CAPP-Compliant Security Event Audit System for Mac OS X and FreeBSD

Robert N. M. Watson
Security Research
Computer Laboratory
University of Cambridge

March 23, 2006
Introduction

• Background
• Common Criteria, CAPP, evaluation
• What is security event audit?
• Audit design and implementation considerations
• Differences between UNIX and Mac OS X
• FreeBSD port
• OpenBSM
Organizations

• Apple Computer, Inc.
  – Tight hardware/software integration, single vendor

• McAfee Research, McAfee, Inc.,
  – Computer security research and engineering
    • Primarily DoD customers, but some commercial

• SAIC
  – Many things, but among them, evaluation lab

• TrustedBSD Project
  – Trusted operating system extensions for FreeBSD
Trusted Operating Systems

- Notions originated in security research and development during 1950's – 1970's
  - Trustworthy and security systems for US military
  - Later, scope expands

- Two focuses
  - Specific security feature sets
  - Assurance


- 1990's–2000's NIAP and Common Criteria (CC)
Role of Evaluations

• Security evaluations controversial
  – Does the evaluation address real security needs?
  – Is the goal more paper or a better product?
  – Do we know more after an evaluation?

• Security evaluations are, however, a reality
  – Cannot sell to US DoD (and others) without evaluation
  – Inclusion of many necessary security features has been driven by evaluation requirements
Common Criteria

• ISO standard and model for security evaluation
  – CC defines vocabulary and processes
  – Protection Profiles define functional requirements
  – Evaluation Assurance Level (EAL) defines assurance target

• Two widely used protection profiles for operating systems
  – CAPP, LSPP
  – Other protection profiles for other sorts of products
## NCSC Orange Book-Derived Protection Profiles

<table>
<thead>
<tr>
<th>Protection Profile</th>
<th>Features/Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Access Protection Profile (CAPP)</td>
<td>Derived from Orange Book C2</td>
</tr>
<tr>
<td></td>
<td>Multiple authenticated users</td>
</tr>
<tr>
<td></td>
<td>Separation of administrative role</td>
</tr>
<tr>
<td></td>
<td>Discretionary access control</td>
</tr>
<tr>
<td></td>
<td>Security event auditing</td>
</tr>
<tr>
<td></td>
<td>Minimal coverage of network concepts</td>
</tr>
<tr>
<td>Labelled Security Protection Profile (LSPP)</td>
<td>Derived from Orange Book B1</td>
</tr>
<tr>
<td></td>
<td>CAPP + Mandatory Access Control (MAC)</td>
</tr>
<tr>
<td></td>
<td>Role-Based Access Control (RBAC)</td>
</tr>
<tr>
<td></td>
<td>Multi-Level Security (MLS)</td>
</tr>
<tr>
<td></td>
<td>Enhanced security event auditing</td>
</tr>
<tr>
<td></td>
<td>Typically shipped with labelled networking</td>
</tr>
</tbody>
</table>
Assurance

- Assurance arguments critical to evaluation
  - Documentation of goals
  - Documentation of assumptions
  - Documentation of system design
  - Argument system implementation matches design
  - Documentation of process

- Assurance is measured in paper
  - For lower EAL, measurements < 1 yard/metre
  - For higher EAL, measurements > 1 yard/metre
Common Criteria Evaluation

• Five easy steps
  1. Select a protection profile, assurance level
  2. Write a security target, evaluation evidence
  3. Add features implementing missed requirements
  4. Write a very large cheque
  5. Work with evaluation lab through testing cycle

• Shortcuts
  – Evaluate to a cut down protection profile (PR)
  – Contract evaluation lab to write your evidence
UNIX and CAPP

- Most commercial UNIX systems meet CAPP requirements with minor configuration tweaks
- Three common extensions required:
  - Enhanced discretionary access control – ACLs
  - Security event audit
  - Authentication and password policy enforcement
- Of these, audit is the most difficult (expensive) to add to a UNIX system
What is Security Event Audit?

- Log of security-relevant events
  - Secure
  - Reliable
  - Fine-grained
  - Configurable

- A variety of uses including
  - Post-mortem analysis
  - Intrusion detection
  - Live system monitoring, debugging
Common Criteria and Audit

• CAPP defines functional requirements
  – Audit will provide comprehensive logging of security events defined in CAPP and security target
  – Reliability and robustness requirements key

• LSPP extends audit to include MAC labelling and decision information
## CAPP Requirements Table

