CS 392/CS 681 - Computer Security

Module 17 - Auditing
**Auditing**

- **Audit**
  - “Independent review and examination of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and operational procedures, and to recommend necessary changes in controls, policies, or procedures.” NSTISSI

- **Audit Trail**
  - “Chronological record of system activities to enable the reconstruction and examination of the sequence of events and/or changes in an event. Audit trail may apply to information in an IS, to message routing in a communications system, or to the transfer of COMSEC material.” NSTISSI
Auditing

- Logging
  - “Logging is the recording of events or statistics to provide information about system use and performance” – text book

- Auditing
  - “Auditing is the analysis of the log records to present information about the system in a clear and understandable manner” – text book
Auditing mechanisms

- Uses of auditing mechanisms
  - Logs provide audit trails to events leading to system compromises
  - Helps determine usage patterns of a system
  - Recorded trail of privileged events can thwart privilege misuse and can complement access control

- Two important questions to address
  1. What information to log?
  2. What information to audit?
Anatomy of an auditing system

- System events are generated by applications that use the logging API.
Anatomy of an auditing system

- **Logger:**
  - Record information to a data storage
  - Could be in binary or text format
  - Quality and quantity is usually decided by the application and system configuration

- **Analyzer:**
  - Analyzes the data logged by the logger
  - Either detects some event or problem or provide feedback on system usage

- **Notifier:**
  - Input from the analyzer sent to notifier and it alerts an analyst or system administrator
Designing an Audit System

- Implementation Consideration
  - Initial state of the system must be proofed secure
  - How much information and what information to log
  - At which level should we log

- Syntactic Issues
  - How to express log entries
  - Log entries should be human and machine readable e.g. grammar based entries, XML
  - Avoid ambiguous entries
Designing an Audit System

- **Log Sanitation**
  - Confidentiality policies and privacy laws mandate sanitation of log information
  - Sanitizers can be cryptographic functions, too
  - Sanitizers should maintain integrity of log data
  - Sanitation may be done before after logging
    - If done before information will not leave system boundaries
    - If done after information will not leave policy boundaries

- **Anonymizing sanitizers**
  - Deleted information such that both originator and recipient cannot reconstruct the deleted information

- **Pseudonymizing sanitizers**
  - Deletes information such that the originator can reconstruct the deleted information
Designing an Audit System

- Application and System Logging
  - An audit mechanism should provide applications a set of APIs, formats to perform logging
  - Also, it should provide isolated APIs for the operating system to perform its logging
  - The difference between application and system logs is the level of abstraction they achieve (e.g.: many system events lead to an application event)

- Correlation Problem:
  - Relating system and applications logs. Given a system log and a corresponding application log how can you relate log entries in them.
Goals of audits

- Two important security goals
  - Identify any violations of a policy: Audit mechanism attempt to log violations in policy but these violations may not be known.
  - Identify particular attempt to breach security: Audit mechanisms attempt to log specific threats to security/policy of a system. Violations are well known.
Auditing procedures

- Auditing to detect violations of a known policy checks to see if a state of the system violates the policy by which it is bound
  - State-based auditing:
    - State-based auditing mechanisms record a system’s state and determine whether a state is secure and complies with the policy
    - Feasible on most distributed systems but infeasible on non-distributed systems
  - Consistent static analysis:
    if a state is obtained while the system is quiescent then we call the state analysis consistent static analysis
  - Inconsistent static analysis:
    if a state is obtained while the system is not quiescent then we call the state analysis inconsistent because the state is a set of different states and not one particular state!
    E.g: file system auditing with TripWire
Auditing procedures

- Transition-based auditing
  - Transition-based systems record actions of a system and given a system’s state determine whether perform this action violates system security
  - Cannot function alone, requires state information
  - Cannot detect system violations if the system begins in a violated state!
  - E.g: network auditing with TCPWrapper

- Hybrid auditing mechanisms
  - Combine both state-based and transition-based auditing
  - E.g: Preventing multiple logins
Auditing procedures

- Auditing to detect known violations of a policy
  - Used to detect known attacks on a host
  - Mostly static rule-based methods
  - E.g: denial of service attacks, Land Attacks etc.
Auditing Mechanisms

- Secure Systems
  - Audit mechanism is integrated with the system
  - Provides added security, integrity to the logs
  - Highly configurable
  - E.g: Secure Syslog, VAX Accounting & Audit

- Non-Secure Systems
  - Designed with accounting in mind
  - May not be part of the system, but an overlay
  - Weak or no cryptographic support
  - E.g: syslog, BSM in SunOS
Unix log files

Usually in /var/log or /var/adm. Below is a sample ...

