Goals for Today

- Learning Objective:
  - Understand the importance of auditing to computer security
  - Explore Linux Audit Framework components
  - Learn the gist of causal analysis of audit logs
- Announcements, etc:
  - MP4 out! **Due May 6th (UTC-11)… 17 days!**
  - **Final Exam — MAY 9TH @ 7PM, SIEBEL 1404**

Reminder: Please put away devices at the start of class

CS 423
Operating System Design: Auditing Frameworks

Professor Adam Bates

Computer Security Technology Planning Study, 1972

"<u>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</u>… 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

- Even in the Multics project, 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.

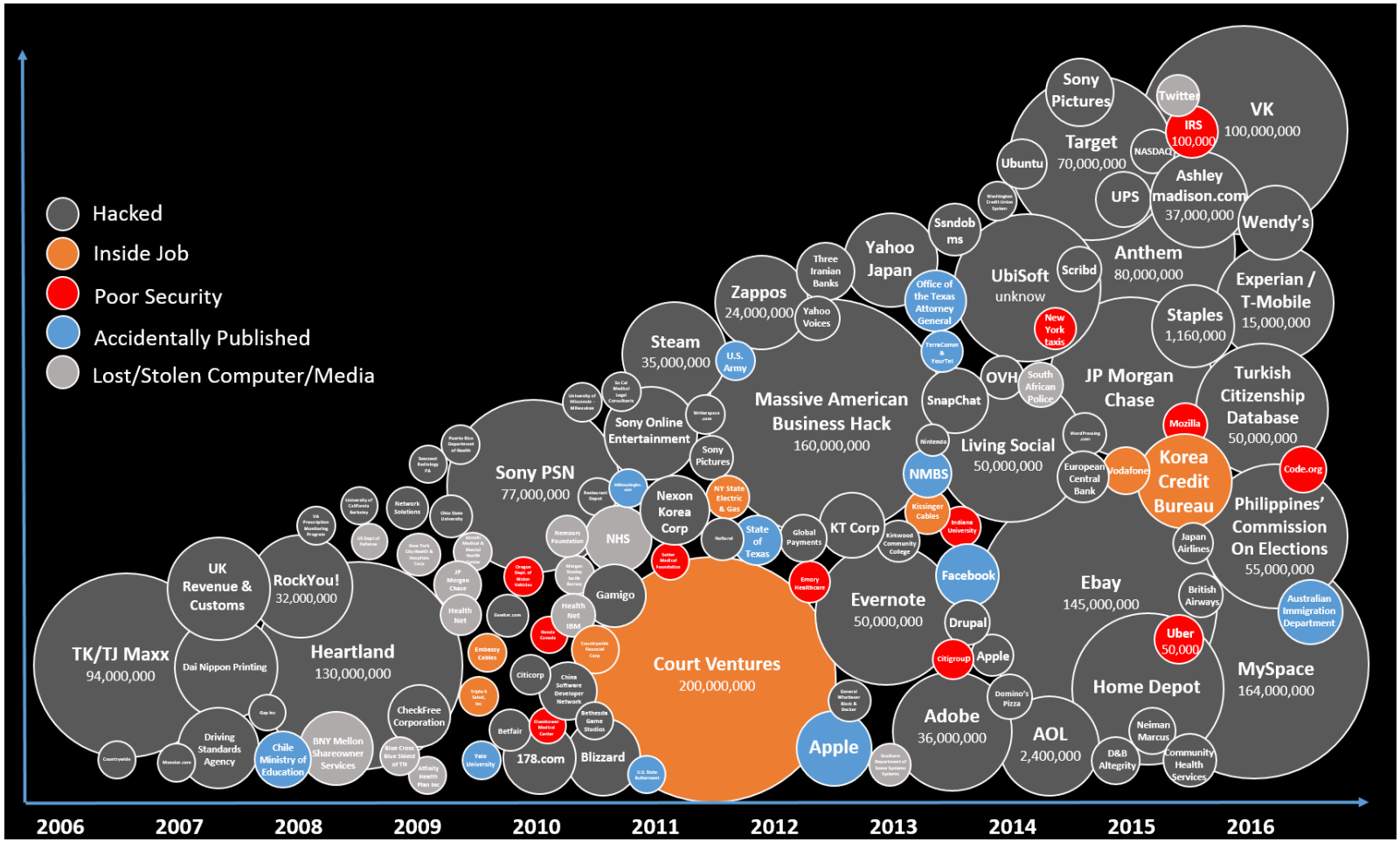
Recent Cyber Attacks

- Equifax (2017)
  - 145 million Americans' data was stolen
- WannaCry (2017)
  - Ransomware attack spanning 150 countries
  - Hackers demanded money to unlock files
- A Yahoo Bombshell
  - 3 billion accounts were stolen
  - Hacked in 2013… didn't find out until 2016!!

