CS 5430

Review

Prof. Clarkson
Spring 2016

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(...H(v)...)))

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); but still can't undetectably add after that truncation
- **Assumption:** M and S share a secret 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. ek = H("encrypt", ak)
2. x = AuthEnc(m; ek; ak)
3. record x in log
4. ak = H("iterate", ak)
```

If M is compromised...

- current value of ak revealed
- previous values not recoverable because hash function is one way

Protocol analysis

- So old ek's cannot be recovered, hence confidentiality of old entries preserved
  - note: M can't read its own log, but that's okay because S is who really wants the log
- And old 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.
- If an entry was made that would reveal attack and that entry is not yet synched to S, attacker has two choices:
  - truncate log and stop sending anything further to S
  - add new log messages that attempt to compensate for the attack
  - note: attacker cannot selectively remove the incriminating entry

EXERCISE: TAMPERPROOF LOGS

REVIEW

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

Manual review

- Enable administrators to explore logs and look for {states,events}
- Log browsing techniques:
  - Flat text [example: last time's syslog]
  - Hypertext [example]
  - DBMS [example: queries in CMS]
  - Graph (nodes might be entities like processes and files, edges might be associations like forking or times) [example]
- Issues:
  - Designers might not have anticipated the right {states,events} to record
  - Visualization, query, expressivity (HCI/DB issues)
  - Correlation amongst multiple logs

Manual review

- Two ideas that might help:
  - Temporal replay: animate what happened when [example]
  - 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

Automated review and response

- **Review:** detect suspicious behavior that looks like an attack, or detect violations of explicit policy
  - Classically used AI techniques like training neural nets, expert systems, etc.
  - Modern research in application of machine learning
- **Response:** report, take action
  - Leads toward *intrusion detection*

Example: tripwire

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 response:** runs (e.g. daily) and reports change of hash
- **Issues:** where to store database, how to protect its integrity, how to protect tripwire itself?

Example: Network monitoring

- **Suspicious behavior:** opening connections to many hosts
- **Automated response:** 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?

INTRUSION DETECTION

Intrusion handling

[Northcutt 1998]

1. **Preparation:** establish procedures and mechanisms in advance
2. **Identification:** detect attack
3. **Containment:** limit ongoing damage
4. **Eradication:** stop attack and block similar attacks
5. **Recovery:** restore system to good state
6. **Follow up:** take action against attacker, identify problems, record lessons learned

Intrusion detection

Intrusion detection system (IDS):

- device for automated review and response
- responds in (nearly) real time
- components:
  - sensors
  - analysis engine
  - countermeasure deployment
  - audit log
- methodology:
  - signature based: recognize known attacks
  - specification based: recognize bad behavior
  - anomaly based: recognize abnormal behavior

Signature-based detection

- Characterize known attacks with signatures
  - e.g., 100 TCP SYN packets received on different ports of same host within 1 second (maybe a portscan)
  - e.g., creating a file while effective user is administrator but actual user is not, then transferring ownership of file to actual user (maybe indication that user managed to improperly escalate privileges)
  - e.g., an email with the subject "Free pictures!" and an attachment "freepics.exe" (maybe contains a virus)
- If behavior ever matches signature, declare an intrusion
- Issues:
  - works only for known attacks
  - signature needs to be robust w.r.t. small changes in attack

Example: Network Flight Recorder (NFR)

[Ranum et al. 1997]

- Three components:
  - *Packet sucker* captures network traffic
  - *Decision engine* uses custom-written filters to extract information from packets
  - *Backend* writes information to disk; packets are discarded
- Queries can be performed over stored information while rest of system continues to process packets
- Backends can trigger alerts to system administrators
- Filters written in domain specific language; provide extensibility
- Similar ideas used in Bro [Paxson 1999], available still as open source IDS

Specification-based detection

- Characterize good behavior of program with a specification
  - e.g., the programs that may be loaded on a given host by a given user
  - e.g., the sequence of system calls that a given program is allowed to make
- If behavior ever departs from specification, declare an intrusion
- Issues:
  - effort to create specifications
  - any program is a potential vulnerability if executed by a privileged user

Example: Distributed Program Execution Monitor (DPEM)

[Ko et al. 1997]

- Generates traces of program execution from log files produced by BSM (Solaris Basic Security Mode auditing)
- Determines whether traces are accepted by grammar that describes good behavior
- Designed for real-time monitoring: able to report violations in hundredths of seconds

Anomaly-based detection

- Characterize normal behavior of system
  - e.g., maximum number of times user will mistype password is 3
  - e.g., number of processes a user launches during daytime and nighttime is between certain bounds which are dynamically adjusted based on past behavior
- If behavior ever departs far enough from normal, declare an intrusion
- Issues:
  - feature identification
  - obtaining data on normal behavior

Example: Haystack

[Smaha 1988]

- Monitors value of some statistic of interest over a time period: a0, a1, a2, ..., an
- Determine lower and upper bounds tL and tU such that 90% of ai values lie between tL and tU
- If next value is outside tL and tU, raise an alarm
- Adaptive: as value of changes over time, detector itself adjusts

Errors

- False positive: raise an alarm for a non-attack
  - makes administrators less confident in warnings
  - perhaps leading to actual attacks being dismissed
- False negative: not raise an alarm for an attack
  - the attackers get in undetected!
- Tradeoff between the two needs to be tunable; difficult to achieve the right classification statistics

Deployment

- So far we've been thinking of host-based deployment
- Network-based IDS:
  - typically a separate machine
  - stealth mode:
    - one NIC faces the network being monitored, no packets ever sent out on it, no packets can be routed specifically to it
    - another NIC faces a separate network through which alarms are sent
- Honeypot:
  - dedicated machines(s) or networks
  - purpose is to look attractive to attacker
  - but actually just a trap: monitored to detect and surveil attacker

Automated response

- **Monitor:** collect (additional) data
  - record additional traffic, system calls, etc.
- **Protect:** reduce exposure of system
  - shut off network connection, make file systems read only or take them offline, etc.
  - degrade response time, e.g. by making system calls take artificially, exponentially longer
  - jail attacker by redirecting to a confined area in which behavior can be controlled and manipulated
- **Alert:** call a human

Counterattack

- **Legal:** file criminal complaint
  - evidence and chain of evidence is important
- **Technical:** damage attacker to stop attack or prevent future attacks
  - Might harm an innocent party
    - what if your counterattack causes you to take out someone the attacker was just spoofing?
    - what if your counterattack degrades the network for everyone?
  - Might expose you to legal liability

Upcoming events

- [today] A3 due
- [next week] A4 out

You are secure from intrusion, secure from yourself; and your hard, restricting shell of individuality is at once dissolved as…you gaze into the vistas of a sunset. – John Muir