android

Android: protecting the kernel

Jeff Vander Stoep

August 26th, 2016

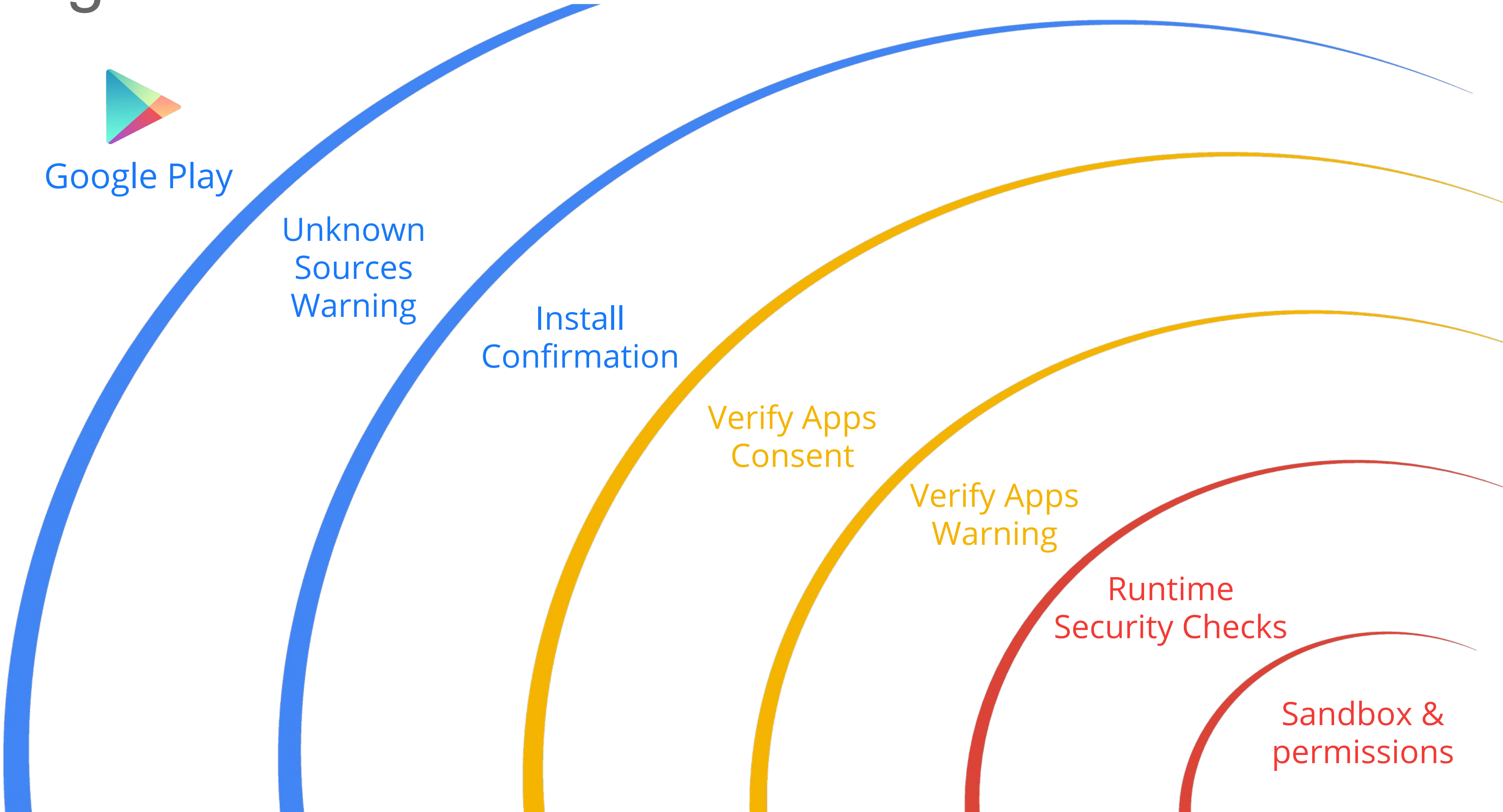
$ whoami

- Jeff Vander Stoep - jeffv@google.com
  @jeffvanderstoep
- Android Security since 2014
- Focus on system hardening
- Software Engineer

