



# **Security-Enhanced Darwin: Porting SELinux to Mac OS X**

**SELinux Symposium 2007**

**Chris Vance**

**Information Systems Security Operation,  
SPARTA, Inc.**



# Results

- **Recall my similar talk last year. What's new?**
  - This year, work backwards - results now, discussion follows
- **MAC Framework will likely be included in Leopard**
  - There were many performance “issues” that needed to be addressed
  - Hopefully intact
  - Took many iterations of re-engineering and maturing various Framework elements
- **Completed coverage of IOKit, devfs, network stack, Mach IPC**
- **SELinux tools now updated regularly**
- **Policy rules still take time to develop**
- **After a brief explanation of the technologies involved, will talk about a few specific technical issues this year**

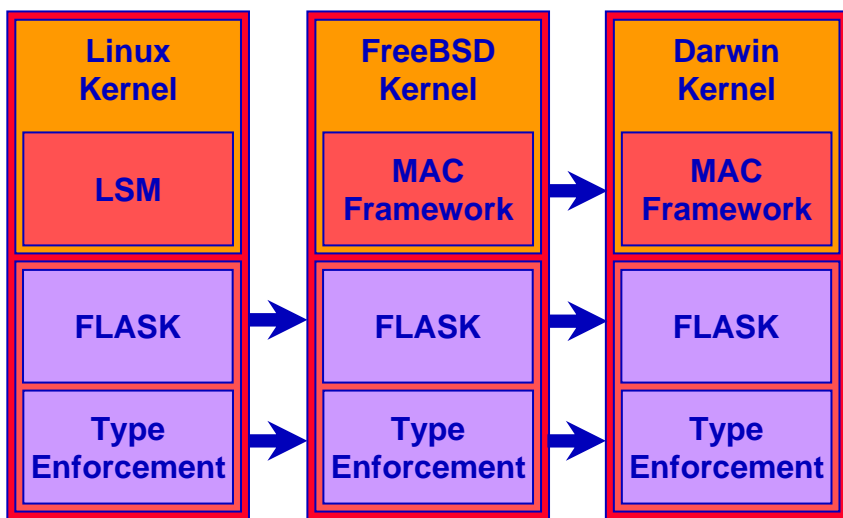


# The Parts Bin

- **Mac OS X as a starting point**
  - Kernel released with Open Source license
  - Kernel audit support
  - Prior version evaluated under CAPP/EAL3
- **BSD code heritage (both user space and kernel)**
  - We understood FreeBSD, Mac OS X was new
  - TrustedBSD MAC Framework from FreeBSD
- **Apple relationship**
- **LSM Framework from Linux for comparison**
- **SELinux provides mature access control**
- **Mach access control research results from DTOS**
- **Kernel debugger and serial console support (aka trial and error)**



# The Plan



- Use MAC Framework to isolate policy from enforcement
- Build on Darwin's source code and structural similarities to FreeBSD
- Port FLASK components from SELinux
- Expand scope for Darwin-specific functionality (Mach IPC, Iokit, etc.)
- Minimize Vendor diffs (OS & SELinux)
- Leverage existing policy & tools
- Aim for near zero performance cost (with serious caveats..)

**Strong, useful security without sacrificing features, performance, or utility**



# Mac OS X

- **Mac OS X is Apple's next generation operating system**
  - Builds on elements of Mach, NeXTStep, FreeBSD, and Mac OS 9, as well as other open source elements such as KDE
  - Continues Apple's tradition of user interface innovation
- **Leopard expected “Spring 2007”**
- **Very user-centric experience**
- **Good support for office application suites, programs people are familiar with, as well as traditional UNIX services**
- **Good virtualization support via VMWare, Parallels**

