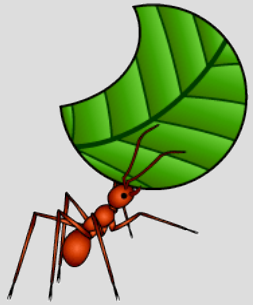
Introduction to TrustedBSD Audit + OpenBSM



Wayne Salamon
(wsalamon@freebsd.org)
Robert Watson
(rwatson@freebsd.org)

Introduction

- What is TrustedBSD?
- What is event auditing?
- CC + CAPP evaluation requirements
- The BSM audit format
- Kernel components
- MAC-Audit integration
- User space components
- Status and Availability

openbsm

TrustedBSD Project

- Trusted system extensions to FreeBSD
  - Announced April, 2000
- Security Infrastructure
  - OpenPAM
  - UFS2, Extended Attributes (EAs)
  - Kernel access control centralization
- Security Functionality
  - Access Control Lists (ACLs)
  - Extensible kernel access control (MAC Framework)
  - Mandatory Access Control (MAC)
  - Event Auditing, OpenBSM

openbsm

What is event auditing?

- Non-bypassable audit log describing security relevant events
- Security-relevant events
  - Controlled operations
  - Authentication related events
  - Security management events
- Appropriate for many uses
  - Post-mortem
  - Intrusion detection
  - Monitoring
- Typically, variable granularity: selection

openbsm

Common Criteria and Audit

- Audit is mandated by common OS security evaluations and standards
  - CC – Common Criteria
  - CAPP – Common Access Protection Profile
  - EAL – Evaluation Assurance Level
  - A variety of other more specific requirements
- CAPP identifies functional requirements
  - Audit will provide comprehensive logging of security events defined to be relevant to CAPP
  - Typically security events identified as part of evaluation process
  - Reliability and robustness requirements also key

openbsm

Excerpt of CAPP Requirements Table

CAPP Requirements Table

CAPP Category		Requirement	Description
5.1.1.1	FAU_GEN.1	Audit Data Generation	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.
5.1.1.2	FAU_GEN.1	Audit Data Generation	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.
5.1.2.1	FAU_GEN.2	User Identity Association	The TSF shall be able to associate each auditable event with the identity of the user that caused the event.
5.1.3.1	FAU_SAR.1	Audit Review	The TSF shall provide authorized administrators with the capabiity to read all audit information from the audit records.
5.1.3.2	FAU_SAR.1	Audit Review	The TSF shall provide the audit records in a manner suitable for the user to interpret the information.
5.1.4.1	FAU_SAR.2	Restricted Audit Review	The TSF shall prohibit all users read access to the audit records, excet those users that have been granted explicit read-access.
5.1.5	FAU_SAR.3	Selectable Audit Review	The TSF shall provide the ability to perform selection of audit data based on the following attributes: (a) user identity, (b) additional attributes.

openbsm

Auditing Basics

- Records describe subject action on object
  - Subjects are either authenticated or non-attributable
- Kernel events are mostly system calls
  - Vast majority relate to Discretionary Access Control (DAC)
  - Wherever an access control decision is made, an audit record may be cut
- User space programs also submit records
  - If appropriately privileged to write to audit log
- Kernel writes to one active log at a time

openbsm

Darwin Audit

- Darwin CAPP Audit
  - McAfee Research under contract to Apple, Inc.
  - In support of Mac OS X CAPP evaluation
- Open Source implementation of
  - Darwin kernel event auditing
  - Darwin user space event auditing
  - Sun's Basic Security Module (BSM) file format and APIs
  - Various Darwin packages, including xnu, bsm, ...
- Under a combination of APSLv2, BSD licenses

openbsm

FreeBSD Audit

- TrustedBSD Project has ported Darwin Audit to FreeBSD 6.x
  - Currently in a development branch
  - Initial merge anticipated in next few weeks
  - FreeBSD 6.0 (experimental feature)
  - FreeBSD 6.1 (production feature)
- OpenBSM
  - Extraction, cleanup, enhancement of BSM include files and libraries
  - Intended to be vendor import for Darwin BSM
  - Portable to other platforms including Linux, Solaris, *BSD

openbsm

BSM – Basic Security Module

- Sun's Basic Security Module (BSM)
  - In Solaris, kernel components, etc.
  - De facto audit API and file format standard
- Where possible, adopted API and file format
  - Some extensions for Darwin events not present in Solaris (etc)
- Permit reuse of applications, tools, docs
  - For example, the BSM code in OpenSSH
- BSM defines a token-oriented record stream
  - Extensible, easily parseable, flexible
  - Consists of tokens and sets of tokens (records)

