Linux Kernel Security Overview

Linux Security Summit Europe 2018
Edinburgh, UK

James Morris *jmorris@namei.org*

$ whoami

- Linux kernel security subsystem maintainer
- Linux kernel engineer at Microsoft
- Previously
  - Netfilter core team member
  - Author of Linux kernel crypto API
  - LSM development team
  - SELinux kernel lead at Red Hat; invented MCS & sVirt
  - Linux kernel manger at Oracle

Overview

- Background
- Major Components
- Resources

Linux Security Model

Discretionary Access Control (DAC)

DAC was inherited from Unix, designed in late 1960s

> “The first fact to face is that UNIX was not developed with security, in any realistic sense, in mind; this fact alone **guarantees a vast number of holes**.”

Dennis Ritchie, “On the Security of UNIX”, 1979

DAC is insufficient for modern security threats:

- does not protect against flawed or malicious code
- does not cover all security critical functions
- superuser violates security model

“It must be recognized that the mere notion of a super-user is a theoretical, and usually practical, blemish on any protection scheme.”

(also from Ritchie 1979)

Linux Kernel Security Extensions

POSIX Access Control Lists (ACLs)

- Extend Unix DAC’s abbreviated ACLs with fine-grained permissions
- So, more DAC

POSIX Capabilities

- Splits superuser into high-level abstractions, e.g. CAP_NET_ADMIN
- Process-based
- Filesystem labels for executables
- Several issues:
  - Privilege overlap and escalation
  - Difficult to reason about overall security policy
  - … and it’s still DAC

Audit

- Implemented to meet government certification requirements (similar to “C2”)
- Actually quite useful, see auditctl(8)

seccomp

- Generalized syscall filter
- Reduces attack surface of kernel
- Not a sandbox, but useful component of one
- Implemented as BPF filters
- Complex...
  - Use libseccomp()

Namespaces

- Private views of global resources:
  - cgroup, ipc, network, pid, user, mount, uts
- Derived from plan9 concepts
- APIs: clone(2), setns(2), unshare(2)
- See also pam_namespace(8)
- Linux containers are namespaces + cgroups

Netfilter

- Hooks in L3 code at packet flow points
- Pluggable:
  - iptables
  - ip6tables
  - conntrack
  - NAT

Cryptography API

- Many types of algorithms:
  - ciphers, compressors, digests, hashes, rngs etc.
- Synchronous and asynchronous APIs
- Zero-copy interface
- Support for crypto hardware
- Userspace API via AF_ALG

- Users include:
  - Disk encryption
  - IPSec
  - Key management
  - Integrity subsystem

Key Management

- Management of keys, keyrings, tokens etc.
- Key attributes: owner, group, permission mask, expiry, payload, type, state, description
- Several types of keys implemented, e.g.:
  - per-process: user, session
  - trusted keys: TPM generated and sealed
  - encrypted keys: kernel generated & encrypted
- Userland API: keyctl(1)

Linux Security Modules (LSM)

- Allows different access control schemes to be plugged into kernel

- Hook API:
  - Mediation of all security-critical interactions in kernel
  - Race-free and with all relevant security information available

- Major: SELinux, Apparmor, Smack
- Minor (stackable): YAMA, Capabilities

Security Enhanced Linux (SELinux)

- All objects & subjects have security labels
- All relevant security interactions mediated via fine-grained generalized permissions
- Flexible security policies: separation of mechanism and policy
- Policy centrally administered, not overridable by user (including root)
- Helps contain breaches via least privilege
- Implemented in Fedora family, Android

Smack

- Simplified MAC (also label-based)
- Smaller code and policy footprints
- Typically seen in embedded space (e.g. Tizen)

Apparmor

- Pathname-based
- Familiar Unix-like configuration files
- Designed for ease of use
- Implemented in Suse, Ubuntu

Platform Security

- Kernel support for platform security features, such as:
  - TPM
  - NX, SMEP
  - SEV, SME
  - SGX
  - TEEs

Integrity Management

- Integrity Measurement Architecture (IMA)
  - Extends secure/trusted boot to the OS
  - Detects if files have been maliciously or accidentally altered
  - Optimally w/ TPM
  - Remote attestation
  - Local appraisal
  - Digital signature support (data authenticity)

- Extended Verification Module (EVM)
  - Protects security xattrs against offline attack

- dm-verity / dm-integrity
  - Transparent block-level integrity verification

Kernel Self Protection (KSP)

- Harden kernel against attack

- Kernel Self Protection Project (KSPP)
  - Started in 2015
  - Politically challenging
  - Initially focused on backporting grsec/PAX to mainline
  - Focus on killing bug classes vs. individual bugs
  - https://kernsec.org/wiki/index.php/Kernel_Self_Protection_Project

Resources

- LSM mailing list
  - http://vger.kernel.org/vger-lists.html#linux-security-module
- Other mailing lists:
  - linux-integrity
  - linux-keyrings
  - oss-security (Openwall)
- Wiki: kernsec.org
- LWN Security
  - http://lwn.net/Security

Questions?