Security Enhanced (SE) Android: Bringing Flexible MAC to Android

Stephen Smalley and Robert Craig
Trusted Systems Research
National Security Agency

Motivation

- Android security relies on Linux DAC.
  - To protect the system from apps.
  - To isolate apps from one another.
  - To prevent bypass of Android permissions.
- DAC shortcomings are well established.
  - Fundamentally inadequate to protect against flawed and malicious applications.
- SELinux can address these shortcomings.

Challenges

- Kernel
  - No support for per-file security labeling (yaffs2).
  - Unique kernel subsystems lack SELinux support.
- Userspace
  - No existing SELinux support.
  - All apps forked from the same process (zygote).
  - Sharing through framework services.
- Policy
  - Existing policies unsuited to Android.

Kernel Support

- Implemented per-file security labeling for yaffs2.
  - Using recent support for extended attributes.
  - Enhanced to label new inodes at creation.
- Analyzed and instrumented Binder for SELinux.
  - Permission checks on IPC operations.

Userspace Support

- xattr and AT_SECURE support in bionic.
- Minimal port of SELinux libraries and tools.
- Labeling support in build and updater tools.
- Policy loading, device & socket labeling (init).
- App security labeling (zygote, dalvik, installd).
- Property service and zygote controls.
- Runtime policy management support.

Policy Configuration

- Enforce a small set of platform security goals.
  - Confine privileged services.
  - Sandbox and isolate apps.
- Key properties:
  - Small, fixed policy.
  - No policy writing for app developers.
  - Invisible to users.

Policy Size & Complexity

	SE Android	Fedora
Size	71K	4828K
Domains	39	702
Types	182	3197
Allows	1251	96010
Transitions	65	14963
Unconfined	3	61

Middleware MAC (MMAC)

- Many attacks occur entirely at middleware layer.
  - Cannot be addressed via kernel layer MAC.
- SELinux userspace object manager model not readily applicable.
  - Binder IPC, multi-stage call chains.
  - checkPermission API.
  - Implications for SELinux policy.
- Required a separate middleware MAC layer.

MMAC mechanisms

- Install-time MAC
  - Enforced by PackageManagerService.
  - Based on app certificate, package name.
  - Can disable even pre-installed apps.
  - Linkage to SELinux policy via seinfo tag.
- Permission revocation
- Intent MAC, Content Provider MAC

Case Studies

- Root exploits.
  - Exploid, RageAgainstTheCage, GingerBreak, KillingInTheNameOf, Zimperlich, mempodroid.
- Flawed apps.
  - Skype, Lookout Mobile, Opera Mobile.
- All mitigated by SE Android.

Case Study: /proc/pid/mem

- /proc/pid/mem
  - Kernel interface for accessing process memory.
  - Write access enabled in Linux 2.6.39+.
- CVE-2012-0056
  - Incorrect permission checking.
  - Induce setuid program into writing own memory.
- Demonstrated by mempodroid exploit.

Mempodroid: Overview

- Some complexity omitted.
- Exploit invokes setuid root run-as program with open fd to /proc/pid/mem as stderr and shellcode as argument.
- run-as program overwrites self with shellcode when writing error message.
- Shell code sets uid/gid to 0 and execs shell or command.

Mempodroid vs SE Android Part 1

- With no specific policy for run-as.
- Write to /proc/pid/mem will still succeed.
- But run-as program runs in caller's security context.
  - Still restricted by SELinux policy.
  - No privilege escalation.
  - But also no support for run-as functionality.

Mempodroid vs SE Android Part 2

- With policy and code changes for run-as.
  - Sufficient to support legitimate functionality.
- Open file to /proc/pid/mem closed by SELinux due to domain transition.
  - No memory overwrite, exploit fails.
- run-as confined to least privilege.
  - Minimal capabilities, required transition.

Case Study: Lookout Mobile

- Security app for Android.
- LOOK-11-001
  - Created files via native calls without setting umask.
  - Leaving them world-readable and -writable.
- Any other app on the device could:
  - Disable or reconfigure Lookout.
  - Read private user data.

SE Android vs Lookout vulnerability

- Classic example of DAC vs. MAC.
  - DAC: Permissions are left to the discretion of each application.
  - MAC: Permissions are defined by the administrator and enforced for all applications.
- All third party apps denied access to files created by other apps.
  - Each app and its files have a unique SELinux category set.

