NSA Security-Enhanced Linux (SELinux)

http://www.nsa.gov/selinux

Stephen Smalley

sds@epoch.ncsc.mil

Information Assurance Research Group

National Security Agency

Operating System Security

- Why secure the OS?
  - Increasing risk to valuable information
  - Information attacks don't require a corrupt user
  - Applications can be circumvented
  - Must process in the clear
  - Network too far/Hardware too close
- Key missing feature: Mandatory Access Control (MAC)
  - Administratively-set security policy
  - Control over all processes and objects
  - Decisions based on all security-relevant information

Why is DAC inadequate?

- Decisions are only based on user identity and ownership
- No protection against malicious or flawed software
- Each user has complete discretion over his objects
- Only two major categories of users: administrator and other
- Many system services and privileged programs must run with coarse-grained privileges or even full administrator access.

What can MAC offer?

- Strong separation of security domains
  - Separate data based on confidentiality/integrity/purpose
- System, application, and data integrity
  - Protect against unauthorized modifications
  - Prevent ill-formed modifications
- Ability to limit program privileges
  - Safely run code of uncertain trustworthiness
  - Prevent exploit of flaw in program from escalating privilege
  - Limit each program to only what is required for its purpose

What can MAC offer?

- Processing pipeline guarantees
  - Ensure that data is processed as required
  - Split processing into small, minimally trusted stages
  - Encryption, sanitization, virus scanning
- Authorization limits for legitimate users
  - Decompose administrator role
  - Partition users into classes based on position, clearance, etc.

MAC Implementation Issues

- Must overcome limitations of traditional implementations
  - More than just Multilevel Security / BLP
  - Address integrity, least privilege, separation of duty issues
  - Complete control using all security-relevant information
- Policy flexibility required
  - One size does not fit all!
  - Ability to change the model of security
  - Ability to express different policies within given model
  - Separation of policy from enforcement
- Maximize security transparency

SELinux provides Flexible MAC

- Flexible comprehensive mandatory access controls integrated into the Linux kernel
- Building on 10 years of NSA's OS Security research
- Application of NSA's Flask security architecture
  - Cleanly separates policy from enforcement using well-defined policy interfaces
  - Allows users to express policies naturally and supports changes
  - Fine-grained controls over kernel services
  - Transparent to applications and users
- Role-Based Access Control, Type Enforcement, optional Multi-Level Security, easily extensible to other models
- Highly configurable

Current Directions

- Transfer to mainline Linux 2.5/2.6 kernel
  - General security framework/hooks (LSM) already merged
  - Reworked SELinux APIs and implementation for merging
  - SELinux module in 2.6.0-test1-mm series
- Kernel Integration Issues
  - API
  - File labeling
  - Initialization
  - Network access controls
  - Coding style / code cleanup

API Changes

- Motivation: Removal of sys_security from 2.5.
  - Required reworking SELinux API to meet kernel developers' criteria.
- SELinux API refactored into three components:
  - Add /proc/pid/attr API for process attributes (in 2.5).
  - Re-use existing xattr API for file attributes (in 2.5).
  - Add selinuxfs pseudo filesystem for security policy API.
  - Support for SELinux extensions for System V IPC and socket IPC to be reinvestigated in the future.
- libselinux encapsulates all three components.

API Changes

- Pass contexts, not SIDs.
- Set-attribute calls instead of extended calls:
  - execve_secure() => setexeccon();execve();
  - open/mkdir_secure() => setfscreatecon();open/mkdir();
  - Implemented via writes to /proc/self/attr/{exec,fscreate}.
  - Cleared explicitly by program or automatically upon exec.
  - Simplifies common case, but requires extra care for:
    - Multi-threaded applications (if not 1-to-1 user-to-kernel).
    - Signal handlers that call execve() or open/mkdir().

API Changes

- Explicit API for obtaining process contexts
  - No longer stat_secure on /proc/pid inodes
  - getcon(), getprevcon(), getfscreatecon(),getexeccon()
  - getpidcon() for other processes
  - Implemented via reads of /proc/pid/attr/*
- File context API layered on top of xattr API
  - [gs]etfilecon, l[gs]etfilecon, f[gs]etfilecon
  - Hides xattr name, handles allocation of context buffers

API Changes

- Security Policy API layered on top of selinuxfs
  - Selinuxfs modeled after 2.5 nfsd, transaction based IO.
  - Removed calls for converting between SIDs and contexts.
  - Added security_check_context.
  - Changed security_load_policy to take (data,size) pair.
  - Renamed calls to reflect elimination of SIDs, clarify meaning, and provide consistency in naming.

File Labeling Changes

- Motivation: Re-use xattr API and support included in 2.5.
- Reworked LSM hooks and added xattr handlers to support use of xattr by security modules (in 2.5).
- Changed SELinux to use xattr when available.
- Added hooks and devpts xattr handler to support setting security labels on ptys (in 2.5).
- Added hook to support /proc/pid inode security labeling based on associated task (in 2.5).

Initialization Changes

- Early initialization for security modules.
  - Required for SELinux to set up security state for all kernel objects.
  - Replaced SELinux-specific patch with a security initcall patch created for LSM by Chris Wright of WireX.
- Initial policy load
  - Reworked API to move initial policy load to userspace.
  - Presently performed via an initrd, may migrate to initramfs.
  - Set up existing superblocks and inodes after initial load.

Network Access Control Changes

- Motivation: Many of the LSM network security fields and hooks rejected for 2.5.
- Retained general socket layer hooks and Unix domain socket hooks.
- Reworking sock_rcv_skb hook and NetFilter hooks to provide subset of original SELinux functionality.
- Revisiting set of network access controls based on experience to date.

Coding style cleanups

- Linux nativization of legacy code
- Consistency with kernel conventions
  - Error return codes
  - Single return paths
- Typedef extermination
- Using kerneldoc
- General code review and cleanup
- Locking review

Future Directions

- Refine locking to enhance scalability
- Further userland integration
- Complete integration into networked environment
  - Integrate with 2.5/6 IPSEC implementation
  - Integrate with NFSv4
- Security-Enhanced X
  - Design report available
- Policy specification and analysis tools
- Platform for application security mechanisms

Questions?

http://www.nsa.gov/selinux/

End of Presentation