Computer Intrusion Detection

Lecture 5

Auditing

Xiangyang Li

Partially based on M. Bishop's book and R. Bace's book

Outline

What is auditing?

How to design an auditing system?

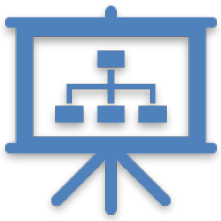
Auditing mechanisms and issues

File Auditing Example: NFSv2

What is Auditing?

Logging

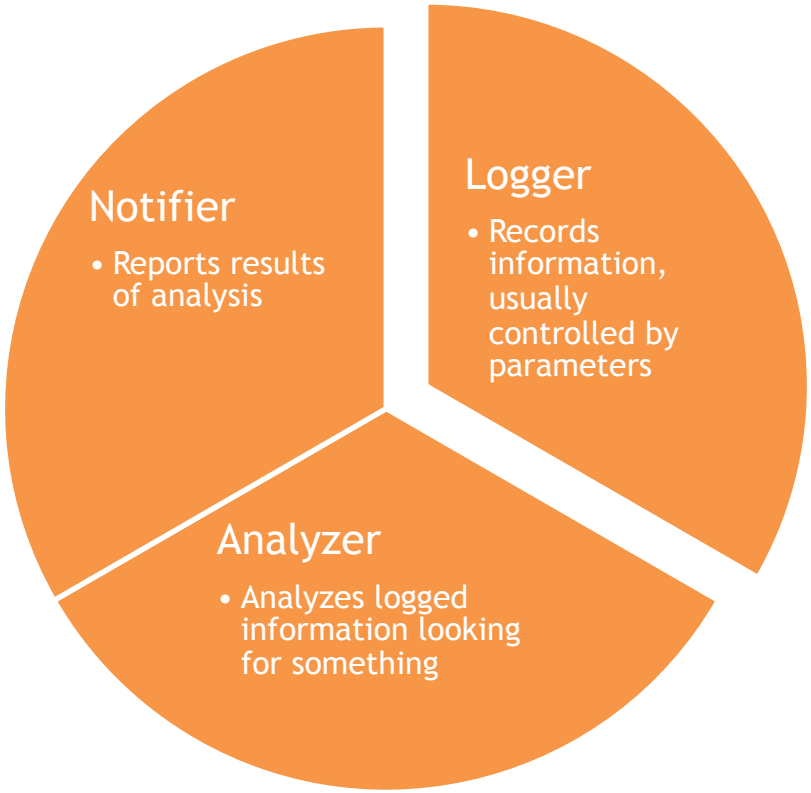
Recording events or statistics to provide information about system use and performance



