EuroBSDCon 2004
October 31, 2004

TrustedBSD MAC Framework on FreeBSD and Darwin

Robert Watson

Principal Research Scientist

McAfee Research

Introduction

- TrustedBSD Project, McAfee Research
- Rationale for Security Extensions
- TrustedBSD MAC Framework
- Porting MAC Framework to Darwin

TrustedBSD Project

- Goal: Trusted system features to FreeBSD
  - Security Infrastructure Dependencies
    - UFS2 + extattr, OpenPAM, NSS, GEOM + GBDE, ...
    - Privilege and structural improvements
  - Security Features
    - Discretionary Access Control Lists (ACLs)
    - Extensible Access Control (MAC Framework)
    - Mandatory Access Control policies (MAC)
    - Security Event Auditing (Audit)
- Secondary Goal: Support organizations evaluating products based on FreeBSD

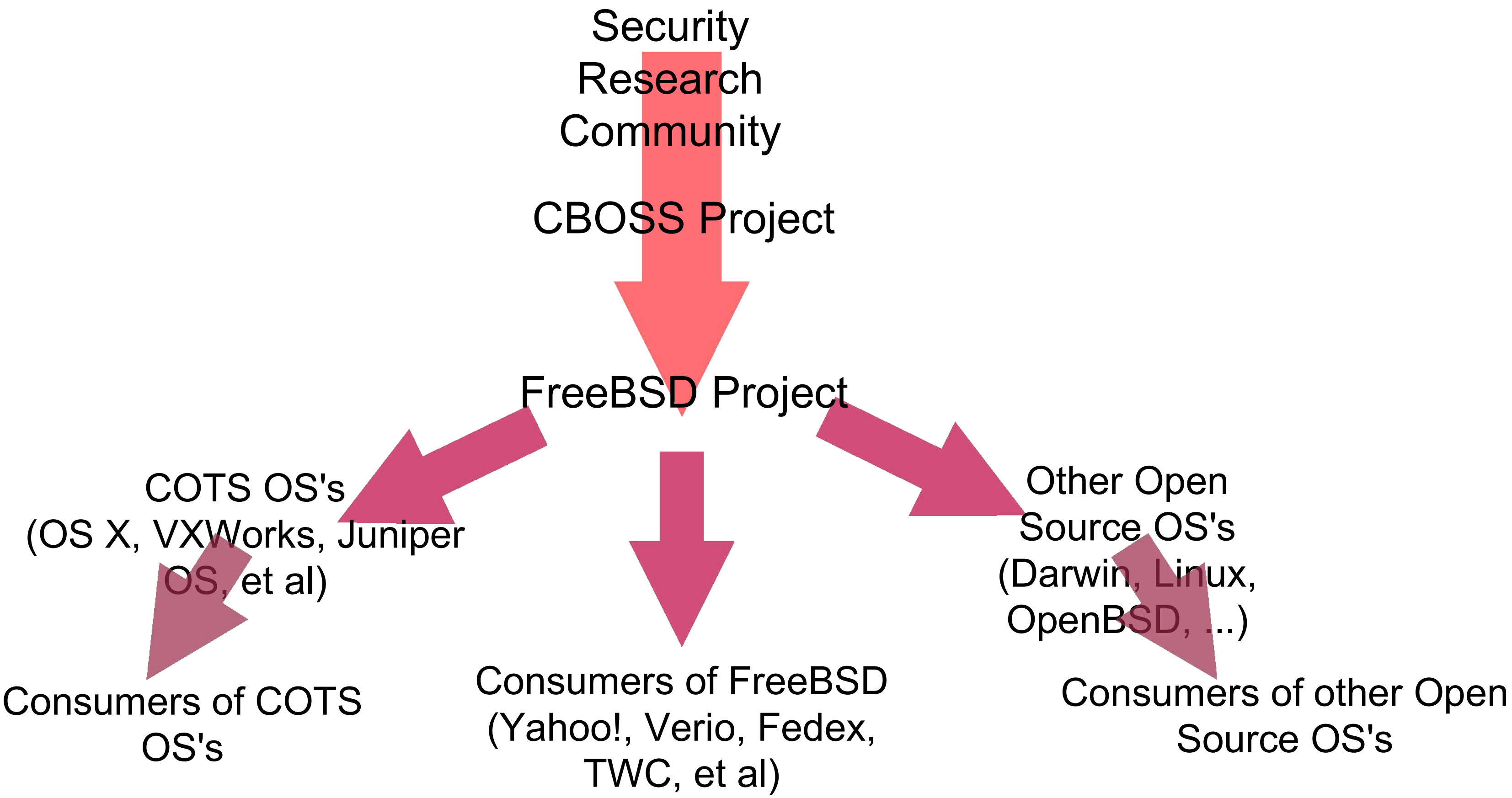
McAfee Research

- A leading commercial security R&D lab
  - Started out as Trusted Information Systems (TIS)
  - Along the way, TIS Labs, NAI Labs, Network Associates Laboratories, and now McAfee Research
  - Primarily R&D sponsored by US government agencies, such as DARPA, NSA, Army, Navy, DHS, VA, and others
  - Also security R&D under contract to commercial customers, such as Apple, Microsoft, and others
  - Defensive technology research into networks, operating systems, distributed systems, wireless, crypto, et al.
