Lessons from the trenches: An inside look at Android security

Nick Kralevich <nnk@google.com>

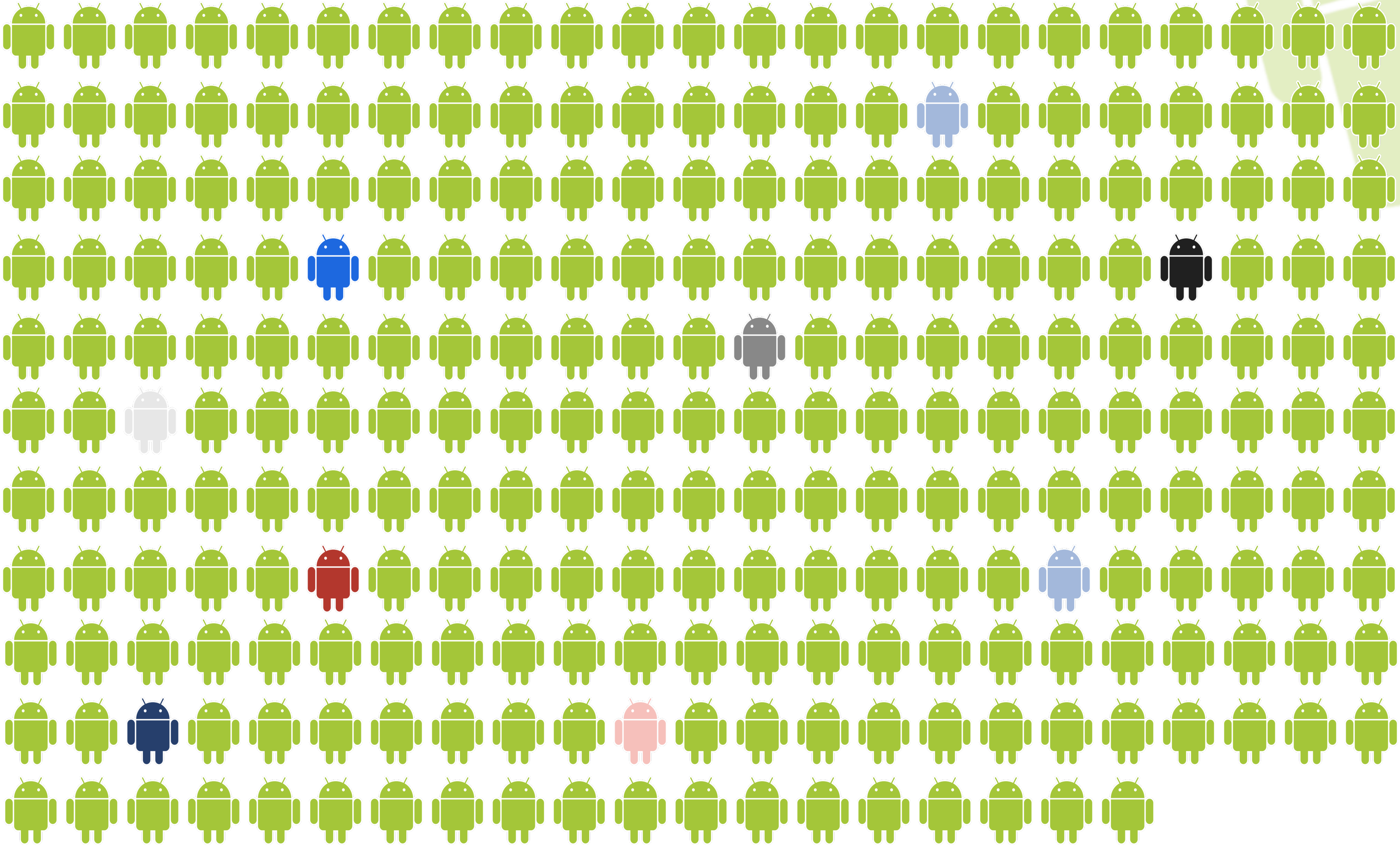
2015-09-09

whoami

whoami

- Nick Kralevich
- nnk@google.com
- Android Security since 2009
- Platform security lead

Just one of many people ...