<table>
<thead>
<tr>
<th>CAPP Category</th>
<th>Requirement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1.1</td>
<td>FAU_GEN.1</td>
<td>Audit Data Generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall be able to generate an audit record of the auditable events listed in column “Event” of Table 1 (Auditable Events). This includes all auditable events for the basic level of audit, except FIA_UID.1’s user identity during failures.</td>
</tr>
<tr>
<td>5.1.1.2</td>
<td>FAU_GEN.1</td>
<td>Audit Data Generation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall record within each audit record at least the following information: (a) Data and time of the event, type of the event, subject identity, and the outcome (success or failure) of the event; (b) additional information specified in Table 1.</td>
</tr>
<tr>
<td>5.1.2.1</td>
<td>FAU_GEN.2</td>
<td>User Identity Association</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall be able to associate each auditable event with the identity of the user that caused the event.</td>
</tr>
<tr>
<td>5.1.3.1</td>
<td>FAU_SAR.1</td>
<td>Audit Review</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall provide authorized administrators with the capability to read all audit information from the audit records.</td>
</tr>
<tr>
<td>5.1.3.2</td>
<td>FAU_SAR.1</td>
<td>Audit Review</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall provide the audit records in a manner suitable for the user to interpret the information.</td>
</tr>
<tr>
<td>5.1.4.1</td>
<td>FAU_SAR.2</td>
<td>Restricted Audit Review</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The TSF shall prohibit all users read access to the audit records, except those users that have been granted explicit read-access.</td>
</tr>
</tbody>
</table>
Audit Basics

- Audit records describe individual events
  - Attributable (to an authenticated user)
  - Non-attributable (no authenticated user)
  - Selected (configured to be audited)
- Most audit events fall into three classes
  - Access control
  - Authentication
  - Security management
- Audit log files are called “trails”
Audit Log Security

- Audit must be non-bypassable
- Right to add records to trail must be controlled
- Setting and viewing the audit configuration must be controlled
- Audit review must be controlled, assignable
- UNIX syslog has none of these properties!
Audit Reliability

• Reliability is key to audit implementation
  – If an event is auditable, selected, and occurs, then it must be audited
  – If an event is auditable, selected, but cannot be audited, it must not occur

• Ability to fail-stop system for predictable loss

• Upper bound on loss in the event of unexpected failure (i.e., power loss)

• UNIX syslog can't do this either
Mapping CAPP Audit into UNIX

- CAPP does not impose a specific OS structure
  - Does require a Trusted Code Base (TCB)
- UNIX structure is layered
  - Operating system kernel (TCB)
  - Operating system user space (TCB)
  - Other operating system user space (user)
- All audit events sourced in TCB
  - Authentication events mostly user space
  - Access control events mostly kernel space
Auditable Events in UNIX

- Access control
  - System calls checking for super user privilege
  - System calls with file system access control checks
    - Including path name lookup!
  - Login access control decisions

- Authentication, Account Management
  - Password changes, successful authentication, failed authentication, user administration

- Audit related events
Mapping CAPP Audit into UNIX

• Typical design choices
  – Audit event stream managed by kernel
  – Most records generated by system calls
  – Other records submitted by system applications using system call; privilege required
  – UNIX DAC permissions protect audit log
  – Helper daemon manages audit configuration, possibly writes audit stream
  – Process state extended with pre-selection masks and audit user ID
Audit and FreeBSD

- FreeBSD is in every sense, a classic UNIX
- All UNIX design choices on previous slide apply
  - Will tell you more in a few minutes
Audit and Mac OS X

- Mac OS X is based on a UNIX kernel
  - Most UNIX audit design choices apply
  - Kernel also offers Mach IPC
- Mac OS X user space relies on extensive IPC
  - UNIX processes cross boundaries with setuid
  - Mac OS X uses IPC to privileged daemons
- Extend Mach message trailers with audit fields
  - Allows privileged daemons to attribute audit events to current subject
Audit and Mac OS X (cont)

• Mac OS X process tree not traditional UNIX
  – UNIX process tree descends from single parent
  – In Mac OS X, user applications launched by a single privileged process (window server)

• Modification to approach that assumes all audit properties can be set at login and then inherited
  – Application launch services had to learn about audit
Modifications to FreeBSD, Mac OS X Kernels

- System call entry pre-selects, allocates record
- System call arguments, return values
- System call exit commits record
- Audit record queue implementation
- Audit event trigger mechanism
- Conversion from internal record to BSM
- Audit system calls
- Mach message trailer audit fields (Mac OS X)
Modifications to FreeBSD, Mac OS X User Space

- Audit library
- Audit trail viewer, reduction tool
- /etc/security audit configuration / databases
- Audit daemon to manage trails, triggers
- Set audit context at user login
- Application launch support for audit (Mac OS X)
- Audit in management tools, daemons
Sample Audit Control Flow

- **access()**: login uthread
  - Audit permission argument
- **Audit result, preselect, commit to record queue, wake up worker**: audit_worker kthread
  - Audit pathname argument
- **access() returns**: 