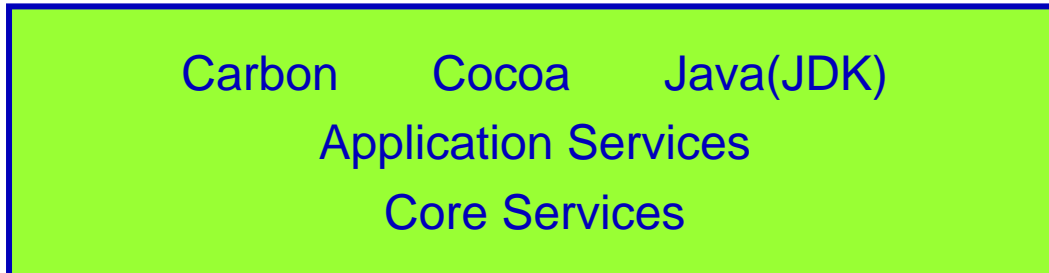


# Mac OS X System Architecture

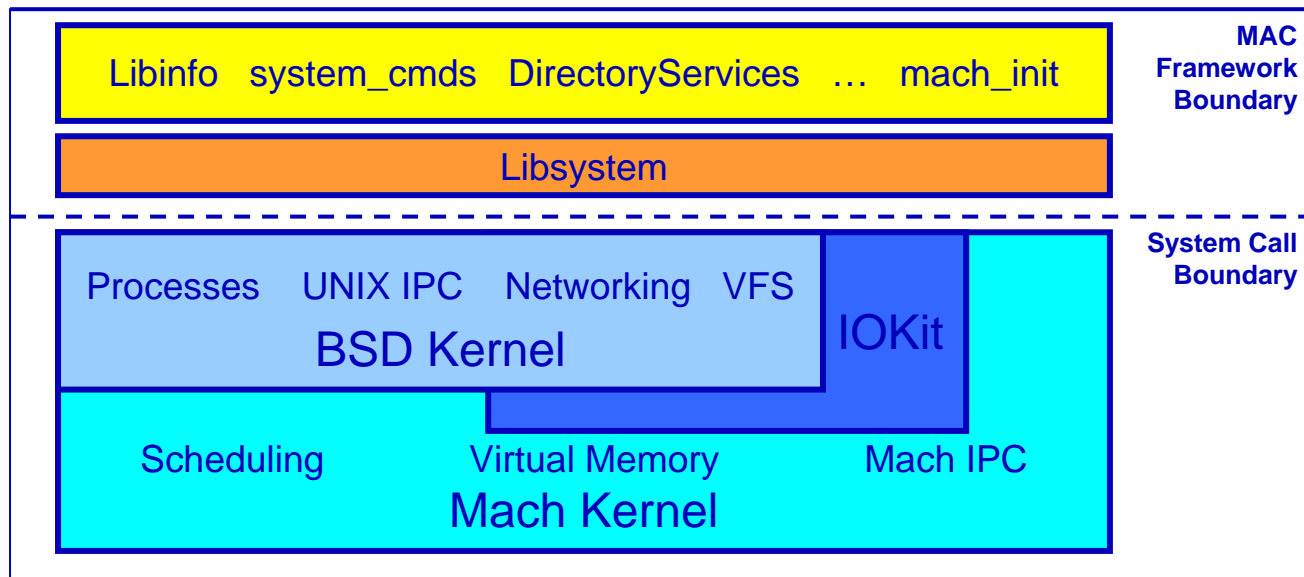
Applications



Closed source frameworks and daemons



Open source libraries and daemons



XNU Kernel



# Unique To Darwin

- **Rich (GUI) applications, desktop integration**
  - Provides motivation to use the system
  - Provides more challenges due to complexity
  - Inter-application messaging is ubiquitous
  - Many closed-source components
- **IOKit object oriented device driver framework**
- **Mach IPC**
  - Critical to secure
  - Performance/efficiency concerns
  - Didn't have to start from scratch
  - Explore DTOS protections for Mach IPC



# Darwin Complexity

- **Three separate system boundaries (IOKit, Mach, BSD) and each one must be adequately secured!**
- **Mach isn't implemented as a microkernel, there is a blending of the lines between BSD and Mach services**
  - BSD is in the kernel address space, not user
  - Threads and the scheduler are Mach constructs while processes are a BSD construct
  - Even worse, virtual memory is shared amongst all three kernel subsystems
- **History showed that the complexity of the Mach microkernel led from DTOS to FLASK**
  - Mach uses lots of opaque pointers and structures, can't poke in like you can on Linux/FreeBSD
  - It's no less complex than it was
  - Yet here we are trying to secure Mach IPC again...

