black hat USA 2017

JULY 22-27, 2017
MANDALAY BAY / LAS VEGAS

Honey, I Shrunk the Attack Surface

Adventures in Android Security Hardening

Nick Kralevich - Blackhat

July 27th, 2017

Agenda

Brief history of Android security

Strategies for dealing with vulnerabilities

Bugs and attack surface reduction efforts

Recognition

The future

Questions?

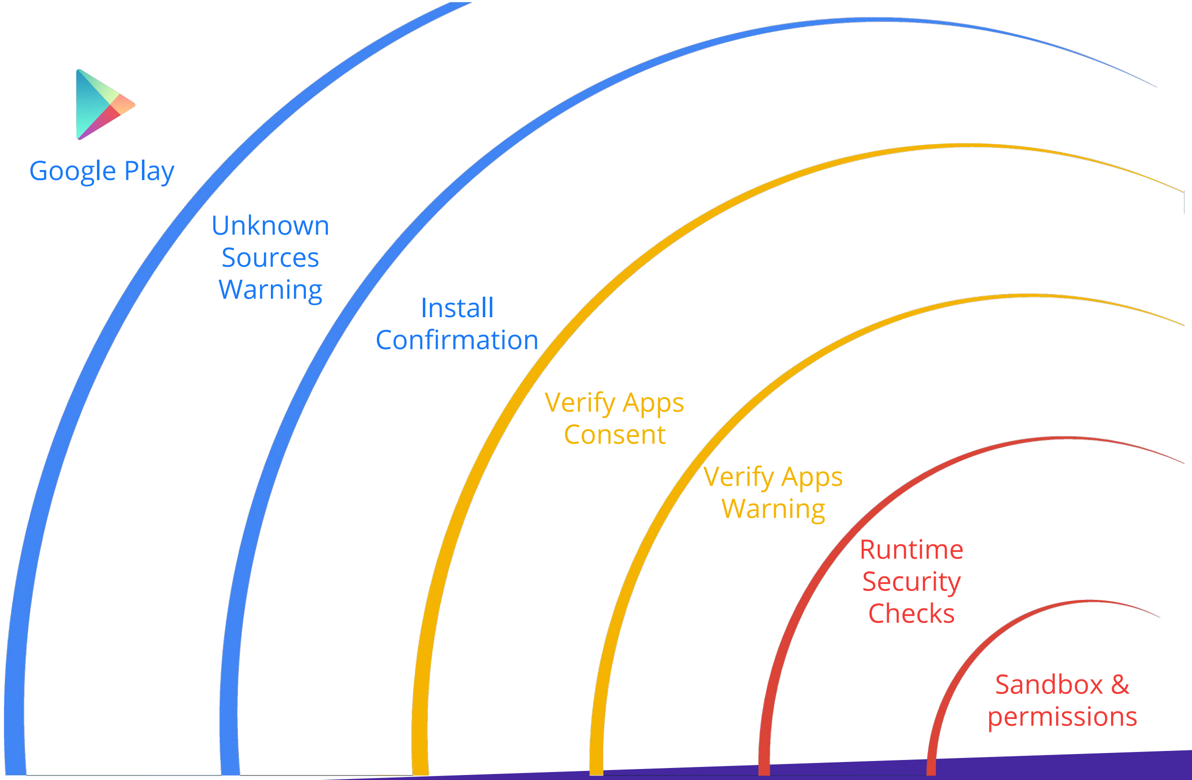
android

Before we begin...

Android security is more than device security...

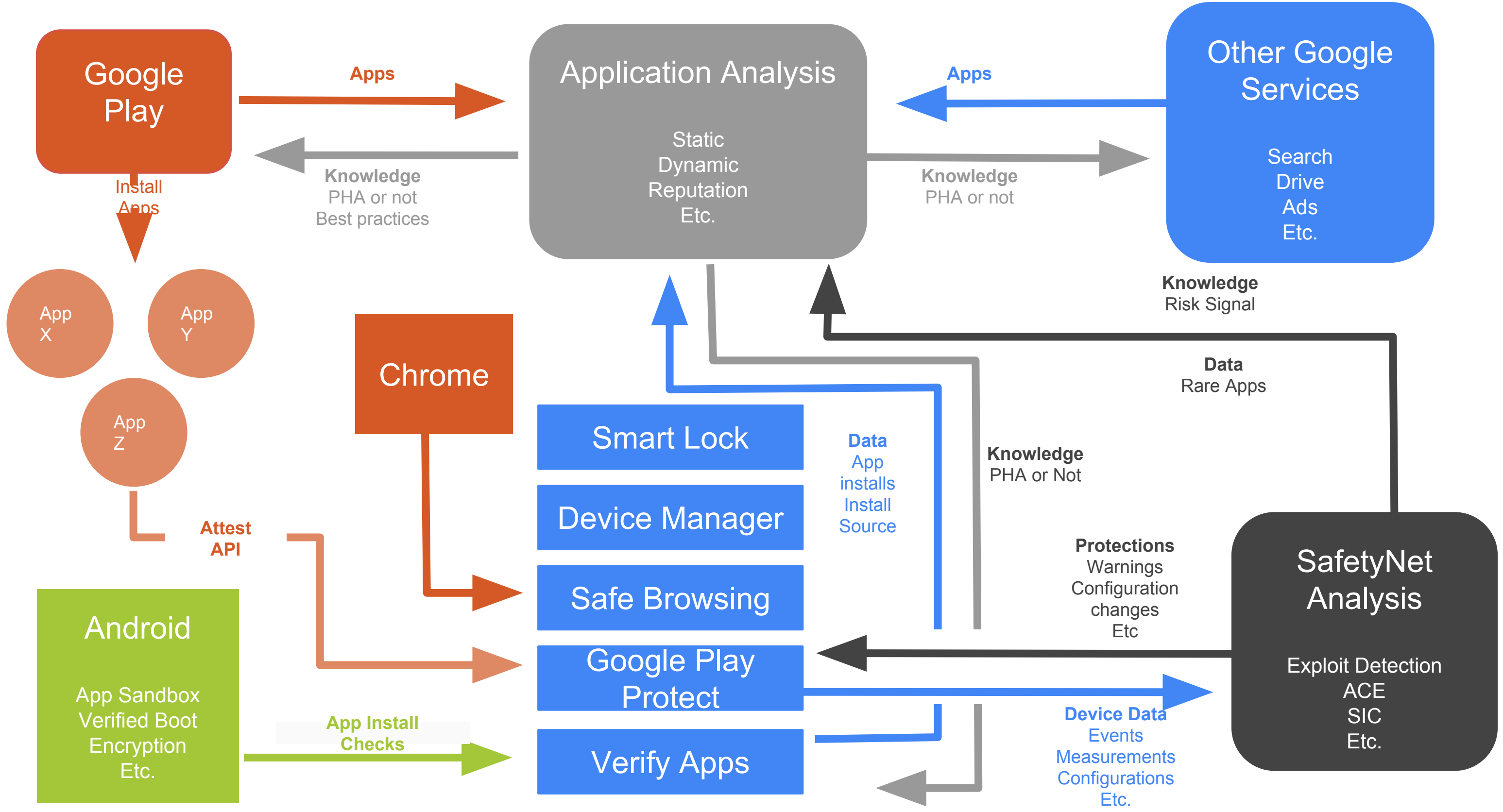
android

Layers of defense even before code gets to the device...



