MAC Framework Port to OpenDarwin
Brown Bag Lunch

Robert Watson, Research Scientist
McAfee Research
MAC Framework port to Open Darwin
Brown Bag Lunch

■ TrustedBSD MAC Framework
  - Extensible kernel access control framework
  - Permits new kernel security policies to be loaded via loadable kernel modules
  - Supports a variety of security policies
  - DARPA-sponsored R&D on FreeBSD platform

■ Follow-on work
  - Port NSA's FLASK/TE from SELinux to FreeBSD as a MAC Framework policy module
  - Port MAC Framework and “SEBSD” module to Darwin
Talk Outline

- Background on CHATS, CBOSS
- Abbreviated design and implementation of TrustedBSD MAC Framework
- Introduction to SEBSBD policy module
- Port of MAC Framework to Darwin, Mac OS X
- Implications for SEBSBD policy module
- New capabilities
- Areas for future work
CBOSS: Community-Based Open Source Security

- DARPA CHATS program under Doug Maughan
  - Create a partnership between leading open source developers and industry security R&D laboratory
  - Additional research and development funding for maturity of MAC Framework, development of SEBSD, port of both to Darwin/Mac OS X

- By improving the security of open source systems, DARPA can impact a wide variety of COTS and research products
  - Rapid technology transfer path of open source
CBOSS Project Overview

- Many extremes in OS security work:
  - Write OS from the ground up
  - Don't change the OS at all
  - Maintain a local version with extensive modifications

- Avoid pitfalls of these approaches by:
  - Leveraging ability to modify open source FreeBSD operating system to provide security extensibility services
  - Working with open source developers to assure knowledge, process, technology transfer
Benefits to the CBOSS Approach

- Support for secure out-of-the-box COTS operating systems
  - Rapid time-to-market of open source already showing concrete benefits
  - Berkeley-licensed open source software rapidly transfers to closed source software products
  - Better support for future security research through extensibility and stronger support infrastructure
  - Long-term improvements in architecture, implementation, process outside of the research community
TrustedBSD MAC Framework
TrustedBSD MAC Framework

- Problem with security extensions for OS's:
  - Expensive to develop, deploy, and maintain
  - Extensions often conflict
  - There is no “one right policy” for all consumers

- Solution:
  - Improve operating system structure to better support security extensions
  - Provide common elements of policies to reduce redundant work by policy implementors
  - Avoid committing to any one specific policy
Why a Framework?

- Support required in the 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 the 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
Kernel MAC Framework Elements

- Policies are encapsulated in kernel modules
- Framework captures common policy elements:
  - Labeling service allows policies to label system objects
  - Policies may perform access control checks to limit access to system objects and services
- When multiple policies are present, their results are (usefully) composed
- Policy-aware and security-aware applications may access and maintain labels on system objects
Kernel MAC Framework

System Call Interface

User Process

User Process

User Process

VFS

Socket IPC

Process Signalling

Pipe IPC

MAC Framework

mac_biba

mac_bsdextended

...

Sebsd
Policy Entry Point Invocation
Policy-Agnostic Labeling Abstraction

MAC Framework

1. Destroy label
2. check file read?
3. EACCES
4. Init label
5. Internalize label
6. OK
7. check relabel file?
8. OK
9. relabel
10. Destroy label

mac_biba

1. label-1
   - jail
   - biba
   - biba/low
2. label-2
   - jail
   - biba
   - biba/high
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
Sample Policy Modules

- mac_test regression test, stub, null modules
- Traditional labeled MAC policies
  - Biba fixed-label integrity, LOMAC floating-label integrity
  - Hierarchical and compartmented Multi-Level Security (MLS)
  - SELinux FLASK/TE “SEBSD”
- Hardening policies
  - File system “firewall”
  - Interface silencing
  - Port ACLs
  - User partitions
Port of the TrustedBSD MAC Framework to Open Darwin
Port of the TrustedBSD MAC Framework to Open Darwin

- Adapt MAC Framework to Darwin
  - Strong research and development motivations
  - Structural and implementation similarities to FreeBSD

- Targeted at Open Darwin 6.6 (Jaguar 10.2.8)
  - Working only from open source components
  - Also “extended” some binary-only components

- Currently R&D Prototype
  - Illustrates concept/approach, but not yet production-quality
  - mac_test, SEBSD policy modules only currently
Strategy: Migrate MAC Framework to Darwin, Port SEBSD as SEDarwin

- Exploit common source code and design roots of FreeBSD and Darwin
  - Port dependencies, such as extended attributes
  - Migrate MAC Framework to Darwin
  - Expand MAC Framework and policies to address Darwin-specific features, such as Mach IPC
    - Requires MAC Framework to sit between various layers
  - Modify Darwin userspace applications
  - Produce adapted MLS, SEBSD TE policies
Architecture of Darwin Kernel (Gross Over-Simplification)

- Mach provides low-level IPC, memory, synchronization primitives
- IOKit provides OO driver infrastructure
- BSD provides high level IPC, networking, storage services
Infrastructure: File System Extended Attributes

- Many policies rely on additional policy-specific security attributes
  - Provide a generic extended attribute meta-data service
  - UFS1 on FreeBSD: extended attributes in per-fs/attribute backing files indexed by inode number
  - UFS2 on FreeBSD: additional file block/extent for per_inode metadata

- Ported UFS1 extended attributes to VFS, HFS+
  - Slower and less tightly integrated, but easier to port
  - No generation number in HFS+ to detect inconsistencies
Additional Infrastructure

