TrustedBSD: Trusted Operating System Features for BSD
Robert Watson, Research Scientist
Host Intrusion Prevention Research Group
McAfee Research
Introduction

Introduction to TrustedBSD feature set

- Background: Trusted Operating Systems
  - Feature sets of interest
  - Role of assurance
- Evolution of the TrustedBSD Project
- Infrastructure to support security features
  - Extended attributes, GEOM, ...
- Security features provided via TrustedBSD/FreeBSD
  - ACLs, MAC, Audit, ...
- Experimental work to port feature set to Darwin/Mac OS X
Background: Trusted Operating Systems

- Notions originated in security research and development in the 1960's and 1970's
  - Desire to support trustworthy and secure systems for military (and later general government, banking, etc)

- Two dimensions of importance:
  - Security feature set
  - Assurance of correct security functionality

- Specifications play an important role
Feature Set: Cx/CAPP

- “Common Access Protection Profile”
- Basic security functionality
  - High level of trust in administrator, hardware
  - Minimal coverage of network concepts
  - Basic notions of users, authentication
  - Separation of administrative role
  - Discretionary protections via Access Control Lists (ACLs)
  - Security event auditing
  - Software life cycle process documentation
Feature Set: Bx/LSPP

“Labeled Security Protection Profile”

Building on C2/CAPP
  - Add mandatory protection, notions of role
    - Typically Biba for integrity, MLS for confidentiality
  - Enhanced security event auditing

Systems frequently also ship with trusted networking extensions
  - CIPSO, MAC integration for IPsec

Compartmented Mode Workstation (CMW)
Assurance

- How can you provide assurance of security?
- Assurance arguments critical to trusted systems
  - Documentation of intent, assumptions of system
  - Documentation that system architecture addresses intent
  - Argument that system is correctly implemented
  - Documentation of software development and maintenance processes
- For lower levels, measured in inches of paper
- For higher levels, development and architectural processes critical to success
Evaluation Process: Common Criteria

- Select a target feature set ("protection profile")
- Select a target assurance level (EALx)
- Contract to an evaluation lab
  - Probably also someone to help with evidence generation

Notes

- Narrow feature sets (cut down PP, context)
- Evaluation process is expensive, but critical to provide software to some audiences (governments, etc).
- Becoming more important as required by more consumers
- Interactions with open source beginning to be understood
Security Infrastructure Features

- Additional infrastructure required
- Problem: cryptographic storage protection
  - Solution: extensible storage framework (GEOM)
- Problem: access control lists and MAC require storage
  - Solution: extended attributes (UFS extattr, UFS2)
- Problem: diverse access control approaches
  - Solution: centralized access control
Mobile computing requires the ability to “revoke” data on mobile computing devices
- Lowest cost solution is a cryptographic transform
- Requires “insertion” of a transform in the storage stack

Rather than implement a one-time transform, provide transformation infrastructure
- GEOM allows “classes” to plug into the storage stack
  • Also used for other services (RAID, partitioning, et al.)
- Cleanly separates storage producers and consumers
- Facilities new security R&D for storage
Infrastructure: Extended Attributes

- New access control models frequently require new meta-data for file system objects
  - Access control lists require storage for list data
  - Mandatory access control requires storage for label data
  - Prevent work when adding more meta-data

- Extended attributes provide (name, value) pairs
  - Name is a character string; value is 0 or bytes of data
  - No semantics for content implied
  - Name spaces indicate protection (system, user)
  - Can be consumed by the kernel or userspace
Infrastructure: EAs on UFS1

- First generation implementation
  - Doesn't modify on-disk layout – facilitates prototyping
  - Allocates “backing files” by attribute name
  - Contains array of attribute data indexed by inode #
  - Requires explicit administrative configuration
  - Administrator-defined bound on max data size
  - Space reservation and efficiency are both issues
  - Works well for fixed-size attributes
  - Concurrency and locality issues for performance
Infrastructure: EAs on UFS2

- Perform roll of on-disk layout version
  - Add additional explicit storage for attributes in new layout
  - Data referenced by inode, stored close to inode
  - Uses normal UFS fragment/block mechanism, but prepared for future use of UFS2 pseudo-extents
  - Tighter integration with soft updates

- While there, also...
  - Bump to 64-bit disk addressing
  - New ABIs for system calls, et al
  - Other misc. bits and pieces
Infrastructure: Centralized Access Control

- Review all kernel access control decisions
- Use explicit monitoring APIs rather than kmem
- Abstract “common” checks
  - vnode access control
  - Inter-process authorization (visibility, signals, debugging, ...)
- SMPng/KSE credential synchronization model
- Not a security feature “per se”
  - However, critical to adding security features
Security Features

- GBDE: Cryptographic Disk Protection
- POSIX.1e Access Control Lists (ACLs)
- OpenPAM
- NSS
- MAC Framework and policy modules
- SEBSD
- SEDarwin
- Audit
GBDE: GEOM-Based Disk Encryption