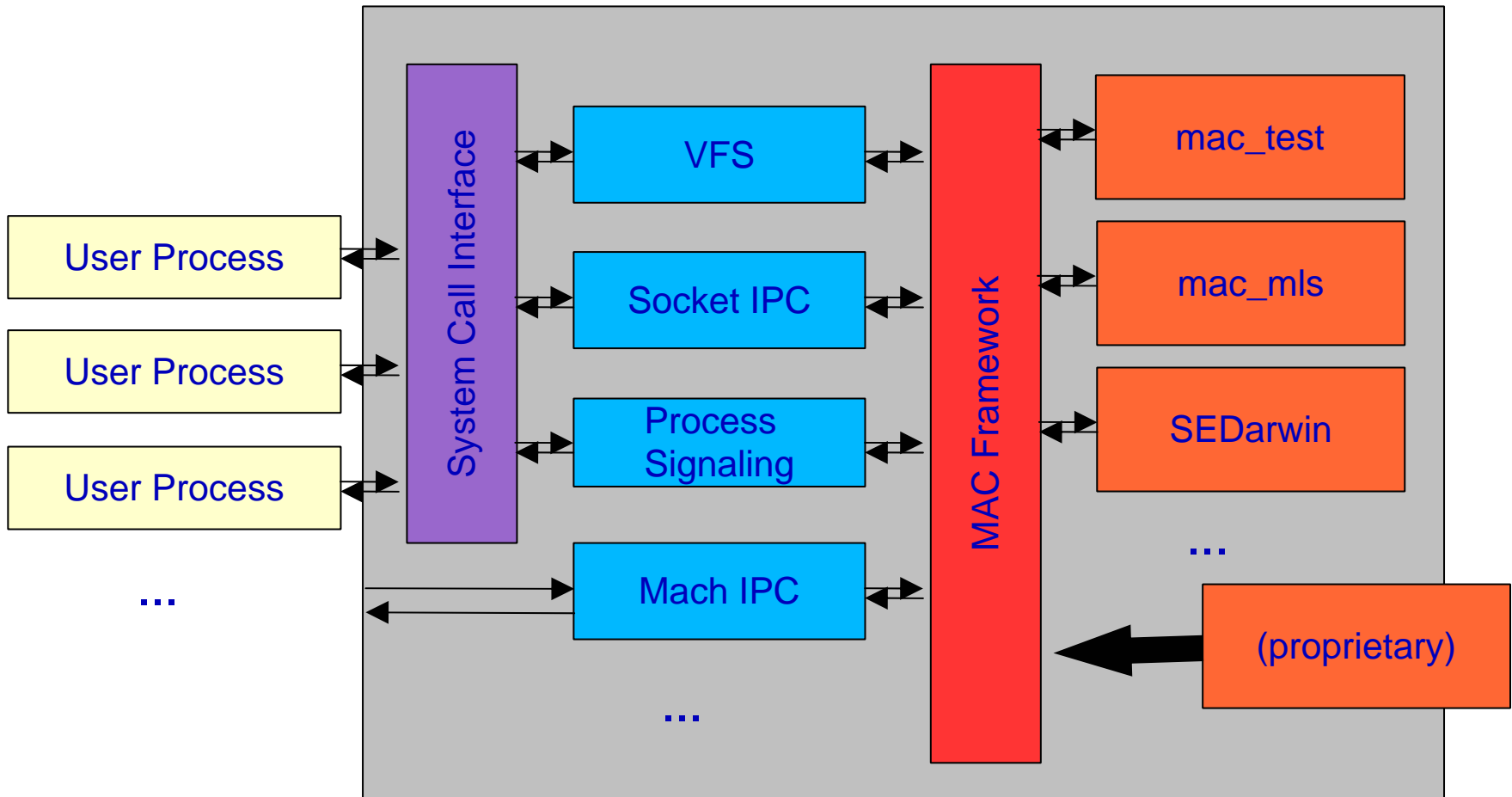




# Security Frameworks

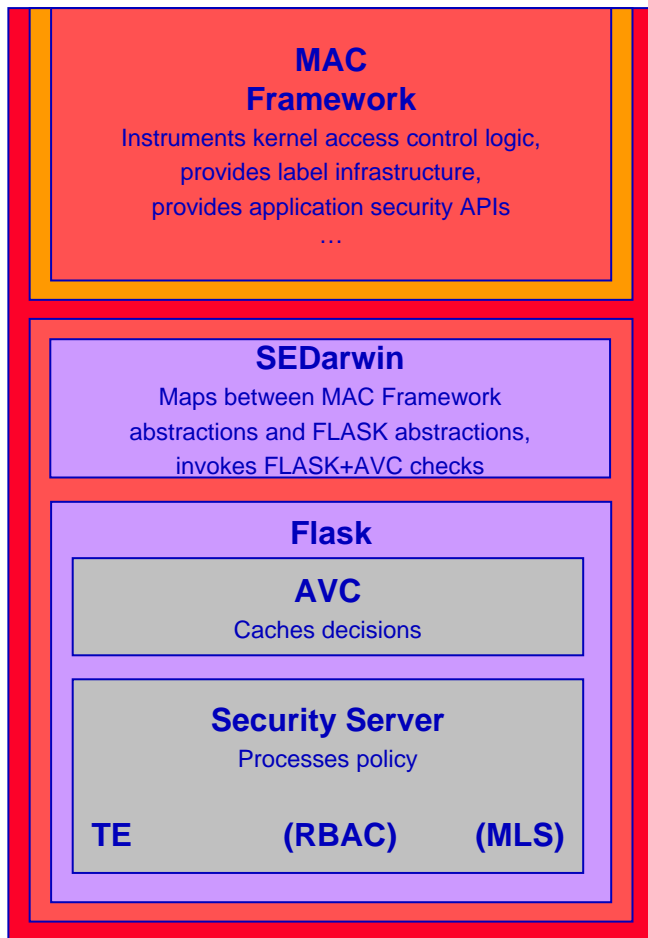
- **We all agree - traditional UNIX security isn't enough**
- **Tight OS integration required for new security services**
  - Frameworks are key to vendor buy-in (FreeBSD, Linux, Apple)
  - Want vendors to support Framework, costs of locally maintaining security extensions are high
  - Framework offers extensibility so that policies may be enhanced without changing the base operating system
  - “Stable” user space APIs more critical than kernel changes
- **Frameworks are tailored to vendor requirements**
  - LSM is lightweight
  - FreeBSD MAC heavier, supports composition, user space policy-agnostic label management
