android

The Art of Defense

How vulnerabilities help shape security features and mitigations in Android

Nick Kralevich

August 4th, 2016

$ whoami

- Nick Kralevich
- Android Security since December 2009
- Android Platform Security Team Lead

Agenda

Quick overview of the Android Security Architecture

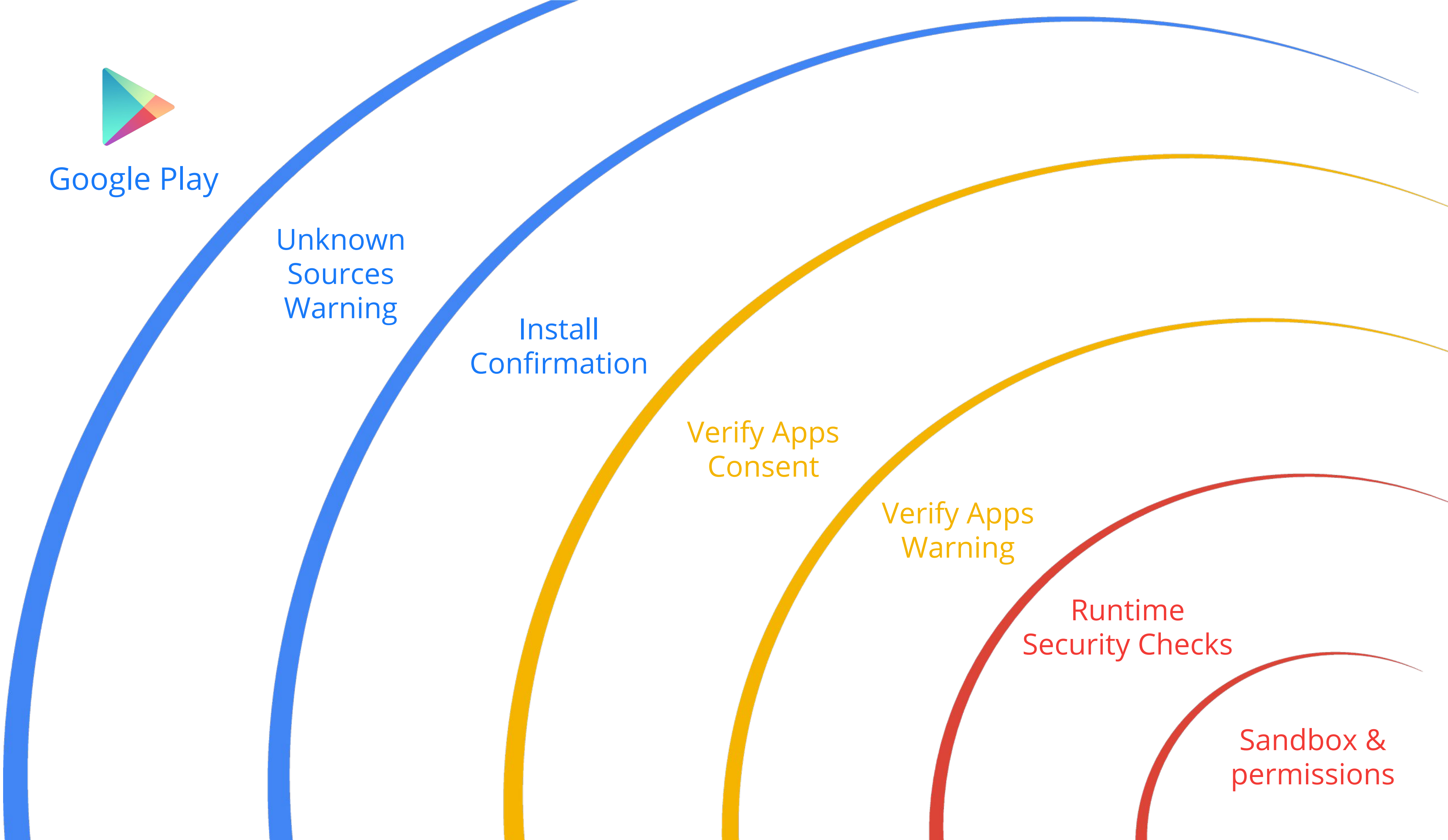
Vulnerabilities that affected Android and Android's response

Where do we go from here?