Recent Cyber Attacks

Every organization wants to keep their name off of this chart!

Societal Impact:



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

Advanced Persistent Threats

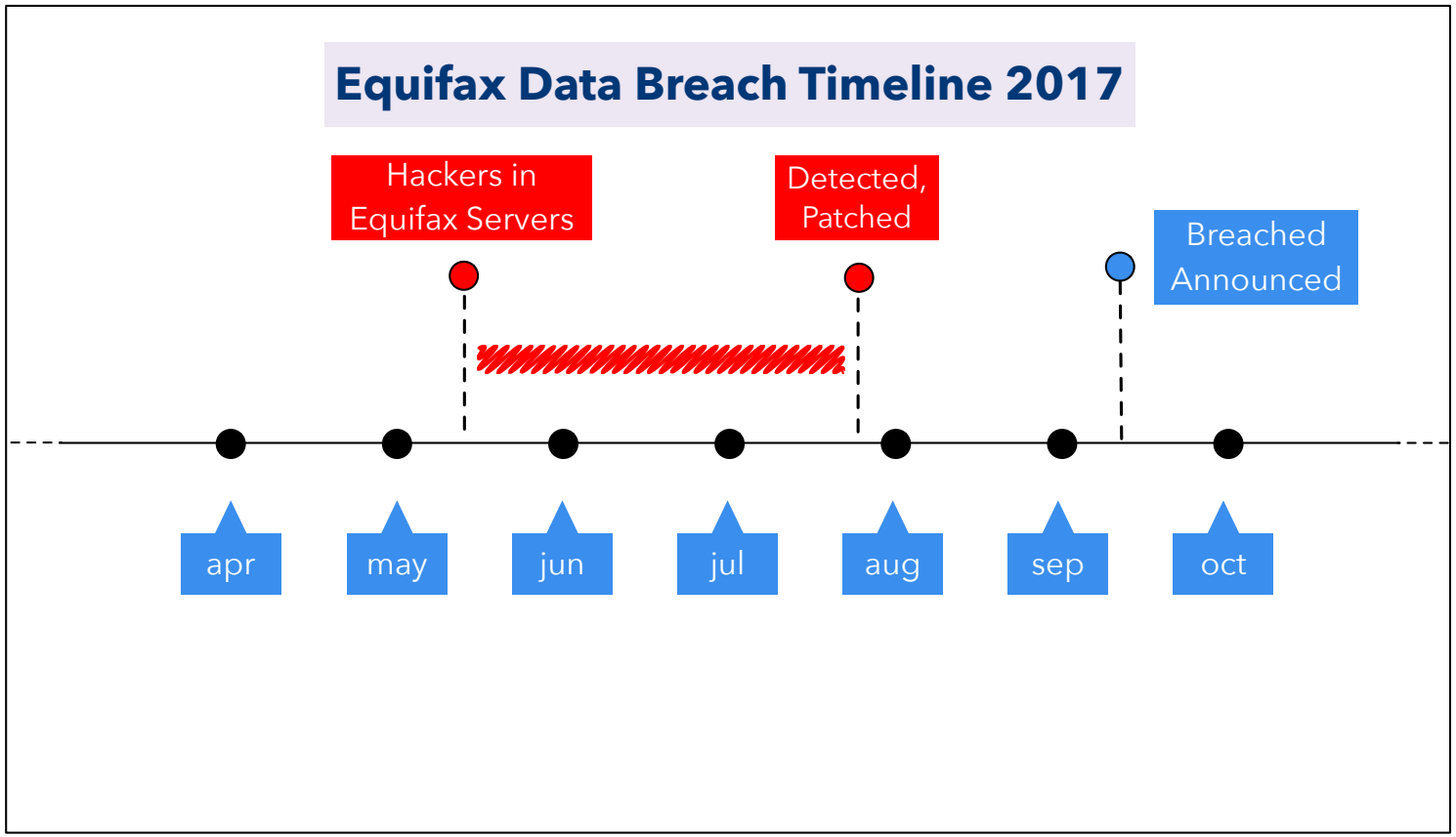
5 Stages of an APT attack:

1. Reconnaissance
   - Understand about the target using social media or company's website
2. Incursion
   - Enters into victim's system using different attack vectors (e.g. social engineering)
3. Discovery
   - The attackers stay low and operate patiently in order to avoid detection
4. Capture
   - Hackers access unprotected systems and capture data over an extended period of time
5. Exfiltration
   - Finally, captured information is sent back to the attack team's home base for analysis

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.