  - Very interested in, and supportive of open source

R&D in Operating System Security Extensibility and Hardening

- DARPA CHATS: CBOSS
  - Composable High Assurance Trusted Systems
  - Community-Based Open Source Security
  - Security extensibility, architecture, tech transfer
    - TrustedBSD MAC Framework
    - PRIVMAN – Library for privilege separation
  - Hardening and infrastructure
    - IPsec, UFS2, cryptographic storage, OpenPAM
  - Various follow-on relating to MAC Framework, FLASK/TE
    - Several other sponsors for follow-ons

CBOSS: Tech Transfer Flow



Security Research Community

CBOSS Project

FreeBSD Project

COTS OS's (OS X, VXWorks, Juniper OS, et al)

Other Open Source OS's (Darwin, Linux, OpenBSD, ...)

Consumers of COTS OS's

Consumers of FreeBSD (Yahoo!, Verio, Fedex, TWC, et al)

Consumers of other Open Source OS's

Outline

- TrustedBSD MAC Framework
  - Framework for operating system access control extension
- Security-Enhanced BSD Prototype
  - Port of NSA's FLASK/TE from SELinux to MAC Framework
- TrustedBSD MAC Framework port to Darwin
  - Apple's open source kernel for OS X operating system
- Security-Enhanced Darwin Prototype
  - SEBSD ported to run on Darwin using MAC Framework