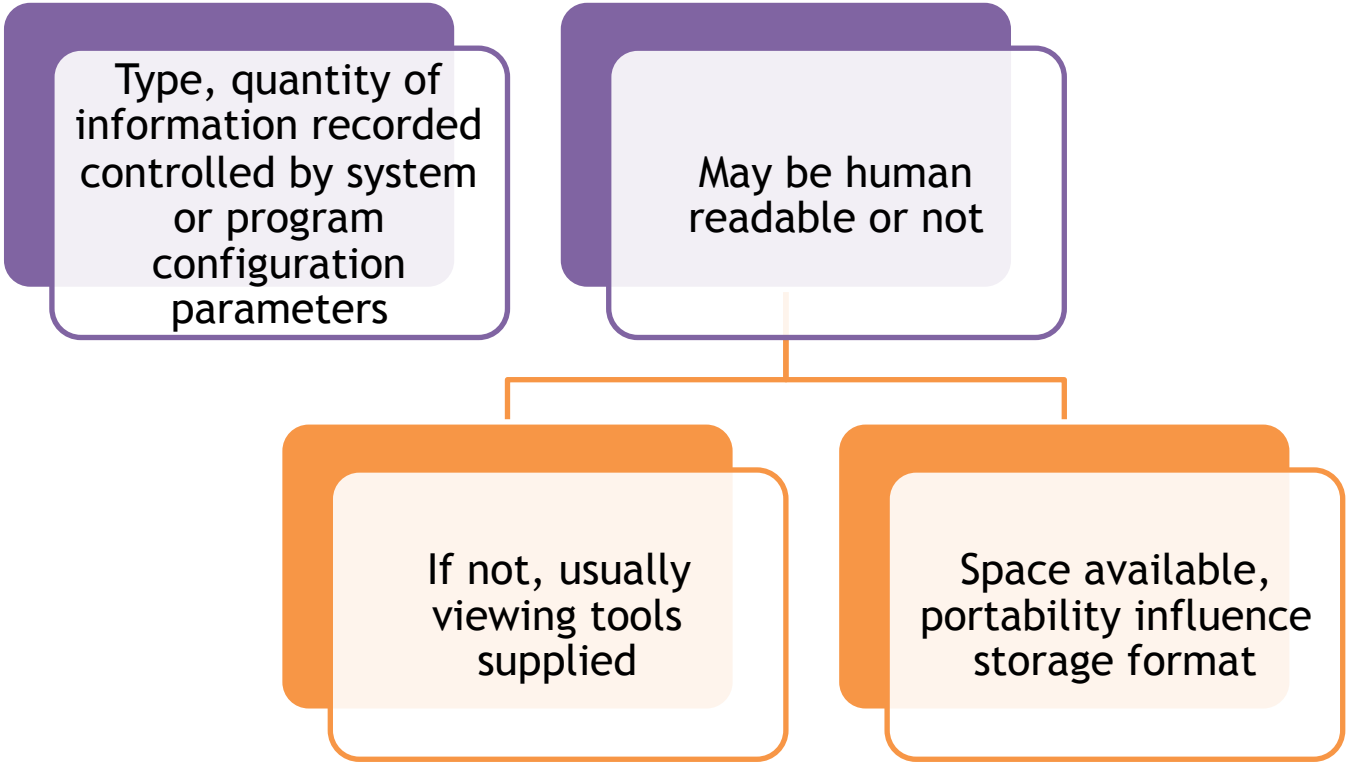
Auditing

Analysis of log records to present information about the system in a clear, understandable manner

Audit System Components



Logger


Type, quantity of information recorded controlled by system or program configuration parameters
May be human readable or not
If not, usually viewing tools supplied
Space available, portability influence storage format

Example: RACF

- Security enhancement package for IBM's MVS/VM
- Logs failed access attempts, use of privilege to change security levels, and (if desired) RACF interactions
- View events with LISTUSERS commands

RACF: Sample Entry

```
USER=EW125004   NAME=S.J.TURNER   OWNER=SECADM
    CREATED=88.004
  DEFAULT-GROUP=HUMRES    PASSDATE=88.004   PASS-
    INTERVAL=30
  ATTRIBUTES=ADSP
  REVOKE DATE=NONE    RESUME-DATE=NONE
  LAST-ACCESS=88.020/14:15:10
  CLASS AUTHORIZATIONS=NONE
  NO-INSTALLATION-DATA
  NO-MODEL-NAME
  LOGON ALLOWED     (DAYS)  (TIME)
  -------------------------------
  ANYDAY                      ANYTIME
    GROUP=HUMRES AUTH=JOIN CONNECT-OWNER=SECADM
                                    CONNECT-DATE=88.004
      CONNECTS= 15  UACC=READ LAST-
    CONNECT=88.018/16:45:06
      CONNECT ATTRIBUTES=NONE
      REVOKE DATE=NONE RESUME DATE=NONE
    GROUP=PERSNL AUTH=JOIN CONNECT-OWNER=SECADM
    CONNECT-DATE:88.004
      CONNECTS= 25 UACC=READ LAST-
    CONNECT=88.020/14:15:10
      CONNECT ATTRIBUTES=NONE
      REVOKE DATE=NONE RESUME DATE=NONE
    SECURITY-LEVEL=NONE SPECIFIED
    CATEGORY AUTHORIZATION
       NONE SPECIFIED
```