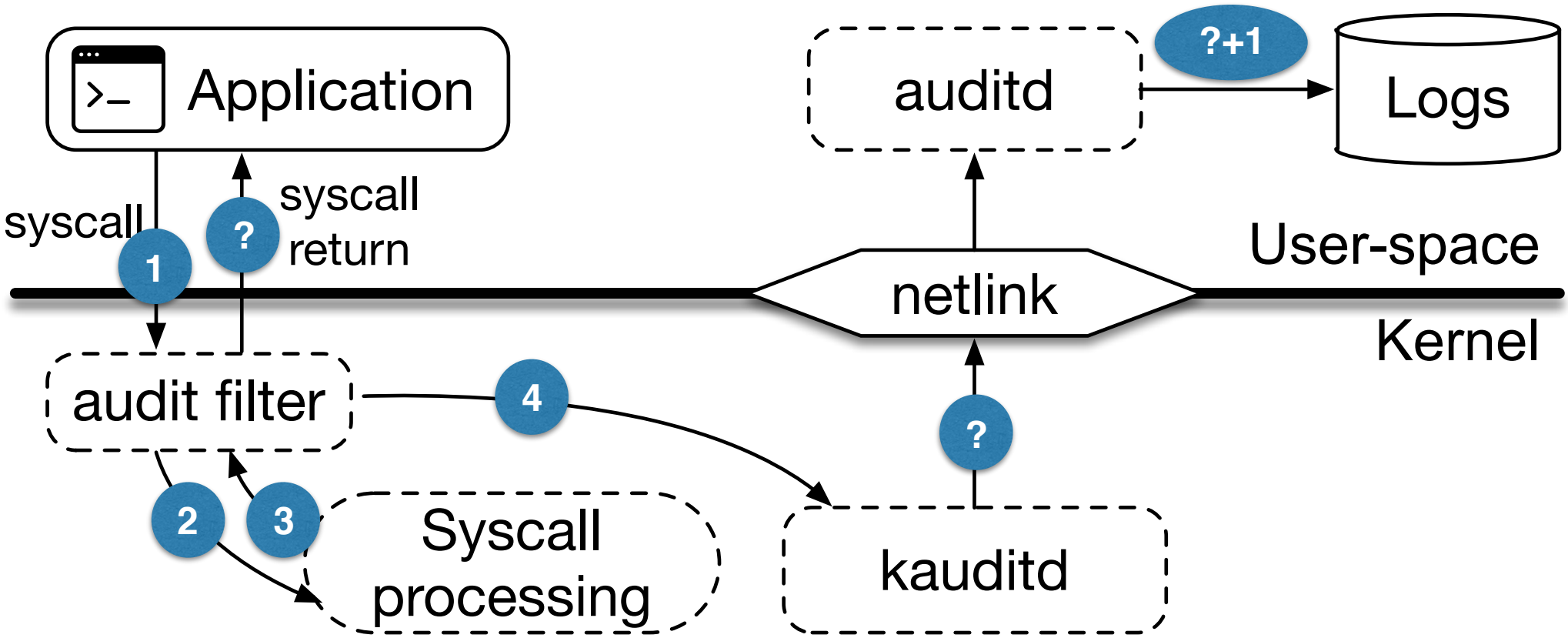
Linux Audit Use Cases

- **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

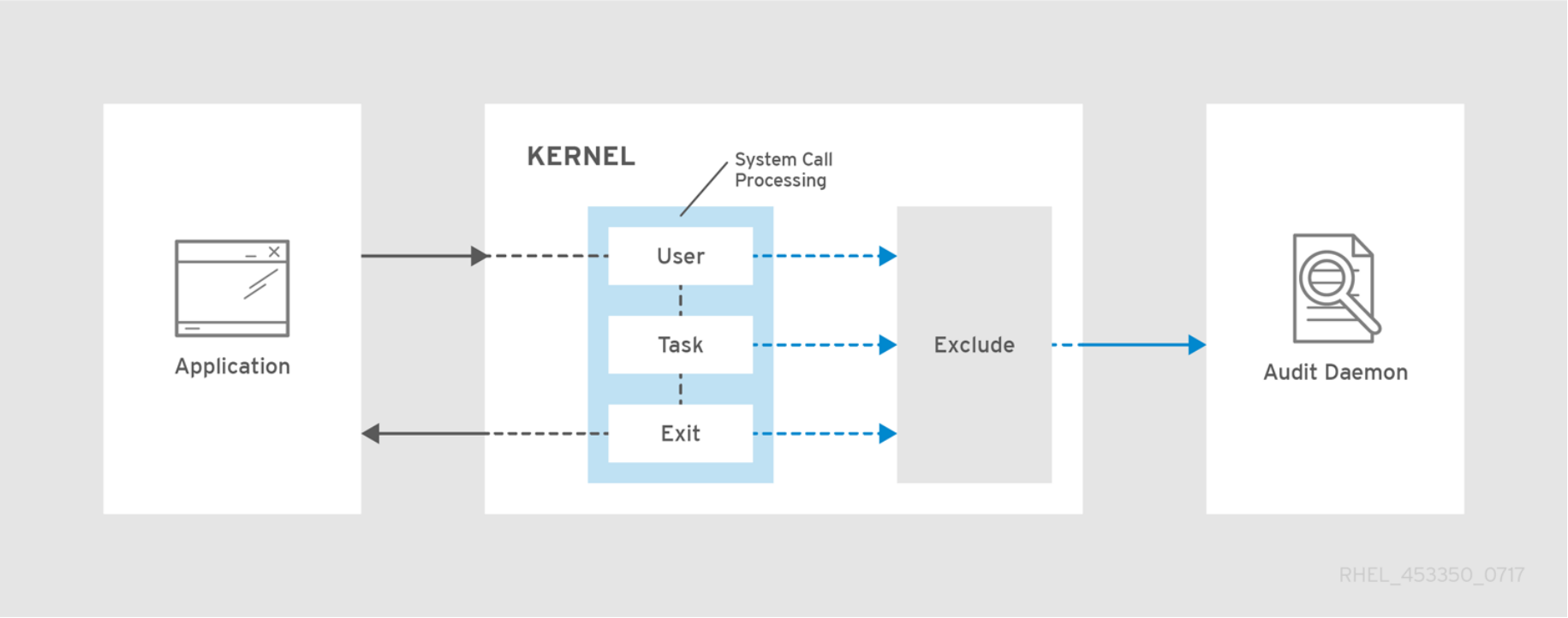
- Auditing hooks around the kernel intercept system calls and records the relevant context
  - Where are audit hooks placed relative to security hooks?
- The auditd daemon ingests kernel events via a netlink socket and writes the audit reports to disk/network.
- Various command line utilities take care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