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

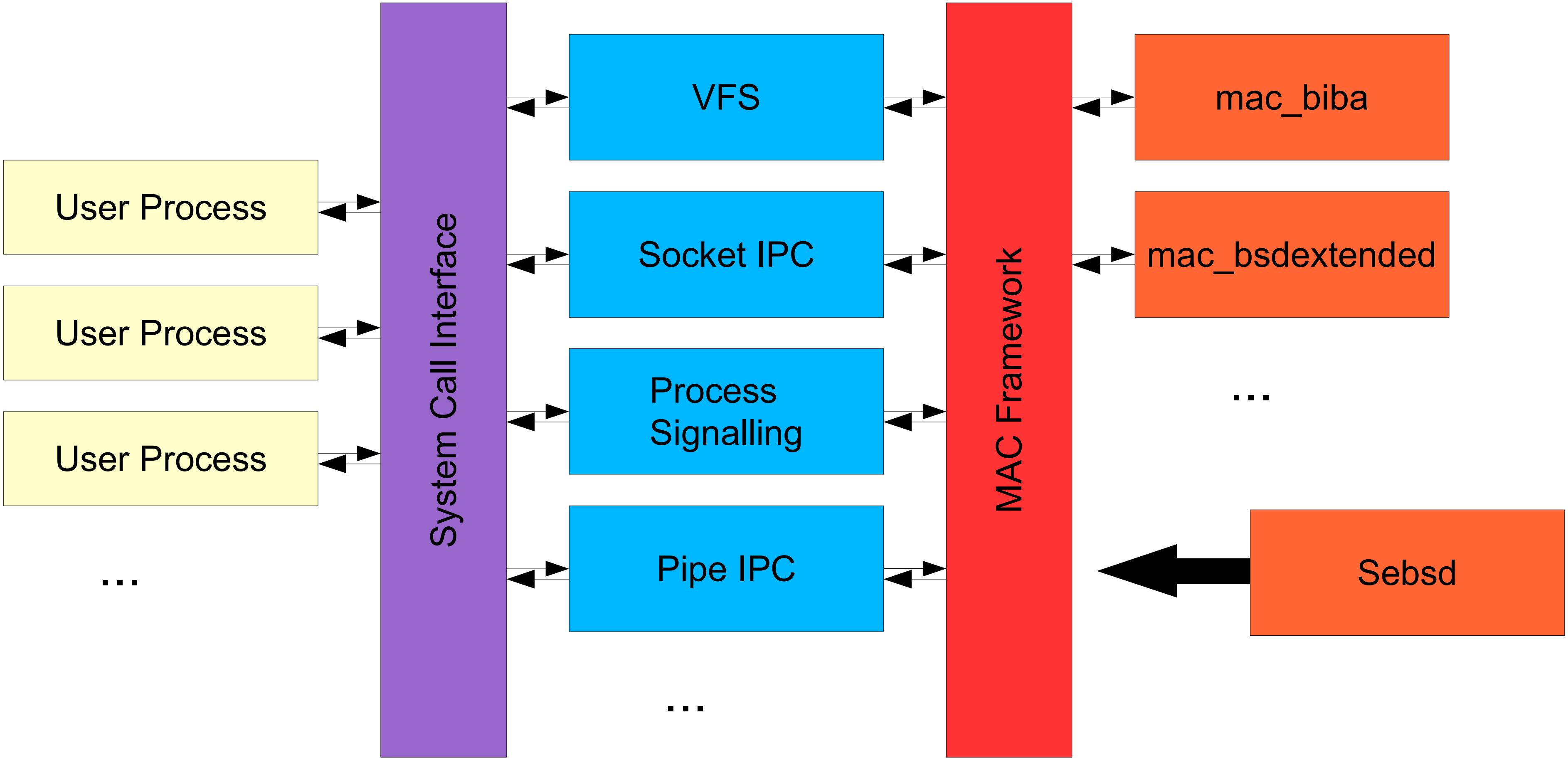
Why a MAC Framework?