Android Security Ecosystem

android

Google Play



Unknown Sources Warning

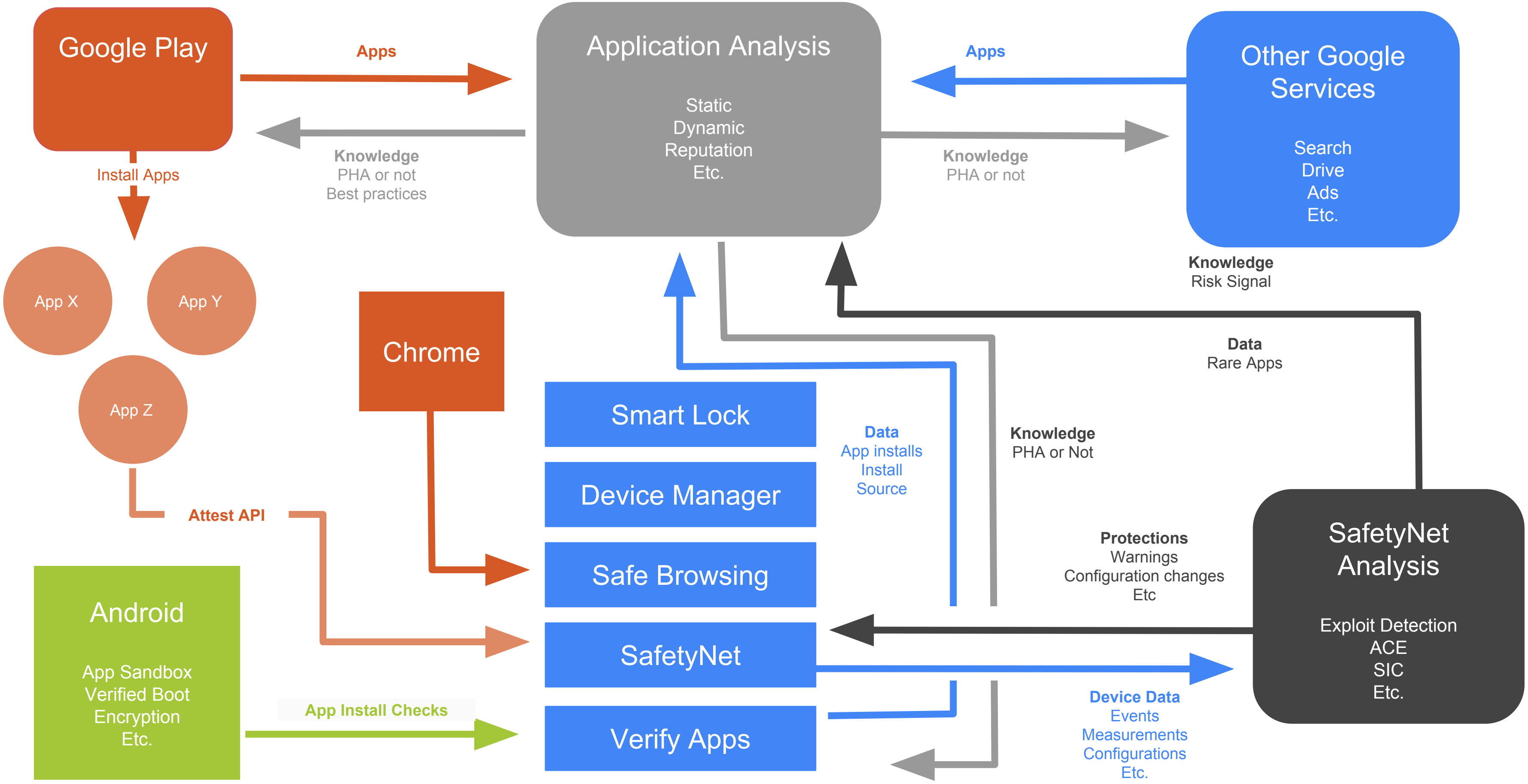
Install Confirmation

Verify Apps Consent

Verify Apps Warning

Runtime Security Checks

Sandbox & permissions



Google Play

Apps

Install Apps

Knowledge
PHA or not
Best practices

Application Analysis

Static
Dynamic
Reputation
Etc.

Apps

Knowledge
PHA or not

Other Google Services

Search
Drive
Ads
Etc.

Knowledge
Risk Signal

App X

App Y

App Z

Chrome

Data
Rare Apps

Smart Lock

Device Manager

Safe Browsing

SafetyNet

Verify Apps

Data
App installs
Install Source

Knowledge
PHA or Not

Attest API

Protections
Warnings
Configuration changes
Etc

SafetyNet Analysis

Exploit Detection
ACE
SIC
Etc.

Android

App Sandbox
Verified Boot
Encryption
Etc.