android

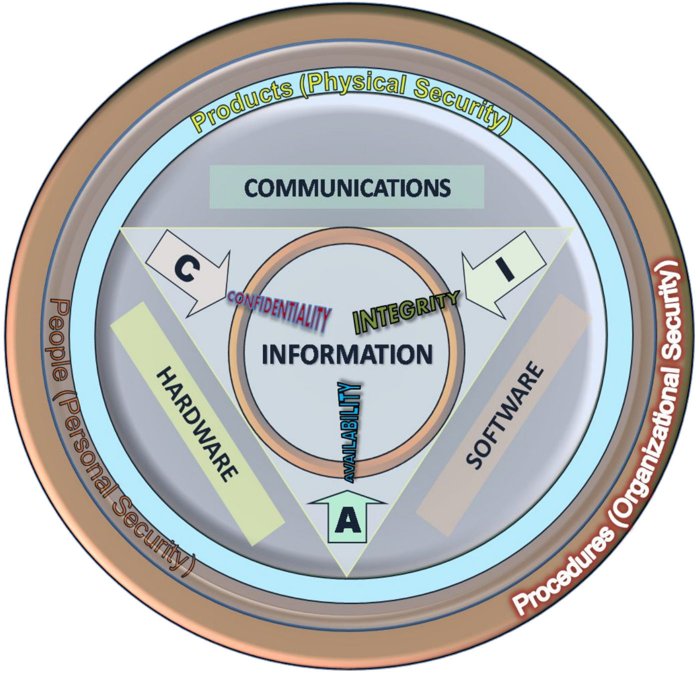
Technology throughout Google working together



Key Principles

Key Android Security Principles

- Exploit Mitigation
- Attack Surface Reduction
  - Exploit Containment
  - Principle of Least Privilege
- Safe by design APIs and interfaces
- Architectural Decomposition



Stepping back in time...

android

10 years ago...

- Windows Vista was released
  - Replaced "administrator-by-default" philosophy of Windows XP
- All desktop OSes
  - No difference between application capabilities and user capabilities (remains mostly true today)
  - User has Administrator / root access (still mostly true today)
- Mobile devices
  - Primarily feature phones
  - Smart devices not widely available

Android enters the picture

- HTC Dream - October 22nd, 2008
  - First commercially available Android device
- Centralized application store
- Application sandboxing
- Memory safe programming language (Java)
- Designed with security in mind
- Strong desire to not repeat the security mistakes of legacy consumer OSes

Early Android Security

- Exploit mitigation technologies were the primary focus
    - -fstack-protector
    - ASLR
    - NX
    - FORTIFY_SOURCE
    - mmap_min_addr
    - Format string vulnerabilities
    - etc...

https://source.android.com/security/enhancements/

Early Android Security

Applications sandboxed using Linux UID technologies. Sandboxing of other processes done on a limited basis.

Global “root” user which was unconstrained and targeted for attack.

IPC boundaries were not consistently defined and enforced.

Security “policy” not auditable.

http://powerofcommunity.net/poc2016/keen.pdf

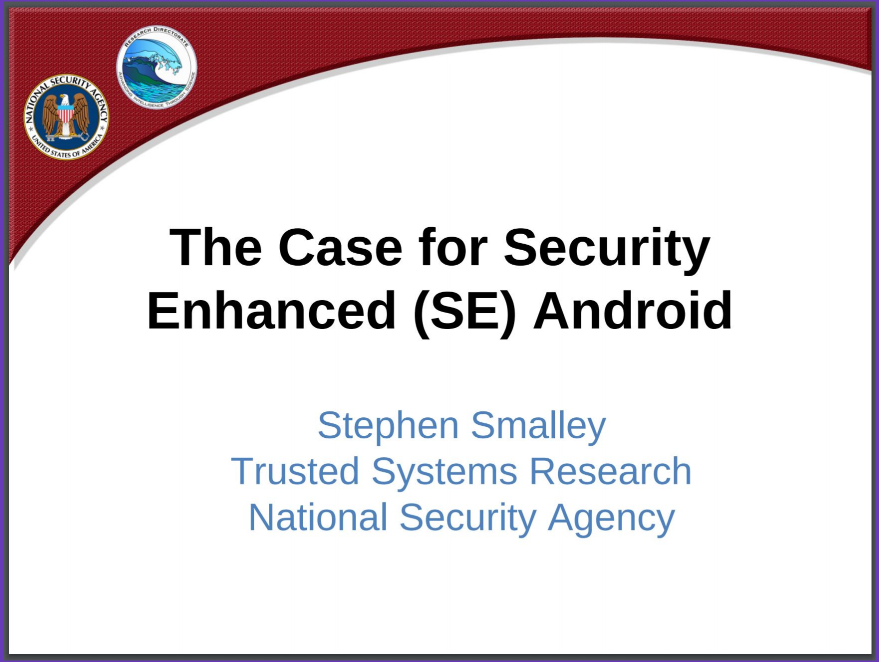
android

Heavy early use of discretionary access control (DAC) tools.

- Address space separation/process isolation
- UID controls
- UNIX permissions
- DAC capabilities
- namespaces
- ...

Greater focus on compartmentalization, attack surface reduction

- Sept 2011
- Proven effectiveness at preventing or mitigating 7 rooting exploits
- Oct 2013: Android 4.4 partially enforcing
- Oct 2014: Android 5.0 fully enforcing



Immediate success in mitigating exploits!

- vold “asec create” exploit (Android 4.4)
- Constrained attack surface mitigated exploit
- Blocked several ways
  - /data/local/tmp directory and file access disallowed
  - No symlink following allowed
  - Mount restrictions

- http://www.androidpolice.com/2014/06/04/android-4-4-3-patch-finally-closes-ancient-vulnerability-shuts-several-serious-security-exploits/
- https://plus.google.com/u/0/+JustinCaseAndroid/posts/7BxgPNc7ZJs?cfem=1

Modern Day Android Security

Every process compartmentalized (including UID=0 processes)

- "root" no longer exists on Android

Principle of least privilege widely deployed

Attack surface limited through tightly controlled IPC boundaries

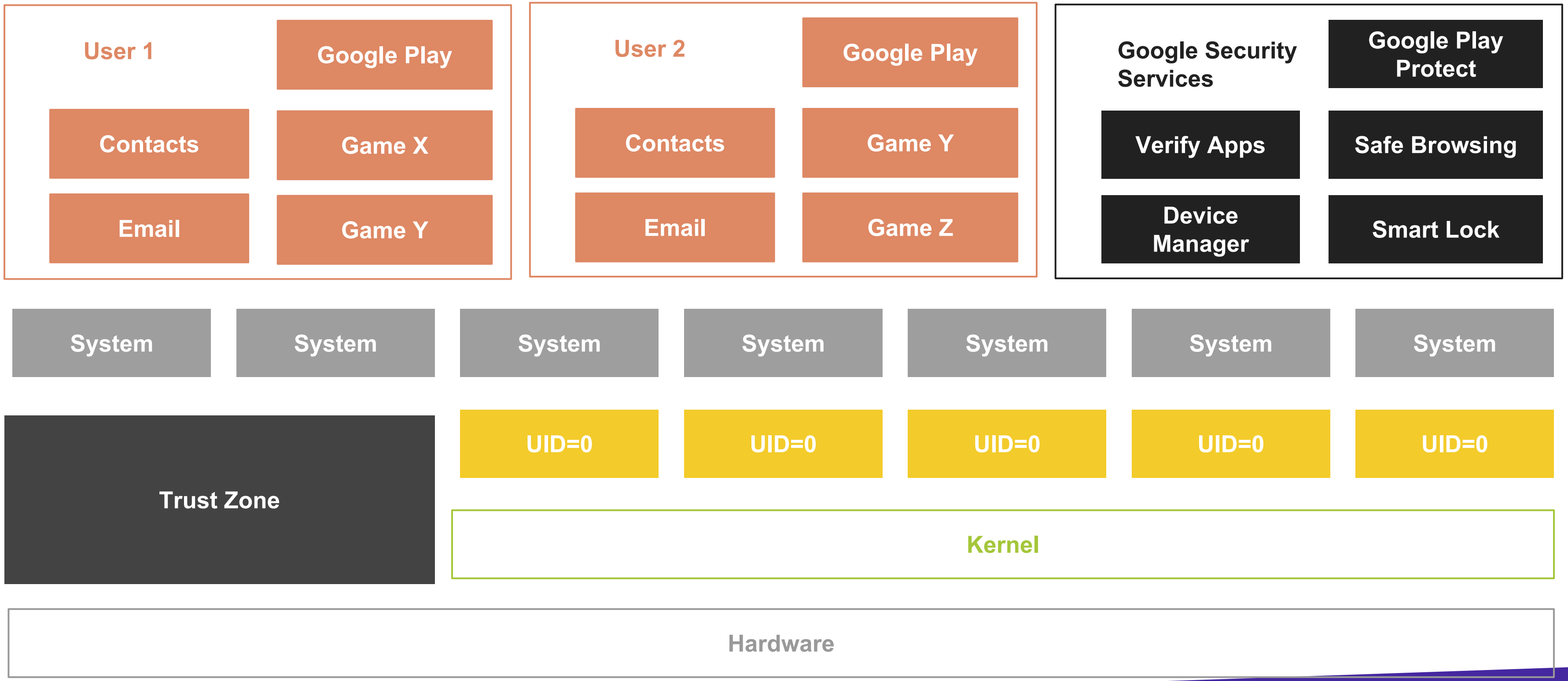
Auditable security policy

Most executable code comes from signed source / cryptographically verified (dm-verity).