First - some good news

- Most kernel bugs discussed are not directly reachable to untrusted code, due to Android's security model. Android Nougat further reduces the attack surface of the kernel.
- New kernel defenses address our biggest category of kernel bugs. (more on this later)

More good news



Agenda

- Kernel bugs in Android
  - Focus on biggest categories (we only have 45 minutes)
- Bugs by cause
  - Mitigations - memory protections
  - Gaps
- Bugs by reachability
  - Mitigations - attack surface reduction
  - Gaps
- Future work

android

Kernel bugs have a long life. Fixing bugs is necessary but not sufficient!

android

Goal:

Use data on security vulnerabilities to prioritize mitigation development and adoption

android

About the data

- Includes kernel bugs January 2014 -> April 2016
- Includes low → critical severity kernel bugs
- Moderate → critical taken directly from public Nexus security bulletins
- Low severity bugs included because the definition of low has changed over time (many bugs previously listed as low considered moderate under new ratings)

“At the operating system level, the Android platform provides the security of the Linux kernel…”

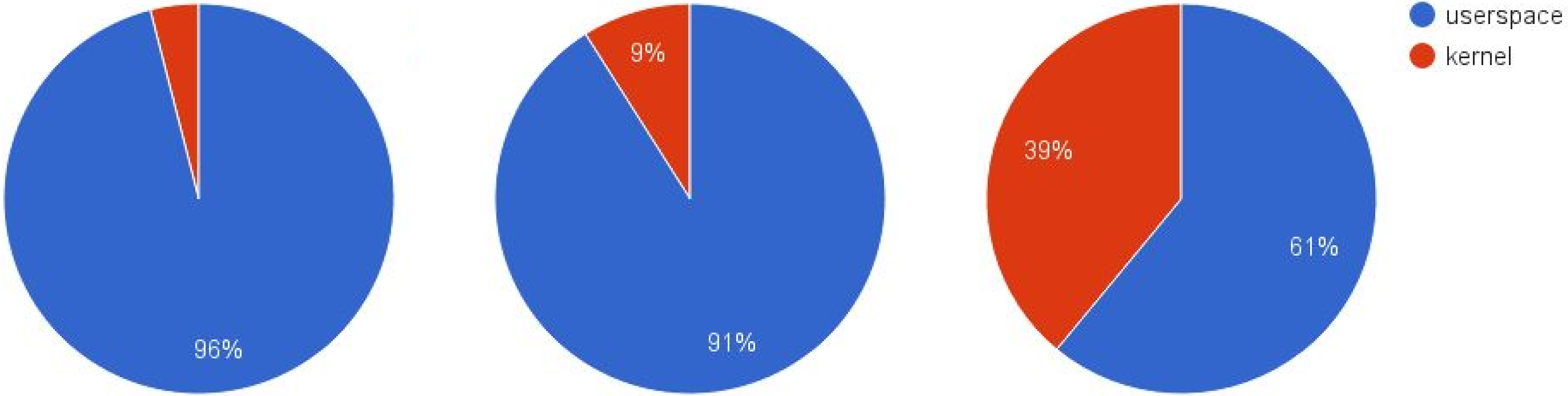
source.android.com

android

Security from the kernel

- Address space separation/process isolation
- unix permissions
- DAC capabilities
- SELinux
- seccomp
- namespaces
- ...