  - Darwin MAC less intrusive code, low performance overhead when not in use (policy provides per process/subsystem masks)
- **Bottom Line: Frameworks for Linux, FreeBSD, Darwin**

# MAC Framework Big Picture





# SEDarwin Policy Module

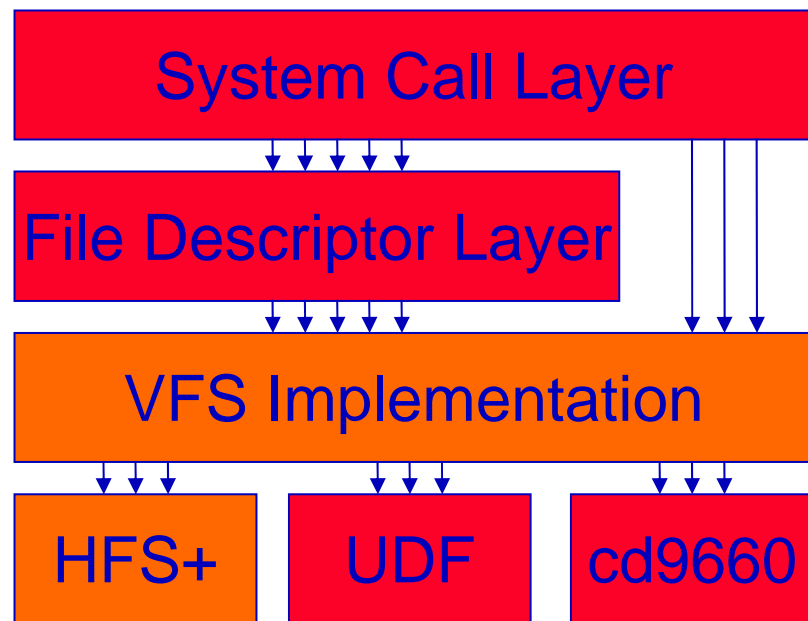


- **Kernel components ported easily**
  - “normal” issues with allocators, locking primitives (no RCU locks), logging, printf, audit, etc.
- **Likewise, user space tools ported easily, rule parsers and compilers**
- **Added a thin compatibility layer to translate sysctls vs. selinuxfs**
- **Policy binary format unchanged**
- **Everything updated regularly (within last couple weeks)**
- **Started with Tresys Reference Policy, still working to develop better rules**