http://powerofcommunity.net/poc2016/keen.pdf

android

Android Today



android

Attack Surface Reduction Examples

android

CVE-2017-6074: DCCP double-free vulnerability (local root)

EDITION: US

ZDNet

VIDEOS SMART CITY WINDOWS 10 CLOUD INNOVATION SECURITY ENTERPRISE IOT MORE NEWSLETTER

MUST READ WHAT TO EXPECT FROM THE WINDOWS 10 CREATORS UPDATE

Linux's decade-old flaw: Major distros move to patch serious kernel bug

Google fuzzer helps find 11-year-old memory-corruption flaw in the Linux kernel.

By Liam Tung | February 23, 2017 -- 14:50 GMT (06:50 PST) | Topic: Security

Networking Protocols

- Only a whitelist of socket families are allowed
  - Netlink Route Sockets
  - Ping Sockets
  - TCP / UDP Sockets
  - Unix stream and datagram sockets
- Whitelist allowed ioctls

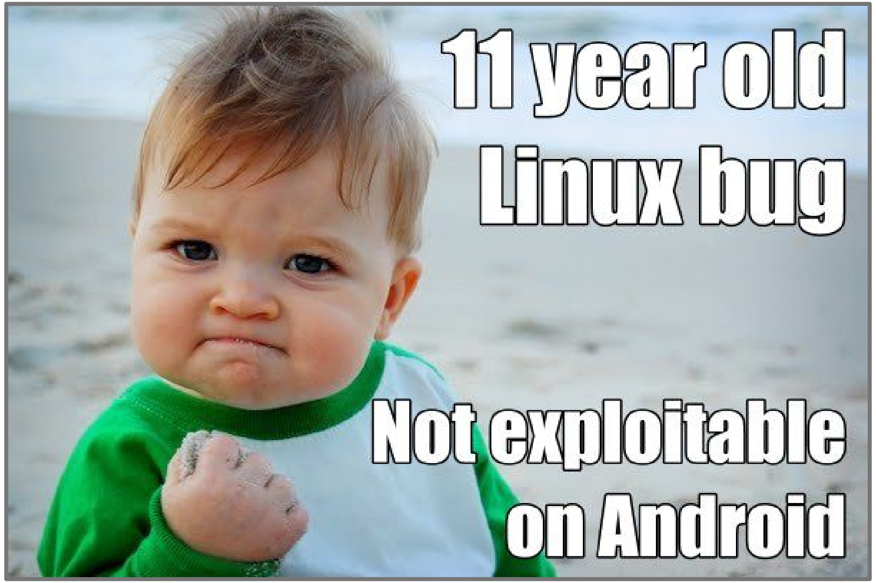
```
# Restrict socket ioctls. Either
# 1. disallow privileged ioctls,
# 2. disallow the ioctl permission, or
# 3. disallow the socket class.

neverallowxperm untrusted_app domain:{ rawip_socket
tcp_socket udp_socket } ioctl priv_sock_ioctls;

neverallow untrusted_app *:{ netlink_route_socket
netlink_selinux_socket } ioctl;

neverallow untrusted_app *:{
  socket netlink_socket packet_socket key_socket
  appletalk_socket netlink_firewall_socket
  netlink_tcpdiag_socket netlink_nflog_socket
  netlink_xfrm_socket netlink_audit_socket
  netlink_ip6fw_socket
  netlink_dnrt_socket netlink_kobject_uevent_socket
  tun_socket netlink_iscsi_socket
  netlink_fib_lookup_socket netlink_connector_socket
  netlink_netfilter_socket netlink_generic_socket
  netlink_scsitransport_socket
  netlink_rdma_socket netlink_crypto_socket
} *;
```

android

CVE-2017-6074: DCCP double-free vulnerability (local root)

Layers of attack surface reduction

- Not compiled into Android common kernels
- Even if compiled in, not reachable due to SELinux restrictions.
- “dodged a bullet” -> “working as intended”



Whitelisted socket families - Other bugs mitigated

- Other bugs blocked
  - **CVE-2016-2059** - Linux IPC router binding any port as a control port
  - **CVE-2015-6642** - Security Vulnerability in AF_MSM_IPC socket: IPC_ROUTER_IOCTL_LOOKUP_SERVER ioctl leaks kernel heap memory to userspace
  - **CVE-2016-2474** - Security Vulnerability - Nexus 5x wlan driver stack overflow
  - etc...

CVE-2017-7184: xfrm kernel heap out-of-bounds access

eWEEK

Ubuntu Linux Falls on Day 1 of Pwn2Own Hacking Competition

By: Sean Michael Kerner | March 16, 2017

The first day of the Trend Micro-sponsored Pwn2Own competition awards $233,000 in prize money to security researchers for exploiting software with previously unknown vulnerabilities.

The Pwn2Own hacking competition began on March 15, and security researchers have already successfully exploited Ubuntu Linux, Microsoft Edge, Apple Safari and Adobe Reader. In total,

CVE-2017-7184: xfrm kernel heap out-of-bounds access

- Compiled into Android kernels
- Requires CAP_NET_ADMIN
    - Available to lots of processes on Android.
- Requires netlink_xfrm_socket
    - Who has it?

CVE-2017-7184: xfrm kernel heap out-of-bounds access

- Reachability:
  - Only available to one process!
  - Effectively unreachable.

```
nnk@nick:/android$ adb pull /sys/fs/selinux/policy
/sys/fs/selinux/policy: 1 file pulled. 8.5 MB/s (451031 bytes in 0.051s)

nnk@nick:/android$ sesearch --allow -c netlink_xfrm_socket -p create ./policy
allow netmgrd netmgrd:netlink_xfrm_socket { nlmsg_write setopt setattr read lock
create nlmsg_read write getattr connect shutdown bind getopt append };
```

TL;DR:

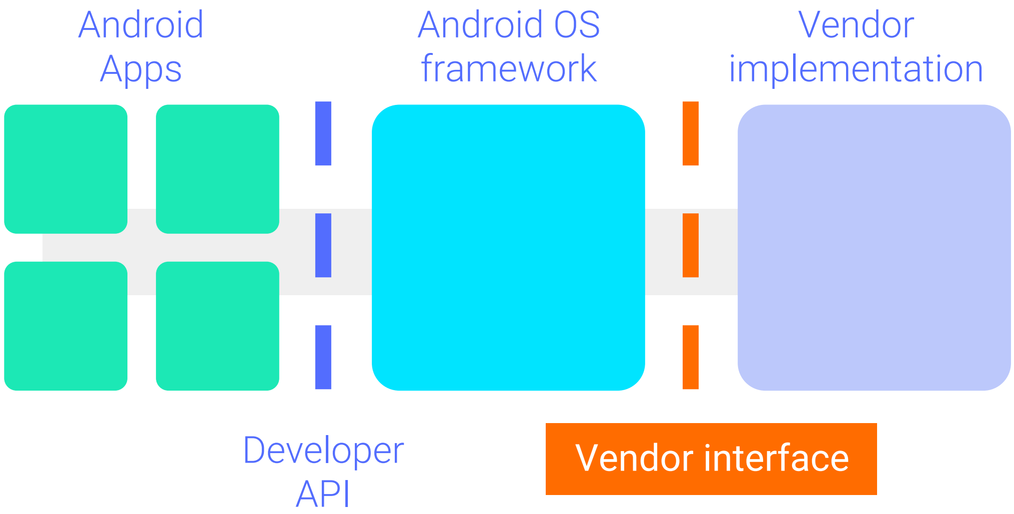
Careful attack surface management kept these bugs from being reachable.