Android security bugs by year



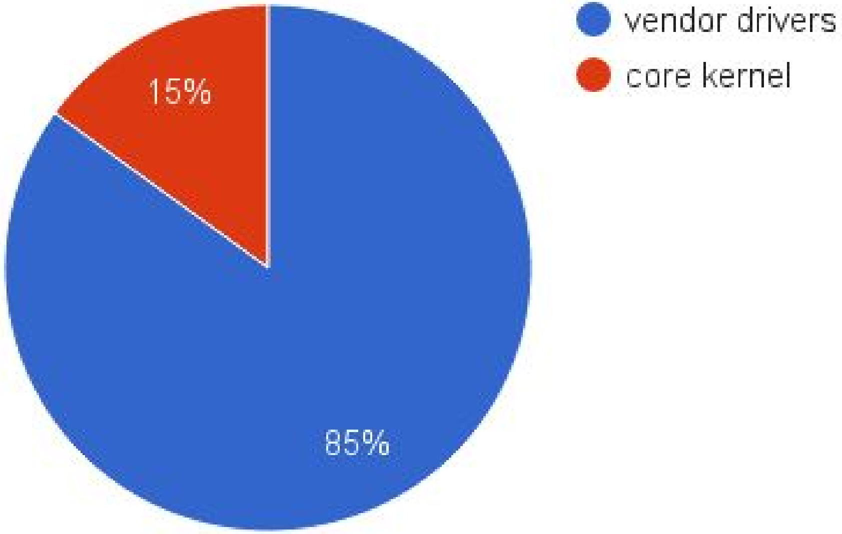
2014 2015 2016

Data: Jan 2014 → April 2016

Why the rise in kernel bugs?

- Lockdown of userspace makes UID 0 significantly less useful.
- 2016 is the first year > 50% of devices in ecosystem have selinux in global enforcing.
- Kernel bugs payout more $$$ (Android vulnerability rewards)

Where Android's kernel bugs are born



Data includes multiple vendors

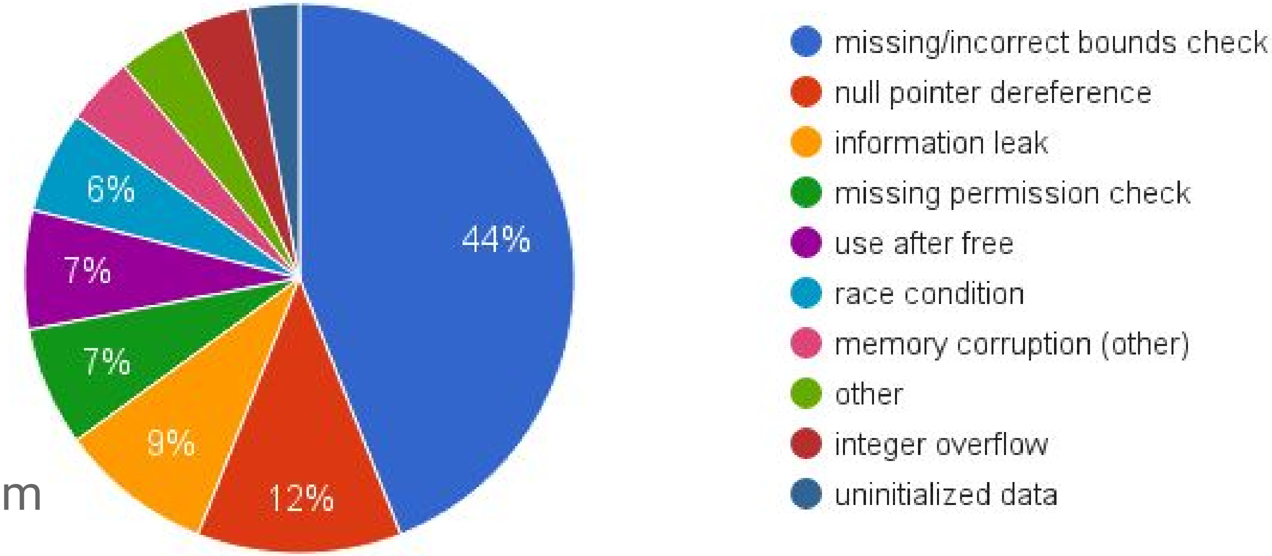
Some vendor drivers are in upstream, others are out-of-tree

Data: Jan 2014 → April 2016

Kernel defenses protect against in-tree AND out-of-tree vulnerabilities

android

What causes Android's kernel bugs?