whoami

millions

lines of code in
Android Open Source

thousands

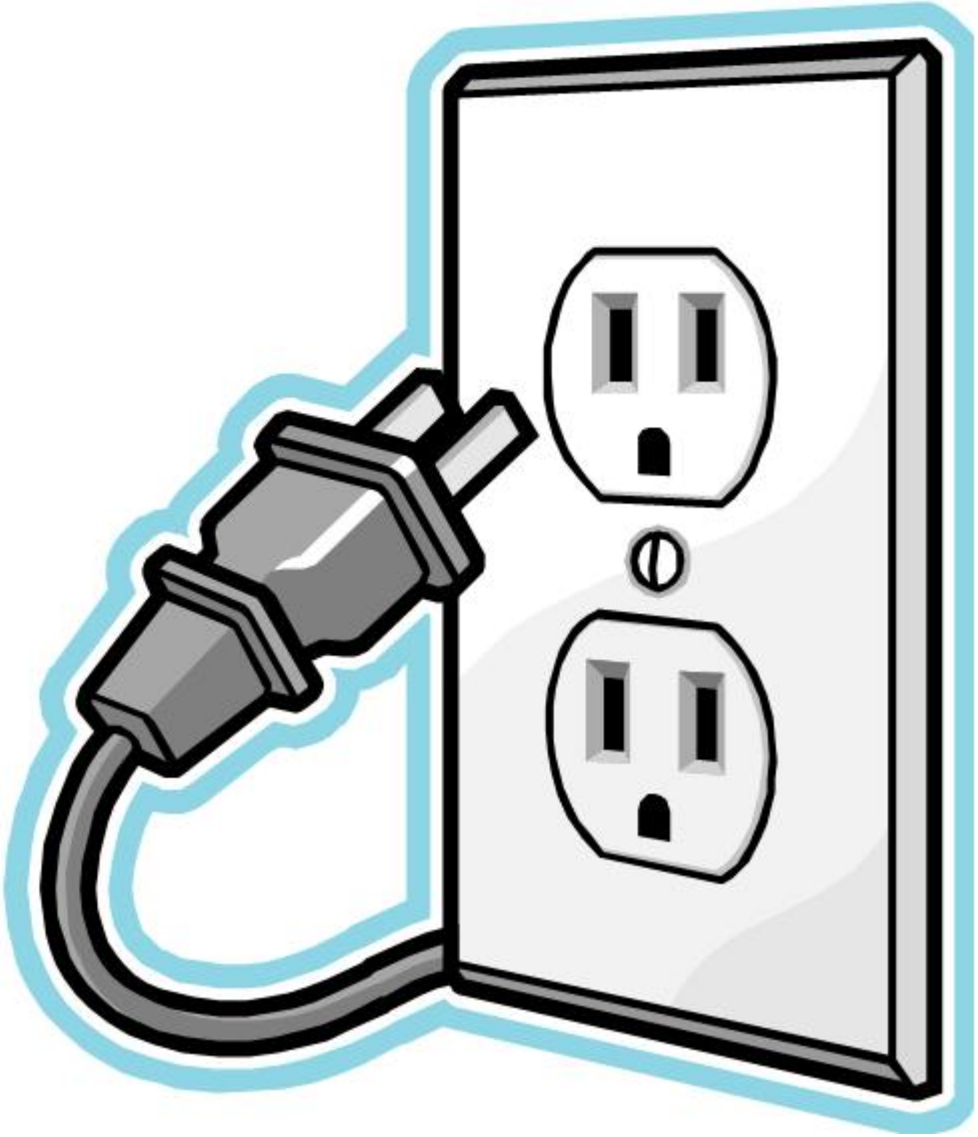
of unique devices

hundreds

of OEMs and
security solutions

What does it mean to be secure?