Attack Surface Management
Android O: Project Treble

android

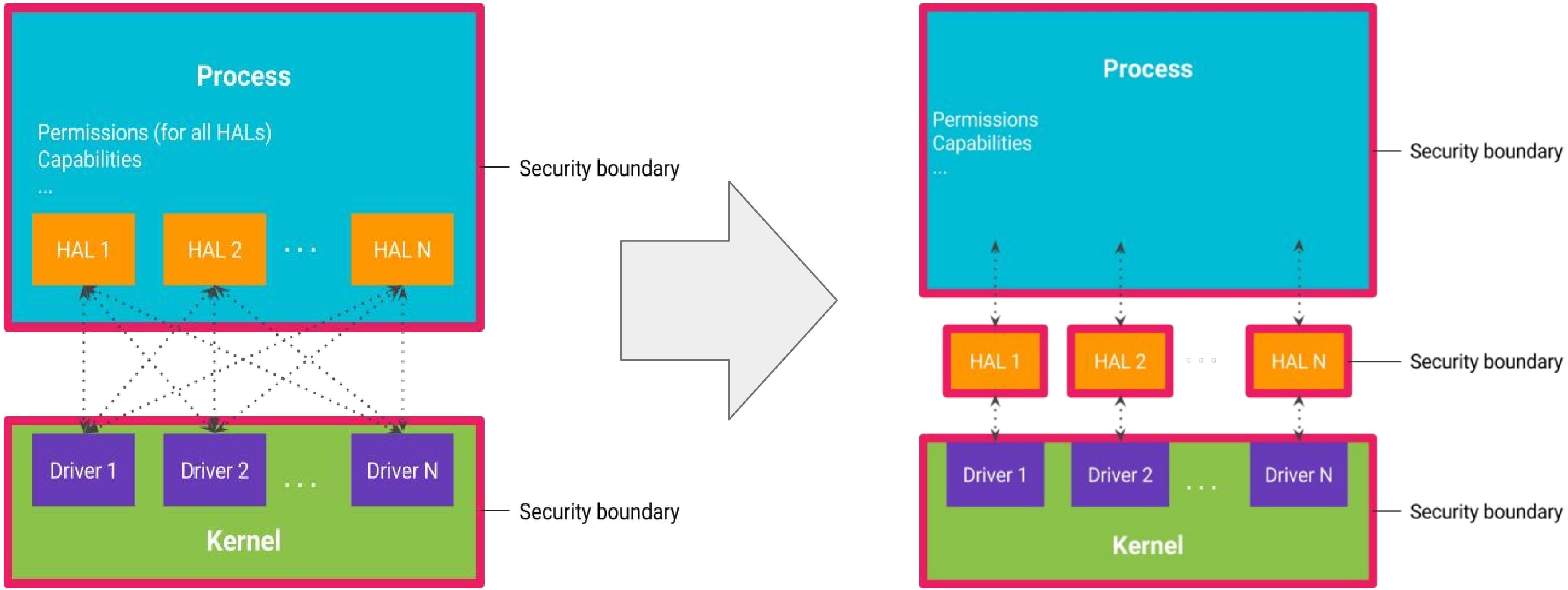
Project Treble

- A modular base for Android
- Allows updating Android without additional work from silicon vendor
- Strong separation and APIs between vendor and Android code



https://android-developers.googleblog.com/2017/05/here-comes-treble-modular-base-for.html

Project Treble - Attack Surface Management



https://android-developers.googleblog.com/2017/07/shut-hal-up.html

Project Treble - Attack Surface Management

- Each HAL runs in its own sandbox
    - Limited to only capabilities needed
- Calling process no longer requires HAL permissions
    - Example: 20 HALs moved out of system_server
- Longer attack chain to the most vulnerable drivers

https://android-developers.googleblog.com/2017/07/shut-hal-up.html

Mediaserver hardening

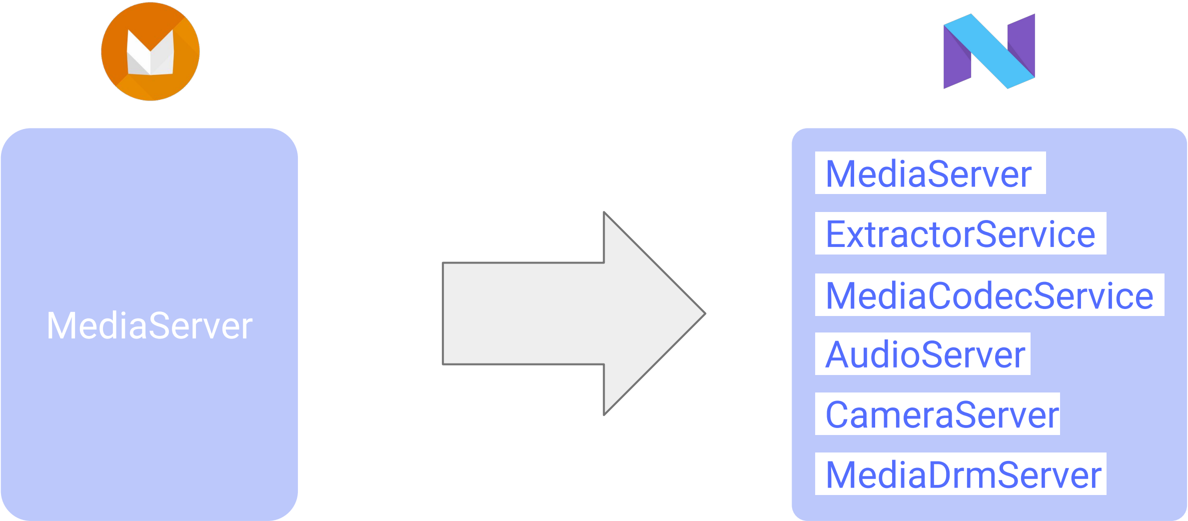
Stagefright

- Series of bugs discovered mid 2015
- Integer overflow in parsing process
- Mediaserver architected for containment with minimal attack surface
- Mediaserver grew up. More features => more capabilities
- Android’s first “successful failure”
    - No evidence of widespread exploitation for 2 years now.

```
meterpreter > # boom! we are now inside the mediaserver process executing in mem
ory!
[-] Unknown command: #.
meterpreter > getuid
Server username: uid=1013, gid=1013, euid=1005, egid=1005
meterpreter > # however... mediaserver is limited both by its privileges (which
are pretty high honestly) and SELinux policy
[-] Unknown command: #.
meterpreter > # we cant even read the shell...
[-] Unknown command: #.
meterpreter > download /system/bin/sh sh
[-] stdapi_fs_stat: Operation failed: 1
meterpreter > #
```

https://twitter.com/jduck/status/756197298355318784

Media Stack Hardening in Nougat



https://www.blackhat.com/docs/us-16/materials/us-16-Kralevich-The-Art-Of-Defense-How-Vulnerabilities-Help-Shape-Security-Features-And-Mitigations-In-Android.pdf

