Overview

Gold standard:

• **Authentication**: bind principals to actions
• **Authorization**: determine whether actions are permitted
• **Audit**: record and review actions
Uses of audit

• **Individual accountability:** deter misbehavior

• **Event reconstruction:** determine what happened and how to recover

• **Problem monitoring:** real-time intelligence
Audit tasks

• **Recording:**
  – what to log
  – what not to log
  – how to log
    • locally
    • remotely
  – how to protect the log

• **Reviewing:**
  – automated analysis
  – manual exploration
WHAT TO LOG
What to log?

Example: US State Department pilot program (1980s)
• Requirements:
  – log every transaction related to protected electronic documents
  – system administrator reviews log daily to search for malicious behavior
• Experiment:
  – test system for 5 users, 10 minutes
  – audit log was a stack of paper over a foot high
  – real system would have been 1000s of users working 24/7
• Lessons learned:
  – logging and review of everything by a human is impractical
  – need to reduce information logged: log reduction
  – need automated review
States vs. events

- **States**: data, *what the system is*
  - backup, or more
  - survive power failures, crashes, attacks
  - memory, disk, network, ...
  - consistent snapshot of distributed system is hard: see CS 5412

- **Events**: actions, *how the system came to be*
  - login, access to protected resource, elevation and attenuation of privileges, ...
  - our focus
  - which events?
Recall: Security requirements

- **Functional requirement:** something system should do
  - e.g., allow people to cash checks
- **Security goal:** something system shouldn't do
  - e.g., prevent loss of revenue through bad checks
- **Security requirement:** constraint on functional requirement to achieve goal
  - e.g., check must be drawn on bank where being cashed, or person cashing must be customer at that bank and deposit in their account
Events to log

• Any event that involves a security requirement
  – Fact that requirement was checked
  – Whether it was met or not
  – The information that led to that decision

• Typically involves the gold standard...
  – whether an action was authorized, or
  – whether a principal was authenticated
Orange Book logging

For minimal C2 level certification:

• Events to log:
  – Use of identification and authentication mechanisms
  – Introduction of objects into a user's address space (e.g., file open, program initiation)
  – Deletion of objects
  – Actions taken by computer operators and system administrators and/or system security officers
Orange Book logging

For minimal C2 level certification:

• What to log:
  – Date and time of the event
  – User
  – Type of event
  – Success or failure of the event
  – For identification/authentication events: origin of request
  – For events involving objects: name of the object
What not to log

• Some information might be too sensitive for log files:
  – plaintext keys, passwords
  – the details of company’s newest shiny product
  – the GPS coordinates of undercover secret agents

• Possibilities:
  – log it anyway, protect the log
  – sanitize log
Sanitization

Protect confidential information in log

• by deleting

• by modifying
  – e.g., replace with user names with pseudonyms, keep separate protected map between names and pseudonyms
Sanitization

• **Before** writing to log:
  – Protects users from system administrators; maybe surveillance warranted only with probable cause
  – Con: have to decide in advance, as part of system design, what information to keep vs. discard

• **After** writing to log:
  – Con: confidentiality of log must be (more) protected
  – Pro: can decide afterwards what information to discard, perhaps even redact logs and send to 3rd party for analysis
EXERCISE: GRADE SYSTEM LOG
HOW TO LOG
Recall: Say what you mean

Main principle: Every message should say what it means

- Interpretation of message should depend only upon content of message
- Hence recipient can recover meaning without needing to assume or supply any context
- Writing down a straightforward English sentence describing the meaning of each step in narration is good practice
Say what you mean

Main principle: Every log entry should say what it means

- Interpretation of log entry should depend only upon content of log entry
- Hence reviewer can recover meaning without needing to assume or supply any context
- Writing down a straightforward English sentence describing the meaning of each log entry is good practice
Log file format

• Keeping log files in standard format enables...
  – Reuse of tools for log analysis
  – Correlation across logs from multiple applications

• Standard formats:
  – Common Log Format (used by web servers)
  – syslog (used by Unix)
    • originated with sendmail
    • became a de facto standard
    • then standardized by IETF: RFC 5424
    • examples: take a look in your local /var/log directory
syslog example message

Mar  6 00:48:29
chardonnay
kernel[0]:
AppleThunderboltNHIType2::prePCIWake - power up complete - took 1624 us
syslog message format

- facility: category
  - kernel, mail, security, printer, clock, ...
- severity
  - emergency, alert, critical, error, warning, notice, informational, debug
- timestamp
- hostname
- application name
- process id
  - no standard meaning; sometime co-opted by application to provide identifier that groups related messages (e.g. a transaction)
- message type
  - also no standard meaning; just a string that can be used for (e.g.) filtering
- message
  - can be structured as key-value pairs, or unstructured
syslog architecture

• **Originators**: source of messages
  – might duplicate to multiple relays

• **Relays**: forward messages
  – might filter or duplicate messages

• **Collectors**: sink of messages
  – might collect from many sources
syslog architecture
Security concerns with syslog

Base syslog protocol has no security goals

• Nothing guarantees C, I, or A
• Recommended to use SSL to protect communication channel
• Nonetheless, receivers are permitted to truncate or drop messages
• Even with SSL, end-to-end integrity of messages from originator to collector not guaranteed
  – Concerns include provenance, message integrity, replays, sequencing, detection of missing messages
  – Digital signatures can help [RFC 5848]
Log space

What happens if log size grows too large?