Linux Audit Framework



Linux Audit Filtering

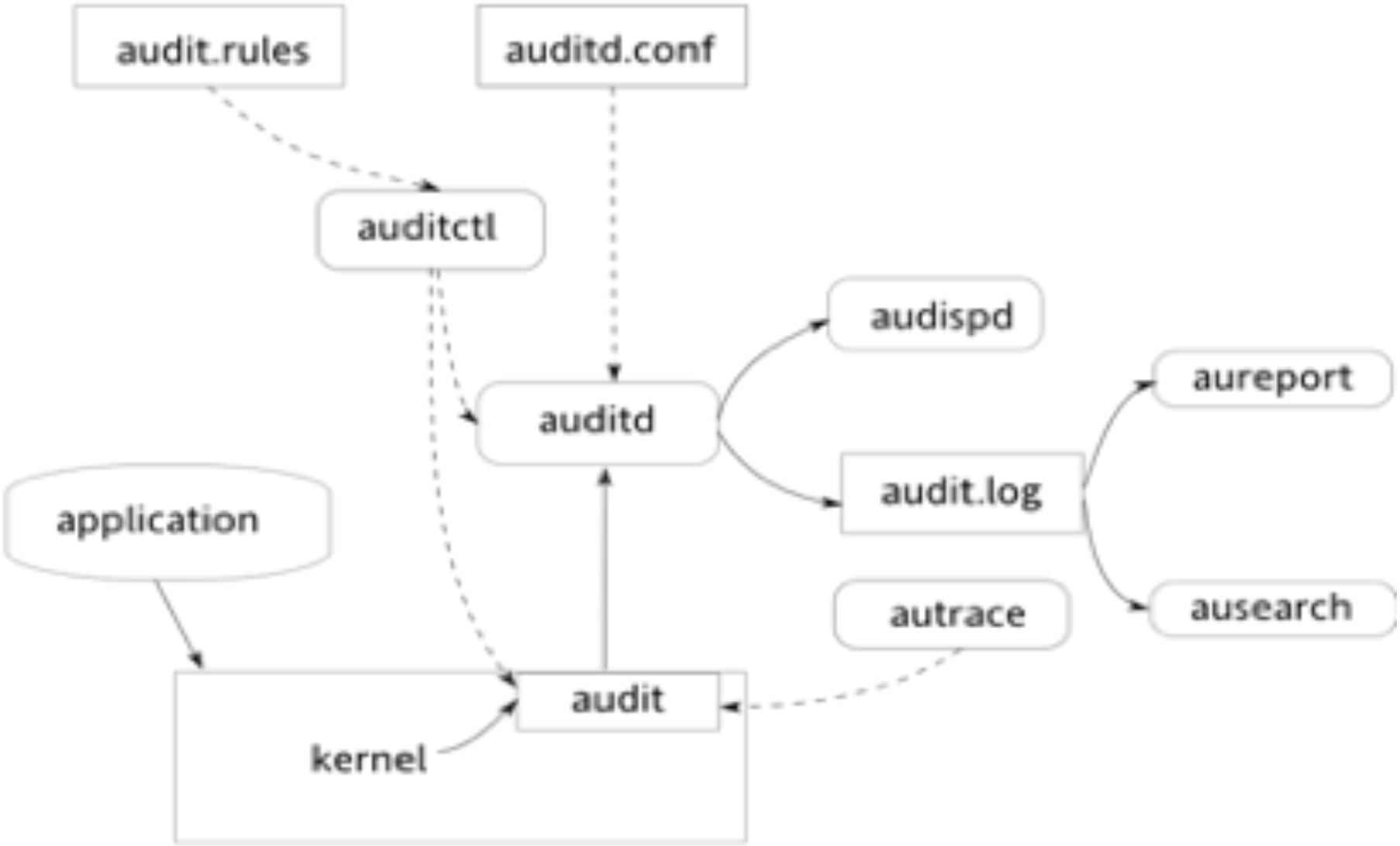
All hooks are defined, but may not be triggered based on active audit configuration….



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

Linux Audit Utilities



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

```
$  sudo aureport -s

Syscall Report
===============================
# date time syscall pid comm auid event
===============================
1. 08/03/2015 15:45:03 313 10285 modprobe -1 52501
2. 08/03/2015 15:45:03 313 10290 modprobe -1 52502
3. 08/03/2015 15:45:03 54 10296 iptables -1 52503
4. 08/03/2015 15:45:03 54 10302 iptables -1 52504
5. 08/03/2015 15:45:03 54 10305 iptables -1 52505
6. 08/03/2015 15:45:03 54 10313 iptables -1 52506
7. 08/03/2015 15:45:03 54 10325 iptables -1 52507
8. 08/03/2015 15:45:03 54 10329 iptables -1 52508
9. 08/03/2015 15:45:03 54 10343 iptables -1 52509
10. 08/03/2015 15:45:03 54 10345 iptables -1 52510
11. 08/03/2015 15:45:03 54 10349 iptables -1 52511
```

