Extensible Kernel Security through the TrustedBSD MAC Framework

Robert Watson, Research Scientist
HIP Group, McAfee Research
Introduction

- Rationale for Security Extensions
- TrustedBSD MAC Framework
- FLASK/TE with SEBSD
- Porting TrustedBSD MAC Framework to Darwin
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 SEBSF, 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
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
- Integration into Darwin/OS X in planning stages
Kernel MAC Framework

- User Process
- User Process
- User Process

System Call Interface
- VFS
- Socket IPC
- Process Signalling
- Pipe IPC

MAC Framework
- mac_biba
- mac_bsdextended

...
Policy Entry Point Invocation
Policy-Agnostic Labeling Abstraction

MAC Framework

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

10. OK

11. relabel

12. Destroy label

Mac_biba

1. Label-1
   - biba/low
   - jail
   - biba

2. Label-2
   - biba/low
   - jail
   - biba
   - biba/high
   - jail.a

3. Label-3
   - biba/low
   - jail
   - biba
   - biba/low
   - jail.b
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

GEOM Storage Framework

Interface Framework

Network Protocols

Socket IPC

SysV msgq, sem

Process/Thread Support

SysV SHM

Pipe IPC

UFS

... devfs, specfs

VFS

File Interface

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
  - Hierarchical 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

- NSA sponsored port of SELinux functionality to the FreeBSD platform
  - Port SELinux policy language and access control model
  - Implement FLASK/TE in a MAC Framework policy module
  - Provide result as open source
SELinux Background

- **FLASK security framework**
  - FLASK provides an access control framework abstraction
  - Initially integrated directly into Linux kernel
  - Now plugged in using “LSM” framework

- **Implements Type Enforcement (TE) Policy**
  - Extensive and comprehensive rule language and policy configuration
  - Mature policy documents privileges for many userspace system components and common applications
  - Also limited MLS, RBAC
SELinux FLASK Abstraction

- FLASK plays a similar role to the TrustedBSD MAC Framework
  - Treats existing system components as “object managers”
  - Abstracts notions of subjects, objects, and methods
  - Label storage using SIDs (Security Identifiers)
  - Differences from MAC Framework are substantial

- Access Vector Cache holds cached computation results for SID and method tuples

- “Security Server” security policy implementation
SELinux Type Enforcement

- Type Enforcement represents the set of permitted actions as rules in terms of:
  - Subjects (processes, generally) assigned domains
  - Objects (files, sockets, ...) assigned types
  - Object methods that may be performed on objects
  - Rules specifying permitted combinations of subject domains, object types, and object methods

- Labeling specification assigns initial labels
- New objects have labels computed from rules
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
Strategy: Migrate MAC Framework to Darwin, Port SEBSD as SEDarwin

- Exploit common source code and design roots of FreeBSD and Darwin
  - Migrate MAC Framework to Darwin
    - And dependencies, such as extended attributes, etc.
  - Migrate SEBSD, MLS, and other policies 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 SEBSD TE policy
Strategy: Migrate MAC Framework to Darwin, Port SEBSB as SEDarwin
Architecture of Darwin (Gross Over-Simplification)

- Mach provides low-level IPC, memory, synchronization primitives
- IOKit provides OO driver infrastructure
- BSD provides high level IPC, networking, storage services
Porting Activities So Far

- Port MAC Framework
  - Focus on getting base functionality up and running
  - Adapt to Mach memory allocation, synchronization
  - Port labeling, access control for devfs
  - Port UFS1 extended attributes to HFS+
  - Port support libraries (libextattr, libmac)
  - Port tools (mac_tools)
  - Adapt base Darwin tools (system_cmds, file_cmds)
  - Port mac_test module
  - Extend MAC Framework to incorporate Mach tasks, IPC
Porting Activities So Far (cont)

- Port SEBSD module
  - Port FLASK, AVC (largely synchronization, allocation issues)
  - Port “Security Server” for Type Enforcement (TE)
  - Port libsebsd
  - Port sebsd_cmds
  - Create minimal policy to get system up and running
  - Create GUI role selection, relabel tools
  - Hook up process, file system labeling and access control
  - Experiment with controls and policy for Mach primitives
Integration of MAC Framework into Darwin Prototype

Mach: Scheduler, VM, IPC

BSD Process/Thread Support

SysV SHM

SysV msgq, sem

HFS+, UFS

... VFS...devfs, specfs

File Interface

Mach/BSD System Call Layer

IOKit Device Driver Framework

Interface Framework

Network Protocols

Socket IPC

Process

Threaded Process

...
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
Conclusion

- TrustedBSD MAC Framework provides flexible, extensible OS access control
- SEBSD is experimental port of SELinux FLASK/TE to FreeBSD using MAC Framework
- Experimental port of MAC Framework to Darwin reveals opportunities, weaknesses in Darwin