How to make a computer secure



Lesson #1:
Security is about compromise

Android Security Philosophy

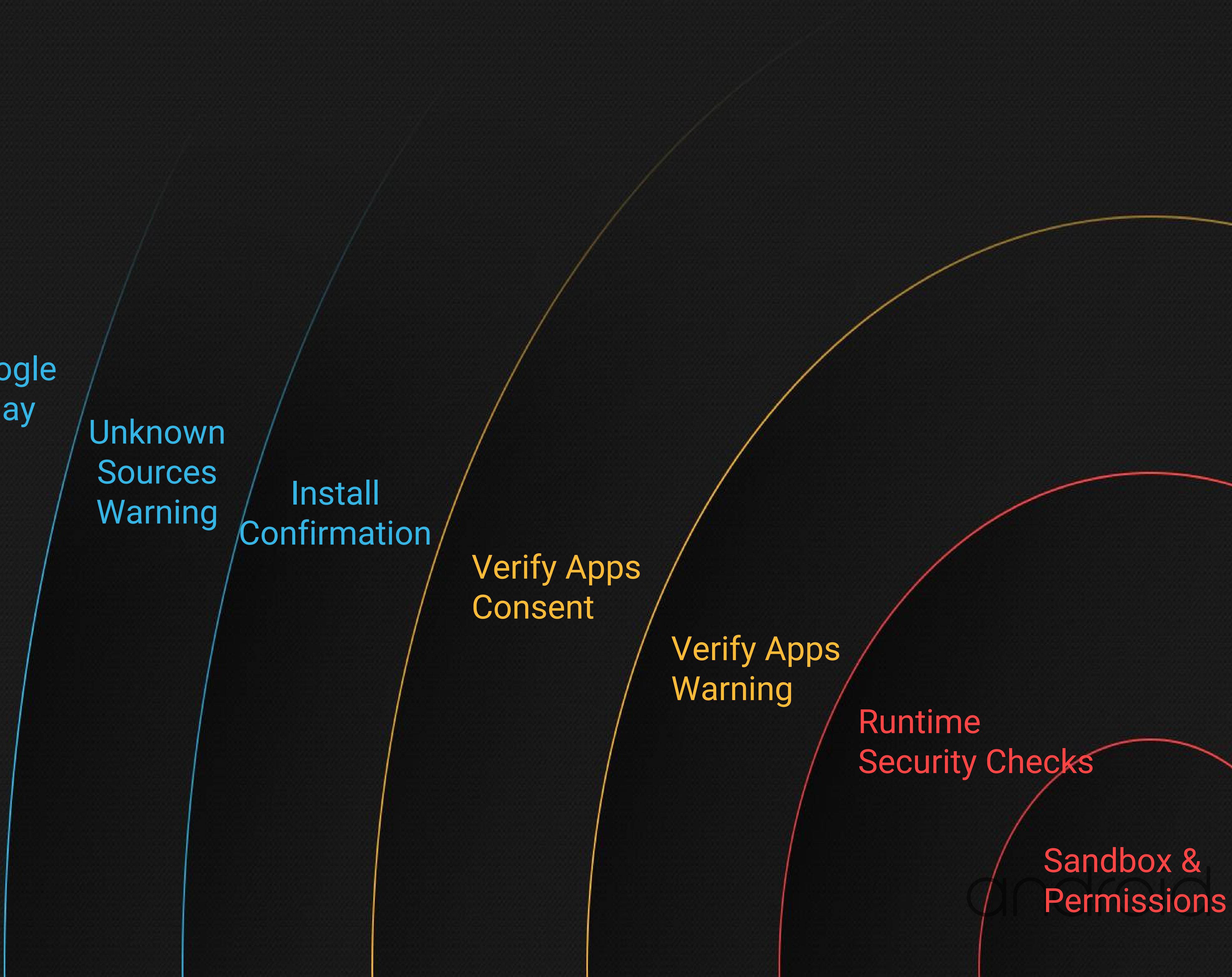
Non-goal

Highly visible, minimally effective, evokes fear.

The goal

Effective security is invisible and evokes calm.