Resources

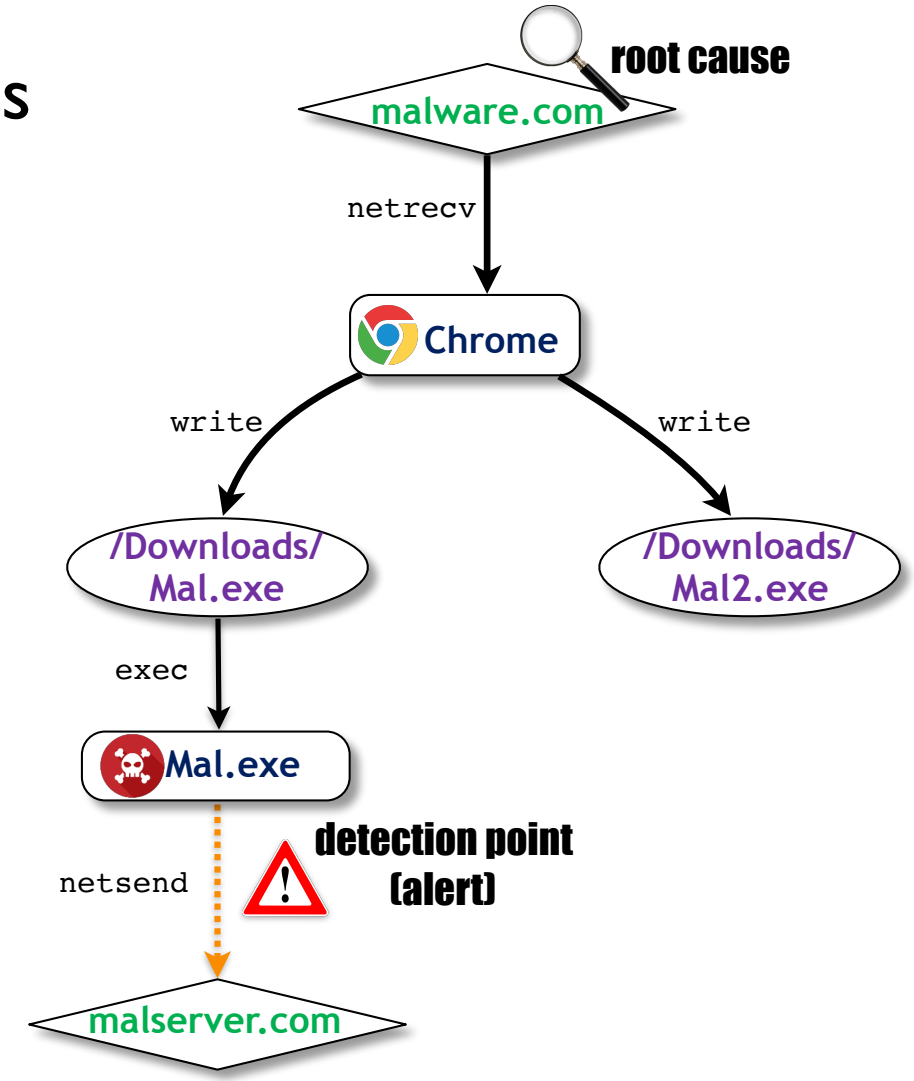
- Audit manual pages
  - There are several man pages installed along with the audit tools that provide valuable information about each utility
- Linux Audit Project:
  - http://people.redhat.com/sgrubb/audit/index.html
- The SPADE Project (for graph-based analysis)
  - https://github.com/ashish-gehani/SPADE

Example Audit Log…


```
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
```

Causal Analysis

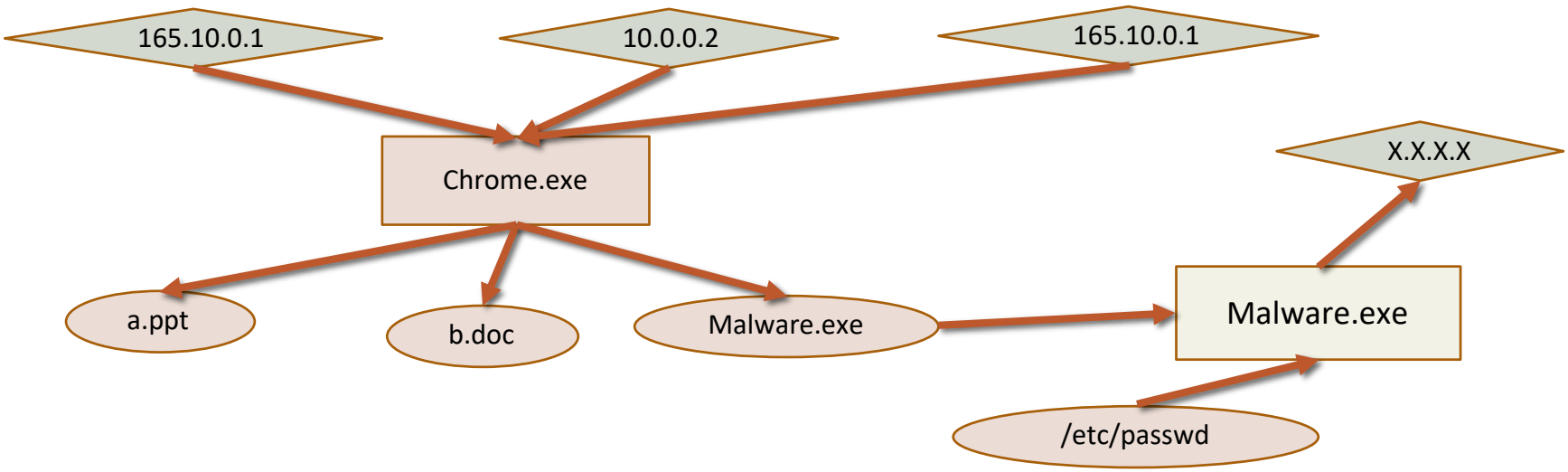
- Idea: Model related log events as a causal relationship graph.
  - Vertices: Files, Processes, etc.
  - Edges: System Accesses (e.g., `read`, `write`, `fork`)
- Backtrace queries identify root cause of a detection point
- Forwardtrace queries identify full attack footprint starting from a root cause.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



