Preservation Decision - Live Analysis

- Physical (volatile) memory is also more persistent than one thinks
  - RAM taken out of computer and scanned for sensitive content



Lest We Remember: Cold Boot Attacks on Encryption Keys Usenix Security 2008

FILE SYSTEM FORENSICS

Common File Systems

UFS: Unix File System

Ext[2,3,4]: Extended file system

FAT[12,16,32]: File Allocation Table

NTFS: New Technology File System

ReFS: Resilient File System

HFS[+]: Hierarchical File System

NTFS Forensics

- To extract forensically relevant information such as
  - Deleted files
  - Event logs
- NTFS Basics:
  - Everything is a file, even the core file system internals
  - The internal files are always hidden from user view

Hidden Internal Files

Filename	Description
$MFT	Master File Table
$MFTMirr	Backup of first 4 records of MFT
$LogFile	Transaction log file
$Volume	Volume related information, usually empty
$AttrDef	$AttrDef Table listing MFT attribute names and numbers
.	Root folder on NTFS
$Bitmap	Map showing which clusters on volume are in use
$Boot	Boot code used during bootstrap
$BadClus	Map of bad clusters
$Secure	Security descriptors and ACLs are listed here
$Upcase	Keeps all lowercase to uppercase character mappings
$Extend	Optional extensions listed here (This is a folder)

Image taken from https://www.slideshare.net/nullowaspmumbai/ntfs-forensics-66460882