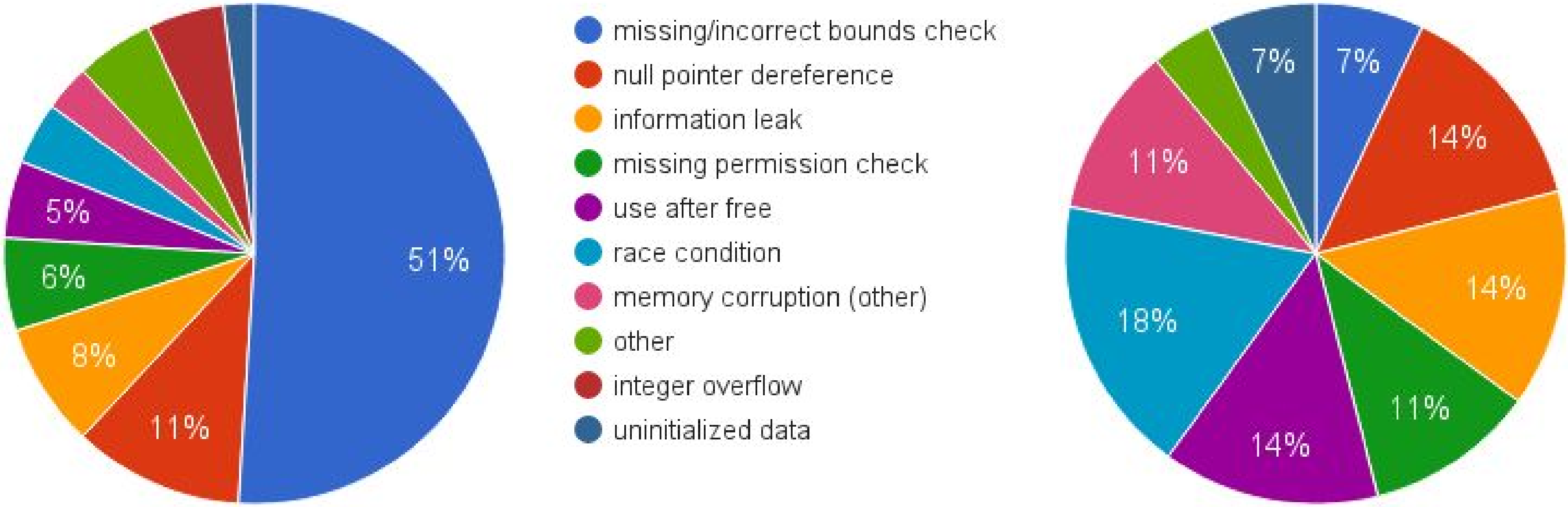


Bugs from upstream kernel and device specific bugs

Data: Jan 2014 → April 2016

android

What causes kernel bugs?



Vendor drivers

Core kernel

Data: Jan 2014 → April 2016

android

Mitigations - missing/incorrect bounds check

- Hardened usercopy
  - Protect against incorrect bounds checking in copy_*_user()
- PAN emulation
  - Protect against kernel access to userspace bypassing hardened usercopy changes.

Landing in upstream kernel!

Mitigations - missing/incorrect bounds check

- Stack protector strong
  - protects against stack buffer overflows
- KASLR (arm64 android-4.4 kernel)
  - Makes code reuse attacks probabilistic
- PXN - make userspace non-executable for the kernel
  - Protects against ret2user attacks
- RODATA - mark kernel memory as read-only/no-execute
  - Makes code non-writeable, and data non-executable

Mitigations - null pointer dereference

- CONFIG_LSM_MMAP_MIN_ADDR
  - Make null pointer dereference unexploitable (just crash)
- PAN emulation also make null pointer dereference non-exploitable

Gaps - code review

Code quality of upstream is significantly better than device specific drivers

- What can be done to enforce better code quality?
  - Compiler changes e.g. integer overflow checking
  - Scripts e.g. checkpatch.pl
  - Runtime changes - e.g. PAN enforce proper use of copy_*_user()
  - KASAN
  - Constification