mediaextractor: seccomp

Significant reduction in syscall attack surface

Architecture	arm	arm64	x86
Allowed syscalls	42	34	42
Kernel syscalls	364	271	373
Percent reduction	89%	87%	88%

```
finit_module(5, "", 0) = ?
ERESTART_RESTARTBLOCK (Interrupted by signal)
--- SIGSYS {si_signo=SIGSYS, si_code=SI_USER,
si_pid=20745, si_uid=2000} ---
+++ killed by SIGSYS +++
Bad system call
```

```
$ cat mediaextractor-arm64.policy
# Organized by frequency of system call
# - in descending order for best performance.
ioctl: 1
futex: 1
prctl: 1
write: 1
getpriority: 1
close: 1
dup: 1
mmap: 1
munmap: 1
openat: 1
mprotect: 1
madvise: 1
getuid: 1
...
```

mediaserver: additional changes

- Signed and unsigned integer overflow protections
- Remove “execmem”
    - No anonymous executable memory
- No loading executable code from outside /system (not new in Nougat)
- Executable content can only come from dm-verity protected partition
- … and more

```
open("/system/lib/libnetd_client.so",
O_RDONLY) = 3
mmap2(NULL, 12904, PROT_READ|PROT_EXEC,
MAP_PRIVATE, 3, 0) = 0xb6d9f000

open("/data/data/com.foo.bar/libnetd_client.
so", O_RDONLY) = 4
mmap2(NULL, 12904, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_FIXED, 4, 0) = -1 EACCES
(Permission denied)

mmap2(NULL, 20,
PROT_READ|PROT_WRITE|PROT_EXEC,
MAP_PRIVATE|MAP_ANONYMOUS, 4, 0) = -1 EACCES
(Permission denied)
```

android

mediaserver - Refactoring results

- Vastly improved architectural decomposition
- Vastly improved separation of privileges
- Riskiest code moved to strongly sandboxed process
- Containment model significantly more robust

https://android-developers.blogspot.com/2016/05/hardening-media-stack.html

“I started working on this exploit on a build of the upcoming Android N release, and anyone sitting near my desk will testify to the increased aggravation this caused me. A lot of general hardening work has gone into N, and the results are impressive.”

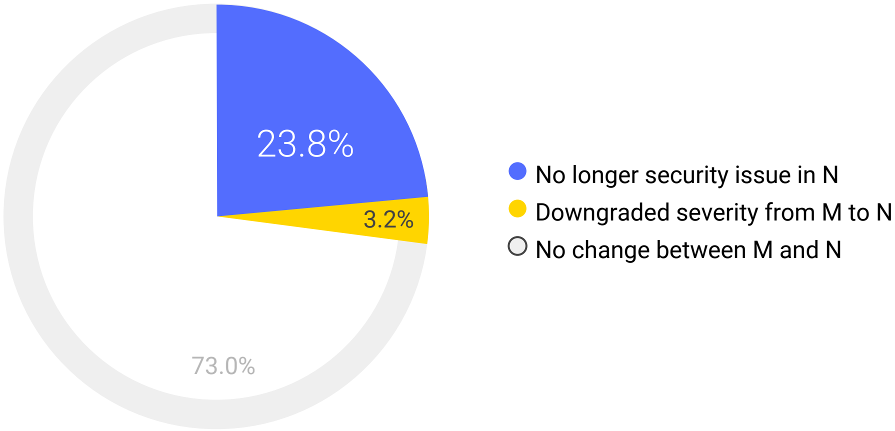
Mark Brand
Google Project Zero

https://googleprojectzero.blogspot.com/2016/09/return-to-libstagefright-exploiting.html

android

Mediaserver hardening effectiveness

Security bulletin bugs
in the media stack
for the first 4 months
of 2017



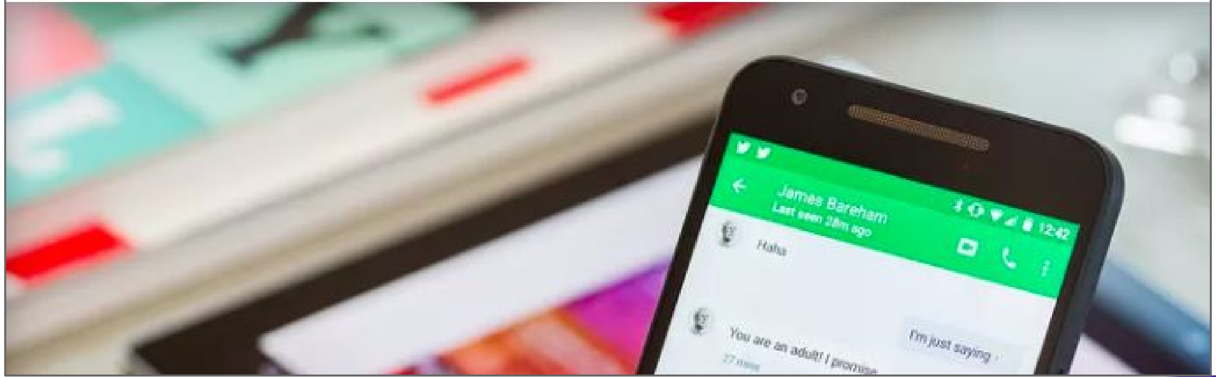
android

Google rebuilt a core part of Android to kill the Stagefright vulnerability for good

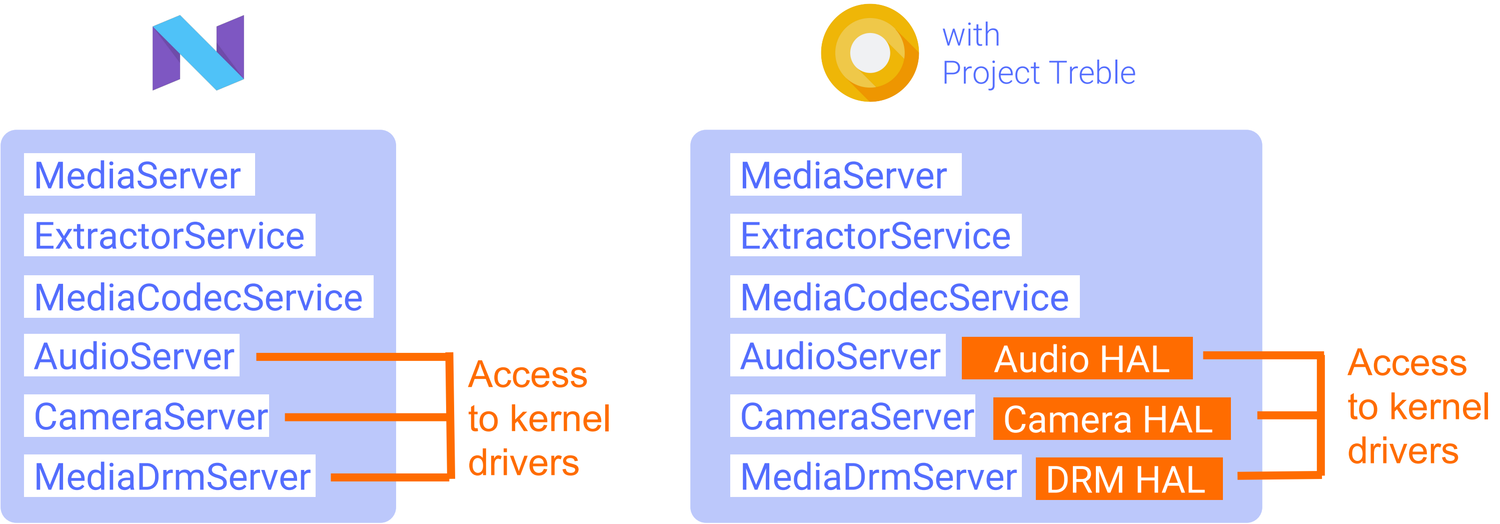
Android 7.0 has a few new security tricks up its sleeve