Analyzer

- Analyzes one or more logs
  - Logs may come from multiple systems, or a single system
  - May lead to changes in logging
  - May lead to a report of an event

Examples

- Using swatch to find instances of telnet from tcpd logs:

```
/telnet/&!/localhost/&!/
*.site.com/
```

- Query set overlap control in databases
  - If too much overlap between current query and past queries, do not answer
- Intrusion detection analysis engine (director)
  - Takes data from sensors and determines if an intrusion is occurring

Notifier

- Informs analyst, other entities of results of analysis
- May reconfigure logging and/ or analysis on basis of results

Examples

- Using *swatch* to notify of *telnet*s

```
/telnet/&!/localhost/&!/
   *.site.com/mail staff
```

- Query set overlap control in databases
  - Prevents response from being given if too much overlap occurs
- Three failed logins in a row disable user account
  - Notifier disables account, notifies sysadmin

Outline

What is auditing?

How to design an auditing system?

Auditing mechanisms and issues

File Auditing Example: NFSv2

Questions

- What do you log?
  - Hint: looking for violations of a policy, so record *at least* what will show such violations
- What do you audit?
  - Need not audit everything
  - Key: what is the policy involved?

Designing an Audit System

Goals determine what is logged

Idea: auditors want to detect violations of policy, which provides a set of constraints that the set of possible actions must satisfy.

Constraint p_i : *action* $\Rightarrow$ *condition*

So, audit functions that may violate the constraints.

Bell-LaPadula Model, Step 1

- Security levels arranged in linear ordering
  - Top Secret: highest
  - Secret
  - Confidential
  - Unclassified: lowest
- Levels consist of *security clearance* $L(s)$
  - Objects have *security classification* $L(o)$

Security Level Example

security level	*subject*	*object*
Top Secret	Tamara	Personnel Files
Secret	Samuel	E-Mail Files
Confidential	Claire	Activity Logs
Unclassified	Ulaley	Telephone Lists

- Tamara can read all files
- Claire cannot read Personnel or E-Mail Files
- Ulaley can only read Telephone Lists

Reading Information

- Information flows *up*, not *down*
  - "Reads up" disallowed, "reads down" allowed
- Simple Security Condition (Step 1)
  - Subject *s* can read object *o* iff $L(o) \leq L(s)$ and *s* has permission to read *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no reads up" rule

Writing Information

- Information flows up, not down
  - "Writes up" allowed, "writes down" disallowed
- *-Property (Step 1)
  - Subject *s* can write object *o* iff $L(s) \leq L(o)$ and *s* has permission to write *o*
    - Note: combines mandatory control (relationship of security levels) and discretionary control (the required permission)
  - Sometimes called "no writes down" rule