App Install Checks

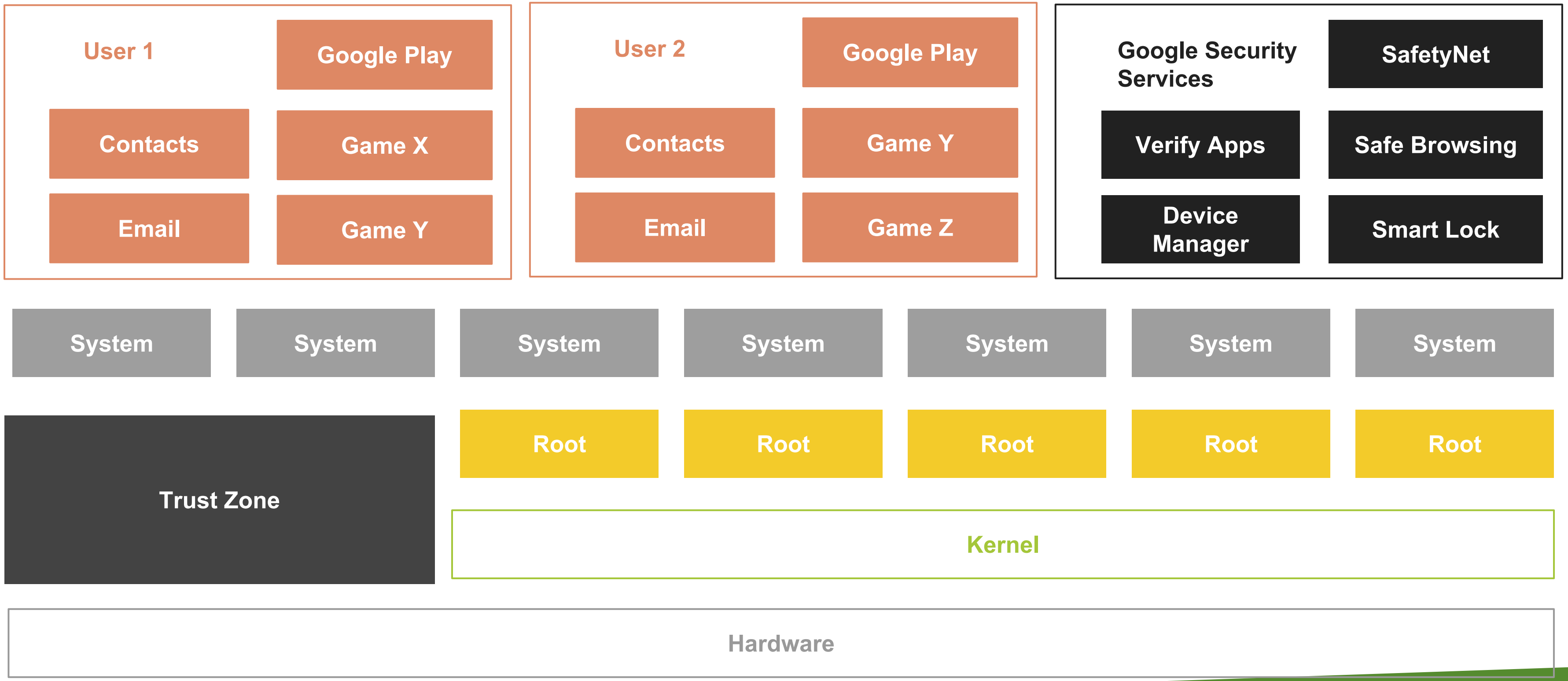
Device Data
Events
Measurements
Configurations
Etc.

Learn More

- https://source.android.com/security/
- Android Security 2015 Annual Report
  - https://security.googleblog.com/2016/04/android-security-2015-annual-report.html
- Android Security State of the Union
  - Black Hat 2015 - Adrian Ludwig
  - https://goo.gl/JrncdF

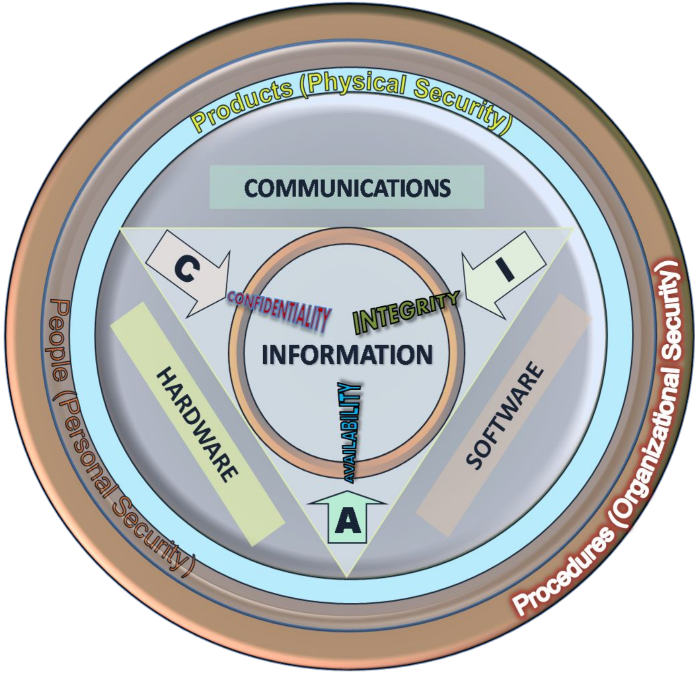
Android Platform Overview

High Level Overview



Key Android Security Principles

- Exploit Mitigation
- Exploit Containment
- Principle of Least Privilege
- Architectural Decomposition
- Attack Surface Reduction
- Safe by design APIs
- Defense-in-depth


Products (Physical Security)
COMMUNICATIONS
C
I
CONFIDENTIALITY
INTEGRITY
INFORMATION
HARDWARE
SOFTWARE
AVAILABILITY
A
People (Personal Security)
Procedures (Organizational Security)

Software Flaws

android

PingPong Root (CVE-2015-3636)