- 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

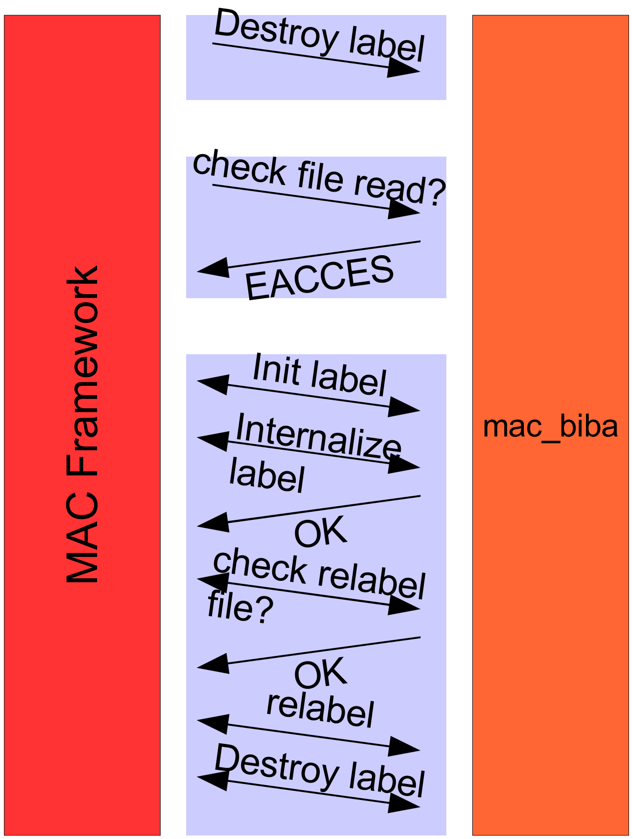
MAC Framework Background

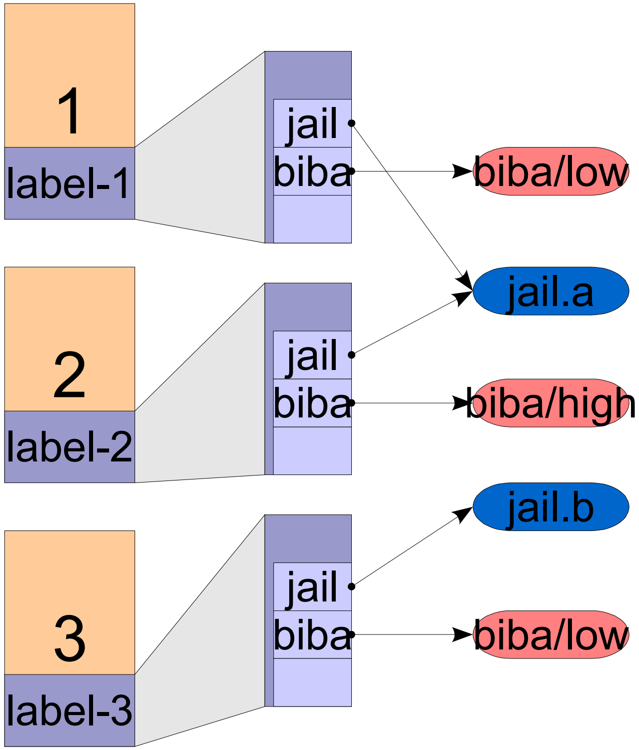
- Extensible security framework
  - Policies implemented as modules
  - Common policy infrastructure like labeling
  - Sample policy modules, such as Biba, MLS, TE, hardening policies, et al.
  - Composes multiple policies if present
  - Also provides APIs for label-aware and possibly policy-agnostic applications
- Shipped in FreeBSD 5.0-5.3
- Considered experimental, but quite usable

Kernel MAC Framework



Policy Entry Point Invocation
Policy-Agnostic Labeling Abstraction





TrustedBSD MAC Framework: Objects

- Broad range of system objects can be labeled and/or controlled
  - Subjects (processes/credentials, NFS clients, etc)
  - IPC (pipes, sockets, SysVIPC, POSIX semaphores)
  - Network objects (mbufs, interfaces, BPF queues, pcbs, IP fragment queues, IPSEC security associations)
  - File system objects (mountpoints, vnodes, devfs nodes, UFS inodes)
- Objects have life cycle entry points, access control check entry points

