Protecting the Android TCB with SELinux

Stephen Smalley
Trusted Systems Research
National Security Agency

Background

- At LSS 2013, SELinux was:
  - shipping in the Samsung Galaxy S4 smartphone, and
  - included in the Google Android 4.3 release,
  - but was in permissive mode by default.
- SELinux went enforcing in:
  - 4.3 updates to S4 and other Samsung devices, and
  - the Google Android 4.4 / KitKat release.

Today's Talk

- Looking at how SELinux has been applied over the past year to protect the Android Trusted Computing Base (TCB).
- All of these changes have been made in the Android Open Source Project (AOSP) master branch.
- Starting from the 4.4 release and then looking at what we expect to be in the upcoming Android “L” release.

The Android TCB

- The Linux kernel.
- Full-root daemons, e.g. init/ueventd, vold, netd, debuggerd, zygote, …
- Non-root daemons with capabilities.
  - Particularly installd and system_server.
  - Those with more limited capabilities, e.g. wpa_supplicant, rild, …
- Other system UID daemons, e.g. servicemanager, surfaceflinger, …
- Subsystem-specific daemons, e.g. sdcard, keystore, …
- To a lesser extent, certain system apps.
  - Applications with platform UIDs, e.g. system, radio, …
  - Platform-signed or /system/priv-app apps.

Android TCB Protection pre-SELinux

- mmap_min_addr, dmesg_restrict, kptr_restrict
- Root daemon minimization.
- Selective capabilities reduction, e.g. installd.
- Per-subsystem UIDs, e.g. radio, nfc, bluetooth, media.
- Per-app UIDs.
- Blocking privilege escalation by apps (NOSUID, CAPBSET_DROP, NO_NEW_PRIVS).
- Read-only /system mount.
- NX/XN, ASLR, PIE, RELRO, FORTIFY_SOURCE, ...

SELinux in Android 4.4 / KitKat

- First Google Android release to ship with SELinux enforcing by default.
- Focused on protecting a set of root daemons.
  - installd, netd, vold, zygote.
  - Protect from misuse, contain damage from exploit.
- Prevented exploitation of a long-standing (since 2010) local root vulnerability in Android vold, fixed in 4.4.3.
  - Similar to various other Android root exploits blocked by SELinux, see our prior papers/presentations.

Android SELinux Distinctives

- Unlike in conventional Linux distributions…
- SELinux enabled and enforcing is mandatory.
  - Required by Android CDD and CTS for >= 4.4.
- There is no generic unconfined domain.
  - But specific domains can be marked with the unconfineddomain attribute.
  - Even such domains are not completely unrestricted by SELinux.

Post-4.4 SELinux Strategy

- In parallel:
  - Shrink the set of unconfined domains.
    - Converging to fully confined.
  - Shrink the set of permissions allowed even to unconfined domains.
  - Targeted improvements to confined domains.
    - Focusing on TCB protection.

Confined Domains

- 4 (out of 47) in Android 4.4.
  - The previously mentioned root daemons.
- 49 (out of 61) in Android L Developer Preview.
  - Including all third party apps.
- 62 (out of 65) in AOSP master.
  - Everything except for kernel, init, init_shell.
- (for Nexus 5 -user build)

Protecting the Kernel

- No domain can map low memory.
- No domain can read/write /dev/kmem or /dev/mem.
- Only init can modify proc security settings (kptr_restrict, mmap_min_addr, dmesg_restrict, …).
- Only init can load/reload SELinux policy.
- No domain can switch SELinux to permissive.
- Not even unconfined domains!

Protecting the Kernel: Loadable Modules

- In Nexus devices, CONFIG_MODULES=n.
- Some devices include module support.
- Removed sys_module from unconfineddomain.
- Only presently allowed to system_server.
  - To support loading of wireless driver.
- Possible future extension: limit module loading to modules from rootfs or /system only.
  - But this requires a newer Android kernel.
  - and a selinux_kernel_module_from_file hook.

Protecting the Kernel: Usermode Helpers

- Only init can set or configure kernel usermodehelpers.
  - e.g. /sys/kernel/uevent_helper,
    /proc/sys/kernel/hotplug,
    /proc/sys/kernel/usermodehelper/bset, ...
- Kernel usermodehelpers can only be executed from the rootfs or /system.
  - Never from /data or /cache.
- Kernel usermodehelpers must transition to a domain other than the kernel domain.
  - Not allowed to retain kernel domain's permissions.

Protecting OS File Integrity

- /system already mounted read-only.
- (Re)mounting filesystems restricted to minimal set of domains and types.
- Write access to /system only allowed to recovery.
- Context= mounts can only be used to assign a non-writable type to a filesystem.
- Writing to block devices restricted to minimal set of domains and types.
- Raw I/O and mknod capabilities restricted to minimal set of domains.
- All of these accesses removed from unconfineddomain.