- Public Disclosure
  - oss-security
- Presented at Black Hat 2015
  - Wen Xu / @K33nTeam
- Result: Kernel code execution

```
diff --git a/net/ipv4/ping.c b/net/ipv4/ping.c
index a93f260..05ff44b 100644
--- a/net/ipv4/ping.c
+++ b/net/ipv4/ping.c
@@ -158,6 +158,7 @@ void ping_unhash(struct sock *sk)
        if (sk_hashed(sk)) {
                write_lock_bh(&ping_table.lock);
                hlist_nulls_del(&sk->sk_nulls_node);
+               sk_nulls_node_init(&sk->sk_nulls_node);
                sock_put(sk);
                isk->inet_num = 0;
                isk->inet_sport = 0;
```

https://www.blackhat.com/docs/us-15/materials/us-15-Xu-Ah-Universal-Android-Rooting-Is-Back.pdf

PingPong Root (CVE-2015-3636)

- An attempt at security hardening made the vulnerable code reachable

```
commit be341cc348257a07c68bcbfdc526835d49283329
Author: Nick Kralevich <nnk@google.com>
Date:   Thu Feb 21 18:36:43 2013 -0800

    init.rc: allow IPPROTO_ICMP support

    Allow userspace programs to create IPPROTO_ICMP sockets.

    This socket type allows an unprivileged program to safely
    send ICMP_ECHO messages and receive the corresponding
    ICMP_ECHOREPLY messages, without relying on raw sockets or
    setuid programs.
```

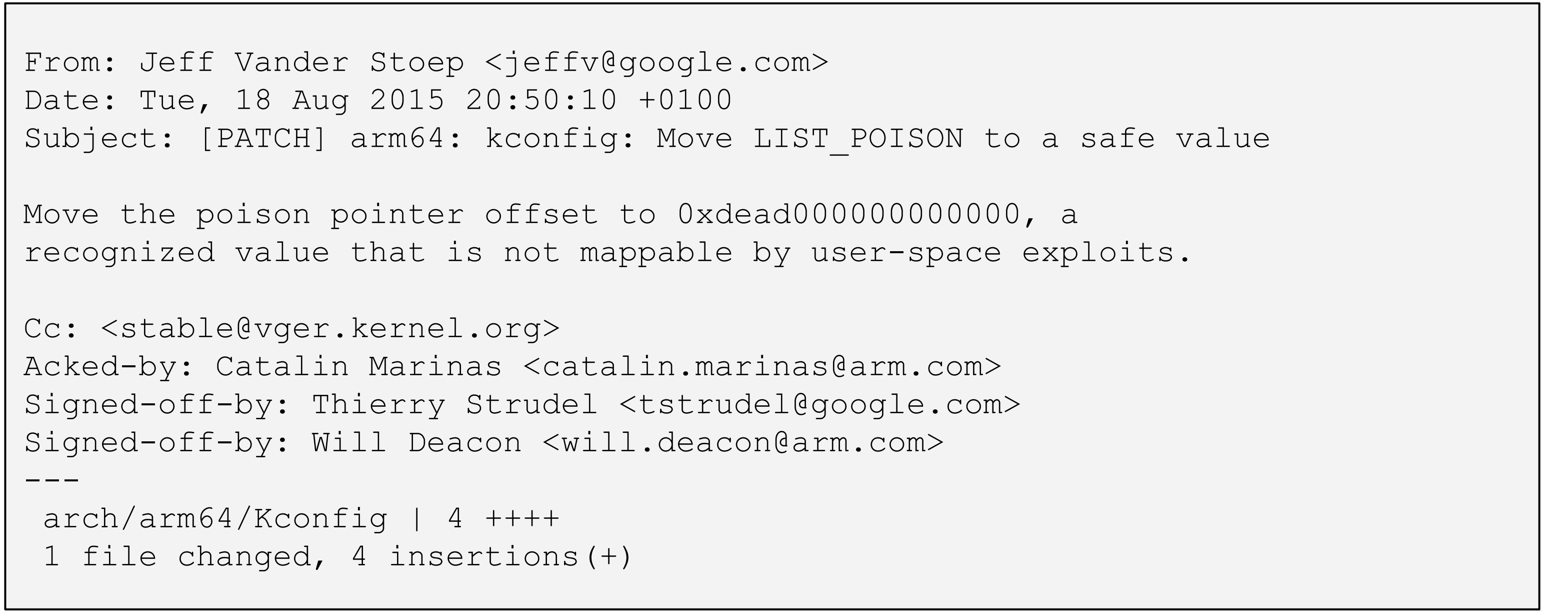
PingPong Root (CVE-2015-3636)

- First priority: **Fix the bug!**
- Next step: How do we protect against similar bugs?

Solely fixing bugs isn’t acceptable.