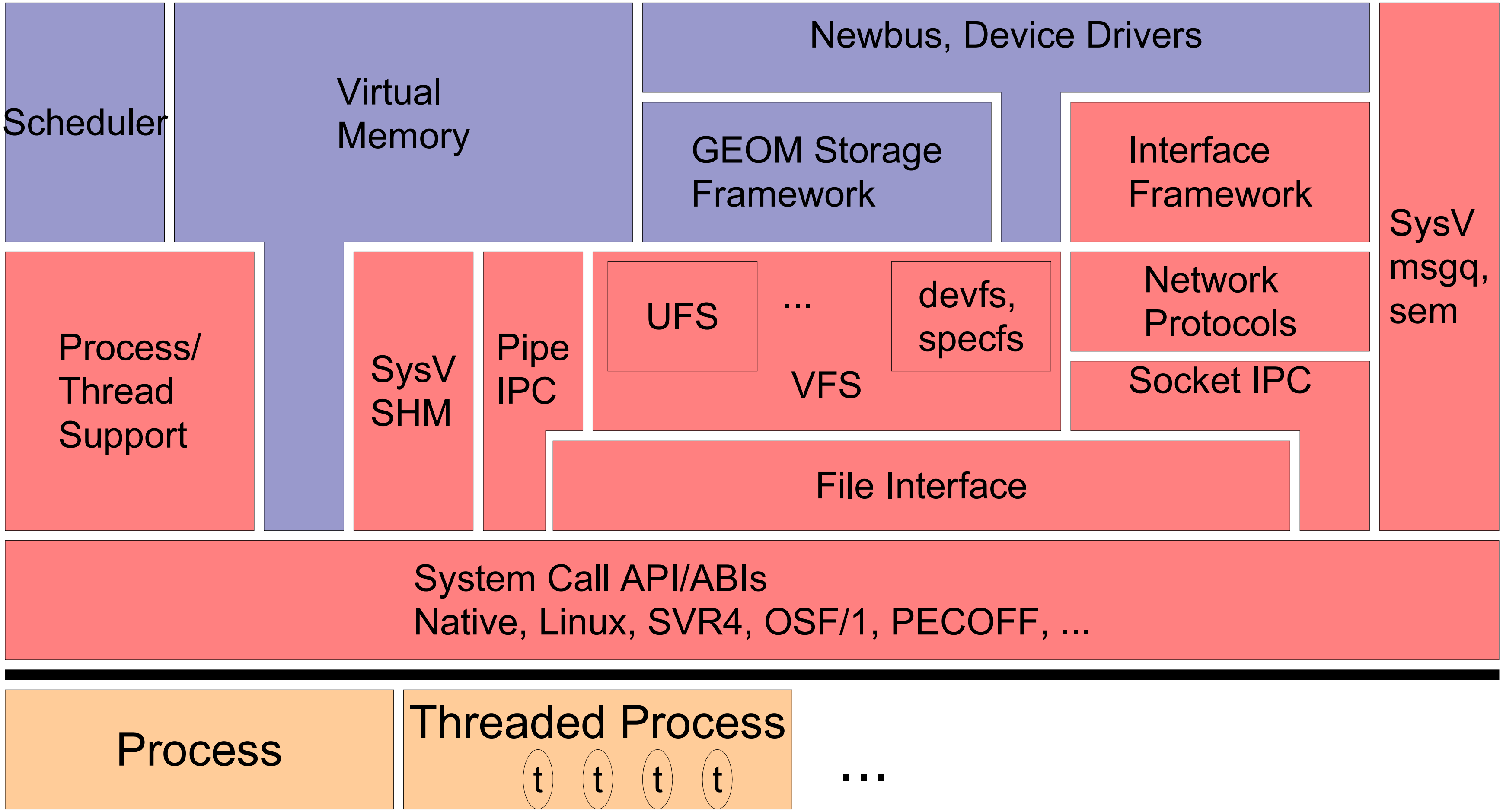
Modifications to FreeBSD to Introduce MAC Framework

- 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.

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
  - Hierarchal and compartmented Multi-Level Security (MLS)
  - SELinux FLASK/TE “SEBSD”
- Hardening policies
  - File system “firewall”
  - Interface silencing
  - Port ACLs
  - User partitions