By Russell Brandom · @russellbrandom · Sep 6, 2016, 1:03p

SHARE TWEET LINKEDIN PIN



Media Stack Hardening Improvements in O



Android O:
Webview Security

android

Webview Security

KitKat

Shipped with the operating system

Lollipop

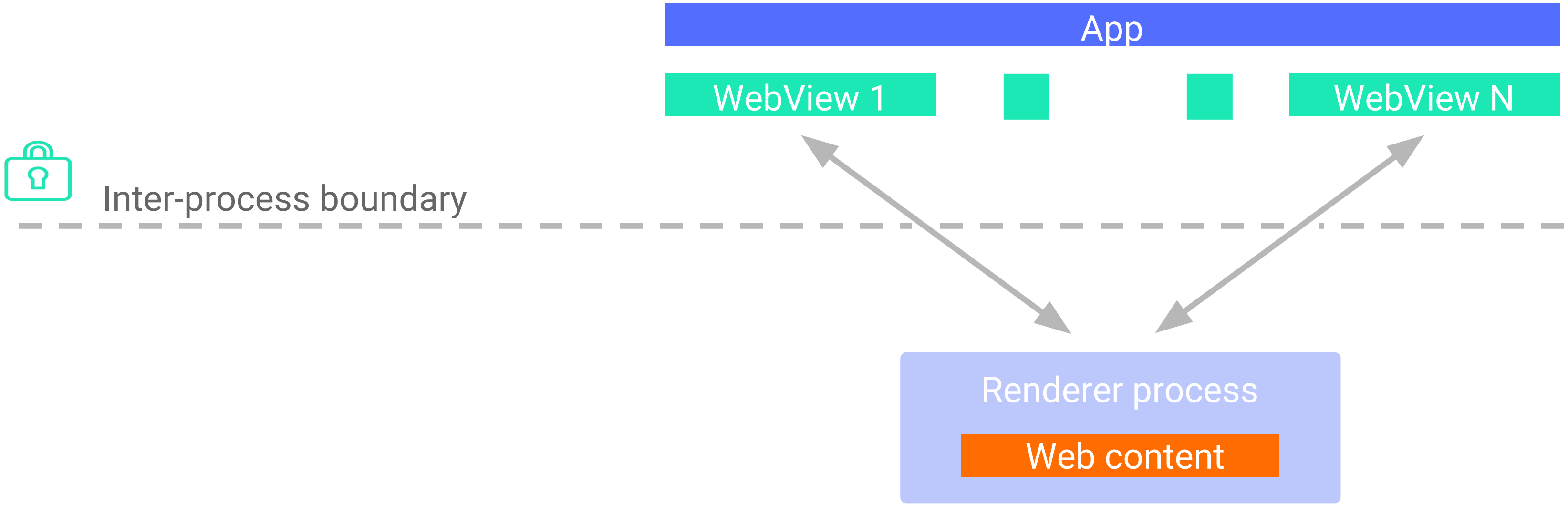
Separate APK updateable via the Play store

O Preview

Renderer in isolated process

Safe Browsing

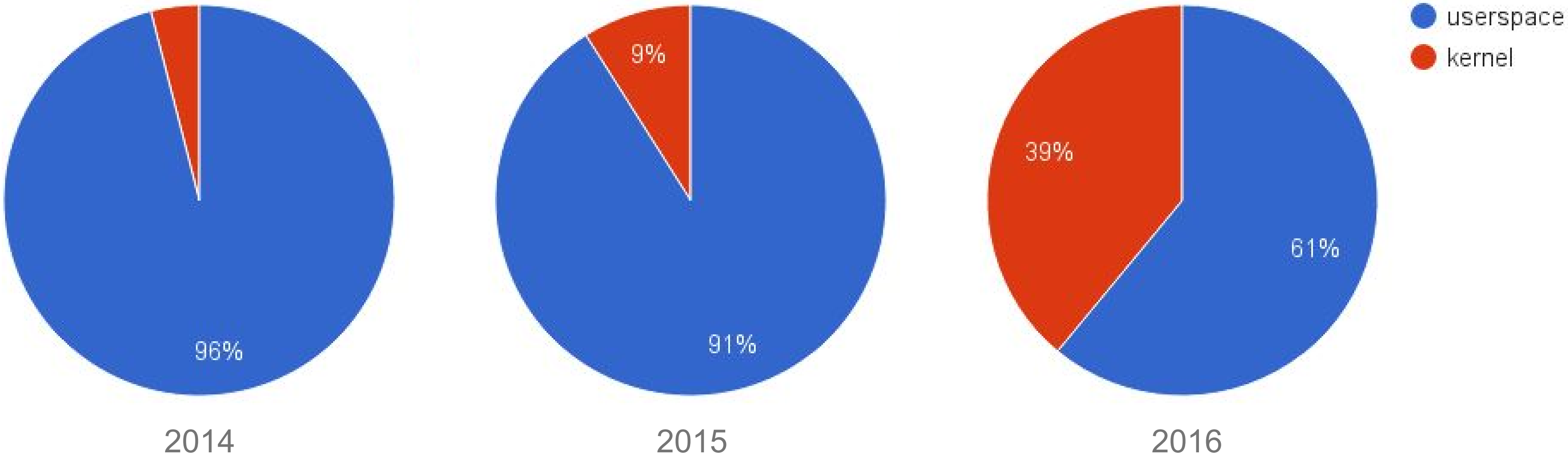
Webview Security



Linux Kernel

android

The kernel is the new target for vulnerability research

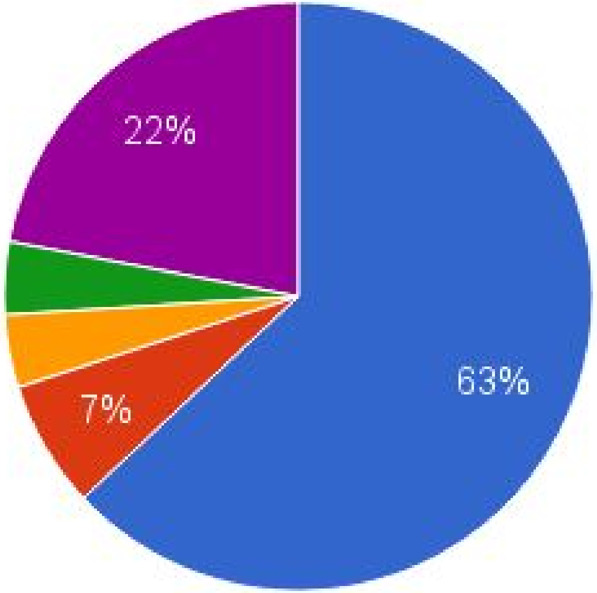


Security bugs reported to Android by year, broken down between userspace and kernel

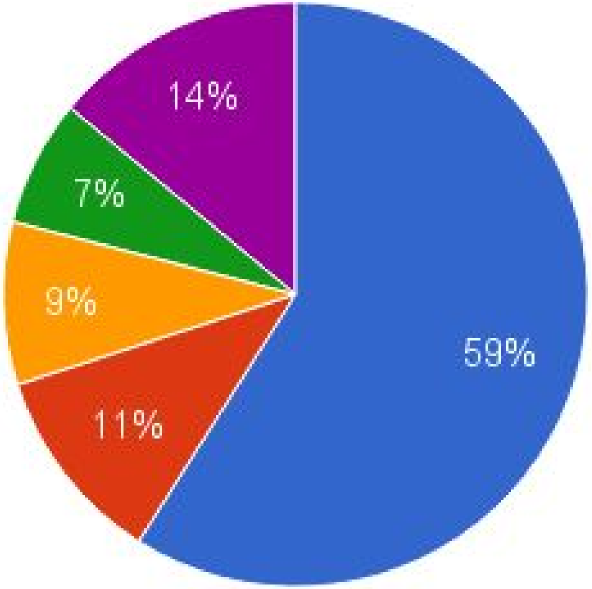
Why the rise in kernel bugs?