PingPong Root - Mitigation

- **Exploit Mitigation** - Move LIST_POINTER out of user-space



```
From: Jeff Vander Stoep <jeffv@google.com>
Date: Tue, 18 Aug 2015 20:50:10 +0100
Subject: [PATCH] arm64: kconfig: Move LIST_POISON to a safe value

Move the poison pointer offset to 0xdead000000000000, a
recognized value that is not mappable by user-space exploits.

Cc: <stable@vger.kernel.org>
Acked-by: Catalin Marinas <catalin.marinas@arm.com>
Signed-off-by: Thierry Strudel <tstrudel@google.com>
Signed-off-by: Will Deacon <will.deacon@arm.com>
---
 arch/arm64/Kconfig | 4 ++++
 1 file changed, 4 insertions(+)
```

PingPong Root - Mitigations

- Disallow access to unusual socket families
  - Bluetooth socket family, AF_MSM_IPC, etc…
  - Backported as CVE-2016-3762. Android Security Bulletin—July 2016
  - Other common socket families were blocked in previous Android versions.
- Whitelist allowable 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
} *;
```

PingPong Root - TL;DR

PingPong Root: 1 bug, 3 mitigations!

Learn more: http://android-developers.blogspot.com/2016/07/protecting-android-with-more-linux.html

PingPong Root - Mitigation

- The mitigations are effective at blocking or reducing the severity of a number of unrelated bugs
  - **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...

Stagefright

- Series of bugs reported by Joshua “jduck” Drake
- Private disclosure with embargo
- Public disclosure via NPR / blog post / PR / ads / etc...
- For this presentation, focusing on CVE-2015-3824
  - MP4 'tx3g' Integer Overflow

https://www.blackhat.com/docs/us-15/materials/us-15-Drake-Stagefright-Scary-Code-In-The-Heart-Of-Android.pdf

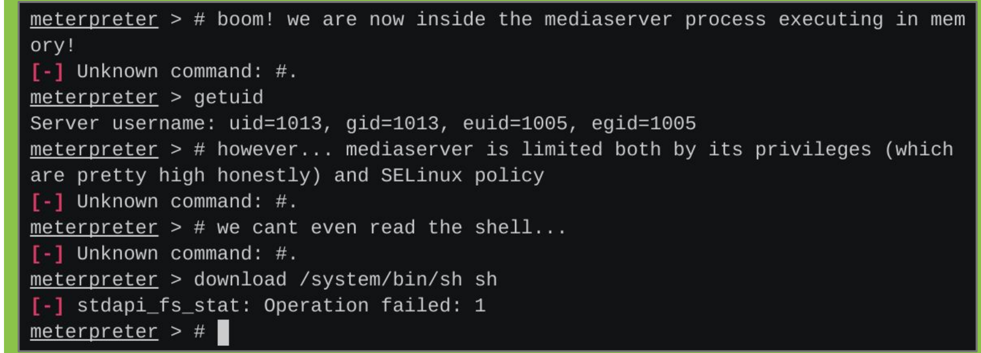
Stagefright - A “successful failure”

- Monthly patching cycle
- Public security bulletins
- No evidence of malicious exploitation
- Exploit mitigations (ASLR, etc) worked as intended and bought time
- Device diversity complicated exploitation and bought time
- Exploit containment (UID sandbox, SELinux) forced vulnerability chaining and bought time
- Widespread patch distribution: 57-89% of population [1]
- Significant architectural improvements (more later)
- Enhanced visibility of Android Vulnerability Rewards Program

[1] Source: Zimperium.com, March 22nd, 2016

Stagefright

- Mediaserver architected for containment
  - “Android: Securing a Mobile Platform from the Ground Up” (Rich Cannings, Usenix Security 2009)
  - Charlie Miller - oCERT-2009-002
- Stagefright exploit was contained
  - Required vulnerability chaining
- Mediaserver grew up. More features => more capabilities



```
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

Stagefright

- First Priority: **Fix the bugs!**
    - 7 patches provided by vulnerability reporter (yay!)

```
@@ -1948,6 +1948,9 @@ status_t MPEG4Extractor::parseChunk(off64_t *offset, int depth) {
         size = 0;
     }

+    if (SIZE_MAX - chunk_size <= size)
+        return ERROR_MALFORMED;
+
     uint8_t *buffer = new (std::nothrow) uint8_t[size + chunk_size];
     if (buffer == NULL) {
         return ERROR_MALFORMED;
```

Stagefright

- Unfortunately, fix was incomplete: CVE-2015-3864

CVE-2015-3824

```
+    if (SIZE_MAX - chunk_size <= size)
+        return ERROR_MALFORMED;
+
     uint8_t *buffer = new (std::nothrow) uint8_t[size + chunk_size];
     if (buffer == NULL) {
         return ERROR_MALFORMED;
```

CVE-2015-3864

```
        size = 0;
     }

-    if (SIZE_MAX - chunk_size <= size) {
+    if ((chunk_size > SIZE_MAX) || (SIZE_MAX - chunk_size <= size)) {
         return ERROR_MALFORMED;
     }
```