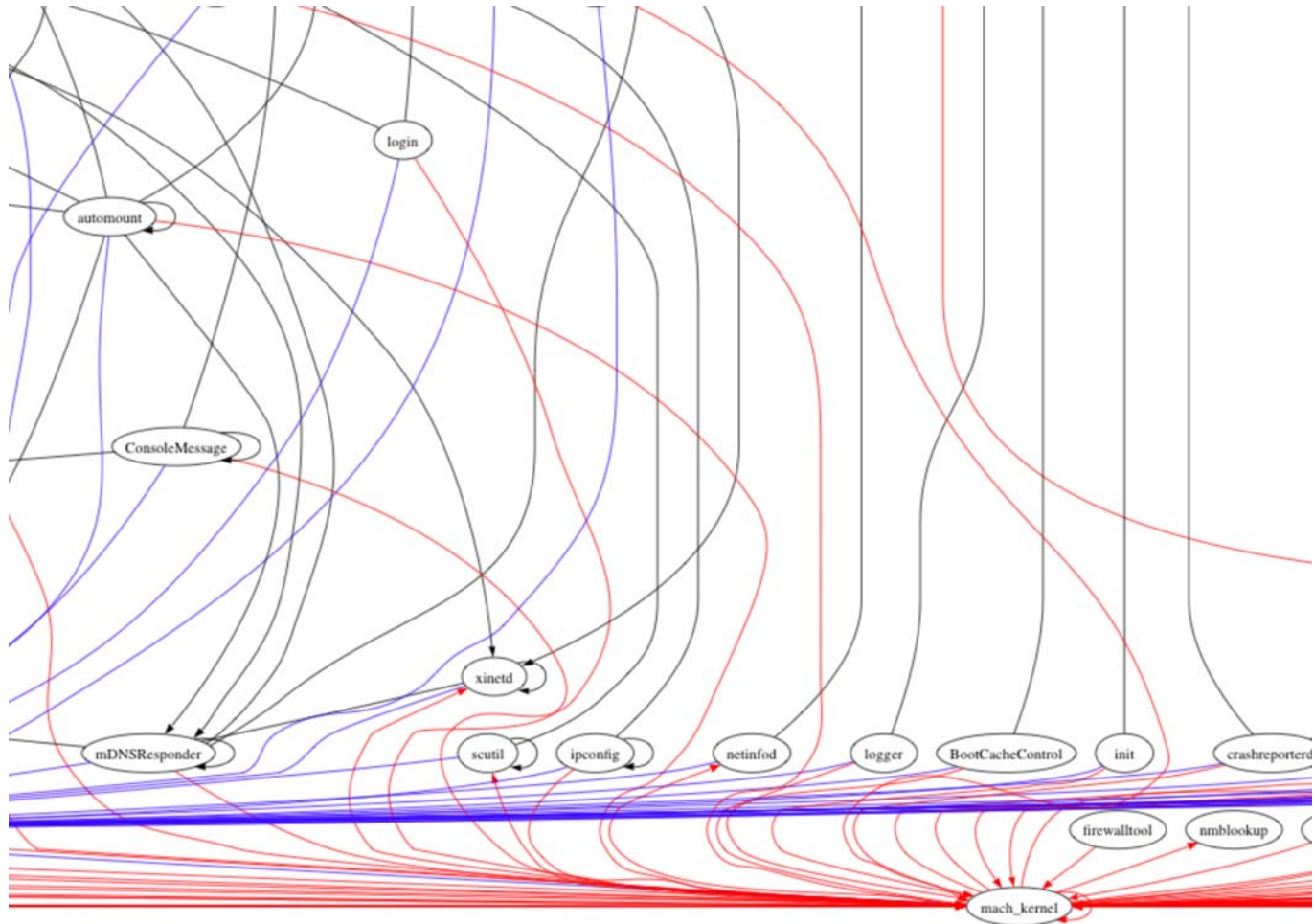
# Securing BSD

- Applied same strategy for most BSD kernel subsystems (network, fd, sysv/posix IPC, etc.)
- Straightforward?
  - Add access control on all access paths
  - Avoid layering and locking violations from vendor code
- Consider performance and encapsulation
- Example: put all access control in VFS
  - Hardly any file system specific code (HFS+, DEVFS, NFS, AFP, etc.)
  - Darwin locking, refcounts, caching made this tricky
  - Don't have sources for all file systems





# Securing Mach IPC





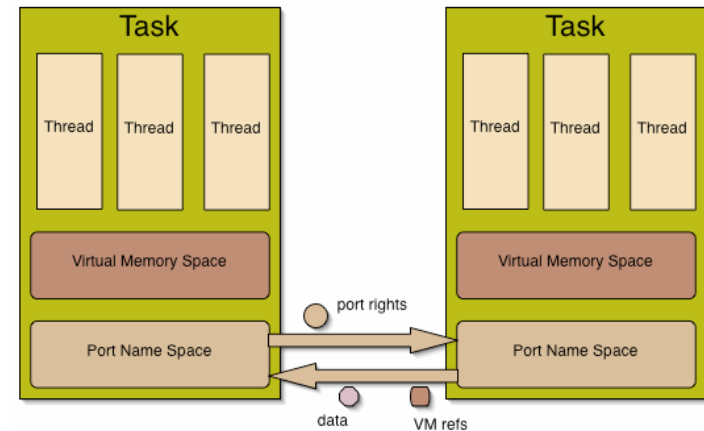
# Introduction to Mach IPC

- **Mach IPC is the primary messaging service on Darwin**
  - Messages are sent to ports
  - Tasks/processes have ports
  - The kernel has ports
- **Ports are unidirectional**
  - Ports have one receiver, many senders
  - Use in pairs for bi-directional communication
- **Messages are structured**
  - vs. socket data streams