Bridging the gap



Google Play

Unknown Sources Warning

Install Confirmation

Verify Apps Consent

Verify Apps Warning

Runtime Security Checks

Sandbox & Permissions

Four pillars of Android Security

- Prevention
- Detection
- Minimization
- Reaction

First pillar of Android Security: Prevention

Traditional approaches to prevention

- Code audits
- Design reviews
- Outreach and education
- Safe by default design philosophy
- “Red team”

Lesson #2:
Always start with a sandbox

A platform for applications

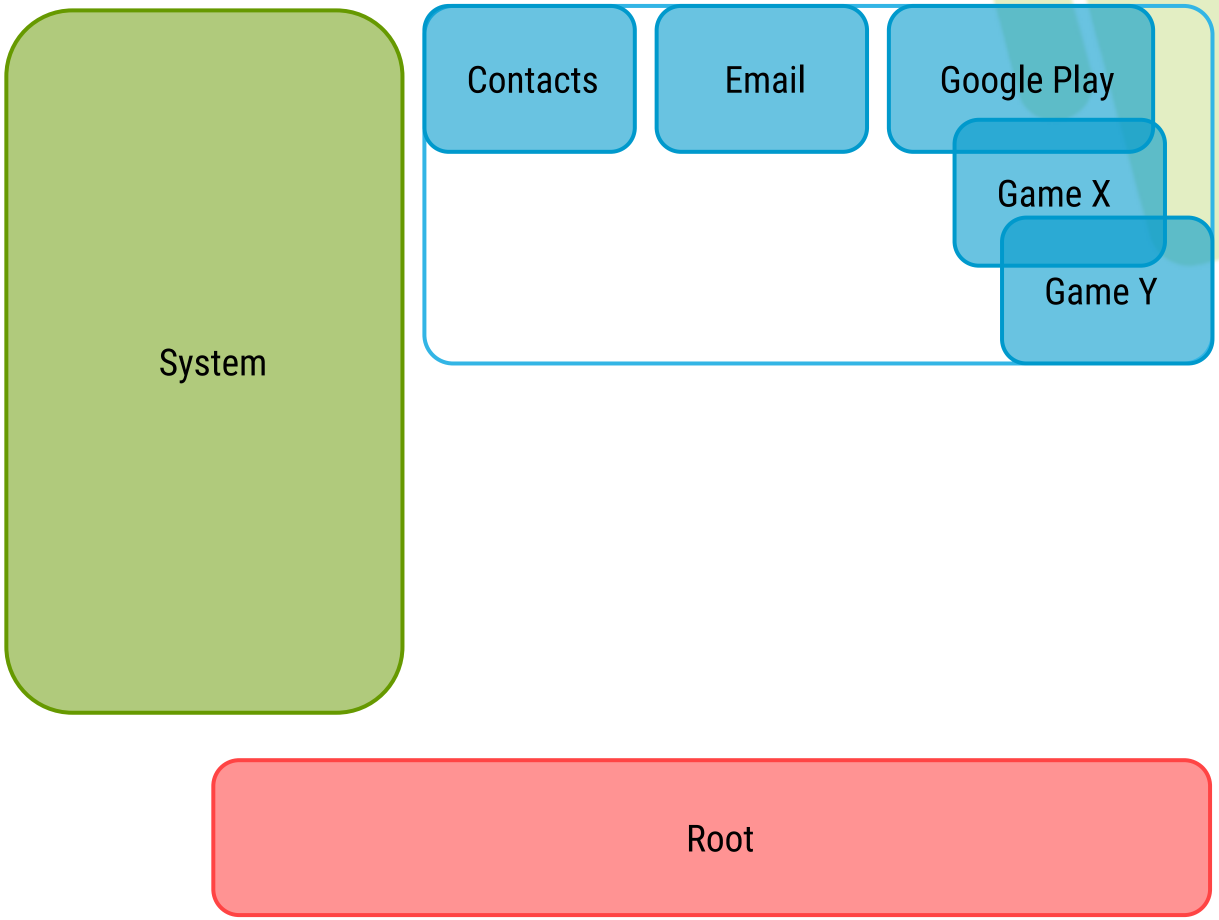
Android Security Evolution

Android verifies application signature and assigns an application sandbox at install time.

Application Sandboxes (including system) isolate data by running each app as it's own UID.

Inter-process communication (IPC) requires mutual request.