Physical Layout of NTFS Volume

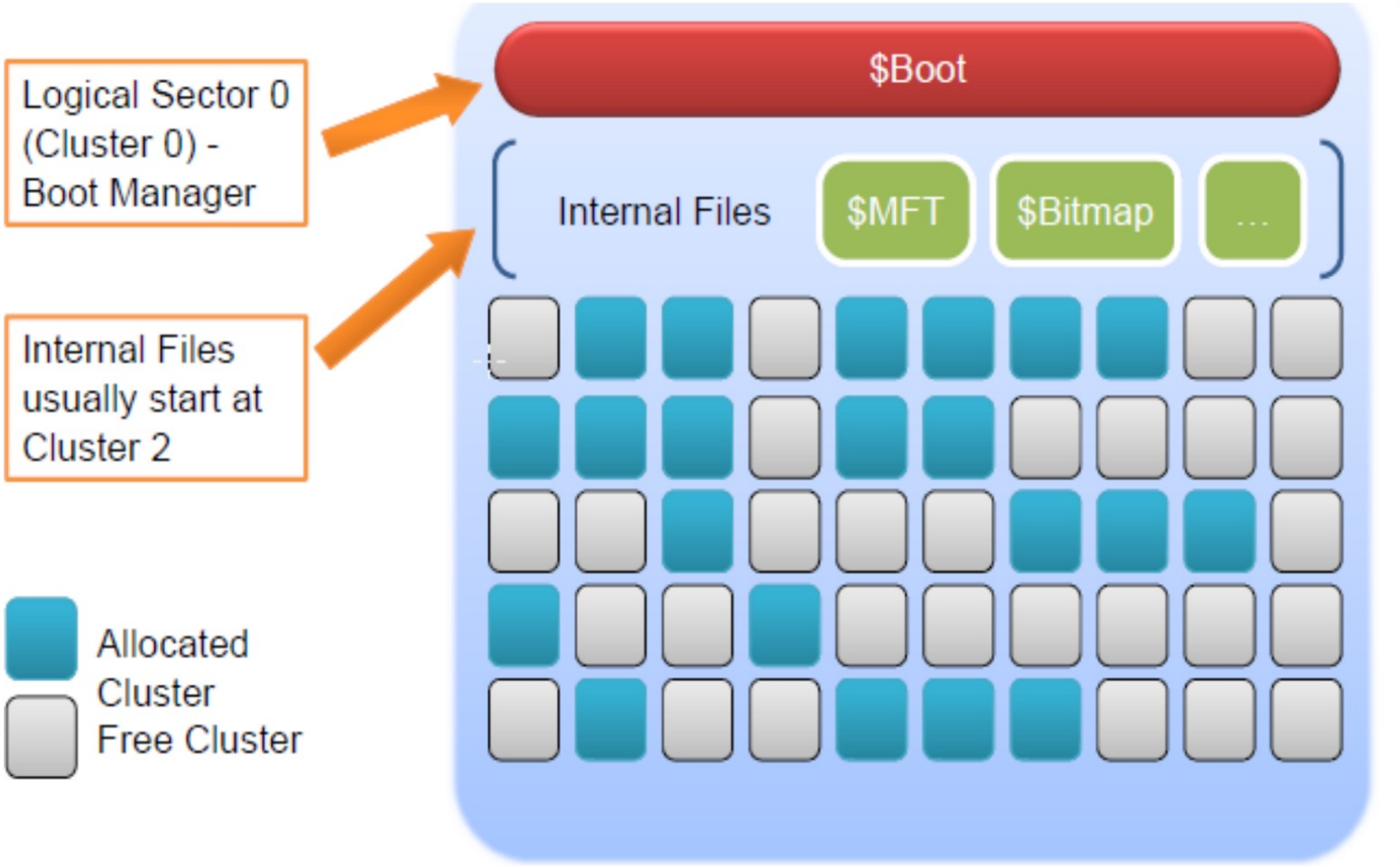


Image taken from https://www.slideshare.net/nullowaspmumbai/ntfs-forensics-66460882

Master Boot Record (MBR)

- A data structure stored on the first sector of the drive.
- Cross-platform industry standard for locating and booting disk partitions
- Used by BIOS to boot (start) the OS
- Contains boot loader code to fetch and execute the operating system.

Master Boot Record (MBR)

- A special boot sector data structure for partitioned disks.
- Data structures describing partition
  - Magic number (for NTFS)
  - Bytes per sector
  - Sectors per cluster
  - Sectors per track
  - Location of master file table ($MFT) and mirror
  - Serial number, checksum
- Bootstrap code (loads operating system)

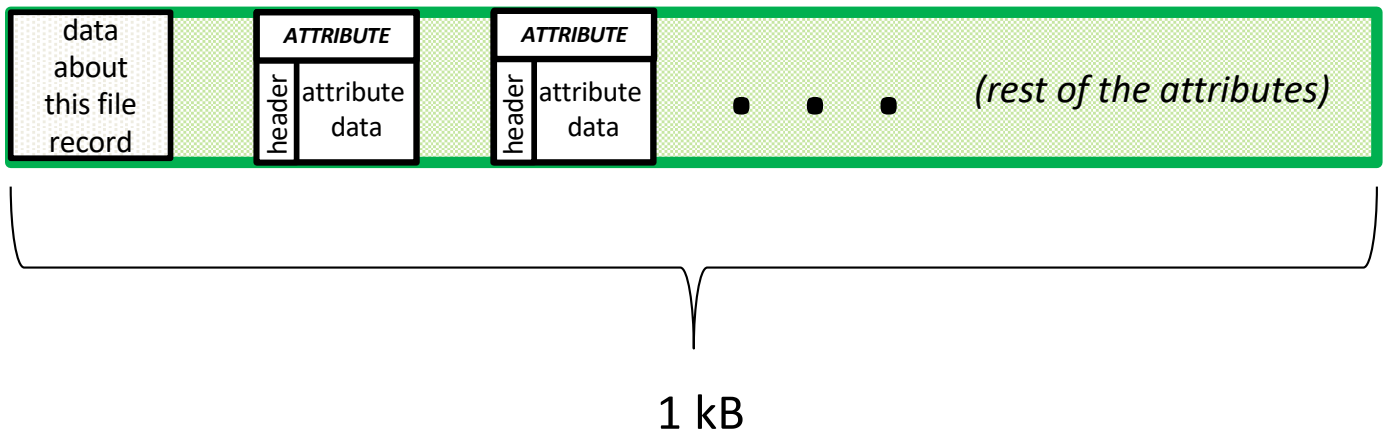
NTFS: Master File Table

- File/folder records are stored in $MFT
- Records are 1 kb
- Records contain ***attributes*** that define the characteristics of a file, including the data itself

MFT File Record Attributes

- Attributes define the characteristics of a file, including the data
- What if file information is too large for record?
  - Resident Attributes vs Non-resident Attributes

a file table record:



File Attributes

- Files in NTFS typically have the following attributes:
  - **$STANDARD_INFORMATION:** Contains MAC times, security ID, Owners ID, permissions in DOS format, and quota data.
  - **$FILE_NAME:** Contains the file name in UNICODE, as well as additional MAC times, and the MFT entry of the parent directory.
  - **$OBJECT_ID:** Identifiers regarding the files original Object ID, its birth Volume ID, and Domain ID.
  - **$DATA:** The raw content data of the file.