[King and Chen, SOSP'03]

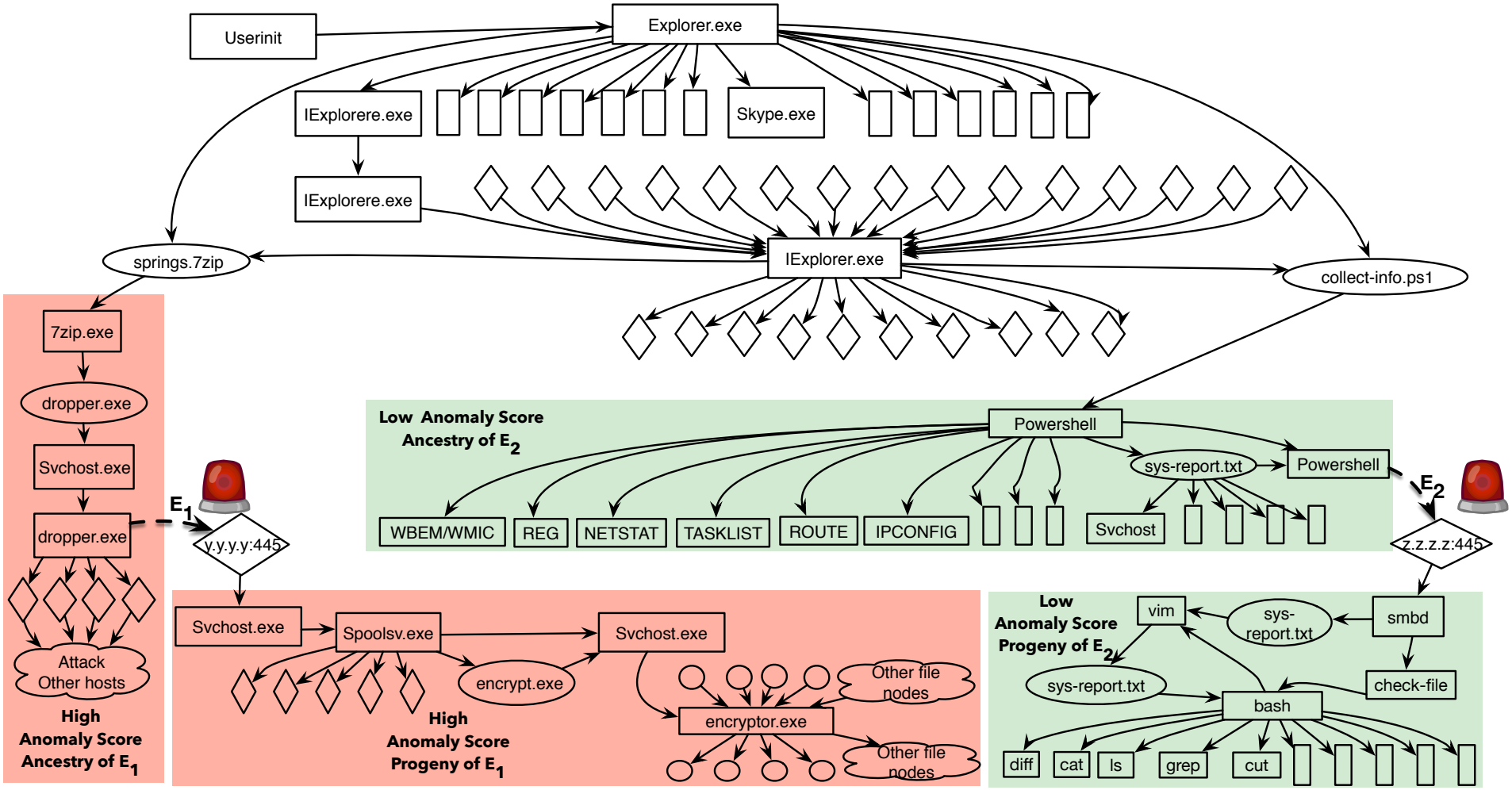
… as a Causal (Provenance) Graph

```
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
```