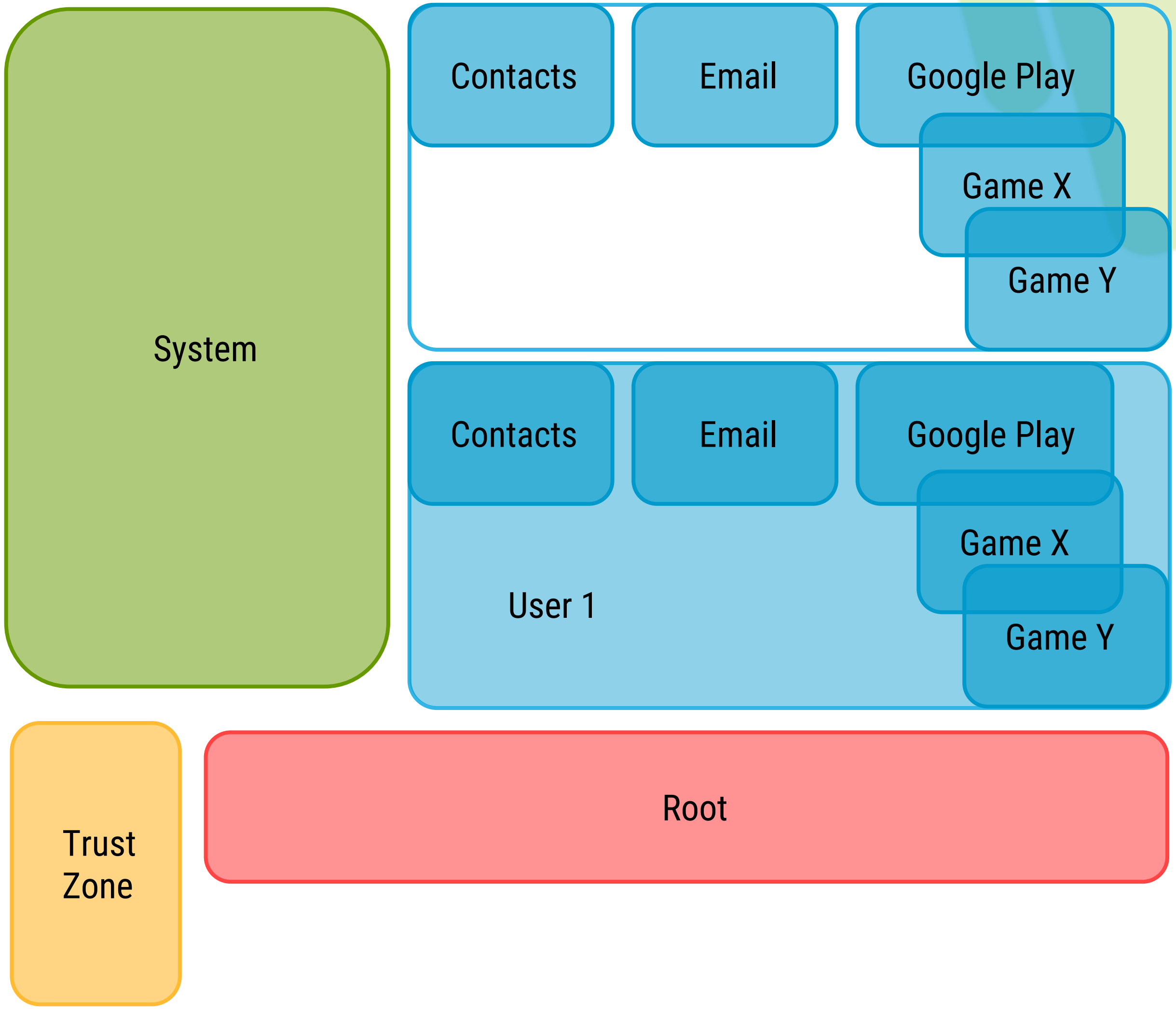
IPC and services may be protected by permissions.



Android Security Evolution – 4.1

Application sandbox extended to groups of applications -- preventing IPC across the user boundary