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lastlog</td>
<td>Records each user’s last successful login. Displayed at login. Overwritten after every login!</td>
</tr>
<tr>
<td>utmp(x)</td>
<td>Records each user currently logged in</td>
</tr>
<tr>
<td>wtmp(x)</td>
<td>Permanent record of user login and logout times. Also shutdowns and startups. Can get very large! /who, finger, w etc. commands obtain data from utmp and/or wtmp. last command displays in human readable form.</td>
</tr>
</tbody>
</table>
| acct     | Records commands run by every user. Command lastcomm displays in human readable form. Can get HUGE!! Processing needed to summarize and save in savacct. UNIX does not log arguments to commands. Hence vi could actually be cc!!!
More log files…

<table>
<thead>
<tr>
<th>xferlog</th>
<th>Logs FTP access</th>
</tr>
</thead>
<tbody>
<tr>
<td>sulog</td>
<td>Logs use of su command</td>
</tr>
<tr>
<td>loginlog</td>
<td>Records bad login attempts</td>
</tr>
<tr>
<td>messages</td>
<td>Messages printed on system console</td>
</tr>
<tr>
<td>Application specific logs</td>
<td>xferlog (WS FTP), access_log (HTTPD) etc. etc.</td>
</tr>
</tbody>
</table>

- Note: Log files can be modified by a wily hacker. Still traces can remain. Also, beware false log entries!!
  - Need mechanism for authenticating them in UNIX.
  - Back up your logs!!
Where and what to log?

- We mentioned that log files are not secured. So where and how to log?
  - To a printer
  - Logging across the network
  - Logging everything everywhere!!

- What to log?
  - Everything or as much as you can.

- What not to log?
  - Anything that may violate privacy. For example, correct passwords, incorrect passwords and also incorrect usernames!!
Syslog facility

- General purpose highly configurable logging facility
- Accepts messages from facilities. Messages have level of importance and syslog.conf file tells syslog how to report message.
- Facilities – mail, lpr, kernel, auth, etc.
- Levels – emerg, err, warning, info, etc.
- How – log file, another host, printer, etc.
  - kern.* /dev/console
  - *.emerg *
  - uucp,news.crit /var/log/spooler
- None of the log files in Unix systems are secure, including those generated by syslog. Secure syslog is designed to be a secured logging mechanism
Swatch

- Monitoring log files can get tedious!
- Swatch – allows you to automatically scan log files for particular entries and take appropriate action like send email.

For example
- `/routed.*bind/ echo 24:00:00 0:16`
  Would cause the following by swatch
  *The following was seen 20 times in the last 24 hrs *
  Routed[9293]:bind: Bad file number.

- Swatch can be used for real time monitoring as well.
  - We can setup swatch to beep a certain number when say, kernel logs a ERROR!
  - This facility is very useful with firewalls and intrusion detection systems
Additional Audit Utilities

- Secure Syslog
- COPS
  - File, directory and device permissions
  - Poor passwords
  - Content, format of password files
  - Contents of /etc/rc* and cron(2), at(2)
  - Integrity of important binaries, text files
- Chkrootkit
  - Checks to see if there are any rootkits installed
Additional Audit Utilities

- Tripwire
  - File adds, deletes, modifications
  - File permissions and properties-ignore, record and check
  - Inode number, number of links
  - User id of owner, group id of owner
  - File type, file size
  - Device number of the disk on which the inode associated with the file is stored
  - Device number of the device to which the inode points. Valid only for device objects.
  - Number of blocks allocated
  - Modification timestamp
  - Inode creation/modification timestamp
  - Growing files-indicates that the file is expected to grow.
  - Shrinking files
  - Access timestamp
Windows Audit

- Windows NT has at least three different logs
  - System event log
    logs system failures, core dumps and other OS related events
  - Application event log
    Applications can add logs using a log API
  - Security event log
    records security related events, such as logins, privilege changes etc.
- Event viewer interprets log formats
Windows Audit

- Highly configurable audit mechanism through Active Directory
- Has nine different categories
  - Audit Account Management
  - Audit logon events
  - Audit object access
  - Audit policy changes
  - Audit privilege use
  - Audit process tracking
  - Audit system events
  - Audit directory services access
  - Audit account logon events
Audit Browsing

1. Text display of events. Auditor may browse through events after post processing.
2. Hypertext display of events. Associated events are interlinked with each other. Most Unix audit tools use hypertext format.
3. Entries are logged to relational databases. Auditor performs queries and/or views of the data.
4. Replay: presents the events in temporal order and highlights temporal relationships.
5. Graphing or visual representation of events (WebTrends NT Audit).
6. Slicing: to obtain a minimum set of events that affect a particular object.
Further reading