

Overview

- As we have seen, assurance is about assigning appropriate levels of trust and not about guaranteeing security
- Assurance therefore does **not** prove that a system cannot enter an insecure state
- It follows that we require **security auditing** to detect when an insecure state is reached and to allow the deduction of how the system reached that state
- Material has been drawn from:
 - *Introduction to Computer Security* by Bishop
 - *Computer Security: Principles and Practice* by Stallings

Logging vs. Auditing

What's the difference between logging and auditing?

- **Logging:** Recording events or statistics to provide information about system use and performance
- **Auditing:** The analysis of log records to present information about the system in a clear and understandable manner
- **Security auditing:** Auditing to detect after-the-fact (a posteriori) violations of security policy

Why Audit?

- **The ultimate goal of auditing is to deter attacks**
- However, security auditing is only effective as a deterrent if the **organisational policy** is such that detected violations of security policy result in:
 1. Identification of the attacker
 2. Subsequent punitive action

What Do We Log?

- A security policy partitions the system into secure and insecure states and defines how a secure state can be differentiated from an insecure one
- Thus, viewing the security policy as a set of constraints dictates that we should log **at least** enough information to detect violation of those constraints
- A constraint is an action plus a condition to be satisfied, e.g. *constraint p_i : action \Rightarrow condition*
- Auditing is about analysis, logging is about gathering the information to perform the analysis

Bell-La Padula Constraints

- In Bell-La Padula we have the **simple security property**, no read-up, which we can translate into the constraint:
 - $S_i \text{ reads } O_j \Rightarrow f_c(S_i) \succeq f_o(O_j)$
- We also have the ***-property**, no write-down, which we can translate into the constraints:
 - $S_i \text{ appends } O_j \Rightarrow f_c(S_i) \preceq f_o(O_j)$
 - $S_i \text{ writes } O_j \Rightarrow f_c(S_i) = f_o(O_j)$

Bell-La Padula Logging

- To check for violations of the Bell-La Padula security policy we must log:
 - The action: read, write, append
 - The security level of the subject: $f_c(S_i)$
 - The security level of the object: $f_o(O_j)$
 - The result: success or failure
- Although not necessary to detect the policy violations, in practice the **identities** of subject and object are also logged in order to identify the attacker and the object of the attack

Audit System Structure and Types

At the highest level an audit system consists of 3 components:

1. A **logger** records information (binary or human readable)
2. An **analyser** checks logged information for policy violations
3. A **notifier** reports the results of the analysis

And 3 types of auditing can be distinguished:

1. **System** level auditing
2. **User** level auditing
3. **Application** level auditing

Audit System Architecture

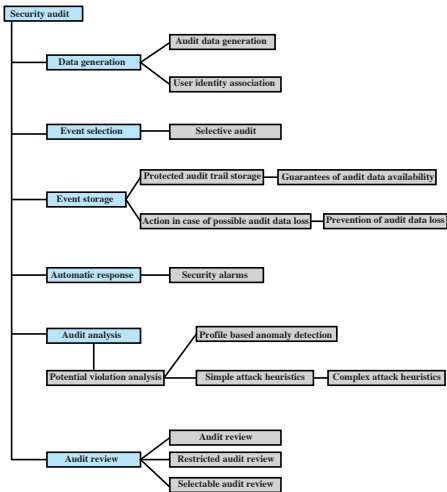


Figure 15.3 Common Criteria Security Audit Class Decomposition

System Level Auditing

- System level logs are generally used to monitor and optimise system performance but can also serve a security audit function
- The operating system or services running on its behalf is responsible for logging system wide events
- Events to be recorded include devices accessed, services activated, machine reboots. . .
- When these events are associated with corresponding subjects then we may have some overlap with user and application level auditing

User Level Auditing

- User level logs can be used to trace the activity of individual users over time and thus can be used to hold users to account for their actions
- Such logs can be usefully fed into analysers that attempt to discern normal from anomalous behaviour
- Events to be recorded include commands issued, files and resources accessed and applications launched

Application Level Auditing

- Application level logs may be used to detect security violations within an application or to detect flaws in the application's interaction with the system
- For an email application the application may log the sender, receiver and any attachments
- For a web application the application may log individual SQL queries executed against a database on behalf of the user

System, User and Application Level Auditing

```

Jan 27 17:14:04 host1 login: ROOT LOGIN console
Jan 27 17:15:04 host1 shutdown: reboot by root
Jan 27 17:18:38 host1 login: ROOT LOGIN console
Jan 27 17:19:37 host1 reboot: rebooted by root
Jan 28 09:46:53 host1 su: 'su root' succeeded for user1 on /dev/tty0
Jan 28 09:47:35 host1 shutdown: reboot by user1
Jan 28 09:53:24 host1 su: 'su root' succeeded for user1 on /dev/tty1
Feb 12 08:53:22 host1 su: 'su root' succeeded for user1 on /dev/tty1
Feb 17 08:57:50 host1 date: set by user1
Feb 17 13:22:52 host1 su: 'su root' succeeded for user1 on /dev/tty0

```

(a) Sample system log file showing authentication messages

```

Apr 9 11:20:22 host1 AA06370: from=<user2@host2>, size=3355, class=0
Apr 9 11:20:23 host1 AA06370: to=<user1@host1>, delay=00:00:02, stat=Sent
Apr 9 11:59:51 host1 AA06436: from=<user4@host3>, size=1424, class=0
Apr 9 11:59:52 host1 AA06436: to=<user1@host1>, delay=00:00:02, stat=Sent
Apr 9 12:43:52 host1 AA06441: from=<user2@host2>, size=2077, class=0
Apr 9 12:43:53 host1 AA06441: to=<user1@host1>, delay=00:00:01, stat=Sent