Developer key store protected from root compromise



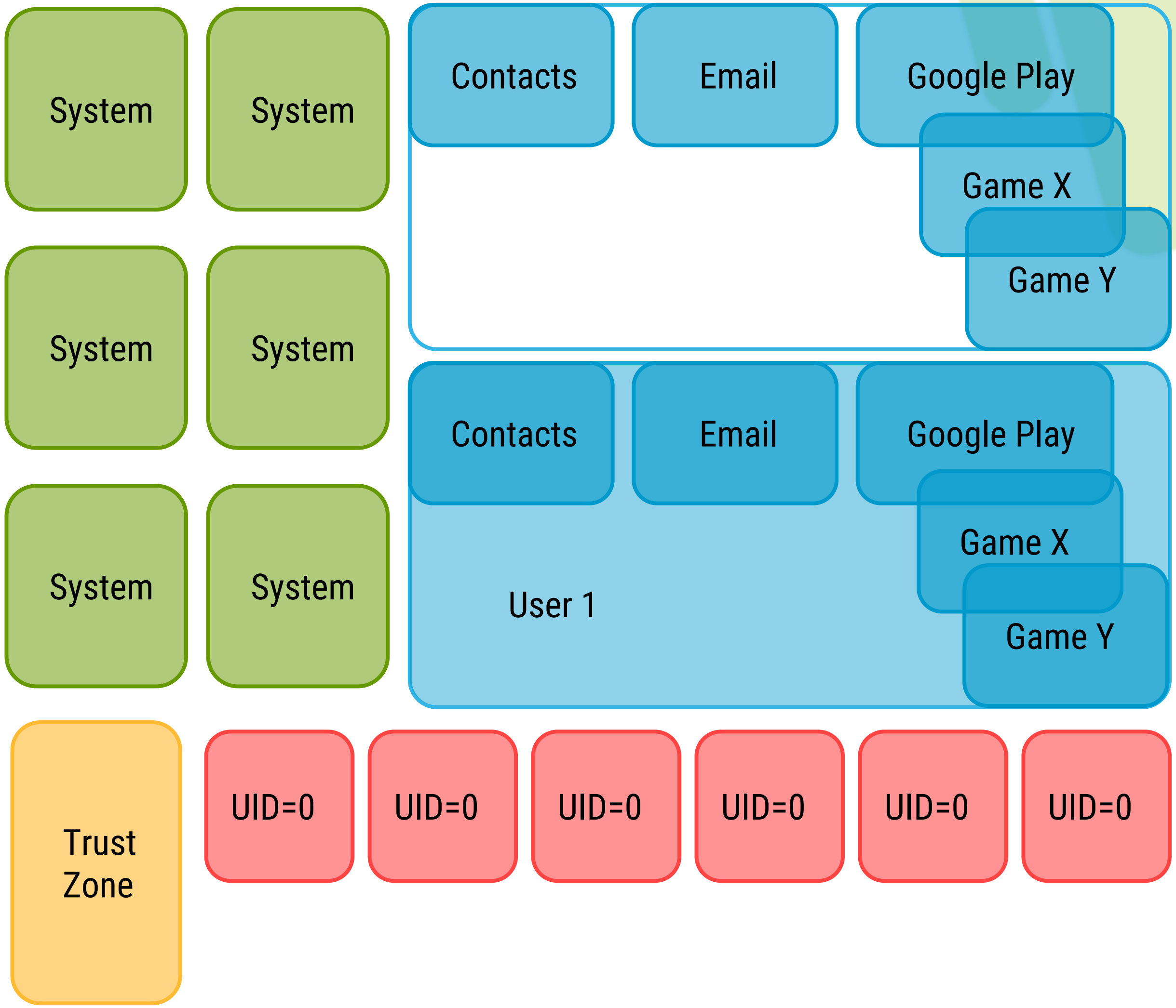
Lesson #3:
Evolve the sandbox as threats emerge

Android Security Evolution – 5.0

Segmentation of system and root UID with constrained SELinux policies

All powerful root no longer exists. Only constrained UID=0

Central security policy allows audit of system & root applications



Android Security Evolution – 5.0

> **Q:** *It might be good for everyone to know: Which Android devices do you find the most secure?*

CunningLogic (aka jcase)

