Linux OS-Level Security

Nikitas Angelinas

SEAGATE

MSST 2015

Agenda

- SELinux
- SELinux issues
- Audit subsystem
- Audit issues
- Further OS hardening

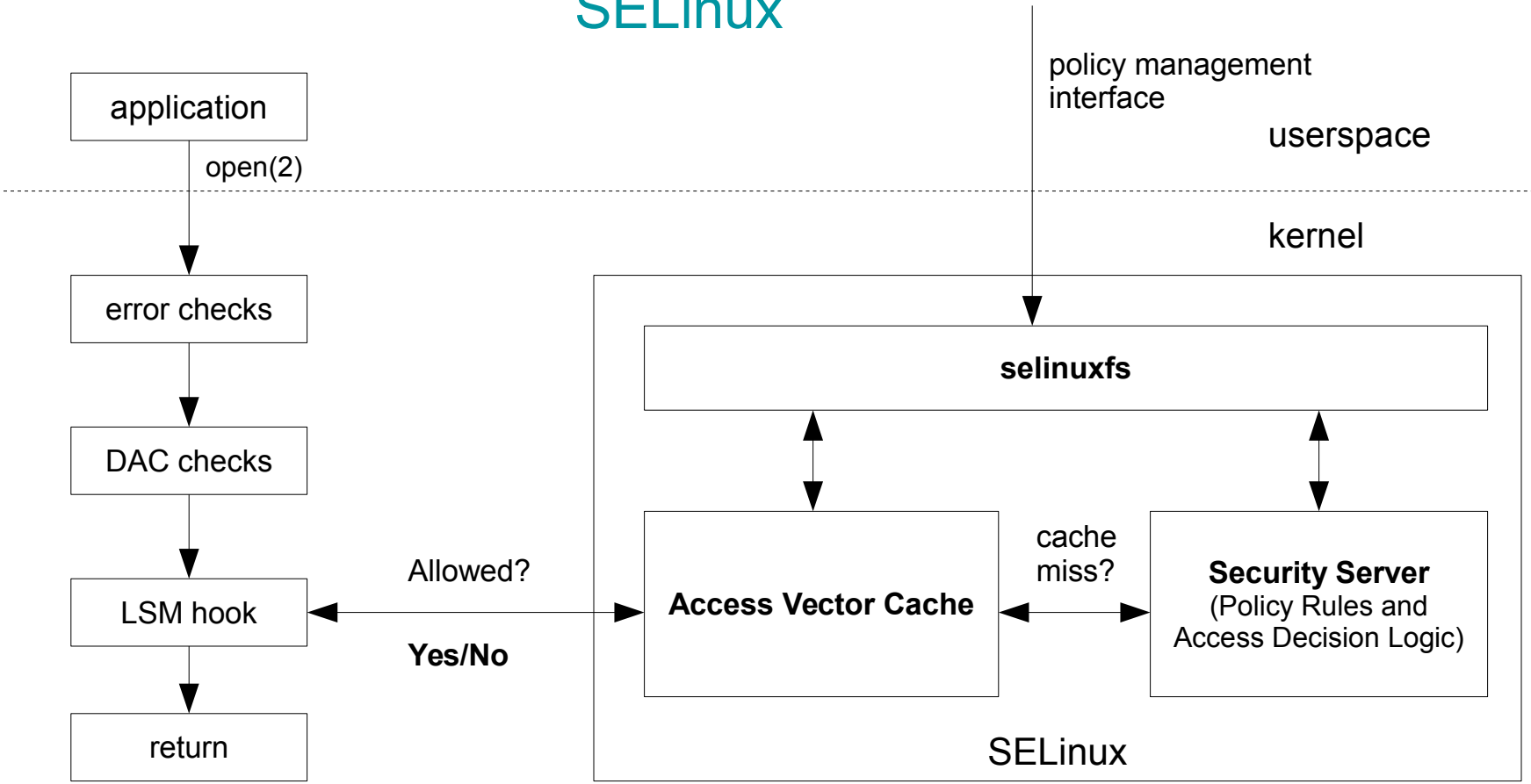
SELinux

- Security-Enhanced Linux
- Is **NOT** a Linux distribution
- A kernel feature
- A Linux Security Module (LSM)
  - security fields in kernel data structures
    - processes, files, directories, sockets, IPC, and other OS objects
  - hooks on all paths from userspace
    - return a binary predicate: **allowed** or **not allowed (e.g. EACCES)**