```

(b) Application-level audit record for a mail delivery system

```

rcp      user1  tty0    0.02 secs Fri Apr 8 16:02
ls       user1  tty0    0.14 secs Fri Apr 8 16:01
clear   user1  tty0    0.05 secs Fri Apr 8 16:01
rpcinfo user1  tty0    0.20 secs Fri Apr 8 16:01
nroff   user2  tty2    0.75 secs Fri Apr 8 16:00
sh       user2  tty2    0.02 secs Fri Apr 8 16:00
mv       user2  tty2    0.02 secs Fri Apr 8 16:00
sh       user2  tty2    0.03 secs Fri Apr 8 16:00
col      user2  tty2    0.09 secs Fri Apr 8 16:00
man      user2  tty2    0.14 secs Fri Apr 8 15:57

```

(c) User log showing a chronological list of commands executed by users

Figure 15.4 Examples of Audit Trails

Logging

- Obviously, at the core of an auditing facility is the initial capture of the data
- This requires that software includes hooks or capture points that trigger the collection and storage of data as preselected events occur
- A trade-off is usually required since the more detailed the logging granularity the bigger the performance hit incurred in checking for activated hooks
- We look at the Windows and Unix facilities available for:
 - System and user level logging
 - Application level logging

Windows Event Logging

- An event in the Windows Event Log is an entity that describes some interesting occurrence in the computer system and each takes the form of a numeric identification code, a set of attributes, and optional user-supplied data
- The data is presented to event “consumers” in XML format
- Windows has three types of event logs:
 - **System event log:** Used by applications running as system services it records system wide events
 - **Application event log:** Events that user-level applications wish to see recorded
 - **Security event log:** For a configurable set of security critical events e.g. logins, sensitive file accesses etc.

Windows Event Log

```
Event Type:      Success Audit
Event Source:    Security
Event Category:  (1)
Event ID:        517
Date:            3/6/2006
Time:            2:56:40 PM
User:            NT AUTHORITY\SYSTEM
Computer:        KENT
Description:     The audit log was cleared
Primary User Name:  SYSTEM          Primary Domain:  NT AUTHORITY
Primary Logon ID:  (0x0,0x3F7)         Client User Name: userk
Client Domain:     KENT           Client Logon ID: (0x0,0x28BFD)
```

Figure 15.5 Windows System Log Entry Example

Windows Auditing Categories

Auditing can be enabled in each of the following categories:

- Account login events
- Account management
- Login events
- Object access
- Policy changes
- Privilege use
- Process tracking
- System events

Unix Event Logging

- The `syslog` (system log) framework provides Unix's general purpose logging mechanism
- The `logger` command line utility adds events to the system log
- The `syslog` library function provides to application programs a means to add events to the system log
- `/etc/syslog.conf` is the configuration file controlling what is to be logged and where it is to be logged (mailer log file, firewall log file etc.)
- `syslogd` is the system daemon that receives events from `syslog` and `logger` and directs them to the appropriate destination according to `syslog.conf`

Unix Event Logging

- Events are logged by **facility** indicating the type of program logging the event and by **level** indicating the importance of the message
- Facilities include **kernel**, **mail**, **authorisation** and levels include **debug**, **information**, **alert**, **emergency**
- The automatic forwarding of events to a central `syslog` server is supported
- Additional third party packages support: robust filtering, log analysis, event response, log file encryption, logging to a database and rate limiting

Example Syslog Messages

```
Mar 1 06:25:43 server1 sshd[23170]: Accepted publickey for server2 from
172.30.128.115 port 21011 ssh2

Mar 1 07:16:42 server1 sshd[9326]: Accepted password for murugiah from
10.20.30.108 port 1070 ssh2

Mar 1 07:16:53 server1 sshd[22938]: reverse mapping checking getaddrinfo for
ip10.165.nist.gov failed - POSSIBLE BREAKIN ATTEMPT!

Mar 1 07:26:28 server1 sshd[22572]: Accepted publickey for server2 from
172.30.128.115 port 30606 ssh2

Mar 1 07:28:33 server1 su: BAD SU kkent to root on /dev/tty2

Mar 1 07:28:41 server1 su: kkent to root on /dev/tty2
```

Figure 15.6 Examples of Syslog Messages

Syslog Facilities & Security Levels

Table 15.5 UNIX syslog Facilities and Severity Levels

(a) syslog Facilities

Facility	Message Description (generated by)
user	User process
kern	System kernel
mail	e-mail system
daemon	System daemon, such as <code>ftpd</code>
auth	Authorization programs <code>login</code> , <code>su</code> , and <code>getty</code>
lpr	Printing system
news	UseNet News system
uucp	UUCP system
cron	<code>cron</code> and <code>at</code>
local0-7	Up to 8 locally defined categories
mark	syslog, for timestamping logs

(b) syslog Severity Levels

Severity	Description
emerg	Most severe messages, such as immediate system shutdown
alert	System conditions requiring immediate attention
crit	Critical system conditions, such as failing hardware or software
err	Other system errors; recoverable
warning	Warning messages; recoverable
notice	unusual situation that merits investigation; a significant event that is typically part of normal day-to-day operation
info	Informational messages
debug	Messages for debugging purposes