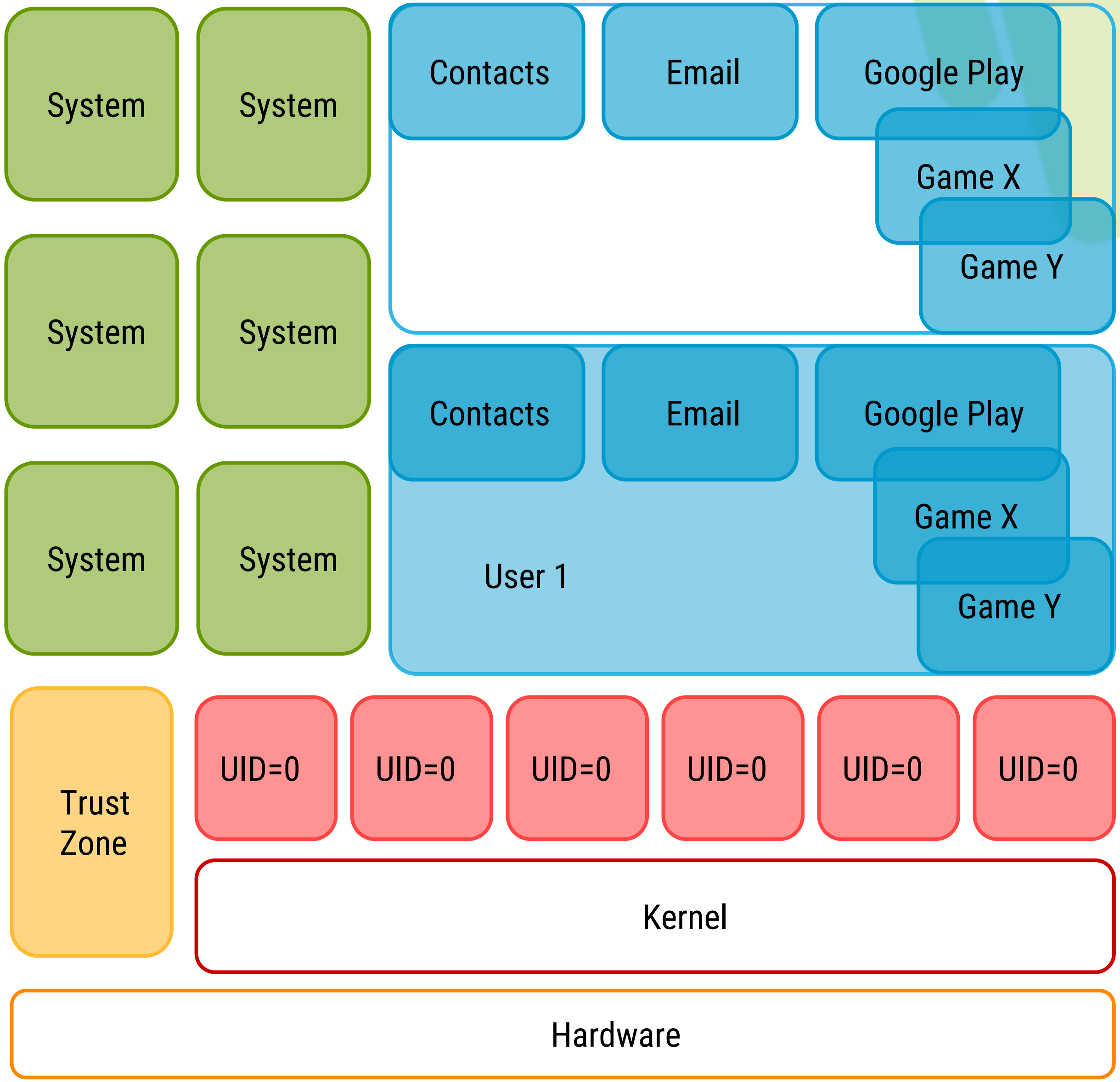
> **A:** *Android 5.x and up is particularly annoying for me to try and root, my go to tactics are often dead due to the strengthened SELinux policies.*

https://www.reddit.com/r/Android/comments/3hhciw/ask_us_almost_anything_about_android_security/

Android Security Evolution

Experimental features in 5.0 provide integrity checking for the full stack.

Supply chain threats are also a focus of research efforts.