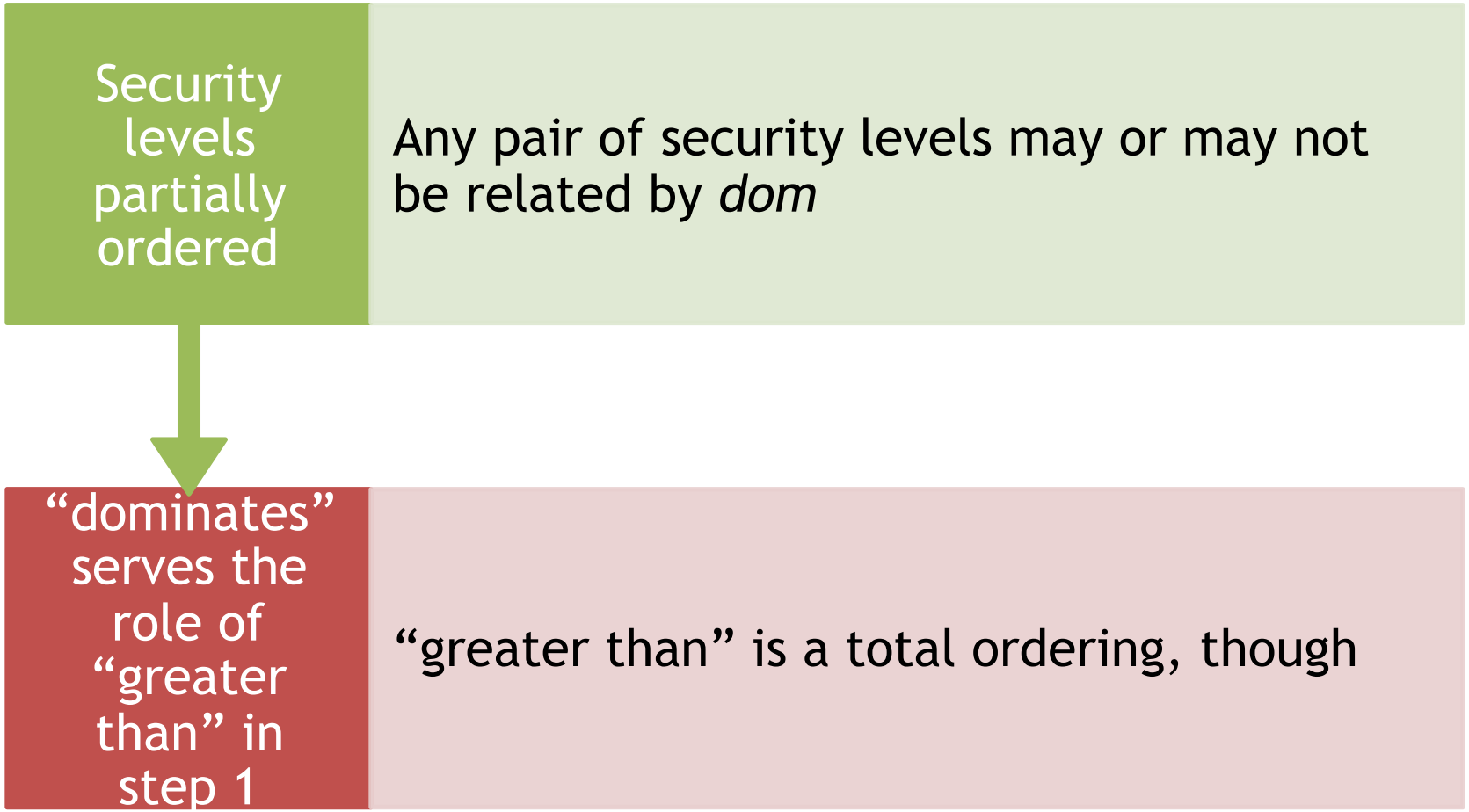
Auditing Requirements: Bell-LaPadula

- What need to be logged?
  - *SS Property: S* reads *O* $\Rightarrow$ *L*(*S*) $\geq$ *L*(*O*)
  - ** Property: S* writes *O* $\Rightarrow$ *L*(*S*) $\leq$ *L*(*O*)
- To check for violations, on each read and write, must log *L*(*S*), *L*(*O*), action (read, write), and result (success, failure)
- Note: need *not* record *S*, *O*!
  - In practice, done to identify the object and the user involved in the violation

Bell-LaPadula Model, Step 2

- Expand notion of security level to include categories
- Security level is (*clearance*, *category set*)
- Examples
  - (Top Secret, { NUC, EUR, ASI })
  - (Confidential, { EUR, ASI })
  - (Secret, { NUC, ASI })

Levels and Ordering



Levels and Lattices

- (A, C) *dom* (A', C') iff $A' \leq A$ and $C' \subseteq C$
- Examples
  - (Top Secret, {NUC, ASI}) *dom* (Secret, {NUC})
  - (Secret, {NUC, EUR}) *dom* (Confidential,{NUC, EUR})
  - (Top Secret, {NUC}) ¬*dom* (Confidential, {EUR})

New Auditing Requirements: Bell-LaPadula

What (else) need to be logged?