Stagefright

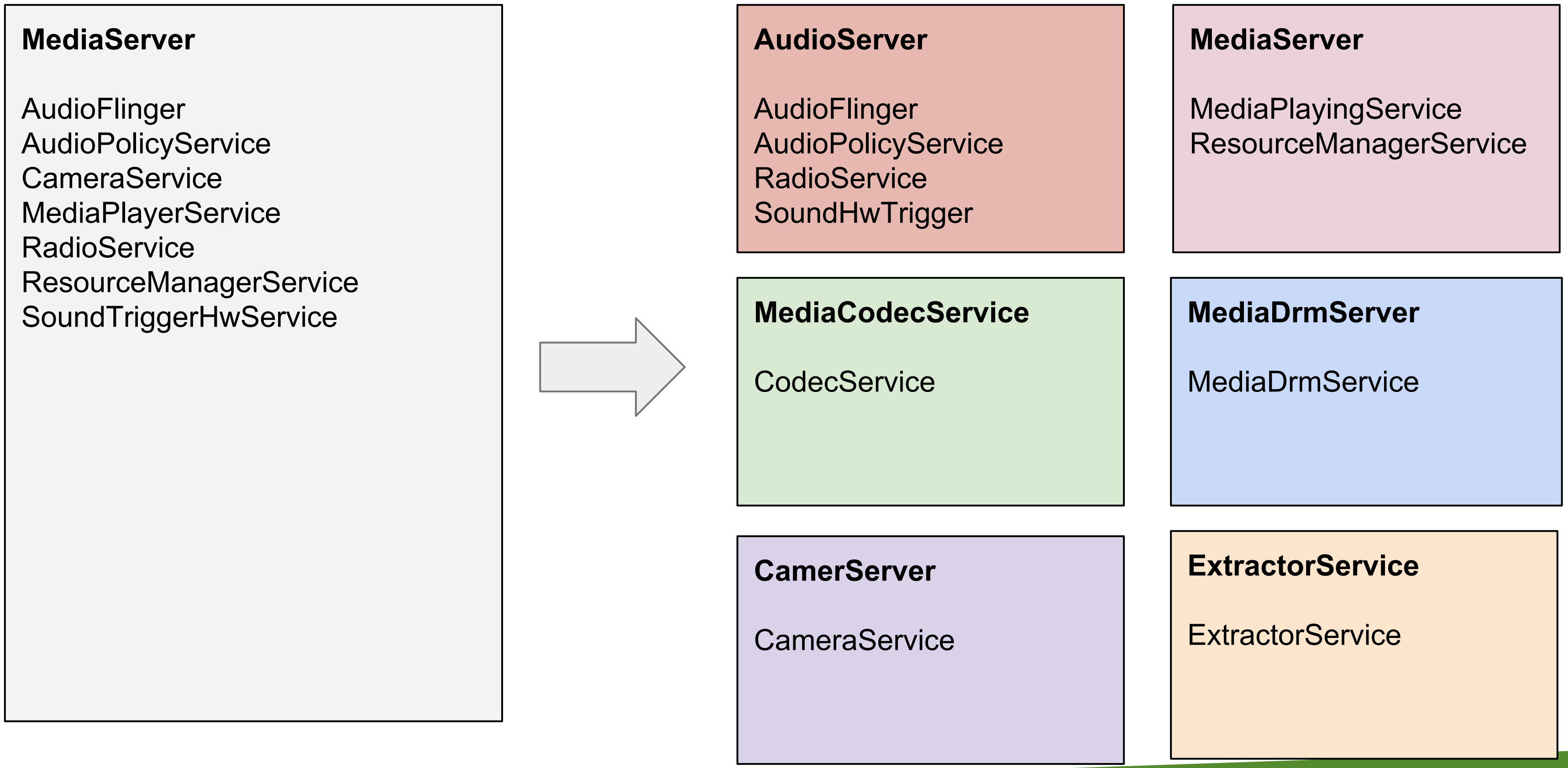
Solely fixing bugs isn’t acceptable.

mediaserver - Architectural Improvements

- Mediaserver refactoring
- Integer overflow protections
- ASLR enhancements
  - Increase kernel randomness
  - Link time randomization
- Mediaserver seccomp
- Remove mediaserver execmem

Android M - Services per process

Android N - Services per process



MediaServer

AudioFlinger
AudioPolicyService
CameraService
MediaPlayerService
RadioService
ResourceManagerService
SoundTriggerHwService