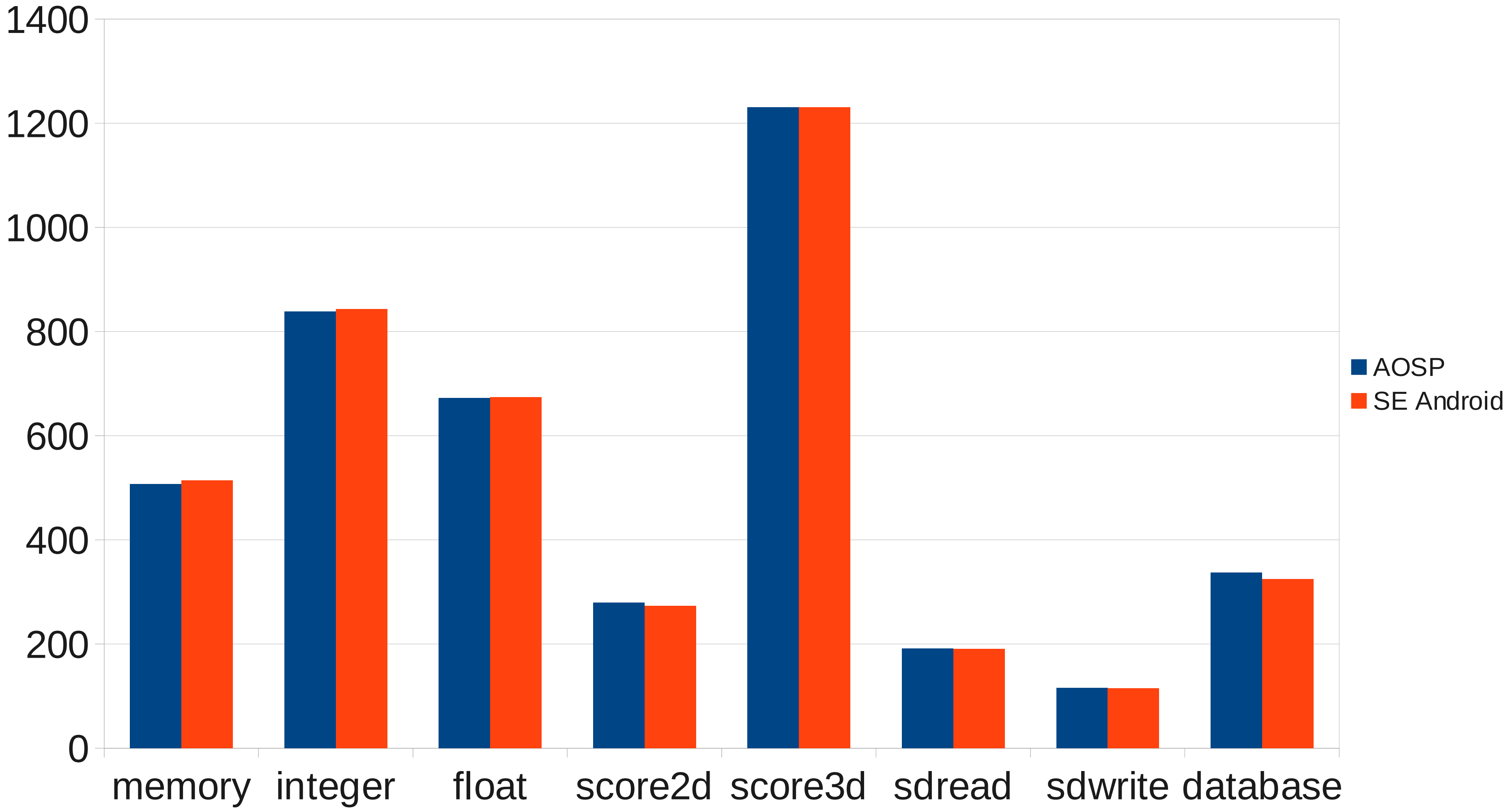
AOSP merging

- 4.1: Changes below dalvik merged, conditional under HAVE_SELINUX.
- 4.2: Many more changes merged, including dalvik and frameworks support, still conditional under HAVE_SELINUX.
- Current master: HAVE_SELINUX guards removed, userspace support unconditional in build.

Size Comparison (maguro, 4.2)

	AOSP	SE ANDROID	INCREASE
boot	4400K	4552K	+152K
system	194072K	194208K	+136K
recovery	4900K	5068K	+168K

AnTuTu (maguro, 4.2)



Related Work

- Android middleware extensions (e.g. Kirin, SAINT, TaintDroid, Porscha, AppFence, IPC Inspection, QUIRE)
  - Depend on underlying kernel protections.
  - SE Android ensures unbypassability of middleware mechanisms.
  - Kirin and SAINT similar to install-time MAC.
- Prior work on SELinux for Android (e.g. Shabtai et al)
  - Good start but did not address many of the challenges, demonstrate effectiveness or merge to AOSP.
- TrustDroid & XManDroid
  - Most similar in goals and approach.
  - MAC for middleware and kernel layers.
  - SE Linux as a better foundation than TOMOYO.

Questions?

- http://selinuxproject.org/page/SEAndroid
- Public SE Android list: Send “subscribe seandroid-list” to majordomo@tycho.nsa.gov.
- NSA SE Android team:
  - seandroid@tycho.nsa.gov