  **login kthread**
  - Dequeue audit record
  - Convert record to BSM
  - Commit to disk

  **audit_worker kthread**
  - Audit preselect, possibly assign record to thread, possibly wait for queue space
BSM APIs and File Formats

- Sun's Basic Security Module (BSM) de facto industry standard
  - File formats
    - Token-oriented audit trail format (almost TLV)
    - Audit configuration and databases
  - APIs
    - Construct, parse, process audit record streams
    - Manage audit state, pre-selection model
- Compatibility with many existing libraries and tools for free
BSM Audit Record Format

- Record header
- 0 or more variable argument tokens...
  (paths, ports, ...)
- Subject token
- Return token
- Trailer token

```
header,129,1,AUE_OPEN_R,0,Tue Feb 21 00:12:23 2006, + 253 msec
argument,2,0,flags
path,/lib/libc.so.6
attribute,444,root,wheel,16842497,11663267,46706288
subject,-1,root,wheel,root,wheel,319,0,0,0.0.0.0
return,success,6
trailer,129

header,108,1,AUE_CLOSE,0,Tue Feb 21 00:12:23 2006, + 255 msec
argument,2,0x6,fd
attribute,444,root,wheel,16842497,11663267,46706288
subject,-1,root,wheel,root,wheel,319,0,0,0.0.0.0
return,success,0
trailer,108
```
Thinking About Audit Reliability

• Correspondence between auditable events and audit records tricky
  – Audit record production is a queue split over several system components
  – Must bound end-to-end queue size based on available storage resources
  – Must bound end-to-end queue size based on maximum permissible loss

• "Fail-stop" must commit remaining records gracefully before stopping
Audit Queuing

User processes

Kernel

Per-thread queue

Audit subsystem queue

File system, Buffer cache

Stable store
Audit Selection

• Potential for audit record volume huge
  – Terabytes/hour on busy, fully audited system

• Two key points for audit record selection
  – Audit pre-selection to limit audit records created
  – Audit post-selection, or reduction, to eliminate undesired records after creation

• Mac OS X and FreeBSD support both models
  – Administrator can apply filters to users at login time
  – Administrator can use tools to reduce trails later
Audit Configuration: Pre-Selection

- Over 350 event types
  - Most of them meaningless individually
- Each event assigned to one or more classes
- Class masks assigned to users

- Over 350 event types
  - Most of them meaningless individually
- Each event assigned to one or more classes
- Class masks assigned to users

0:AUE_NULL:indir system call:no
1:AUE_EXIT:exit(2):pc
2:AUE_FORK:fork(2):pc
3:AUE_OPEN:open(2) - attr only:fa
4:AUE_CREAT:creat(2):fc
5:AUE_LINK:link(2):fc
6:AUE_UNLINK:unlink(2):fd
7:AUE_EXEC:exec(2):pc,ex
8:AUE_CHDIR:chdir(2):pc
...

0x00000000:invalid class
0x00000001:fr:file read
0x00000002:fw:file write
0x00000004:fa:file attribute access
0x00000008:fm:file attribute modify
0x00000010:fc:file create
0x00000020:fd:file delete
0x00000040:cl:file close
0x00000080:pc:process
0x00000100:nt:network
...

root:lo:no
audit:lo:no
test:all:no
www:fr,nt,ip:no
...
FreeBSD Port

• FreeBSD Operating System
  – BSD-licensed 4.4BSDlite2 derivative OS
  – Widely used in high-end embedded, networking, ISP, server spaces.
  – One of the source code bases for Mac OS X

• More classic UNIX operating system

• Common code base makes it an easy target

• Currently present in FreeBSD 7.x development tree, will be merged as of 6.2 release
Changes Made Porting to FreeBSD

• Endian-independent implementation
  – Also now important on Mac OS X
• Discard Mac OS X mach trailer support
• Add 64-bit token support
  – Also now important on Mac OS X
• Significant clean-up, debugging, documentation
• Largely different user space integration
• Introduce audit pipes
Audit Pipes

- Historically, audit for post-mortem analysis
- Today, for intrusion detection / monitoring
- Audit pipes provide live record feed
  - Lossy queue
  - Discrete audit records
  - Independent streams
OpenBSM

- BSD-licensed BSM library, tools, docs
- Portable across many platforms
- Implements Sun BSM with some extensions
- Foundation for FreeBSD, future Mac OS X use
- http://www.OpenBSM.org/
Conclusion

- Security event auditing is critical to successful security evaluation
  - Some argue audit is a critical security feature
- Complex reliability requirements
- Complex security requirements
- Open source common to FreeBSD, Mac OS X
- API/file format compatibility with Solaris