openbsm

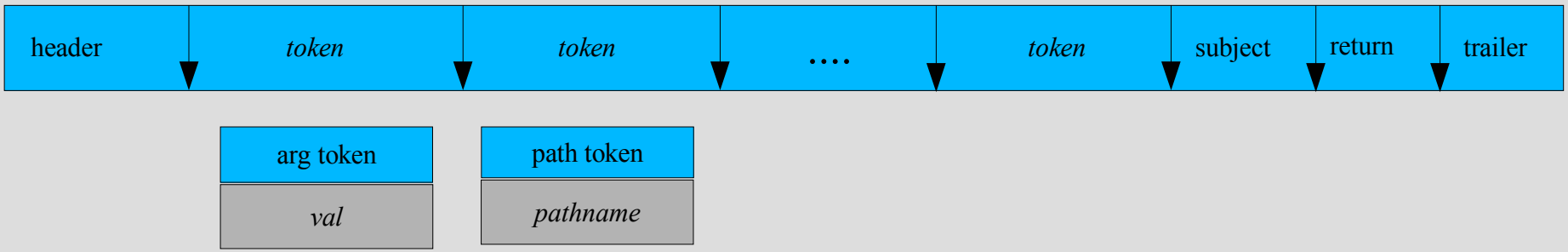
Audit File Stream Format

- Audit file streams consist of
  - Audit file identifier token
  - Stream of audit event records
  - Audit file identifier token
- This permits logs to be combined while maintaining log boundaries
  - Files may be concatenated
  - Files may be streamed
- Record consists of
  - Series of typed tokens describing an event

openbsm

Audit Record

- Audit records consist of a series of tokens
  - All records contain header, subject, return, trailer
  - Additional tokens of various types contain event-specific arguments
    - Ie., path names, file attributes, signal numbers, etc.



openbsm

Audit Records

- Header token contains event type, timestamp, total length of record, etc.:
  - `header,98,1,open(2) - write,creat,trunc,0,Fri Jul 9 21:43:59 2004, + 15 msec`
- Subject token contains user IDs, invariant audit ID, PID, session ID, terminal info:
  - `subject,audit,root,audit,wsalamon,audit,752,751,67108866,0.0.0.0`
- Return token contains system call success/failure and return value:
  - `return,success,3`
- Two object tokens for a file in same record:
  - `path,/private/var/run/utmp`
  - `attribute,100644,root,daemon,234881029,0,0`

openbsm

Audit Selection

- CAPP (and practicality) require the ability to select audit records
  - Must be able to audit all security-relevant events
  - Doesn't mean you (end-user) want to
- TrustedBSD Audit follows Solaris model
  - Pre-selection occurs early in system call to decide if a record may be required
  - Post-selection occurs at the end of the system call to decide if record was required
- Event masks are associated with each process, evaluated twice for each event
- Reduction tools also available

openbsm

Audit Events & Classes

- Kernel audit events are associated with system calls
- Audit classes are used to manage classes of related audit events
- Events are mapped to 1..n classes
  - Mapping is configured by control files
  - Loaded into kernel by audit daemon
  - Event classes include “file read”, “file write”, ..., “network”, ... “System V IPC”, ... “exec”, ...
- Processes have associated class masks for success and failure

openbsm

Audit Components

- Kernel components
  - Audit system calls, event management, logging, etc.
- Audit daemon
  - Configures audit system parameters, manage audit log rotation, send warnings
- BSM library
  - APIs for creating and parsing audit records
- Tools to display, reduce the audit log
- Modifications to login, passwd, etc, to audit user space events of interest