Remove Tranquility

- New commands to manipulate security level must also record information
  - *S* reclassify *O* to $L(O')$ => $L(O) \leq L(S)$ and $L(O') \leq L(S)$
- What need to be logged?
  - Log $L(O)$, $L(O')$, $L(S)$, action (reclassify), and result (success, failure)
  - Again, need not record *O* or *S* to detect violation

Another Example: Chinese Wall Model

- Subject S has $COI(S)$ and $CD(S)$
  - $CD_H(S)$ is set of company datasets that S has accessed
- Object O has $COI(O)$ and $CD(O)$
- Simple Security Constraint
  - S reads $O \Rightarrow COI(O) \neq COI(S) \vee CD(O) \in CD_H(S)$
  - Record $COI(O)$, $COI(S)$, $CD_H(S)$, $CD(O)$, action (read), and result (success, failure)

Defining Policy

- Now we know the importance of policy.
- Consider this example
  - A hospital deploys a database system for patient records. The system consists of a centralized DB server accessed by client systems in the hospital. Clients access the information through a network of connected PCs and via wireless PDAs
- What sorts of policy statements can we make about the hardware? Software? Users?

Defining Policy (cont'd)

- Possible statements
  - The DB server software will be kept up to date
  - Unused network services (ports) on the DB server will be disabled
  - Wireless access will employ strong cryptographic protocols
  - Users are prohibited from examining records of patients not in their care
- Machine readable policy is very hard problem
  - Particularly for misfeasance (i.e. insiders)

Converting Policy to Rules

- Policy states a philosophy of protection.
- Procedure states the specific goals that are associated with the philosophy of protection.
- Rule states the specific task that accomplishes the goal, along with
  - responsible party
  - time interval
  - reporting requirement
  - means of enforcing compliance

Monitoring Policy

- Implemented in an IDS
- Example:
  - Security policy statement

  "Access to patient financial information is restricted to the accounting clerk".
  - Monitoring policy statement

  "If patient financial information is accessed and subject is not a member of the group "accounting-clk," then generate an alert message."

Mapping Policy to Configuration - Causes of Security Problems

System design and development	System management	Trust allocation
• Inadequate development process / quality assurance • Errors/bugs	• Failure to create adequate policies • Failure to maintain (patches, etc.)	• Protocols with inadequate authentication • Failure to create adequate policies

Outline

What is auditing?

How to design an auditing system?

Auditing mechanisms and issues

File Auditing Example: NFSv2

Implementation Issues

Show non-security or find violations?

Former requires logging initial state as well as changes

Defining violations

Does “write” include “append” and “create directory”?

Multiple names for one object

Logging goes by *object* and not name

Representations can affect this (if you read raw disks, you’re reading files; can your auditing system determine which file?)

Syntactic Issues

- Data that is logged may be ambiguous
  - BSM: two optional text fields followed by two mandatory text fields
  - If three fields, which of the optional fields is omitted?
- Solution: use grammar to ensure well-defined syntax of log files