Lesson #4: Establish strong security standards

Security Standards – SELinux assertions

- No unlabeled files
- No ptrace
- No device node creation
- No raw I/O
- No mma
- No mac_
- No settin
- No access to /data/security and /data/misc/keystore
- No /dev/mem or /dev/kmem access
- No /proc usermode helpers
- No ptrace of init
- No access to generically labeled /dev/block files
- Restrictions on mounting filesystems

- No execute of files from outside of /system
- No access to /data/properties
- No writing to /system or rootfs

services

access

- No apps acquiring capabilities
- No raw app access to camera, microphone, NFC, radio, etc.
- No app-generic socket access
- No app/proc access to different security domains
- No access to GPS files
- Cannot disable SELinux

Currently ~250 rules

Second pillar of Android security: Minimization

Why minimization?

- Impossible to fix every bug
- Impossible to find every bug
- Robustness in failure
- Maintain the integrity of the system

Lesson #5:
Account for human error

A quiz

Is this statement true?

$$x + 1 > x$$

A quiz

Is this statement true?

$$x + 1 > x$$

Not if you’re a programmer...

Compiler Hardening

- ASLR
- No eXecute Memory
- FORTIFY_SOURCE
- Read-only Relocations
- Stack Canaries
- Non-PIE binaries banned

Compiler Hardening – research

- Research
  - Integer overflow protections
  - CFI (Control Flow Integrity)
  - Safe Stack
  - -fstack-protector-strong

Lesson #6: Encourage safe languages

Language Choice

- Android standardized on memory safe languages
- Native code specifically discouraged:

> *Notably, using native code on Android generally does not result in a noticeable performance improvement, but it always increases your app complexity.* ***In general, you should only use the NDK if it is essential to your app—never because you simply prefer to program in C/C++.***

https://developer.android.com/tools/sdk/ndk/index.html

Language Choice – research

- Our industry needs to discourage memory unsafe languages
  - Too risky and error prone
- Early research on C replacements for Android
  - Suggestions welcome!

Principle of least privilege

"Every program and every user of the system should operate using the least set of privileges necessary to complete the job."

J. H. Saltzer and M. D. Schroeder, “The protection of information in computer systems”, pp. 1278-1308, Proceedings of the IEEE 63, number 9, September 1975

Case Study – libstagefright

- Designed with containment in mind
  - UID sandbox
  - SELinux sandbox
- Exploit mitigations effective
  - ASLR
  - SELinux no-exec rules

Third pillar of Android Security: Detection

Lesson #6:
Keep your ears to the ground

Multiple methods of discovering bugs

- security@android.com
- Android bug database
- Academic research / journals
- Automated monitoring of forums
- Failed exploit detection
- Android Security Rewards Program

Android Security Rewards Program

Severity	Bug	Test case	CTS / patch	CTS+Patch
Critical	$2,000	$3,000	$4,000	$8,000
High	$1,000	$1,500	$2,000	$4,000
Moderate	$500	$750	$1,000	$2,000
Low	$0	$333	$500	$1,000

Android Security Rewards Program

- $10K - local to kernel
- $20k - remote to kernel
- $20k - local to trustzone
- $30k - remote to trustzone

Up to $38,000 per issue

https://g.co/AndroidSecurityRewards

Fourth pillar of Android Security: Reaction

Lesson #7:
Have an update strategy

Updates

nexus

- Monthly Security Updates
- Monthly Security Bulletins
- 3 years from device availability

Not just about OS updates...

- 3rd party apps are important too
- 1.6 million apps in Google Play
- Identified security vulnerabilities
  - OpenSSL
  - Private Keys in Apps
  - Apache Cordova Update
  - Exposed Credentials
- All of them are getting fixed

There is no such thing as perfect security.

Lesson #8:
Strive for accurate risk assessments

On risk

Vulnerability	News Headline	Unique APKs	Peak exploitation after public release (per install)	Exploitation before public release (absolute)
Master Key	99% of devices vulnerable	1231	< 8 in a million	0
FakeID	82% of Android users at risk	258	<1 in a million	0

Source: Google Safety Net Data

On risk

As an industry, we should provide better data about actual risk and focus more attention on calming users while protecting them.

https://static.googleusercontent.com/media/source.android.com/en//devices/tech/security/reports/Google_Android_Security_2014_Report_Final.pdf

Closing

In closing

- Android grew up in the Internet age, and learned from 40 years of digital security experience.
- Robust, sophisticated, multi-layer security model.
- Open platform ensures Android will continue to evolve to meet new threats.

Questions?

Nick Kralevich
nnk@google.com

security@android.com