Attack surface reduction

Remove or restrict access to entry points into the kernel

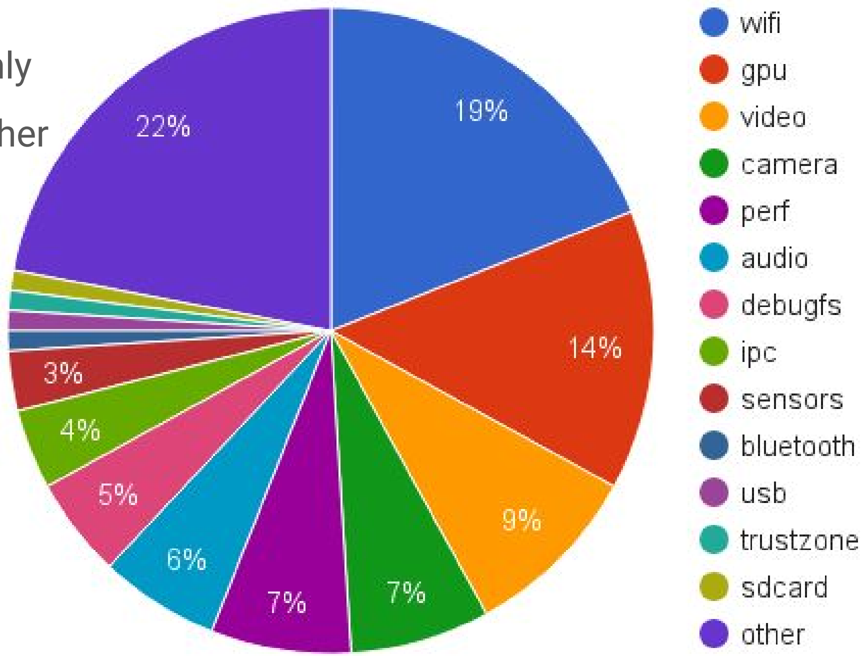
Attack surface reduction

Gate access to kernel provided developer tools in developer settings.

android

How are kernel bugs reached - driver/subsystem

Includes many bugs only reachable by root or other privileged processes.

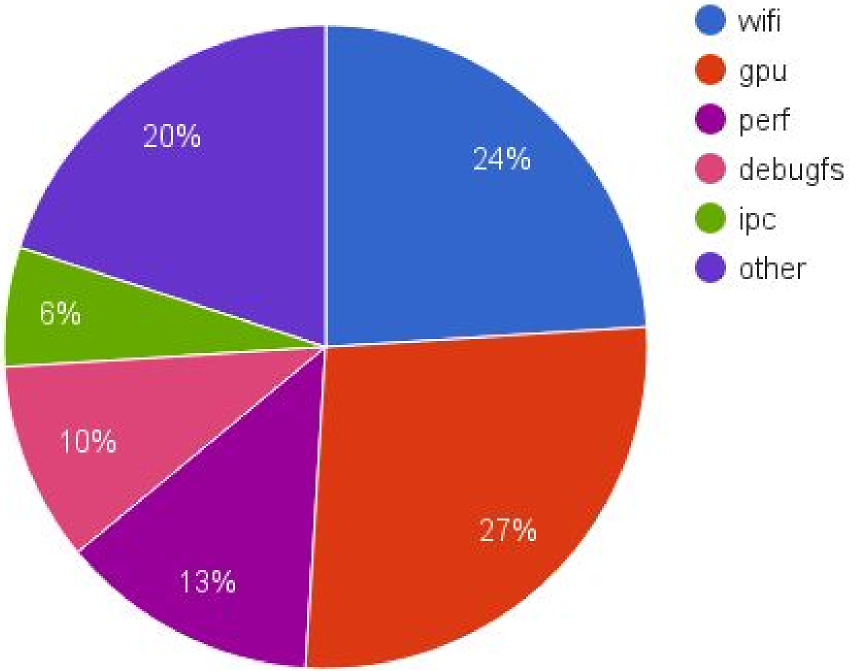


Data: Jan 2014 → April 2016

Bugs reachable by unprivileged apps

Fix all bugs, but prioritize mitigation development for bugs that are reachable by apps

More on this later...