AudioServer

AudioFlinger
AudioPolicyService
RadioService
SoundHwTrigger

MediaServer

MediaPlayingService
ResourceManagerService

MediaCodecService

CodecService

MediaDrmServer

MediaDrmService

CamerServer

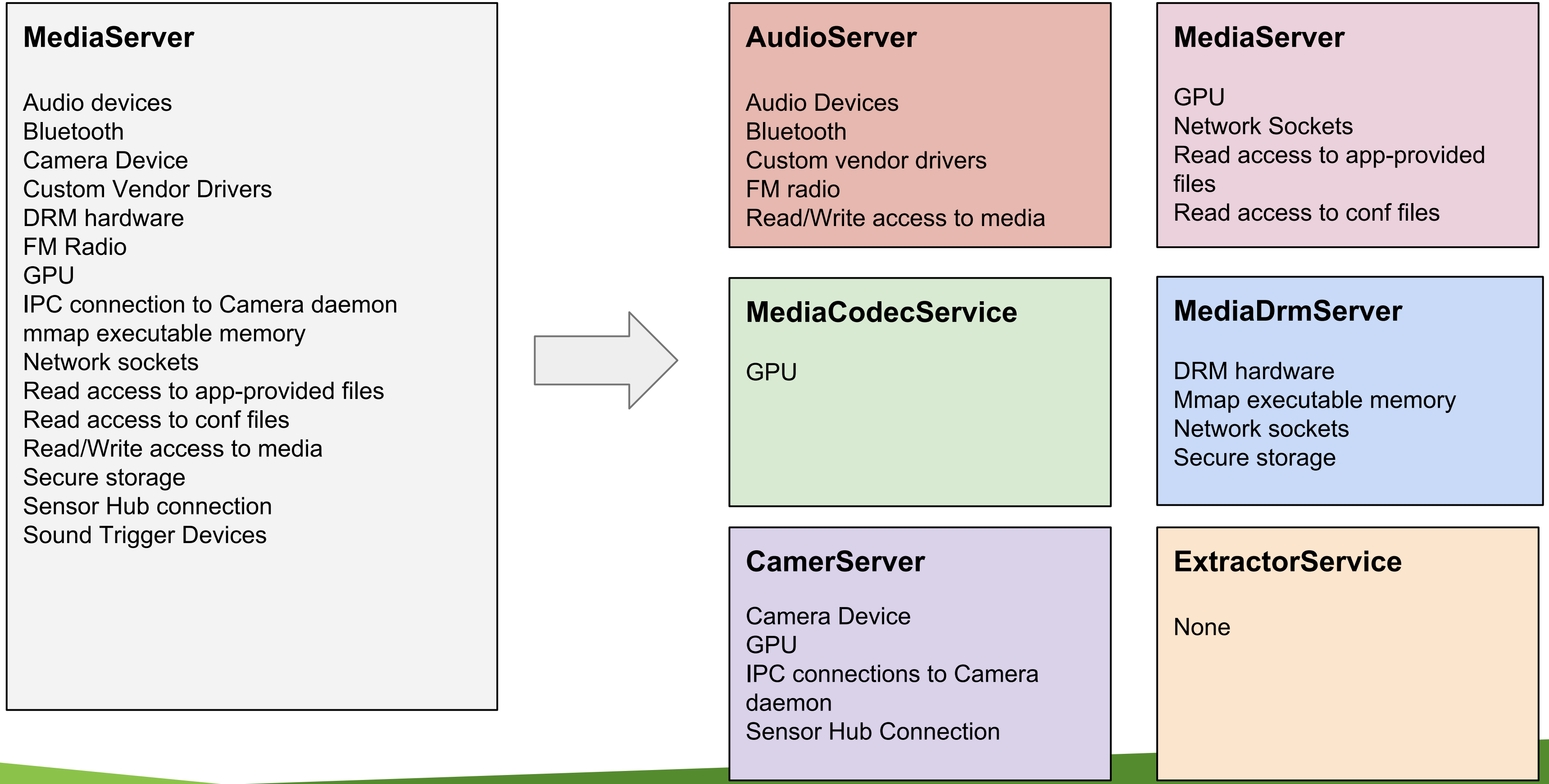
CameraService

ExtractorService

ExtractorService

Android M - Capabilities per process

Android N - Capabilities per process



MediaServer

Audio devices
Bluetooth
Camera Device
Custom Vendor Drivers
DRM hardware
FM Radio
GPU
IPC connection to Camera daemon
mmap executable memory
Network sockets
Read access to app-provided files
Read access to conf files
Read/Write access to media
Secure storage
Sensor Hub connection
Sound Trigger Devices

AudioServer

Audio Devices
Bluetooth
Custom vendor drivers
FM radio
Read/Write access to media

MediaServer

GPU
Network Sockets
Read access to app-provided files
Read access to conf files

MediaCodecService

GPU

MediaDrmServer

DRM hardware
Mmap executable memory
Network sockets
Secure storage

CamerServer

Camera Device
GPU
IPC connections to Camera daemon
Sensor Hub Connection

ExtractorService

None

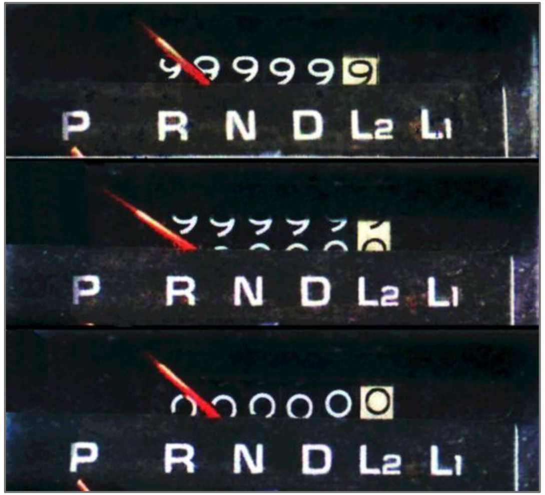
mediaserver - Refactoring results

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

Everyone is safer!