- Lockdown of userspace makes UID 0 significantly less useful.
- 2016 is the first year > 50% of devices in ecosystem have selinux in global enforcing.
- Android Vulnerability Rewards: Critical bugs payout more $$$.
    - … and kernel bugs tend to be high or critical severity

How are kernel bugs reached - syscall (before mitigations)



100% of perf vulns introduced in vendor customizations



all bugs

bugs reachable by apps

Data: Jan 2014 → April 2016

```
commit fa1aa143ac4a682c7f5fd52a3cf05f5a6fe44a0a
Author: Jeff Vander Stoep <jeffv@google.com>
Date:   Fri Jul 10 17:19:56 2015 -0400

    selinux: extended permissions for ioctls

    Add extended permissions logic to selinux. Extended permissions
    provides additional permissions in 256 bit increments. Extend the
    generic ioctl permission check to use the extended permissions for
    per-command filtering. Source/target/class sets including the ioctl
    permission may additionally include a set of commands. Example:

    allowxperm <source> <target>:<class> ioctl unpriv_app_socket_cmds
    auditallowxperm <source> <target>:<class> ioctl priv_gpu_cmds
```

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

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
- Remove default access to /sys
  - App access to files in /sys must be whitelisted
  - 38,000 files to 500 files **(98% reduction)**

android

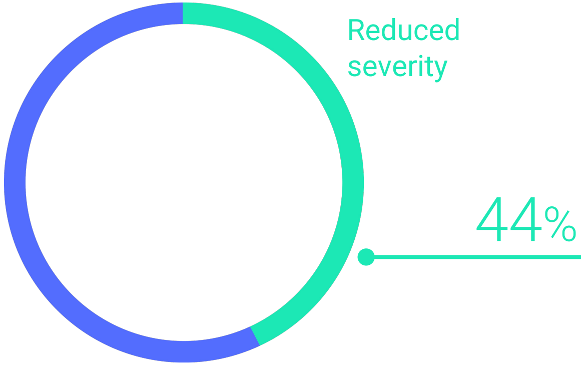
Impact of mitigations

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)

SELinux Effectiveness

SELinux reduced severity of almost half of kernel bugs
(Android security bulletin data for Jan-Apr, 2017)



Other Attack Surface Reductions

- Restricted /proc/PID visibility (hidepid=2, credit CopperheadOS)
  - Limit visibility between Android processes
  - Prevents popups, notification spam, and phishing
  - Addresses UI State Inference attacks
- DAC capabilities removal
  - Kernel module loading, writes to /system, most root capabilities
- Whitelist of /proc files (new in Android O)
  - 4400 files -> 2500 files (remainder mostly in /proc/sys/net)
- Hardlink removal

Recognition

android

Good reviews from attackers :-)

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

CunningLogic (aka jcase)

Now ~70% of Android devices!

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.*

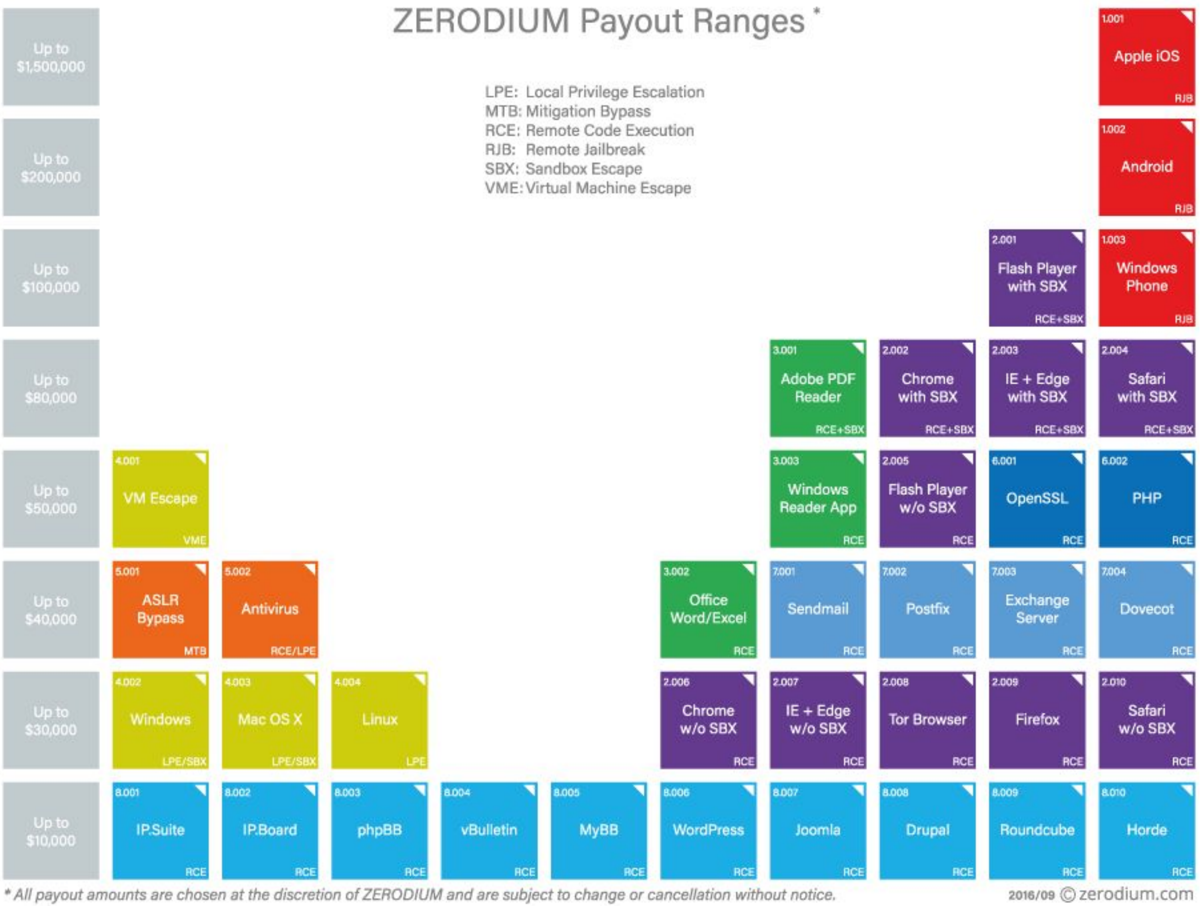
https://www.reddit.com/r/Android/comments/3hhciw/ask_us_almost_anything_about_android_security/

Good reviews from attackers :-)