- Storage encryption using key or random key
  - Intended to be resilient to cryptographic attack
  - Appropriate for use on notebooks, for swap devices, etc.
- Performed at block level, not file system level
- Created using GEOM class; once instance per encrypted storage device
- Auto-configuring, subject to key availability
- Details covered in GBDE session yesterday.
- Implementation by Poul-Henning Kamp
POSIX.1e Access Control Lists (ACLs)

- Enhanced “discretionary” access control
  - Administrator/owners of objects control object protections
  - Extension of permission model permits new entries
    - Additional users, additional groups
    - Mode compatibility through “mask” entry

- Based on POSIX.1eD17 draft standard
  - Specification never finalized for a variety of reasons

- Model selected due to compatibility concerns
  - On the whole, API-compatible with IRIX, Linux
  - Semantics similar but syntax non-identical to Solaris
OpenPAM

- Pluggable Authentication Modules (PAM)
- FreeBSD used linux-pam derivative
- Desire for fresh implementation
  - More complete integration required
  - XSSO standards compliance, Solaris compatibility
  - Strong portability goals
  - Security audit and review
  - More complete set of modules
- OpenPAM integrated into FreeBSD 5.x
NSS – Name Service Switch

- NSS permits directory services to be plugged
  - Similar to PAM for password file, group file, etc
  - Allows new directory services to be plugged in as modules
    - LDAP particularly of interest
  - Requirement for extensibility so new database types and databases can be added easily
- Current implementation uses shared libraries
  - On-going work to support IPC to NSS daemon for caching, reduced cost

- Integrated into FreeBSD 5.x
MAC Framework and Policy Modules

- Addresses two requirements
  - Mandatory Access Control (MAC) policies
  - Extensible/flexible kernel policy mechanism

- Allows extension of kernel access control model
  - Policies encapsulated in kernel or loadable modules
    - Compile-time, boot-time, and run-time extension
  - Modules can instrument critical access decisions in kernel
  - Provides common infrastructure, such as labeling, APIs
  - Automatic composition of multiple policies
Rationale for Security Extensions

- Common FreeBSD deployment scenarios
  - Banks, multi-user ISP environments
  - Web-hosting cluster, firewalls
  - “High-end embedded”

- Many of these scenarios have requirements poorly addressed by traditional UNIX security
  - OS hardening
  - Mandatory protection
  - Flexible, manageable, scalable protection
Why a MAC Framework?

- Support required in operating system for new security services
  - Costs of locally maintaining security extensions are high
  - Framework offers extensibility so that policies may be enhanced without changing base operating system

- There does not appear to be one perfect security model or policy
  - Sites may have different security/performance trade-offs
  - Sites may have special local requirements
  - Third party and research products
MAC Framework Background

- Extensible security framework
  - Policies implemented as modules
  - Common policy infrastructure like labeling
  - Sample policy modules, such as Biba, MLS, TE, hardening policies, et al.
  - Composes multiple policies if present
  - Also provides APIs for label-aware and possibly policy-agnostic applications

- Shipped in FreeBSD 5.0 to 5.2, 5.2.1
Kernel MAC Framework

- User Process
- System Call Interface
  - VFS
  - Socket IPC
  - Process Signalling
  - Pipe IPC
- MAC Framework
  - mac_biba
  - mac BSD extended
  - Sebsd
Policy Entry Point Invocation
Policy-Agnostic Labeling Abstraction

MAC Framework

Destroy label
check file read?
EACCES
Init label
Internalize label
OK
check relabel file?
OK
relabel
Destroy label

mac_biba

1
label-1
jail
biba
biba/low
jail.a
jail.b
biba/high

2
label-2
jail
biba

3
label-3
jail
biba
biba/low
Modifications to FreeBSD to Introduce MAC Framework

- A variety of architectural cleanups
  - Audit and minimize use of privilege
  - Centralize inter-process access control
  - Centralize discretionary access control for files
  - Clean up System V IPC permission functions
  - Prefer controlled and explicit export interfaces to kmem
  - Combine *cred structures into ucred; adopt td_ucred
  - Correct many semantic errors relating to credentials
  - Support moves to kernel threading, fine-grained locking, SMP
Modifications to FreeBSD to add the MAC Framework (cont)

- Infrastructure components
  - Add support for extended attributes in UFS1; build UFS2

- Actual MAC Framework changes
  - Instrument kernel objects for labeling, access control
  - Instrument kernel objects for misc. life cycle events
  - Create MAC Framework components (policy registration, composition, label infrastructure, system calls, ...)
  - Create sample policy modules
  - Provide userspace tools to exercise new system calls
  - Modify login mechanisms, user databases, etc.
List of Labeled Objects

- **Processes**
  - Process credential, process

- **File System**
  - Mountpoint, vnode, devfs directory entries

- **IPC**
  - Pipe IPC, System V IPC (SHM, Sem, Msg), Posix IPC

- **Networking**
  - Interface, mbuf, socket, Inet PCB, IP fragment queue, Ipsec, security association
Integration of MAC Framework into FreeBSD

Scheduler

Virtual Memory

Newbus, Device Drivers

System Call API/ABIs
Native, Linux, SVR4, OSF/1, PECOFF, ...