# Mach Port Rights

- Ports have “rights” associated with them
- If you have the right, you can do what it represents
  - Capability model
  - All or none, message types, content, meaning are ignored
- **Two types:**
  - send/receive msgs
  - transfer rights
- **Port rights can be transferred in messages**
- **Root can request task ports (task\_for\_pid)**
  - kill/suspend process, create threads, read/write memory, etc.





# Securing Mach IPC

- **Messages shouldn't flow freely between processes and each other, or to and from the kernel**
- **Fine-grained in kernel access control**
  - Add Mach per-method access permissions
  - Reduce root privilege
- **Provide support for security-aware user space Mach services**
- **Example service: bootstrap nameserver**
  - Lets you lookup services by name
  - Holds send rights for services, passes them out to processes
  - Need to control who gets send rights
  - Need better control over registration to avoid spoofing





# Mach Kernel Access Controls

- **Add object labels and manage creation, deallocation**
- **Verify per-method access permissions at object usage**
- **Synchronize Mach Tasks/Threads labels with BSD Process labels**
- **Add new Mach server to handle label operations and provide generic access checks**
  - Similar to interfaces provided with selinuxfs on Linux
- **Modify SELinux policy to add port label support**

```
class mach_port {  
    relabelfrom  
    relabelto  
    send  
    recv  
    make_send  
    make_send_once  
    copy_send  
    move_send  
    move_send_once  
    move_recv  
    hold_send  
    hold_send_once  
    hold_recv  
}
```

```
allow kexd_t self:mach_port { copy_send make_send_once send };  
allow kexd_t coreservicesd_t:mach_port hold_send;  
allow kexd_t init_t:mig_bootstrap { bootstrap_look_up bootstrap_register  
    bootstrap_status };
```

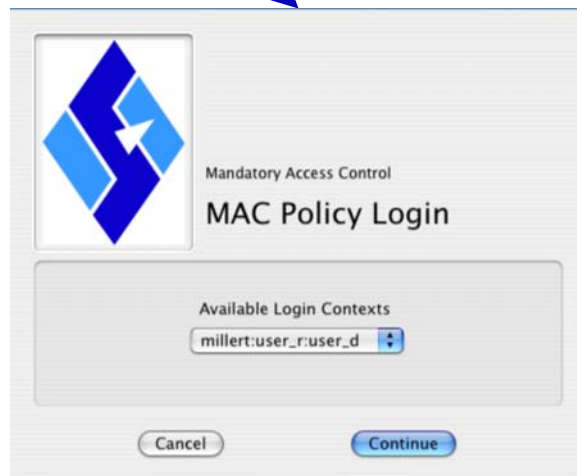


# Application Interfaces

- **Need to communicate Mach labels with user space**
- **The MAC Framework's policy-agnostic design complicated Mach label support**
  - Multiple policies, no fixed label formats
  - MAC Framework internalizes/externalizes labels as strings (too expensive for Mach IPC)
  - Can't simply attach a SID like DTOS did
- **Associate abstract "label handle" with messages**
  - Implemented as a Mach port
  - Basically a reference (ala file descriptor) to an in-kernel label
- **Provide services to get/set labels from the handle**
- **Provide the label handles to use space upon request**
- **Leverage Mach Interface Generator (MiG) to simplify request/verification/deallocation of label handles**
  - protocol generator, like rpcgen