Where next for the 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

SEBSD: Security-Enhanced BSD

- 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

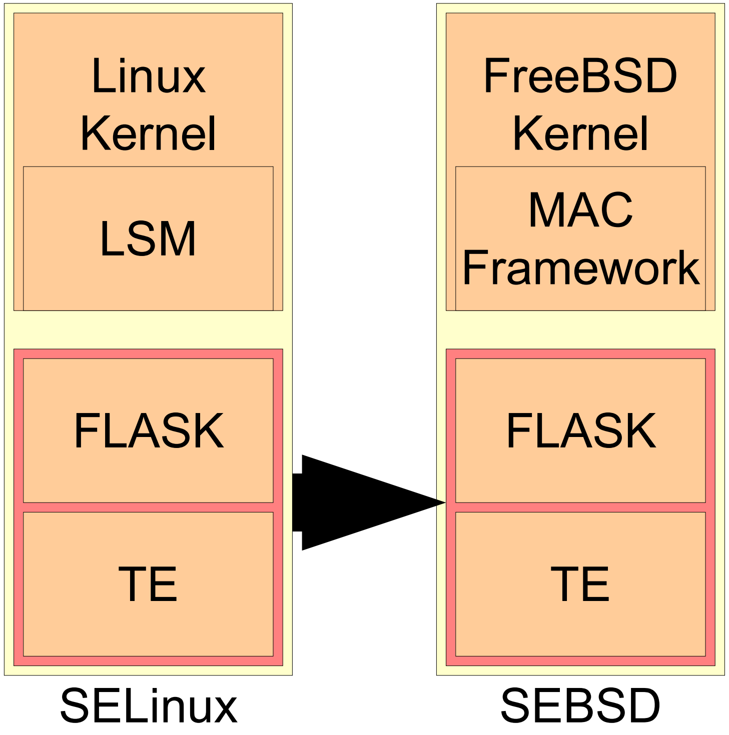
SELinux Type Enforcement

- 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

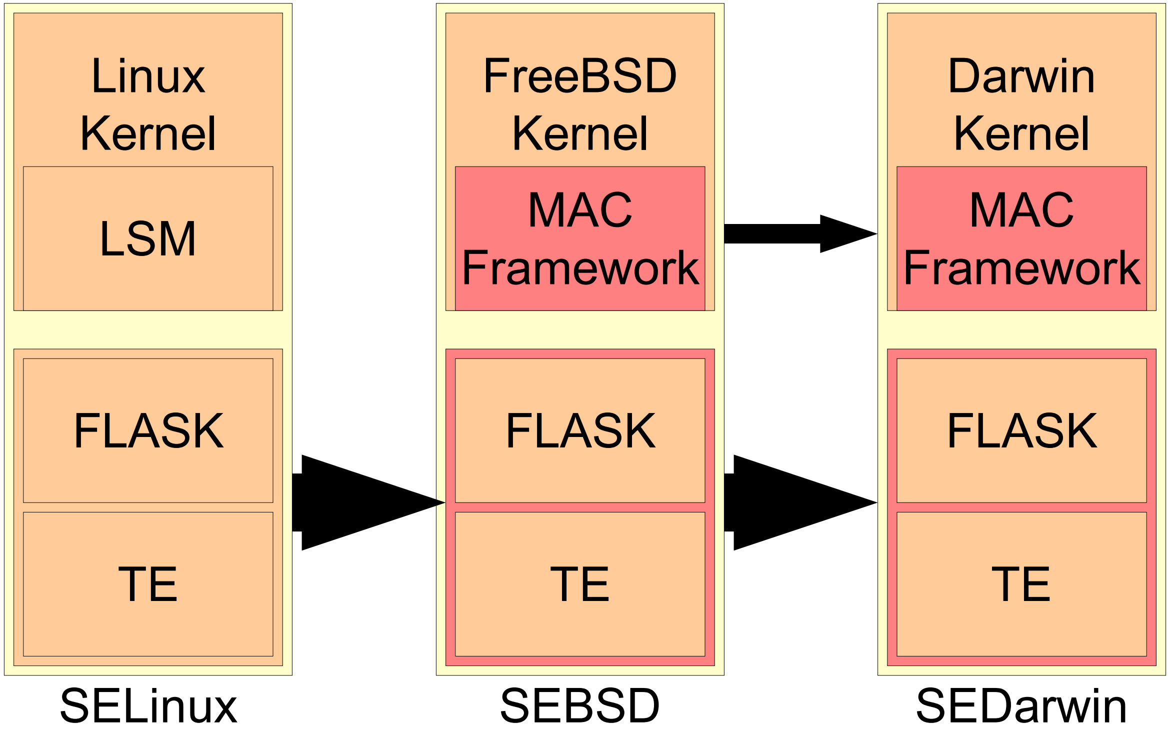
Current Status of SEBSD