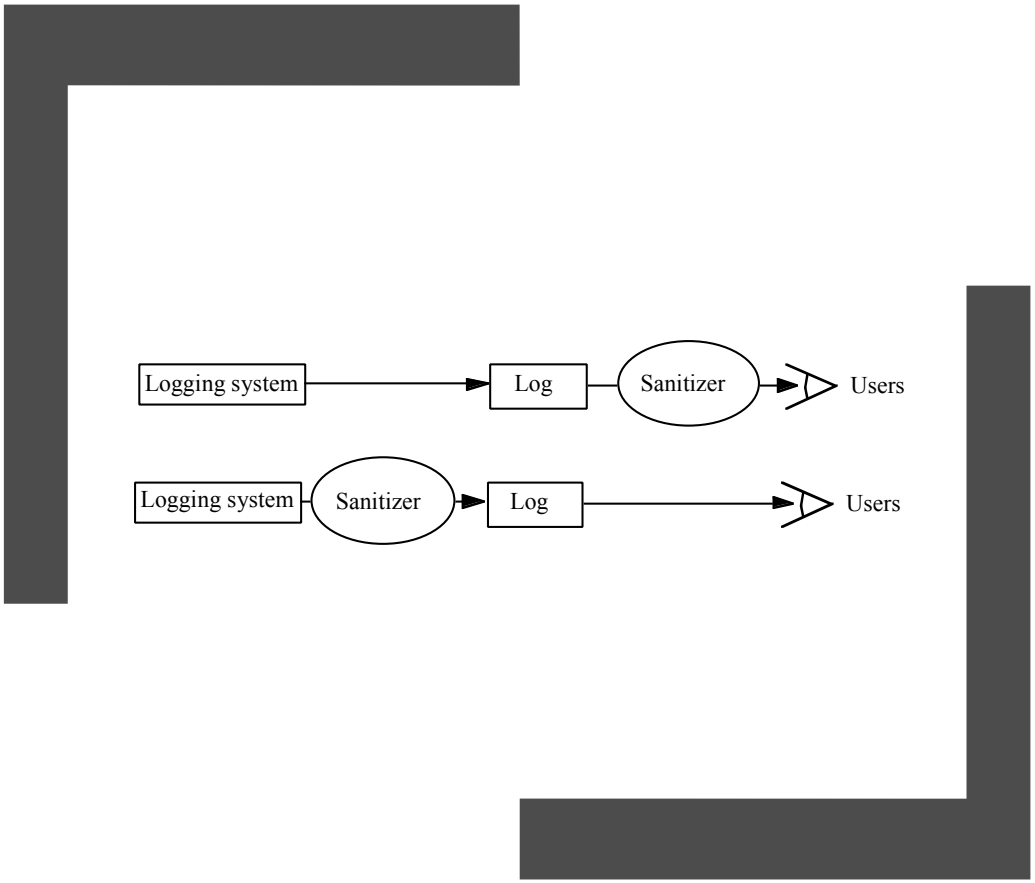
More Syntactic Example

- Unknown user uses anonymous *ftp* to retrieve file "/etc/passwd"
- Logged as such
- Problem: *which* /etc/passwd file?
  - One in system /etc directory
  - One in anonymous *ftp* directory /var/ftp/etc, and as *ftp* thinks /var/ftp is the root directory, /etc/passwd refers to /var/ftp/etc/passwd

Log Sanitization

- *U* set of users, *P* policy defining set of information *C*(*U*) that *U* cannot see; log sanitized when all information in *C*(*U*) deleted from log
- Two types of *P*
  - *C*(*U*) can't leave site
    - People inside site are trusted and information not sensitive to them
  - *C*(*U*) can't leave system
    - People inside site not trusted or (more commonly) information sensitive to them
    - Don't log this sensitive information

Logging Organization



- Top prevents information from leaving site
  - Users' privacy not protected from system administrators, other administrative personnel
- Bottom prevents information from leaving system
  - Data simply not recorded, or data scrambled before recording

Example

- Company wants to keep its IP addresses secret, but wants a consultant to analyze logs for an address scanning attack
  - Connections to port 25 on IP addresses 10.163.5.10, 10.163.5.11, 10.163.5.12, 10.163.5.13, 10.163.5.14, 10.163.5.15
  - Sanitize with random IP addresses
    - Cannot see sweep through consecutive IP addresses
  - Sanitize with sequential IP addresses
    - Can see sweep through consecutive IP addresses

Detect Violations of Known Policy

- Goal: does system enter a disallowed state?
- Two forms
  - State-based auditing
    - Look at current state of system
  - Transition-based auditing
    - Look at actions that transition system from one state to another

State-Based Auditing

- Log information about state and determine if state allowed
  - Assumption: you can get a snapshot of system state
  - Snapshot needs to be consistent
  - Non-distributed system needs to be quiescent
  - Distributed system can use Chandy-Lamport algorithm, or some other algorithm, to obtain this

Example

- File system auditing tools
  - Thought of as analyzing single state (snapshot)
  - In reality, analyze many slices of different state unless file system quiescent
  - Potential problem: if test at end depends on result of test at beginning, relevant parts of system state may have changed between the first test and the last
    - Classic TOCTTOU flaw