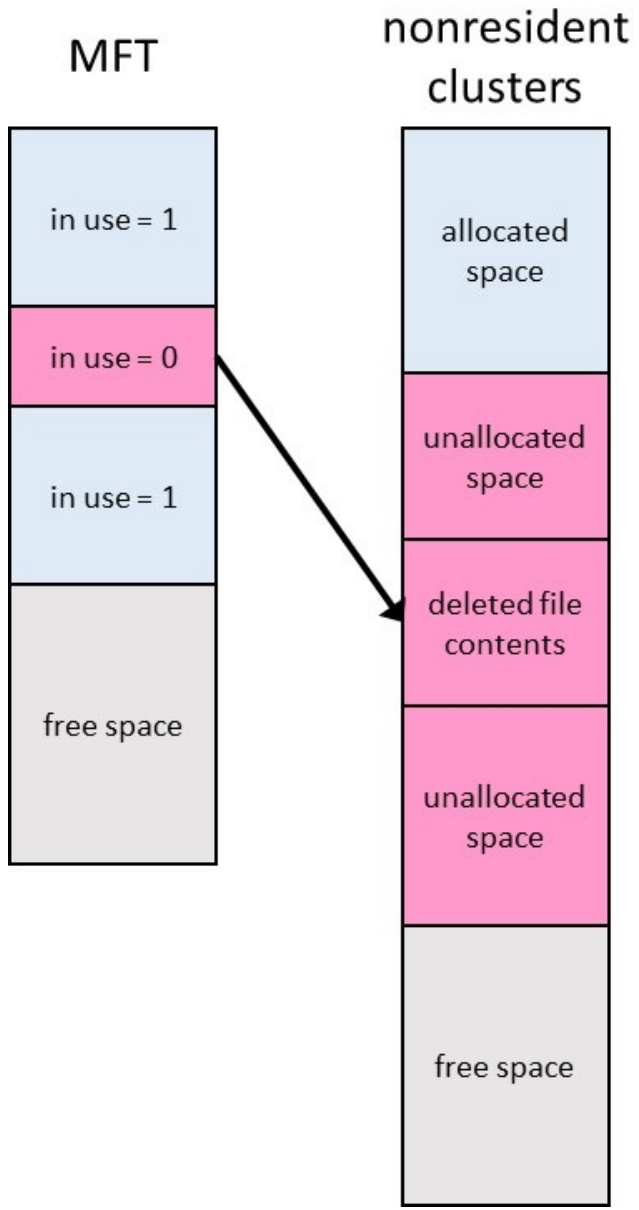
Deleting a File

- When a file is deleted the IN_USE flag is cleared from the MFT entry, but the attribute contents still exist.

- Can be used to recover path of deleted file.

Recovery with File Record

- File record intact.
- File contents intact.



FILE CARVING

File Carving

- Modern file systems tend to overwrite metadata for deleted files
- Recovery of files ***without*** file system metadata
- Can be done without ***any*** metadata
- Can carve many different kinds of media

Recovery vs. Carving

- Recovery: file system metadata are intact; use them to find file (“undelete”)
- Carving: pulls the ***raw*** bytes from the media