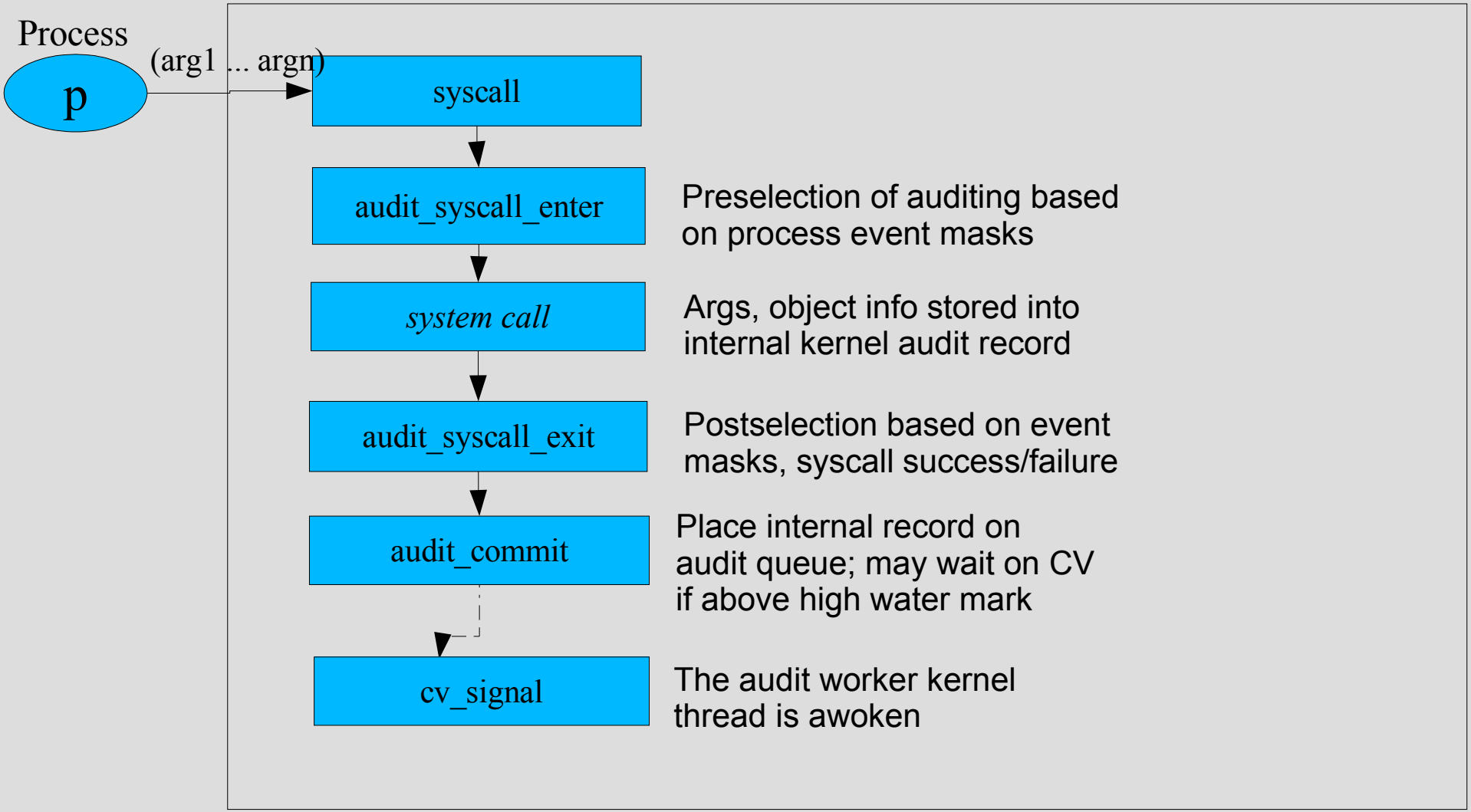
openbsm

FreeBSD Kernel Changes

- Imported Darwin 7.x kernel audit code
  - Mach portions removed
- Modified syscall.master files to include audit event associated with system call
- System calls instrumented to collect argument information
- Use special file instead of Mach messages to communicate with user space audit daemon