SELinux

- SELinux is a labeling system
- Every process has a label
- Every OS object has a label
- Policy rules control access between processes and objects, based on their labels
- Policy rules are enforced by the kernel
- The objective is to contain misbehaving/compromised userspace applications

SELinux


application
open(2)
error checks
DAC checks
LSM hook
return
Allowed?
Yes/No
policy management
interface
userspace
kernel
selinuxfs
Access Vector Cache
cache
miss?
Security Server
(Policy Rules and
Access Decision Logic)
SELinux

SELinux

- Mandatory Access Control
    - vs. Discretionary Access Control (file ownership, permissions)
- Labels are of the form user:role:type:**level**
    - e.g. process: /usr/sbin/sshd → system_u:object_r:sshd_exec_t:s0
    - e.g. file: /root/ssh/* → root:object_r:ssh_home_t
- SELinux implements LSM hooks and offers:
    - Role-based Access Control
    - Type Enforcement
    - **Multi-Level Security**

SELinux labels and rules

- ls -Z /usr/sbin/sshd -rwxr-xr-x. root root system_u:object_r:**sshd_exec_t**:s0 /usr/sbin/sshd
- ls -Z /root/.ssh/id_rsa -rw-------. root root root:object_r:**ssh_home_t** /root/.ssh/id_rsa
- ls -Z /etc/ssh/sshd_config -rw-------. root root system_u:object_r:**etc_t** /etc/ssh/sshd_config
- ls -Z /var/run/sshd.pid -rw-------. root root system_u:object_r:**sshd_var_run_t** /var/run/sshd.pid
- ps -eZ | grep ssh system_u:system_r:**sshd_t** 5142 ? 00:18:57 sshd

- sesearch --allow -t sshd_var_run_t /etc/selinux/clip/policy/policy.24 | grep sshd_t

allow **sshd_t sshd_var_run_t** : file { ioctl read write create getattr setattr lock append unlink link rename open } ;

- sesearch --allow -t sshd_exec_t /etc/selinux/clip/policy/policy.24 | grep sshd_t

allow **sshd_t sshd_exec_t** : file { ioctl read getattr lock execute execute_no_trans **entrypoint** open } ;

SELinux policies

- Policy rules defined in a high-level language
    - Conditionals, booleans, m4 support, etc.
- Compiled in userspace
- Inserted in the kernel
- Policies are stackable
- Large collection of off-the-shelf policies
- Custom software will need custom policy rules
    - audit2allow(1) can help