- 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

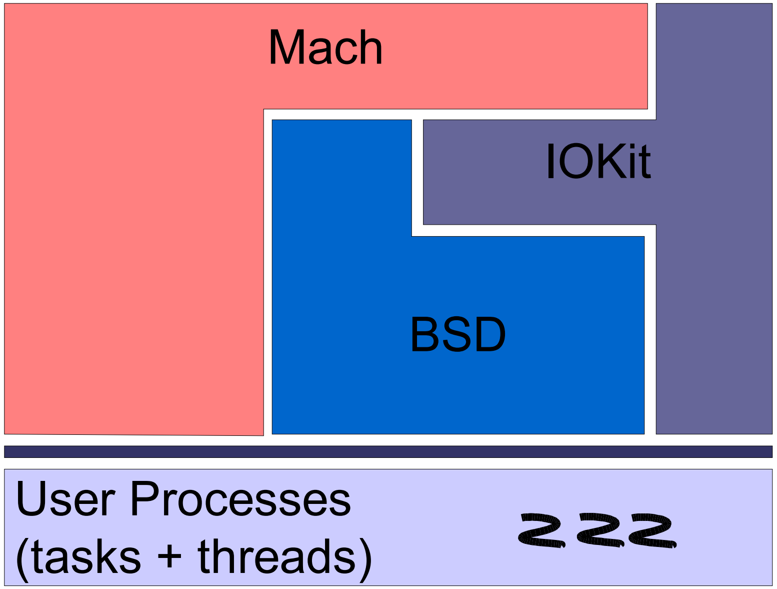
Strategy: Migrate MAC Framework to Darwin, Port SEBSD as SEDarwin

- 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 across various layers
  - Modify Darwin userspace applications
  - Produce adapted SEBSD TE policy

Strategy: Migrate MAC Framework to Darwin, Port SEBSD as SEDarwin (et al)