Investigating Security Alerts

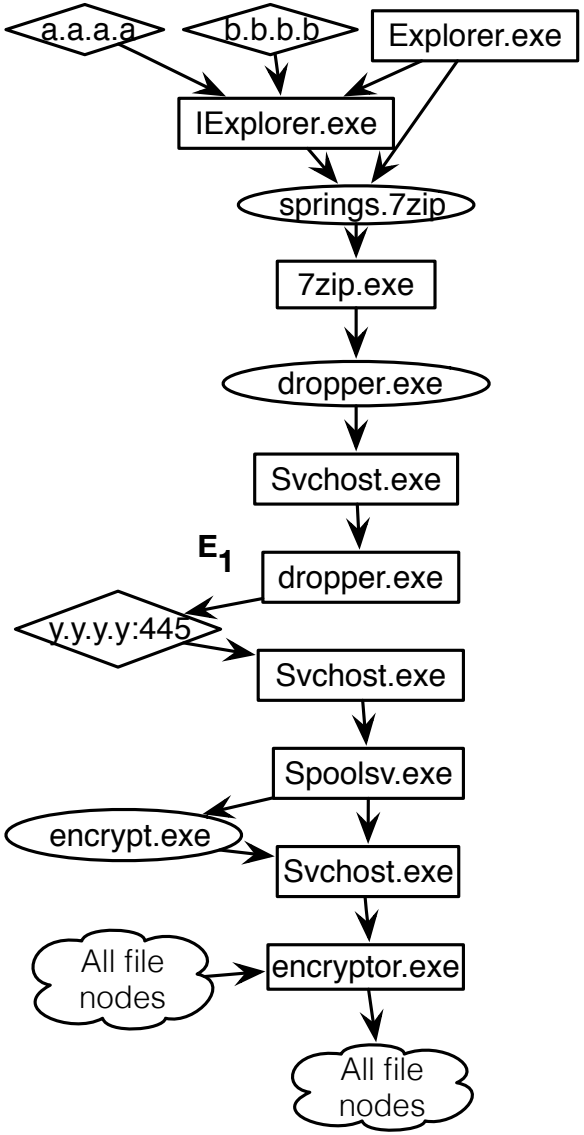
We can use provenance analysis to investigate security alerts fired by other monitoring products:



WannaCry attack scenario

Investigating Security Alerts

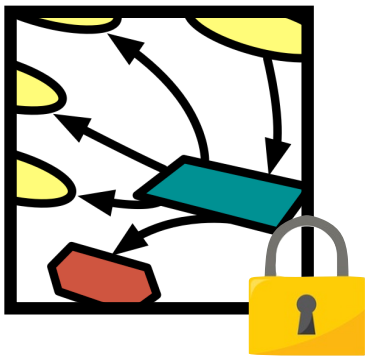
- Challenges in system auditing:
  - "Dependency Explosion"
  - Semantic Gap issues
- The literature has many tricks for extracting precise attack graphs from large system graphs.



(The same) WannaCry attack scenario

Shameless Ad for My Research

LOG INTEGRITY

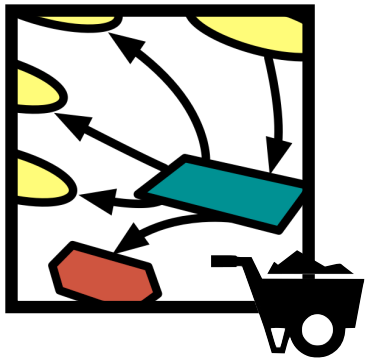


Assumption In this paper, like other existing provenance systems built on audit logging, we trust the kernel and the user-space processing daemon. [Wang et al., ACSAC'18]

RAPID7 metasploit ?

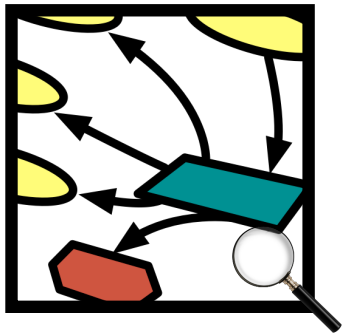
Protecting audit logs to bring our threat model more in line with real world attackers

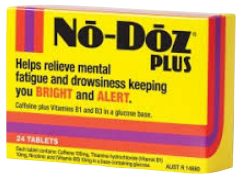
LOG FIDELITY



Enriching audit logs by incorporating application semantics into causal analysis.

LOG ANALYSIS





Extracting more insights from provenance graphs with minimal human intervention.