• **Halt** system
• **Overwrite** previous entries
• **Stop** logging
Review

• Audit is needed when prevention fails
  – By **design**: infeasible to prevent bad thing, so detect it instead
  – By **accident**: attacker breaches system despite countermeasures, so figure out afterwards what went wrong

• Analysis might be automated or manual
Automated review

• Detect suspicious behavior that looks like an attack
• Or detect violations of explicit policy
• Leads toward intrusion detection and machine learning
Automated review

Example: network monitoring

• suspicious behavior: opening connections to many hosts

• automated review: router reconfigures to isolate suspicious host on its own subnet with access only to (e.g.) virus scanner download, notifies administrators

• issues: false positives? false negatives?
Automated review

Example: open source tool tripwire
• policy: certain files shouldn't change
  – want to detect, e.g., rootkits
• state snapshot: analyzes filesystem, stores database of file hashes
• automated review: runs (e.g. daily) and reports change of hash
• issues: where to store database, how to protect its integrity, how to protect tripwire itself?
Manual review

• Enable administrators to explore logs and look for \{states, events\} that designers might not have anticipated
• Complications:
  – HCI/DB issues: visualization, query, expressivity
  – Correlation amongst multiple logs and entries in same log
• Log browsing techniques:
  – Flat text
  – Hypertext
  – DBMS
  – Graph (nodes might be entities like processes and files, edges might be associations like forking or times)
Manual review

• Two ideas that might help:
  – Temporal replay: animate what happened when
  – Slice: minimum set of log events that affect a given object
    • Idea comes from *program slice*: debugging technique that reveals program statements that led to current value of variable

• Research example:
SECURING THE LOG
Securing the log

• Good practice: limit access to log files
  – Least Privilege
  – Append-only access for most users: no read, rename, delete permission

• Limitations:
  – Once attacker compromises host, logs on that host are compromised too
  – Cryptography doesn't help: nowhere to put the keys that attacker can't access (absent a hardware solution)
  – But can protect log entries made before host is compromised
    • Offline copies: protect archived log files with encryption and MACs, physical security
    • Online copies: similar ideas...
Securing the log

• **Threat:** attacker who compromises host that stores log
• **Harm:** log can be read, modified, deleted
• **Vulnerability:** log protected only by access control mechanisms on host
• **Countermeasure:** cryptography: iterated hashing: \( H(H(H(\ldots H(v)\ldots)))\)
Securing the log

• **System:**
  – machine M maintains a local log
  – periodically M synchs log to trusted remote log server S
  – might be very long periods between synch: if short periods are possible, no real need for this protocol

• **Threat:** attacker might completely compromise M, but not S

• **Goals:** assume attacker compromises M at time t...
  – Contents of log messages entered before t are not disclosed to anyone who can read log at M (Confidentiality)
  – Contents of log messages and their sequence before time t cannot be changed in a way that is undetectable by S (Integrity)
Securing the log

• **Weaknesses (non-goals):** after time $t$...
  
  – Attacker can read and modify new log messages (Confidentiality+Integrity)
  
  – Attacker can truncate from log any messages not yet synched (maybe even from before $t$) to $S$ (Availability)

• **Assumption:** $M$ and $S$ share a secret authentication key $ak$
Protocol

M, to record message m in log:
1. $ek = H("encrypt", ak)$
2. $x = AuthEnc(m; ek; ak)$
3. record x in log
4. $ak = H("iterate", ak)$

Simplified from [Schneier and Kelsey 1999]
Protocol analysis

M, to record message m in log:
1. \( \text{ek} = H("encrypt", \text{ak}) \)
2. \( \text{x} = \text{AuthEnc}(m; \text{ek}; \text{ak}) \)
3. record x in log
4. \( \text{ak} = H("iterate", \text{ak}) \)

If M is compromised...
- current value of \( \text{ak} \) revealed
- previous values not recoverable because hash function is one way
- so old \( \text{ek} \)'s cannot be recovered, hence confidentiality of old entries preserved
- and old \( \text{ak} \)'s cannot be recovered, so any changes to past log can be detected by S when log next synched
- but from now on attacker could fabricate new messages, read new messages, etc.
Upcoming events

• [Wed] A3 due

It only takes one audit to ruin your day.
– Kathy Burlison