SELinux in Android Lollipop and Marshmallow

Stephen Smalley
Trusted Systems Research
National Security Agency

Background

- At LSS 2014, we looked at how SELinux had been applied to protect the Android Trusted Computing Base (TCB).
- Starting with selective root daemon confinement in Android 4.4 KitKat.
- Moving toward full confinement and TCB protection.
- Culminated in the Android 5.0 Lollipop release.

Today's Talk

- Review final state of SELinux in Android 5.0 Lollipop release and updates.
- Look at advances in SELinux expected in the upcoming Android 6.0 Marshmallow release.
- Discuss the state of Android & upstream SELinux.
- Summarize ongoing and future work.

Android 5.0 Lollipop

- Officially announced mid-October 2014.
- Released to AOSP on November 4, 2014.
- First shipped on the Nexus 6, 9, and Player.
- Currently running on ~18% of active Android devices.
- First official Android release to ship with SELinux enforcing for all processes.
  - Mandated by Android 5.0 CDD, tested by CTS.

SELinux in Android 5.0 Lollipop

- All system services and apps are confined.
  - Including root daemons.
- Only two domains are “unconfined”.
  - kernel and init
- Even these two domains are not completely unrestricted by SELinux.
  - TCB protection goals are applied universally.
  - No domain/process is all-powerful.

Protecting the Android TCB via SELinux

- Nothing can map low memory or access /dev/{k}mem.
- Only init can set sensitive kernel settings/policy.
- Only recovery can write to / or /system (the OS).
- Native services can only execute from / and /system.
- Only debuggerd can ptrace others.
- Apps cannot write to most netlink sockets.
- Apps cannot write to most service sockets.
- No reading/following untrusted symlinks.

Service-specific Protection via SELinux

- Android keystore
  - Provides secure storage of keys.
- SELinux kernel-enforced guarantees:
  - Nothing can ptrace the keystore.
  - Nothing else can open /data/misc/keystore files.
- SELinux userspace access control:
  - Keystore checks SELinux policy for client requests.
  - Sensitive operations restricted via policy.

Android 5.0 CTS SELinux Tests

- Enforcing for all.
- Policy satisfies neverallow rules.
- Core system services running in their domains.
- Only init in the init domain.
- Only kernel threads in the kernel domain.
- Nothing running in recovery or su domains.
- No policy booleans.

Lollipop Updates

- 5.0.1 through 5.1.1
- Carry forward all of the SELinux protections.
- 5.1 CTS has improved neverallow checker.
  - Check all domains, not just AOSP domains.
  - Back-ported from AOSP master.

Android 6.0 Marshmallow (“M”)

- Next major release of Android.
- 3 Developer Previews released.
- Final release expected Q3 2015.
- Includes SELinux changes made to AOSP master since Lollipop was forked.

SELinux in Android 6.0 Marshmallow

- ioctl whitelisting (see separate talk)
- Reinforcing Android multi-user
- Strengthening the Chrome sandbox
- Locking down the Binder
- Policy Hardening
- CTS enhancements

Android Multi-User

- First introduced in 4.2 for tablets.
- Supported on phones starting with 5.0.
- Also the basis for restricted and managed profiles, including Android for Work.
- User identity encoded as part of the UID.
- Middleware enforces certain cross-user restrictions.
- Kernel enforces the usual DAC restrictions.

SELinux and Android Multi-User

- Goal: Reinforce Android multi-user separation transparently, without complicating policy.
- Map user identity to unique MLS level (using categories), assign to app processes and files.
- MLS constraints prevent communications across different levels.
  - except via Binder, which is mediated by middleware

SELinux and Android Multi-User (cont'd)

- Apps running for different users are automatically assigned different levels.
- SELinux blocks sending signals, accessing /proc/pid, opening app data files, or communicating via local sockets across levels.
- No per-user or per-app policy configuration required; just using MLS and categories.

The Chrome for Android sandbox

- Combines multi-process architecture with UID isolation.
- App service components can be declared with a process=”name” and an isolatedProcess=”true” attribute.
- Android will run such services in a separate process and UID from the main app.
- This process has no Android permissions and the usual DAC restrictions, i.e. cannot read or write the files of the main app unless they are world accessible.

SELinux and the Chrome sandbox

- Goal: Strengthen the sandbox beyond DAC.
- Isolated service processes already assigned their own domain, isolated_app.
- Removed specific accesses from isolated_app.
  - No direct open of app data files.
  - No GPU device access.
  - No keystore permissions.

Locking down the Binder

- Binder is the central IPC primitive for Android.
- Binder has a top-level name service, known as the Binder context manager.
- On Android, this is the servicemanager process.
- The servicemanager checks SELinux policy for requests.
  - add (register new service by name)
  - find (look up a service by name)
  - list (list all registered service names)

Locking down the Binder (cont'd)

- Lollipop shipped with add permission restricted.
- Marshmallow locks down find and list permissions.
- Prevents apps from looking up arbitrary binder services.
- If you cannot look up the service binder reference, you cannot call the service.
  - Binder references are kernel-managed capabilities.
- Chrome sandbox / isolated_app restricted to only minimal required services.

Policy Hardening

- Forced init to transition domains on exec.
  - Separate domains added for helper programs and all services, even oneshot services.
- Locked down block device access.
  - Protecting critical partitions from direct access.
  - Limiting each domain to only needed partitions.
- Removed unconfined domain.
  - Even init and kernel no longer use it.
- Many more neverallows.

CTS Enhancements

- Test validity of all device SELinux configuration files.
  - Correctness and inclusion of AOSP definitions.
- Test labeling of running CTS app and its files.
  - Runtime state of device, not just configuration.
- Augmented testing of service domains.
  - Reverse mapping, additional services, hostside.
- Test MLS attributes.

Android & Upstream SELinux

- Android SELinux refreshed from upstream.
- Complete copy of upstream SELinux code imported under external/selinux, used for libsepol, checkpolicy.
- Work ongoing to synchronize upstream libselinux and Android fork.
- Policy tools imported into Android for use by Android developers (audit2allow, sesearch, seinfo).
  - Breaks dependency on build host OS supporting Android SELinux policy version.

SELinux in Android vs Linux distributions

- > 60% of active Android devices are running a version of Android that has SELinux.
  - 4.3 Jelly Bean (4.7%), 4.4 KitKat (39.3%), 5.x Lollipop (18.1%)
- In Android 4.4 and later, SELinux is always enabled and enforcing.
- As of 5.0 and later, Android has a more secure default policy than Linux distributions.
  - Fully enforcing, only kernel and init “unconfined”, TCB protection goals.
- Yet Android has a much smaller and simpler policy than Linux distributions.
  - Roughly 5% the number of rules, 10% the number of types.
- Could a similar policy be developed for Linux distributions?

Ongoing and Future Work

- Enable apps to opt into stronger protections.
  - Sandboxing, isolation, file protection
- Investigate new runtime permissions feature.
  - when Android 6.0 source is released
- Improve SELinux tooling for Android.
- Further userspace policy enforcement.
- Kernel self-protection

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
- ToDo list: https://bitbucket.org/seandroid/wiki/wiki/ToDo