Process/Thread Support

SysV SHM

Pipe IPC

GEOM Storage Framework

Interface Framework

VFS

File Interface

sysV msgq, sem

Network Protocols

Socket IPC

File System Interface

UFS

devfs, specfs

GEOM Storage Framework

File Interface

Newbus, Device Drivers

Process

Threaded Process

System Call API/ABIs
Native, Linux, SVR4, OSF/1, PECOFF, ...

Process

Threaded Process
Where Next for the TrustedBSD MAC Framework

- Continue to research and develop TrustedBSD MAC Framework on FreeBSD
  - Enhanced support for IPsec
  - Improve productionability of policy modules
  - Continued R&D for SEBSD
  - Integrate with Audit functionality
Sample Policy Modules

- **mac_test** regression test, stub, null modules
- Traditional labeled MAC policies
  - Biba fixed-label integrity, LOMAC floating-label integrity
  - Hierarchal and compartmented Multi-Level Security (MLS)
  - SELinux FLASK/TE “SEBSD”
- Hardening policies
  - File system “firewall”
  - Interface silencing
  - Port ACLs
  - User partitions
SEBSD: Security-Enhanced BSD
Port of FLASK/TE from SELinux

SELinux based on:
- NSA's FLASK architecture
  - Developed on FLUX, a Mach/BSD microkernel
  - Access control abstraction based on subjects, objects, sids
- Type Enforcement policy language
  - Similar to Domain and Type Enforcement (DTE)
  - Subjects assigned domains, objects types
  - Rule language permits subject methods on objects
  - Domain transitions occur on selected binaries
- Policy file determines nature and granularity of policy
MAC Framework Modifications Required for SEBSD

- Framework parallel to LSM in construction
  - Similarity between LSM and MAC Framework simplify implementation; differences simplify it further

- Provides stronger label manipulation and management calls
  - Don't need a number of the system call additions required to run FLASK on Linux

- Removed notion of SID exposed to userspace since mature APIs for labels already existed
  - This approach later adopted in SELinux, also.
Creating SEBSD Module from Largely OS-Independent FLASK/TE

- At start
  - SELinux tightly integrated FLASK/TE into Linux kernel
  - Over course of SEBSD work, similar transformation was made with LSM

- MAC Framework plays similar role to LSM for SEBSD
Current Status of SEBSD

- Kernel module “sebsd.ko” functional
  - Most non-network objects labeled and enforced for most interesting methods
  - File descriptor, privilege adaptations of MAC Framework complete

- Userspace experimental but usable
  - Libsebsd port complete, ports of SELinux userland programs completed as needed (checkpolicy, newrole, ...)
  - Adapted policy allows many applications to run
    • Few changes needed for third party applications, mostly change required for base system components
SEBSD: Implementation

- Fairly straightforward to port FLASK/TE
  - FLASK/TE originally developed on BSD
  - Encapsulated FLASK/TE into MAC Framework module

- Some enhancement to MAC Framework
  - Requires labeling, access control for file descriptors
  - Requires greater policy control over superuser privilege
  - Required tighter integration into user space components

- In many ways easier on FreeBSD than Linux
  - MAC Framework infrastructure critical (labels, APIs, tools)
  - FreeBSD locking much better defined
SEDarwin: Security-Enhanced Darwin Port of MAC Framework, SEBSD

- Currently experimental work
  - Ported extended attributes, MAC Framework to XNU
  - Ported SEBSD module and simple sample TE policy
  - Modified some user space applications
  - Explored applying mandatory protections to Mach
  - Now porting other policies, improving maturity

- Many lessons learned concerning Darwin
  - Build environment, architectural similarities and differences, HFS+ issues, closed source pieces, working with Apple, windowing systems, Mach, ...
Security Event Auditing

- Fine-grained security event auditing
  - Create a detailed audit log of security events
    - Postmortem
    - Intrusion detection
  - Required by various security standards
    - Including Orange Book, Common Criteria

- Detailed audit of result of many event classes
  - Access to controlled objects (files, network, etc)
  - Authentication events
  - System configuration events
Implementation Requirements

- Process properties (audit ID, session, ...)
- System calls to set properties on login
- System calls to configure audit support
- Instrument kernel events to generate audit trail
- System calls to submit user audit records
- Modifications to user applications (login, et al)
- Kernel record queue, queue limits, disk drain
- User databases and library
- Applications for printing, parsing, managing
Audit Implementation

- McAfee Research implemented Audit on Mac OS X/Darwin platform under contract
  - Uses Solaris BSM API, user interfaces, trail format

- Currently porting implementation to FreeBSD
  - Subject to code drops, licensing from Apple

- Hard problems to solve, however, include
  - How to generate file paths to use in audit records for UFS
  - Problems solved in HFS+ due to different name properties

- Work in progress; 6.x/5.4 time frame
Conclusion

- TrustedBSD Project active
  - Steady stream of features applied to FreeBSD 4.x, 5.x, and upcoming 6.x branches
  - Some features quite mature (GEOM, UFS2, extended attributes, OpenPAM, NSS, ACLs)
  - Other features in the process of maturing (MAC Framework, MAC policies)
  - Others in early development (Audit)

- Information at http://www.TrustedBSD.org/
- Feel free to join lists, post messages, pitch in!