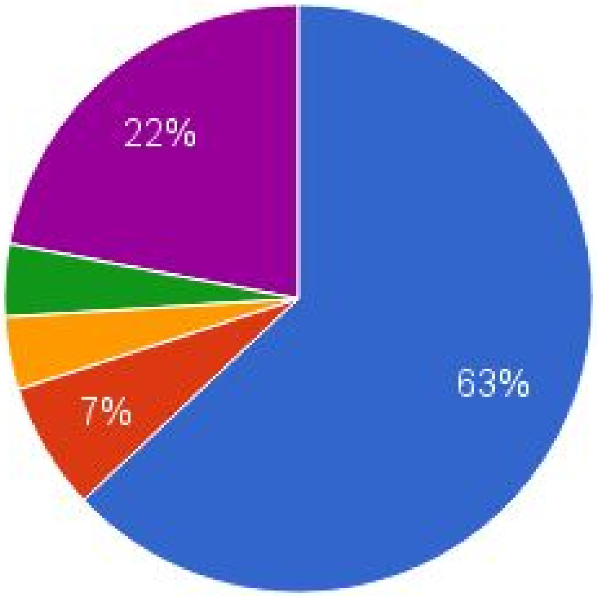


Data: Jan 2014 → April 2016

Case study: Wifi driver bugs

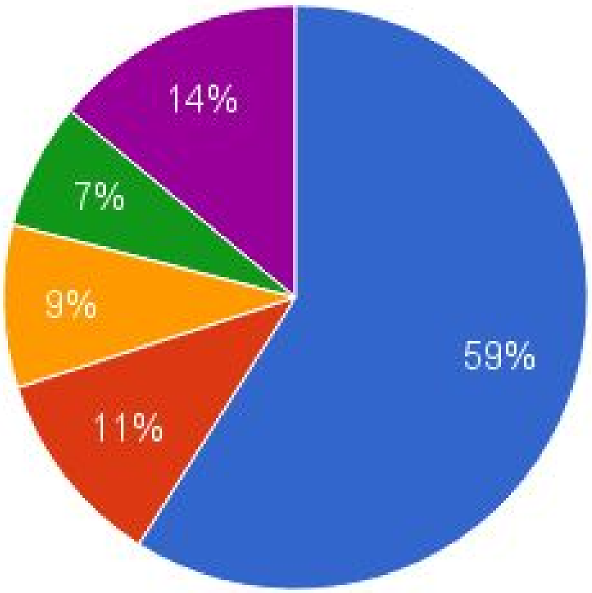
- Every app-reachable bug should have been protected by a CAPABLE(CAP_NET_ADMIN) check.
- Relying on developers to correctly implement in-code checks is risky.
- Better to have privileged behavior guarded by auditable security policy.
- Many wifi driver bugs were reachable via local unix sockets! Add strong policy around all socket types.

How are kernel bugs reached - syscall (before mitigations)



all bugs

100% of perf vulns introduced in vendor customizations



bugs reachable by apps

Data: Jan 2014 → April 2016

android

Mitigations - attack surface reduction

Ioctl command whitelisting in SELinux

- Wifi
    - Originally hundreds of ioctl commands → 29 whitelisted safe network socket ioctls
    - Blocks access to all bugs without restricting legitimate access.
    - Unix sockets: wifi ioctls reachable by local unix sockets :(Hundreds → 8 whitelisted unix socket ioctls
    - No ioctls allowed on other socket types including generic and netlink sockets
- GPU
    - e.g. Shamu originally 36 -> 16 whitelisted commands
    - Ioctl commands needed varies by device but < 50% needed seems consistent across KGSL drivers

android

Mitigations - attack surface reduction

- Restrict access to perf
  - Access to perf_event_open() is disabled by default.
  - Developers may re-enable access via debug shell
- Remove access to debugfs
  - All app access to debugfs removed in N
- Remove default access to /sys
  - App access to files in /sys must be whitelisted
- Seccomp required for all devices (minijail shoutout!)

Impact of mitigations

Because most bugs are driver specific, effectiveness of mitigations varies across devices. In general most previously reachable bugs were made unreachable

- Case study of bugs reachable by apps on Nexus 6 (Shamu)
  - 100% of wifi bugs blocked
  - 50% of GPU bugs blocked
  - 100% of debugfs bugs blocked
  - 100% of perf bugs blocked (by default)

Gaps - Attack surface reduction

- Need more/better controls over kernel feature accessibility.
  - Controls allow us to do what's best for both Linux developers and users of Linux based products.
- Argument inspection for seccomp

android

Future work

Kernel devs, we need more/better safety features (seat belts)!

Sometimes seat belts are inconvenient…

Those “other” categories - potential attack surface reduction

Architecture	syscalls provided by kernel	syscalls in bionic	reduction (%)
arm	364	204	44
arm64	271	198	27
x86	373	203	46
x86_64	326	199	39

android

Those “other” categories - Memory safety

Where do we go from here?

- Kernel self protection project - get involved!
- Principle of Least Privilege
- Attack Surface Reduction
- Defense-in-depth
- Continue to find/fix bugs!

QUESTIONS ?