Protecting System Services: Ptrace

- Only debuggerd is allowed to ptrace other domains.
- Most domains have no ptrace access at all.
  - Not allowed by default intra-domain.
  - Not even allowed to unconfined domains.
- ptrace to init and keystore prohibited by neverallow.
- ptrace by app to non-app prohibited by neverallow.

Protecting System Services: W^X

- rootfs files cannot be written at all.
- /system can only be written from recovery.
- For most domains, no execute to files outside of the rootfs or /system partitions.
- For most domains, no PROT_EXEC anonymous mappings or modified private file mappings.
- Removed from unconfineddomain as well.

Protecting System Services: Malicious Input

- Preventing malicious symlink attacks.
  - No reading symlinks created by apps or shell.
  - Not even by unconfineddomain.
- Protecting against malicious socket IPC.
  - Preventing use of netlink sockets and daemon sockets by apps or shell.

Protecting Data File Integrity

- installd is responsible for various aspects of /data management, including upgrading the layout, populating the dalvik cache, creating app data directories.
- As a result, our original policy (and 4.4) allowed it fairly sweeping write access to directories and files under /data.
- In AOSP master, installd is restricted to only the /data file types and permissions needed and files it creates are placed in their own type.
- Can no longer write to arbitrary system data.

Protecting Data File Integrity (cont'd)

- system_server is responsible for managing large parts of /data at runtime.
- Our original policy confined it but allowed it to read/write all types under /data. Android 4.4 shipped it unconfined.
- In AOSP master, system_server is restricted to only the /data file types and permissions needed.
  - No longer can access e.g. keystore, tee, drm data.
- One of only two confined domains that can write to the generic system_data_file type.

Protecting Security-Critical Data Files

- /data/property: Persistent system property store.
  - Can only be read/written by init.
- /data/misc/keystore: Certificate and (encrypted) key store.
  - Can only be read/written by keystore.
- Not even allowed to unconfined domains.

Protecting User Data from Rogue Daemons

- Removing access to app data files.
  - Also helps protect daemons from malicious inputs.
  - Where necessary, try to avoid open access.
- Removing SDcard access.
  - Also protects daemons from being killed by unsafe ejection.
- Limiting daemons to specific partitions.

Other Unconfined Lockdown

- Removed from unconfineddomain, but may be allowed to specific domains.
- No process operations on other domains (e.g. signals).
- No transitions to other domains.
- No executing other programs without transitioning.
- No internet access.
- No syslog access.
- No audit capabilities.
- No MAC capabilities.

Preserving TCB Protection Goals

- What policy takes away, policy can give back again..
- Particularly need to consider device-specific policy and OEM customizations.
- Preserving in AOSP: neverallow rules
- Preserving in devices: Android CTS test
  - SELinuxTest
  - SELinuxDomainTest

More than Status Quo Encapsulation

- Removing need for Linux capabilities from various daemons.
  - e.g. clatd, various qcom daemons
- Removing need for kmem access.
  - rmt_storage, rewritten to use uio.
- Detecting and eliminating descriptor leaks.
  - Sockets/pipes inherited across fork/exec.
- Detecting and eliminating text relocations.

Going Beyond the Kernel

- SELinux is extensible for userspace policy enforcers (aka userspace object managers).
- Just define object classes and permissions in policy and call selinux_check_access() from your own code.
- Handles mapping classes/permissions from strings to values, auditing denials, caching decisions, flushing cache on policy reload, …

Going Beyond the Kernel (cont'd)

- In 4.4, applied to property service and zygote.
- In AOSP master:
  - Extended to servicemanager, keystore, drmserver, debuggerd.
  - Replacing legacy hardcoded UID-based ACLs.
- Benefits:
  - Configurable, centralized policy.
  - Distinguish at finer granularity than UID.
  - Can control even root processes.
  - Centralized, uniform audit.

Overcoming Practical Challenges

- Handling upgrades from non-SE (unlabeled) devices and from older policies.
  - Automatic relabel, triggered by change in file_contexts configuration.
- Enabling efficient, complete labeling of sysfs.
  - Prune tree walk as early as possible (partial matching support in selabel).
  - Check/relabel on add/change/online uevents.

Overcoming Practical Challenges (cont'd)

- Labeling devices based on stable names.
  - Daemon needs read/write access to a specific partition.
  - Partition number and thus real device node name can vary across devices.
  - Partition name is stable.
  - Label-by-symlink support using /dev/block/platform/<device>/by-name symlinks.
  - Best match support in selabel.

What's Next

- Automating new device bringup
- New APIs for apps (isolation, sandboxing, ...)
- Hardening Android multi-user
- Enterprise Ops / IntentFirewall
- Improved CTS testing
- Further policy hardening

Questions?

- Send email to seandroid-list-join@tycho.nsa.gov to join the public SE for Android mailing list.
- Private email just to our SE for Android team: seandroid@tycho.nsa.gov
- Source code: https://bitbucket.org/seandroid
- Project page: http://seandroid.bitbucket.org