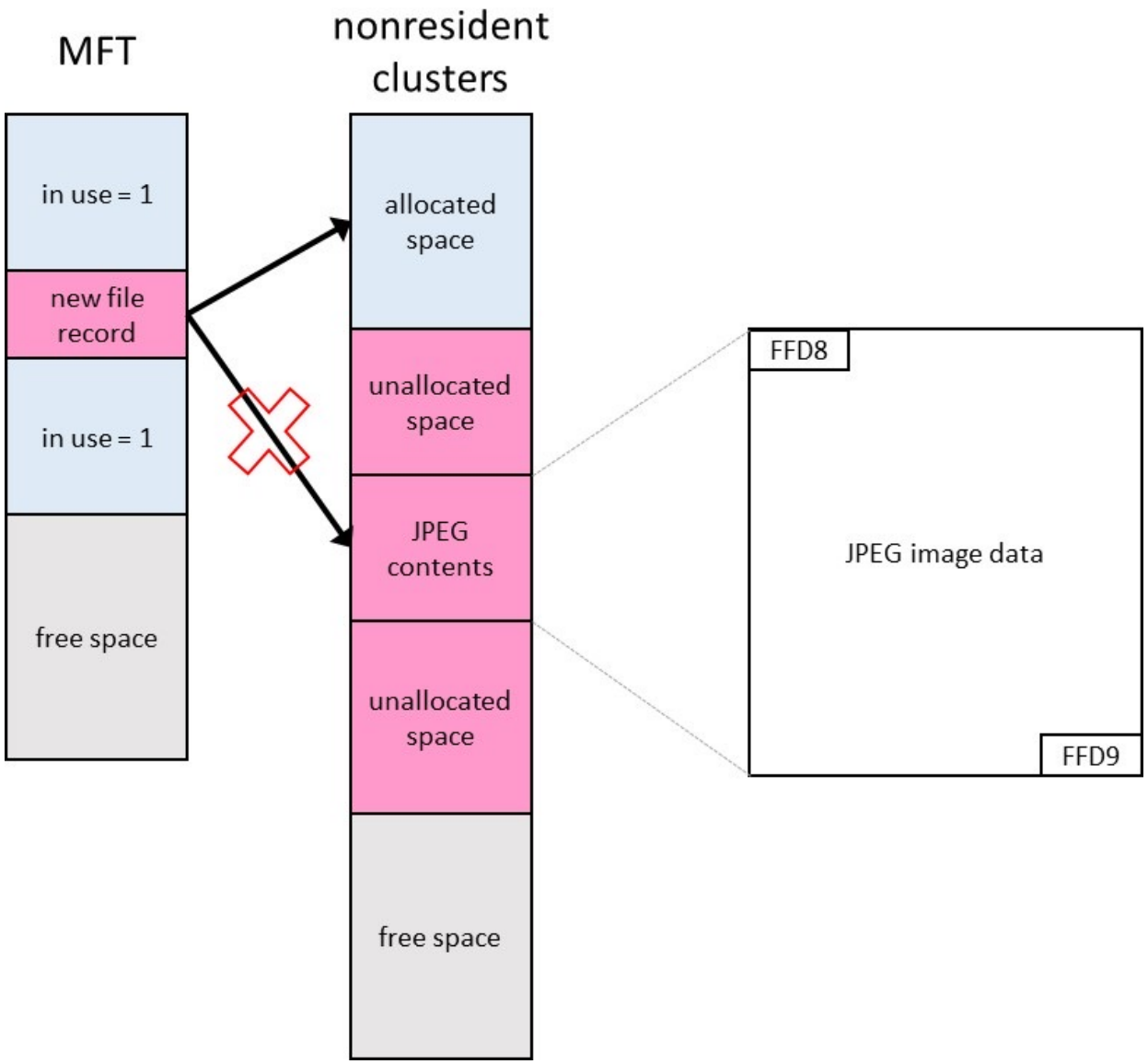
Basic File Carving Techniques

- Header-footer carving
- File-structure-based carving
- Content-based carving

Header-Footer Carving

- Many file types have standard headers and footers stored inside
- Example: A JPEG starts with “Start of image” header 0xFFD8 and ends with “End of image” footer 0xFFD9
- Carve out everything between JPEG header and footer $\rightarrow$ image file

Carving a JPEG



Header-Footer Carving Problems

- Header or footer might be a common string.
  - E.g., the header of an MP3 is “mp”
  - Produces false positives
- The beginning of the file might be missing
  - If file was deleted, might get squashed by a newer file
- The footer might be missing
  - Big output!
- The disk could be fragmented

File Structure Carving

- Use internal file metadata if available
- Find cluster size
- Read entire cluster and hunt for internal signatures (in addition to header/footer)
- Example: Foremost, PhotoRec

Content-Based Carvers

- Look for statistical signatures indicating language or file content
- Machine-learning/statistic-based
- Semantic carvers

Resources

- Audit manual pages
  - There are several man pages installed along with the audit tools that provide valuable information about each utility
  - Linux Audit Project:
    - http://people.redhat.com/sgrubb/audit/index.html
  - The SPADE Project (for graph-based analysis)
    - » https://github.com/ashish-gehani/SPADE