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

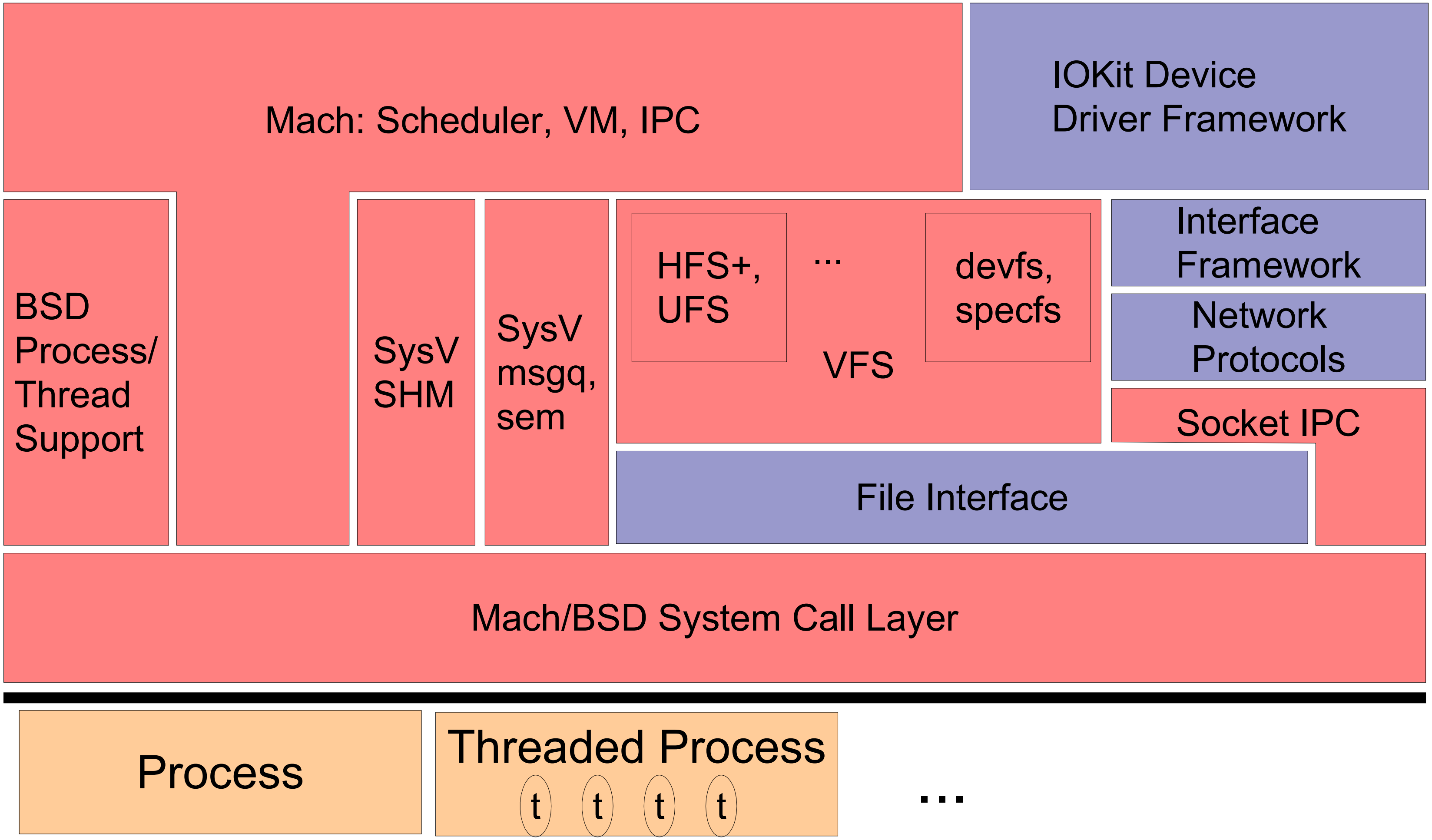
Two Generations of Port So Far

- Experimental prototype on Jaguar
  - Basic proof of concept:
    - Process, VFS labeling at BSD layer
    - Experimental work to explore Mach/BSD relationship
    - Experimental work to introduce Mach controls
- Forward-port to Panther
  - Bring port forward
  - Correct substantial omissions (features, rigor)
  - Move towards high levels of usability
  - Draw useful conclusions regarding Mach, etc.

Technical Elements of Port So Far

- Focus on getting base functionality running
  - Adapt to Mach memory allocation, synchronization
  - File system extensions, labeling, access control
  - Port support libraries/tools (libextattr, libmac, mac_tools)
  - Adapt base Darwin tools (system_cmds, file_cmds)
  - Port mac_test, mac_mls, SEBSD
  - Adapt to login environment differences
  - Extend MAC Framework to incorporate Mach tasks, IPC
  - Various other IPC work, such as sockets, System V IPC

Integration of MAC Framework into Darwin Prototype



On-going Darwin Work

- Working with various customers to improve productionability of system
- Porting additional FreeBSD elements over, such as network stack
- Integrating with Audit framework
  - Porting Audit to FreeBSD
- Developing test tools and environments
- Collaborating with Apple to identify base OS requirements to do future work of this sort

Some Impressions of Darwin

- A very interesting experience
  - Apple's use of FreeBSD greatly facilitates our work, but also general research/development on Mac OS X
  - Unique blend of Mach and BSD components offers opportunities and substantial challenges
    - Mach IPC used extensively: cannot be overlooked!
  - One of the biggest practical challenges was reproducing development environment outside Apple
  - Customers love “Local extensions + Microsoft Office”
  - Complexity of Mac OS X environment substantial

Impressions of Darwin (cont)

- Apple can be a strong partner in open source
  - Still figuring out aspects of how to be open source, though
  - Very open to requests for change and help
- ABIs present a serious issue
  - Work towards kernel ABI/API stability via documented promises and resilient approaches will be important

Conclusion

- A lot of exciting work going on
  - TrustedBSD has brought many features to FreeBSD
    - Many more to follow, including Audit, more MAC support
  - Port to Darwin offers both opportunities for research and substantial benefits for FreeBSD/TrustedBSD work
  - SEBSD/SEDarwin bring experimental SELinux FLASK/TE functionality to FreeBSD and Darwin
  - Successful transfer to *BSD, Darwin, Linux, !open source
- More information
  - http://www.TrustedBSD.org/