Transition-Based Auditing

- Log information about action, and examine current state and proposed transition to determine if new state would be disallowed
  - Note: just analyzing the transition may not be enough; you may need the initial state
  - Tend to use this when specific transitions *always* require analysis (for example, change of privilege)

Example

- TCP access control mechanism intercepts TCP connections and checks against a list of connections to be blocked
  - Obtains IP address of source of connection
  - Logs IP address, port, and result (allowed/blocked) in log file
  - Purely transition-based (current state not analyzed at all)

Land Attack Detection

- Must spot initial Land packet with source, destination addresses the same
- Logging requirement:
  - source port number, IP address
  - destination port number, IP address
- Auditing requirement:
  - If source port number = destination port number and source IP address = destination IP address, packet is part of a Land attack

Outline

What is auditing?

How to design an auditing system?

Auditing mechanisms and issues

File Auditing Example: NFSv2

NFS Version 2

- Mounting protocol
  - Client kernel contacts server's mount daemon
  - Daemon checks client is authorized to mount file system
  - Daemon returns *file handle* pointing to server mount point
  - Client creates entry in client file system corresponding to file handle
  - Access restrictions enforced
    - On client side: server not aware of these
    - On server side: client not aware of these

File Access Protocol

- Process tries to open file as if it were local
- Client kernel sends file handle for element of path referring to remote file to server's NFS server using LOOKUP request
- If file handle valid, server replies with appropriate file handle
- Client requests attributes with GETATTR
  - Client then determines if access allowed; if not, denies
- Iterate above three steps until handle obtained for requested file
  - Or access denied by client

Site Policy

1. NFS servers respond only to authorized clients
2. UNIX access controls regulate access to server's exported file system
3. No client host can access a nonexported file system

Resulting Constraints

1. File access granted ⇒ client authorized to import file system, user can search all parent directories, user can access file as requested, file is descendent of server's file system mount point
    - From P1, P2, P3
2. Device file created or file type changed to device ⇒ user's UID is 0
    - From P2; only UID 0 can do these actions
3. Possession of file handle ⇒ file handle issued to user
    - From P1, P2; otherwise unauthorized client could access files in forbidden ways
4. Operation succeeds ⇒ similar local operation would succeed
    - From P2; mount should fail if requester UID not 0

NFS Operations

- Transitions from secure to nonsecure state can occur only when NFS command occurs
- Example commands:
  - MOUNT *filesystem*
    - Mount the named file system on the requesting client, if allowed
  - LOOKUP *dir_handle file_name*
    - Search in directory with handle *dir_handle* for file named *file_name*; return file handle for *file_name*

Logging Requirements

1. When file handle issued, server records handle, UID and GID of user requesting it, client host making request
   - Similar to allocating file descriptor when file opened; allows validation of later requests
2. When file handle used as parameter, server records UID, GID of user
   - Was user using file handle issued that file handle?—useful for detecting spoofs

Logging Requirements

3. When file handle issued, server records relevant attributes of containing object
   - On LOOKUP, attributes of containing directory show whether it can be searched
4. Record results of each operation
   - Lets auditor determine result
5. Record file names used as arguments
   - Reconstruct path names, purpose of commands

Audit Criteria: MOUNT

1. Check that MOUNT server denies all requests by unauthorized clients to import file system that host exports
   - Obtained from constraints 1, 4
   - Log requirements 1 (who requests it), 3 (access attributes—to whom can it be exported), 4 (result)
2. Check file handle comes from client, user to which it was issued
   - Obtained from constraint 3
   - Log requirement 1 (who issued to), 2 (who is using)

Audit Criteria: LOOKUP

3. Check that directory has file system mount point as ancestor and user has search permission on directory
   - Obtained from constraint 1
   - Log requirements 2 (who is using handle), 3 (owner, group, type, permissions of object), 4 (result), 5 (reconstruct path name)