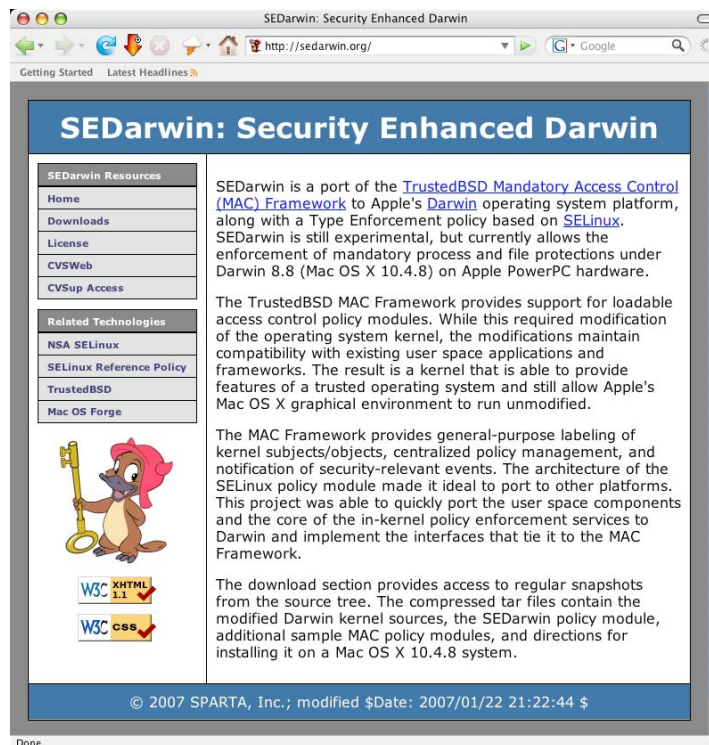


# Login



# SEDarwin: For More Information

- **Email:**
  - <Christopher.Vance at SPARTA.com>
  - TrustedBSD Mailing lists  
<http://lists.freebsd.org/mailman/listinfo/trustedbsd-discuss>
- **Web:**
  - <http://sedarwin.org/>
  - <http://trustedbsd.org/>



The screenshot shows a web browser window titled "SEDarwin: Security Enhanced Darwin" with the URL "http://sedarwin.org/". The page content includes:

- SEDarwin Resources:** Home, Downloads, License, CVSWeb, CVSup Access.
- Related Technologies:** NSA SELinux, SELinux Reference Policy, TrustedBSD, Mac OS Forge.
- Introduction:** SEDarwin is a port of the TrustedBSD Mandatory Access Control (MAC) Framework to Apple's Darwin operating system platform, along with a Type Enforcement policy based on SELinux. SEDarwin is still experimental, but currently allows the enforcement of mandatory process and file protections under Darwin 8.8 (Mac OS X 10.4.8) on Apple PowerPC hardware.
- TrustedBSD MAC Framework:** Provides support for loadable access control policy modules. While this required modification of the operating system kernel, the modifications maintain compatibility with existing user space applications and frameworks. The result is a kernel that is able to provide features of a trusted operating system and still allow Apple's Mac OS X graphical environment to run unmodified.
- MAC Framework:** Provides general-purpose labeling of kernel subjects/objects, centralized policy management, and notification of security-relevant events. The architecture of the SELinux policy module made it ideal to port to other platforms. This project was able to quickly port the user space components and the core of the in-kernel policy enforcement services to Darwin and implement the interfaces that tie it to the MAC Framework.
- Downloads:** The download section provides access to regular snapshots from the source tree. The compressed tar files contain the modified Darwin kernel sources, the SEDarwin policy module, additional sample MAC policy modules, and directions for installing it on a Mac OS X 10.4.8 system.

At the bottom of the page, there is a copyright notice: "© 2007 SPARTA, Inc.; modified \$Date: 2007/01/22 21:22:44 \$".