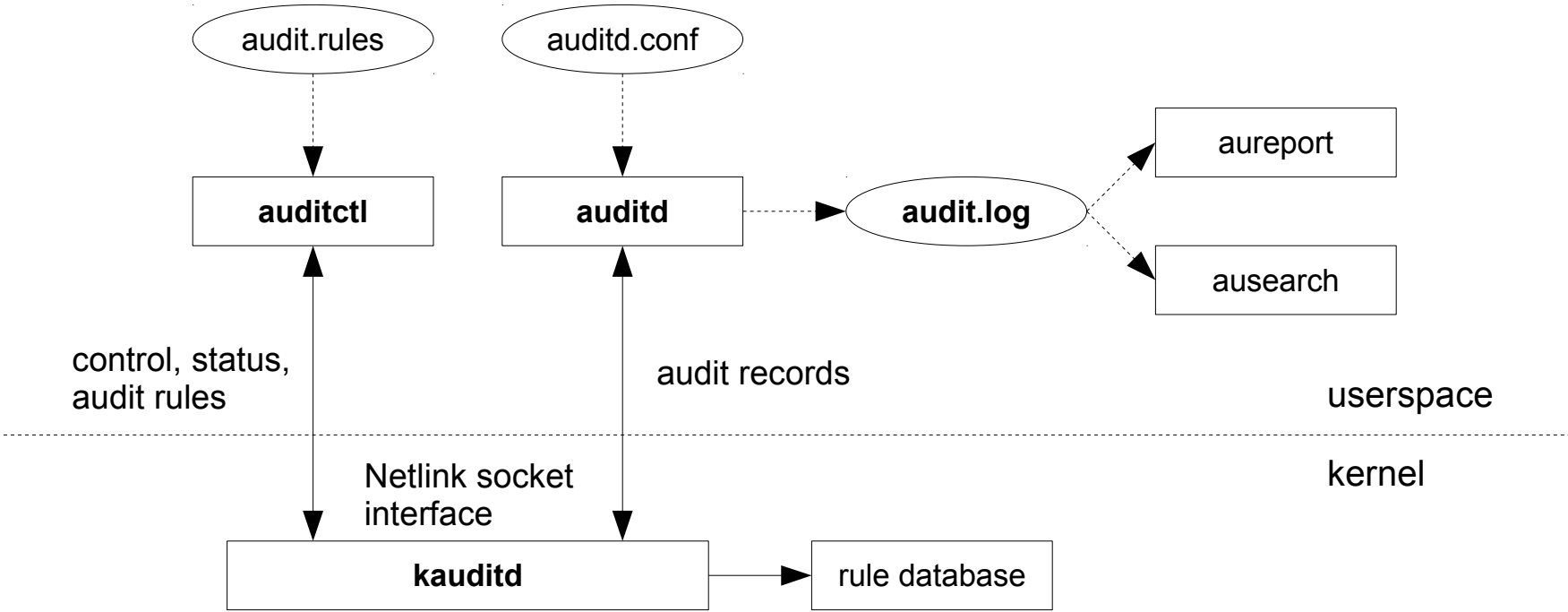
SELinux issues

- SELinux I/O checks and auditing occur at the VFS
  - Networked filesystems might not use the VFS
  - Exporting SELinux hooks is a software layer violation
  - LSM hooks add overhead
- Potential for policy mismatch in a distributed environment
- Writing policy rules can be tedious, error-prone
- Can be difficult to verify
- Object stores

Audit subsystem

- Generates a system audit trail
    - timestamp, user, LSM labels, authentication details, etc.
- Meets or exceeds CAPP, LSPP, RSBAC, NISPOM, FISMA, PCI-DSS, etc. certifications
- Highly capable
    - Watches on individual files and syscalls
    - Filters on uid, syscall #, LSM labels, and other attributes
    - Integrated with other kernel subsystems
    - Message types: PATH, SYSCALL, USER_*, DAEMON_*, AVC, etc.
    - Log rotation
    - Email notifications

Audit subsystem



Audit issues

- Audit logs are not generated by remote fs operations
- Additional tools needed for log aggregation
- Log integrity might mean blocking operations
    - Possible performance impact
        - Mitigate with solid-state storage

Putting the two together

- SELinux is a user of audit
    - SELinux denials log AVC-type messages

 # ausearch -m avc -ts recent

type=SYSCALL msg=audit(1431684482.612:67019): arch=c000003e **syscall=2** success=yes **exit=9** a0=4076db a1=50000 a2=3cdf a3=3b8ff28d60 items=0 ppid=120622 pid=120628 auid=0 uid=0 gid=0 euid=0 suid=0 **fsuid=0** egid=0 sgid=0 fsgid=0 tty=(none) ses=11085 **comm="updatedb"** exe="/usr/bin/updatedb" subj=system_u:system_r:locate_t:s0-s15:c0.c1023 key=(null)

type=AVC msg=audit(1431684482.612:67019): **avc: denied** { read } for pid=120628 **comm="updatedb"** name="audit" dev=dm-0 ino=15583 **scontext=system_u:system_r:locate_t:s0-s15:c0.c1023 tcontext=system_u:object_r:auditd_log_t:s15:c0.c1023** tclass=dir

Kernel hardening

- SELinux only handles userspace
    - Kernel vulnerabilities
        - assume root, LSM reset
- Reduce kernel attack surface
    - Remove unnecessary drivers, device nodes, filesystems, protocols, syscalls, etc.
    - seccomp BPF/libseccomp
- Reduce predictability and harden
    - Use non-stock distribution kernels
    - Restrict /proc/kallsyms, System.map permissions
    - kASLR/base address randomization
    - kernel config options: NX, etc
    - grsecurity, PaX

Best practices

- Minimise
    - Remove cruft
    - Read-only virtual filesystems
    - Least privileges for daemons
- Make good use of:
    - permissions, PAM, capabilities, namespaces, containers, etc.
- Disable module loading or whitelist
- ASLR/NX for userspace
- Timely updates (can be a challenge for some sites)
    - Rely on OS distributor or follow CVEs
- Pentest internally and externally

Thank You!

Linux OS-Level Security

Nikitas Angelinas

SEAGATE

MSST 2015