- Port "sbuf" abstraction to Darwin kernel
  - Safe string manipulation for labels
- Port strsep() and other string management
- Port FreeBSD SMPng condition variables
  - Wrap Mach wait_queue_t with higher level synchronization "struct cv" primitive
- Modify BootX to preload non-binary data
  - Pre-load policy file during boot before file systems are mounted
Adapt MAC Framework to Mach Synchronization and Allocation

- Because we interact directly with Mach, must make use of Mach-level primitives
- All MAC Framework components assumed to run free of funnel
  - Except where kernel structures are covered by funnel

- Synchronization
  - FreeBSD and Mach mutexes basically the same
  - Condition variable changes

- Memory allocation
  - Move to Mach kmem allocator, zone allocator
BSD and Mach Layers

- TrustedBSD MAC Framework on FreeBSD addresses only BSD-layer primitives
  - BSD processes
  - BSD VFS
  - BSD networking
  - BSD IPC

- Initial port addressed only BSD components
  - Maintain labels and access control only on BSD objects
  - Now addressing Mach components
Mach Integration

- Add additional initialization stages

- Added label and label lock to Mach Task
  - Maintain BSD process label as “primary” label for tasks bound to processes
  - Maintain BSD process label as “primary” label for tasks bound to processes
  - Need a leaf lock to avoid violating lock orders

- Add label to Mach port
  - Export label information using message trailers
  - Still exploring how to do this best/fastest/simplest/...
mach_init and bootstrap namespace

- **Mach_init name space critical to security**
  - Ability to spoof services throughout desktop/system
  - Especially with untrusted root

- **Export label management, access decision functions to userspace**
  - Allow user processes to maintain labels on objects
  - May request access control decisions using passed labels, message trailer

- **Exploring “object handle” object attached to Mach port**
Integration of MAC Framework into Darwin Prototype

Mach: Scheduler, VM, IPC

BSD Process/Thread Support

SysV SHM

SysV msgq, sem

Mach/BSD System Call Layer

HFS+, UFS

VFS

devfs, specfs

File Interface

IOKit Device Driver Framework

Interface Framework

Network Protocols

Socket IPC

Process

Threaded Process

...
System Calls, Support Libraries

- Extended attribute system calls, support library
- Userland tools for extended attributes
- MAC label system calls, support library
- Userland tools for manipulating MAC labels
- Modifications to libkvm, ps, ls, ifconfig
Modifications to Login Window

- Closed source component in Mac OS X
- Provided modified exec(), setlogin() library calls for loginwindow
  - Track logins and required label changes
  - Intercept execution step to perform necessary relabel operations
- Lack of inheritance from login session creator presents substantial obstacle
  - Not just for MAC, also for resource limits, audit, etc.
Issues and Concerns

- Lack of unified build infrastructure for Darwin
  - Challenging to build and maintain extensive modifications

- Serious binary compatibility issues
  - Drivers when expanding data structures for labels
  - Had to back off initial port of network stack components

- Mach wait_queue primitive much weaker

- Mach IPC
  - Mach IPC primitives very weak, semantically
  - Requires applications to be much more involved in access control than in traditional UNIX system
Additional Issues and Concerns

- HFS+ lacks generation numbers
  - May break NFS, also prevents us from checking consistency of attributes with file system objects
  - Prefer that HFS+ had a native extended meta-data service

- Source code for loginwindow not available
  - Jaguar substantially less mature than Panther
  - While there have been improvements to the login process and credential management, still much to be done

- Jaguar applications behave poorly on failure
Additional HFS+ concerns

- TrustedBSD MAC Framework splits ownership of label management with file system
  - Performs access control checks at cross-file system layer
  - Some file systems provide per-label storage
  - Other file systems rely on VFS layer labeling

- Darwin offers a number of stronger file system system calls
  - Permits more direct reading, manipulating of disk catalog
  - Requires MAC Framework to become more involved in HFS+, not to mention layring issues
Things That Went Quite Well

- MAC Framework ported quite well for basic BSD objects
  - Little or no problem porting for processes, vnodes
  - Some improvement in Darwin abstractions required
  - Tools and libraries “just worked”

- SEBSD slid over without a hitch
  - Once MAC Framework pieces were there, it “just worked”

- Many applications do behave well, especially if they don't treat HFS+ as “special”
Capabilities: High Level Overview

- Insufficient time to discuss policies in detail
- Policies may implement a variety of models
  - Policies attach to the system during boot or at run-time
  - May implement controls on interesting subsets of system
  - May instrument access control decisions associated with a variety of system services in kernel and user space
  - May maintain additional security labels on system objects
- Information flow policies largely focus of .mil
- Hardening, discretionary policies also possible
Where We're Going Next

Four dimensions into which we are expanding the work:

- Expand coverage of BSD objects – IPC, etc
- Expand into the Mach space: a “must do” to provide comprehensive enforcement, but will raise synchronization and performance issues
- Expand SEBSD policy from minimalist demo policy to useful enforcement policies
- Port MLS policy module
- Explore impact of this work on OS X more extensively
Conclusion

- MAC Framework provides access control extensibility on FreeBSD
- Proof-of-concept port of MAC Framework to Darwin at early prototype phase
  - Labeling of some but not all objects
  - Enforcement of some but not all access methods
  - Limited file system support, application integration
- Successfully running with ported SEBSD policy module, others underway
How to Get the Code

- http://opensource.nailabs.com/
- http://www.TrustedBSD.org/
- Strict version requirements currently:
  - Jaguar 10.2.8 + Development Kit
- Source via CVSUP
  - Follow bootstrap_instructions.txt very carefully
  - Mach components very much in flux