**End.**

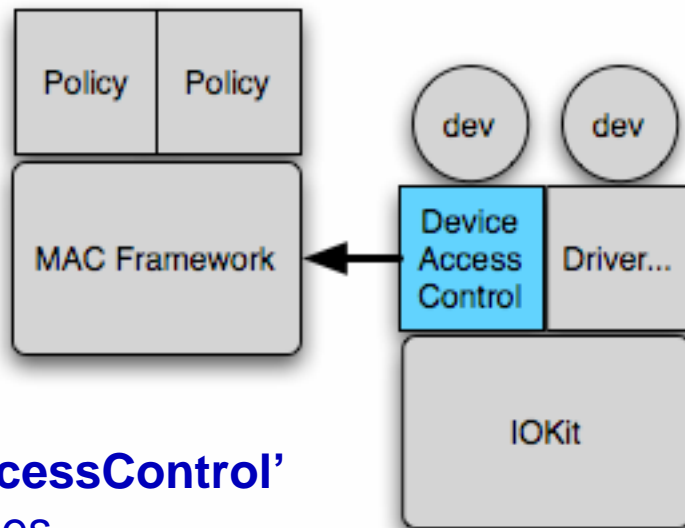


# Securing IOKit

**A pictorial example of securing a subsystem**



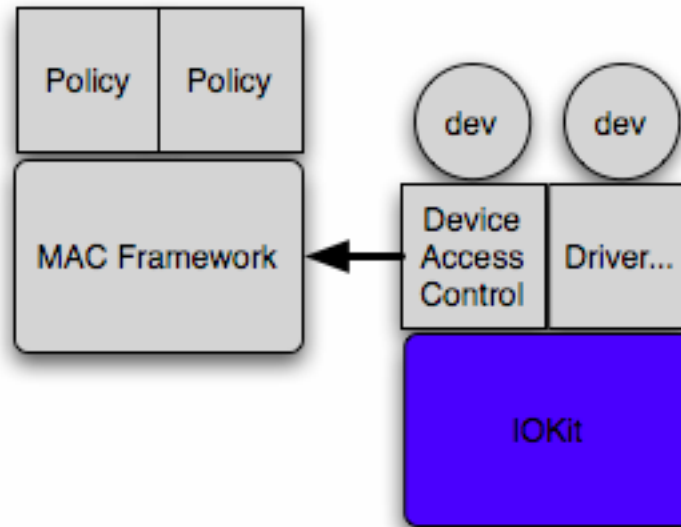
# IOKit Operation



- **Special IOKit driver ‘DeviceAccessControl’**
  - USB and Firewire personalities
  - Special plist key ‘IOKitForceMatch’
  - Store device metadata in a dictionary
- **Ask Policy to decide if device permitted**
- **Attach “dummy” driver to devices that are not permitted, making them unavailable to normal matching code**
- **Allow permitted devices to be probed, matched, attached as normal**



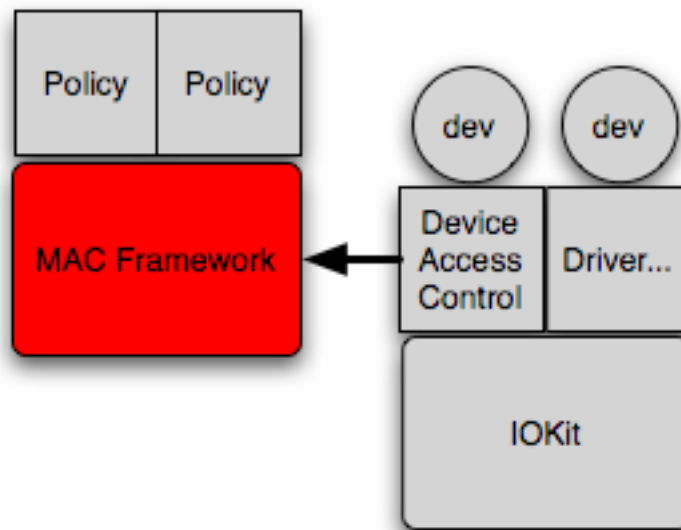
# IOKit Operation



- **Modification to IOKit's priority matching code. Short circuit election when 'IOKitForceMatch' is present in driver.**



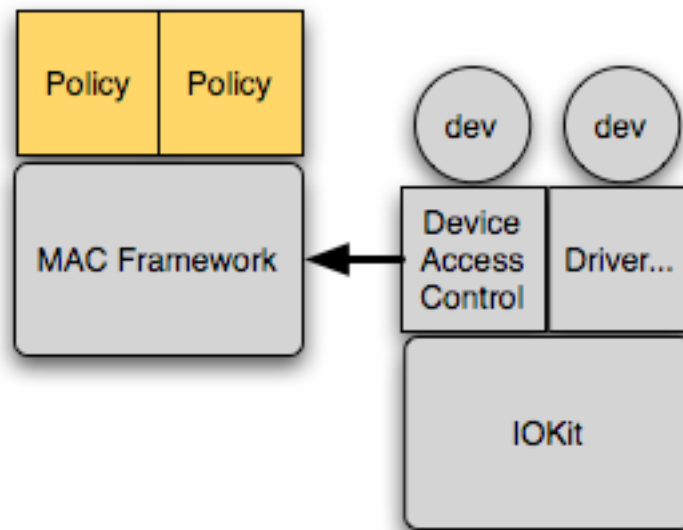
# IOKit Operation



- **Framework entrypoint**

- `int mac_iokit_check_device(int devtype, struct module_data *mdata)`

# IOKit Operation



- **Policy entrypoint**

- `typedef int mpo_iokit_check_device_t(int devtype, struct mac_module_data *mdata);`