openbsm

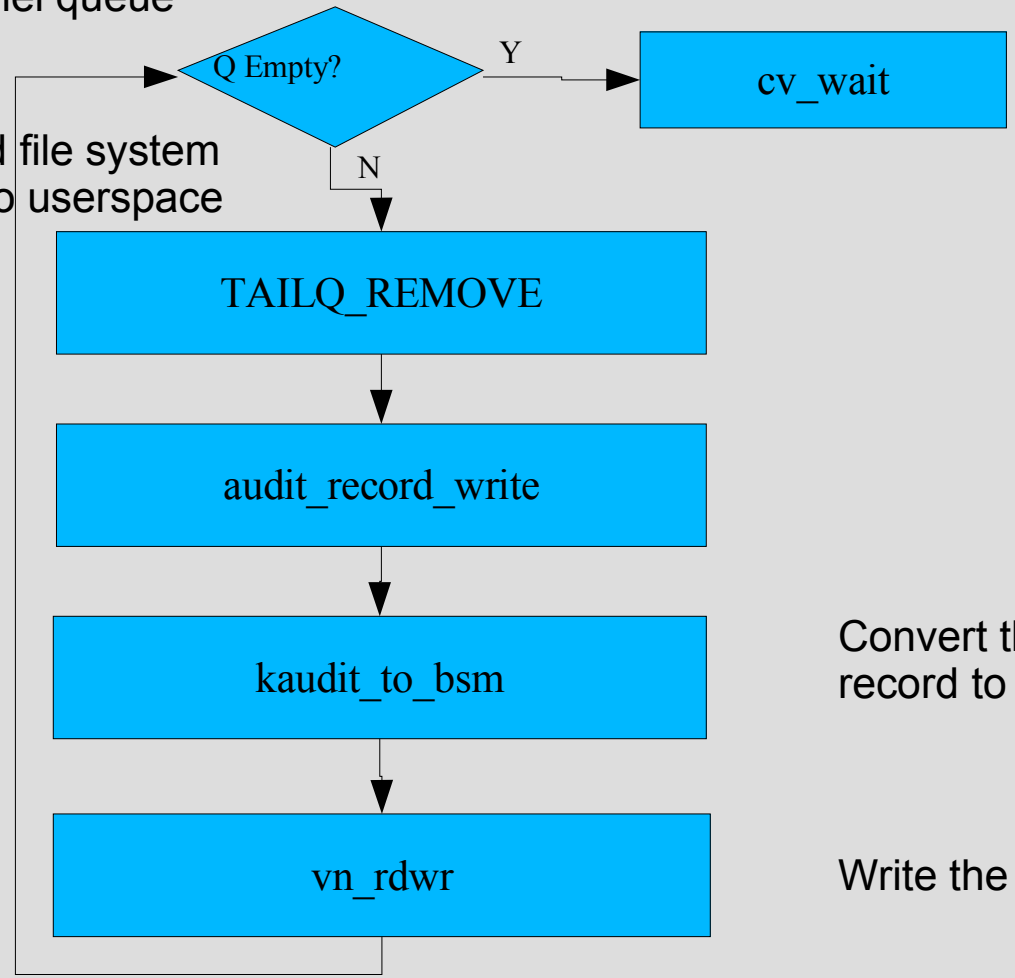
Kernel Flow Diagram



openbsm

Audit Worker Kernel Task

Remove the audit record from the kernel queue



Will check log and file system full, send trigger to userspace

Convert the internal audit record to BSM format

Write the audit record

openbsm

Audit Record Generation

- On system call entry, if pre-selection succeeds, kernel audit record is allocated on thread
  - Preselection based on audit masks associated with the process and class of event
- System call stores parameters, object info into kernel audit record
- On system call exit, post-selection decides whether to commit audit record
  - Based on result of call (success/failure) and process selection mask

openbsm

Audit Record Generation cont.

- Checks made, triggers sent if:
  - Filesystem free space falls below configured limit
  - Filesystem full
  - Audit log size over configurable maximum
- Kernel audit records converted to BSM format before writing
- Tokens are generated for the audit record based on type of record
- Header, subject, and trailer tokens added
- Audit record written

openbsm

MAC/Audit Integration

- The Audit system will pull subject/object labels from the policies when storing other subject/object information for auditing
  - A new interface in the MAC framework for policies to return audit-specific labels
  - Policies can also push ancillary data to the Audit system for inclusion in the currently audited system call using `mac_audit_text()`
- Audit information is placed in text tokens within the audit record

openbsm

Audit Daemon

- Audit daemon loads the event->class mapping into kernel on startup
- Sets audit configuration parameters in kernel
- Manages audit start, suspension, and termination
- Is also responsible for audit log rotation and generating warnings
- Receives triggers from the kernel via the `dev/audit` special file

openbsm

BSM Library

- BSM library ported with minimal changes
- Provides an API for generating BSM tokens and audit records
  - That can then be included in the audit trail via the audit() system call
- API for parsing an audit trail and presenting the information in human-readable form
- The OpenBSM project has been created to centralize changes to BSM library
- www.openbsm.org

openbsm

Audit Tools

- The audit log can be examined by using tools ported from Darwin:
  - `auditreduce`: Select records from audit log based on user ID, date, event, etc.
  - `praudit`: Present audit records in human-readable form
- Example:
  - `auditreduce -m AUE_OPEN_WC /var/audit/20040710003835.20040710014658 | praudit`
- Should be compatible with existing BSM tools

openbsm

Cool OpenBSM Logo by Jennifer Dodd

openbsm

Status and Availability

- Most of the core kernel components are in a TrustedBSD branch
- BSM library and audit reduction tools ported
- Audit daemon ported
- Several system calls audited
- Investigate defining new audit tokens for MAC auditing needs
- First public drop of OpenBSM available in the next few days
  - http://www.OpenBSM.org/
  - http://www.TrustedBSD.org/

openbsm

Future Directions

- Further MAC/Audit integration
- Complete system call coverage
- Complete login/userland audit events
- Remove interim kernel audit record and use BSM token format throughout
- Performance analysis
- Test, test, test
- Produce OpenBSM 1.0 release
- Merge to FreeBSD CVS

openbsm