Comparison

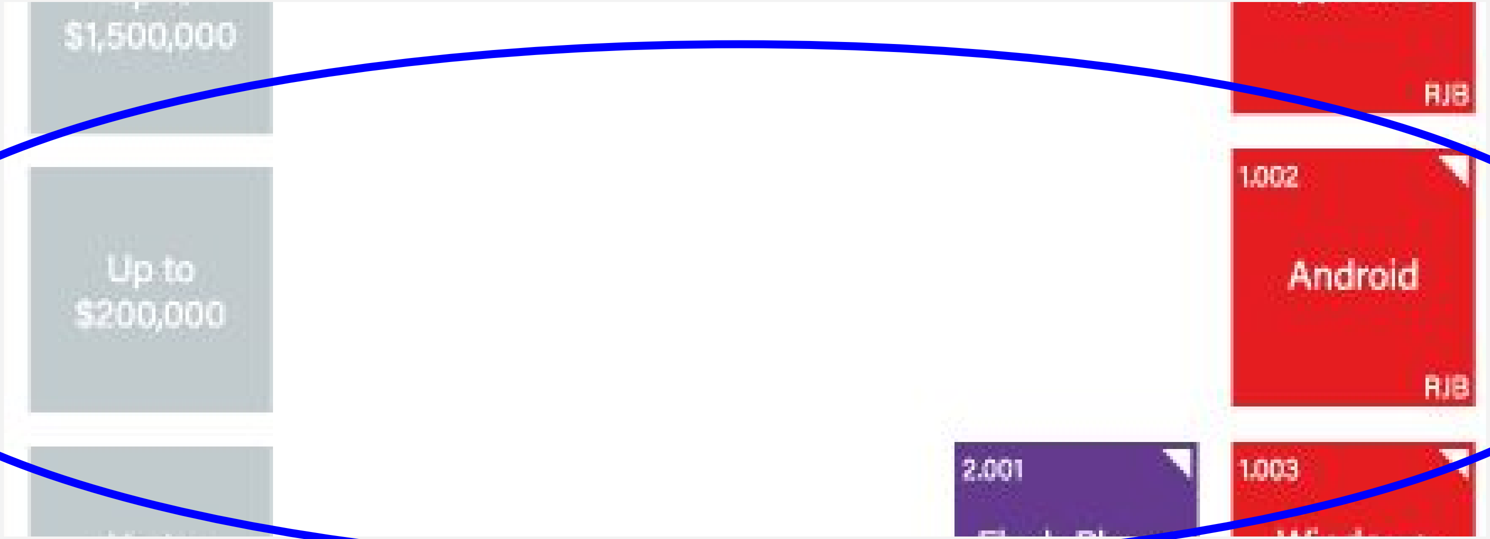
- Both platforms share lot of traits. They both implement a sandbox policy in files that specify it and can be audited
- In general between the 2, the Chromium Android sandbox feels stronger because it exposes a smaller attack surface.
- On Android we have more layer of sandboxing:
  - Android sandbox, chrome is an application, it's restricted by its DAC sandbox
  - IsolatedProcess, the render processes run in their own unprivileged process
  - Restrictive SELinux policy isolated_app.te

https://papers.put.as/papers/macosx/2016/sandbox_defcon.pdf


ZERODIUM Payout Ranges *
LPE: Local Privilege Escalation
MTB: Mitigation Bypass
RCE: Remote Code Execution
RJB: Remote Jailbreak
SBX: Sandbox Escape
VME: Virtual Machine Escape
Up to $1,500,000
1.001 Apple iOS RJB
Up to $200,000
1.002 Android RJB
Up to $100,000
2.001 Flash Player with SBX RCE+SBX
1.003 Windows Phone RJB
Up to $80,000
3.001 Adobe PDF Reader RCE+SBX
2.002 Chrome with SBX RCE+SBX
2.003 IE + Edge with SBX RCE+SBX
2.004 Safari with SBX RCE+SBX
Up to $50,000
4.001 VM Escape VME
3.003 Windows Reader App RCE
2.005 Flash Player w/o SBX RCE
6.001 OpenSSL RCE
6.002 PHP RCE
Up to $40,000
5.001 ASLR Bypass MTB
5.002 Antivirus RCE/LPE
3.002 Office Word/Excel RCE
7.001 Sendmail RCE
7.002 Postfix RCE
7.003 Exchange Server RCE
7.004 Dovecot RCE
Up to $30,000
4.002 Windows LPE/SBX
4.003 Mac OS X LPE/SBX
4.004 Linux LPE
2.006 Chrome w/o SBX RCE
2.007 IE + Edge w/o SBX RCE
2.008 Tor Browser RCE
2.009 Firefox RCE
2.010 Safari w/o SBX RCE
Up to $10,000
8.001 IP.Suite RCE
8.002 IP.Board RCE
8.003 phpBB RCE
8.004 vBulletin RCE
8.005 MyBB RCE
8.006 WordPress RCE
8.007 Joomla RCE
8.008 Drupal RCE
8.009 Roundcube RCE
8.010 Horde RCE
* All payout amounts are chosen at the discretion of ZERODIUM and are subject to change or cancellation without notice.
2016/09 © zerodium.com

android

Second highest exploit cost!



Price List Changelog

Changes of Sep. 29, 2016

Product / Exploit Type	New Price	Previous Price
[illegible]	[illegible]	[illegible]
Android 7 (Remote Jailbreak)	**$200,000**	$100,000
[illegible]	[illegible]	[illegible]
MS Edge + IE (RCE) + Sandbox Escape	**$80,000**	$50,000
Safari on Mac (RCE) + Sandbox Escape	**$80,000**	$50,000
OpenSSL or PHP (RCE)	**$50,000**	$40,000
MS Windo[illegible]	[illegible]	[illegible]
MS Offic[illegible]	[illegible]	[illegible]

2x increase in exploit cost!

pwn2own

Category	Phone	Price (USD)	“Master of Pwn” Points
Obtaining Sensitive Information	Apple iPhone	$50,000	10
	Google Nexus	$50,000	10
	Other Android	$35,000	7
Install Rogue Application	Apple iPhone	$125,000	23
	Google Nexus	$100,000	20
	Other Android	$60,000	15

- http://blog.trendmicro.com/presenting-mobile-pwn2own-2016/

pwn2own

- Price parity among the major mobile operating systems
- Smaller attack surface increases complexity and cost of finding an exploit

Phone	Price (USD)
Apple iPhone	$50,000
Google Nexus	$50,000
Other Android	$35,000
Apple iPhone	$125,000
Google Nexus	$100,000
Other Android	$60,000

pwn2own successes

Contest	Core Android Platform Bug	Additional Notes
2009 pwn2own	NO	All mobile devices unexploited
2010 pwn2own	NO	iPhone 3GS compromised. No Android compromised
2011 pwn2own	NO	Google Stays Strong
2012 pwn2own	NO	Non-Android device specific parsing bug - NFC delivered
2013 pwn2own	NO	Non-Android device specific bug
2014 pwn2own	YES	1. NFC triggered remote leak of Bluetooth MAC address 2. DHCP code execution (partial win)
2015 pwn2own	NO	Chrome exploit -> Google Play Install - No OS compromise
2016 pwn2own	NO	Chrome exploit -> Google Play Install - No OS compromise

android

No success from the Project Zero prize

ZDNet

VIDEOS SMART CITY WINDOWS 10 CLOUD INNOVATION SECURITY DEVOPS MORE NEWSLETTERS ALL WR

MUST READ HOSPITALS ACROSS ENGLAND HIT BY RANSOMWARE CYBER ATTACK, SYSTEMS KNOCKED OFFLINE

Didn't we offer you enough? Google's $350,000 Project Zero prize attracts junk entries

Was Google's Project Zero prize too difficult or was the prize just too small?

By Liam Tung | March 31, 2017 -- 11:34 GMT (04:34 PDT) | Topic: Security

http://www.zdnet.com/article/didnt-we-offer-you-enough-googles-350000-project-zero-prize-attracts-junk-entries/

Accelerating bug discovery

https://www.extremetech.com/mobile/250316-google-increases-android-bug-bounties-much-200000

“... no researcher has claimed the top reward for an exploit chains in 2 years ...”

	Old Amount	New Amount
Remote chain to TrustZone or Verified Boot compromise	$50,000	**$200,000**
Remote to Kernel	$30,000	**$150,000**

https://security.googleblog.com/2017/06/2017-android-security-rewards.html

android

Wikileaks: CIA Hacking Tools Revealed

“Furthermore, when SELinux became common on Android, this became more problematic since the radio SELinux context that rild started with was too restrictive for the implant to function.”

https://wikileaks.org/ciav7p1/cms/page_28049453.html

Future

android

Future: Global Seccomp Whitelist

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

Future Attack Surface Reduction

- Take better advantage of Treble - system / vendor split
- Continued reduction in /proc files
- Removal of useless /dev files
  - Faster boot time, less kernel code, less attack surface
- Stronger IPC controls
- System Properties
- Finer grain attack surface reduction for applications
- Scale back shared data stores

Takeaways

android

Takeaways

- Attack surface management is critical to preventing or mitigating unknown bugs.
- Android has invested significantly in reducing attack surface and containing processes.
- Vulnerabilities will never go away, but they can be contained and managed.

“Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away.”

- Antoine de Saint-Exupery - 1939

android

THANK YOU

security@android.com

Nick Kralevich
nnk@google.com