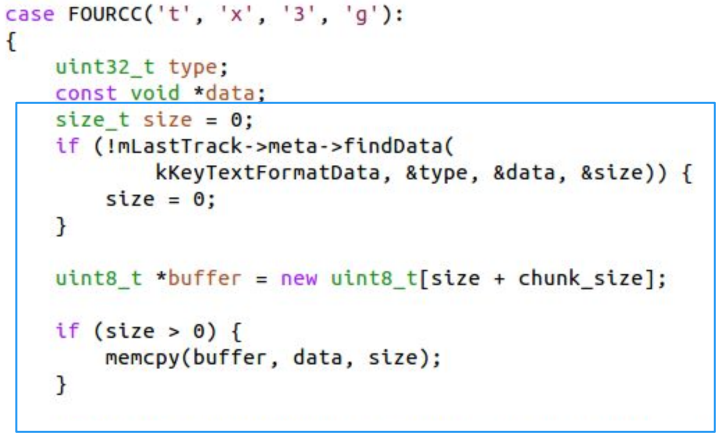
Stagefright - Integer Overflow Protections

- Majority of stagefright bugs were integer overflow
- In C & C++:
  - For unsigned values: the result is taken modulo 2^{bits}
  - For signed values: the result is undefined

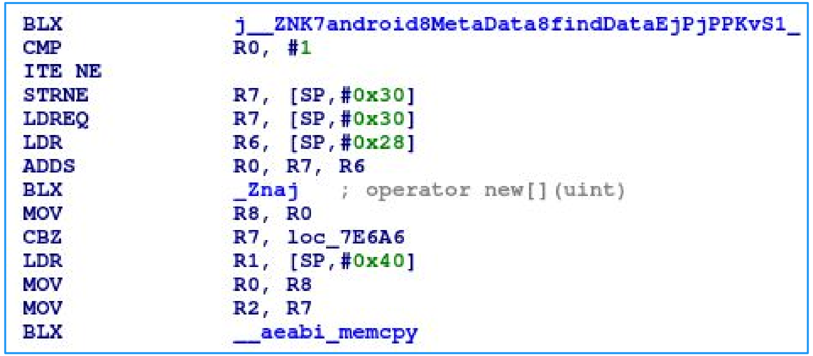


UBSan to the rescue!

Stagefright before patch

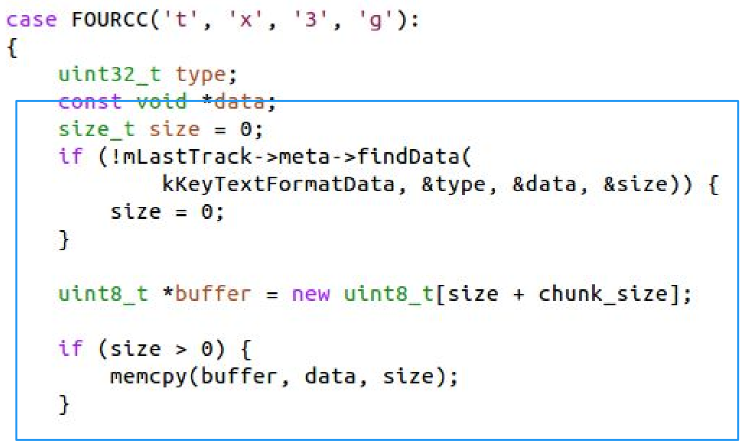


```
case FOURCC('t', 'x', '3', 'g'):
{
    uint32_t type;
    const void *data;
    size_t size = 0;
    if (!mLastTrack->meta->findData(
            kKeyTextFormatData, &type, &data, &size)) {
        size = 0;
    }

    uint8_t *buffer = new uint8_t[size + chunk_size];

    if (size > 0) {
        memcpy(buffer, data, size);
    }
```

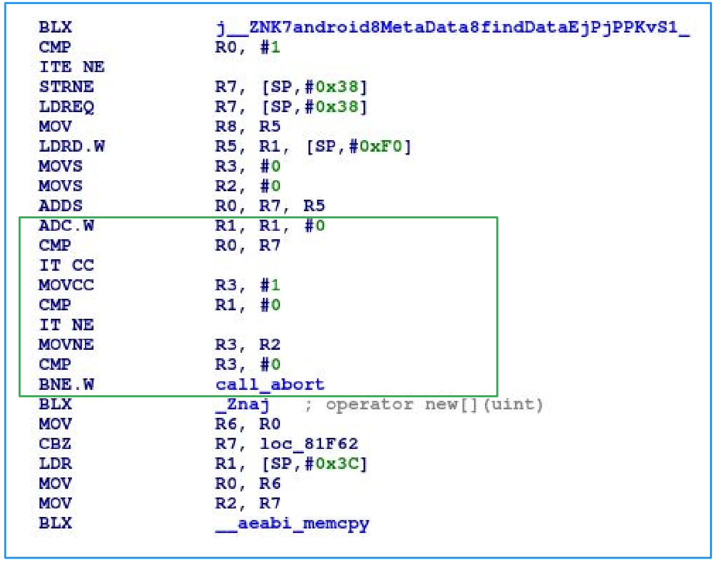


```
BLX             j__ZNK7android8MetaData8findDataEjPjPPKvS1_
CMP             R0, #1
ITE NE
STRNE           R7, [SP,#0x30]
LDREQ           R7, [SP,#0x30]
LDR             R6, [SP,#0x28]
ADDS            R0, R7, R6
BLX             _Znaj   ; operator new[](uint)
MOV             R8, R0
CBZ             R7, loc_7E6A6
LDR             R1, [SP,#0x40]
MOV             R0, R8
MOV             R2, R7
BLX             __aeabi_memcpy
```

Stagefright before patch v1, sanitized

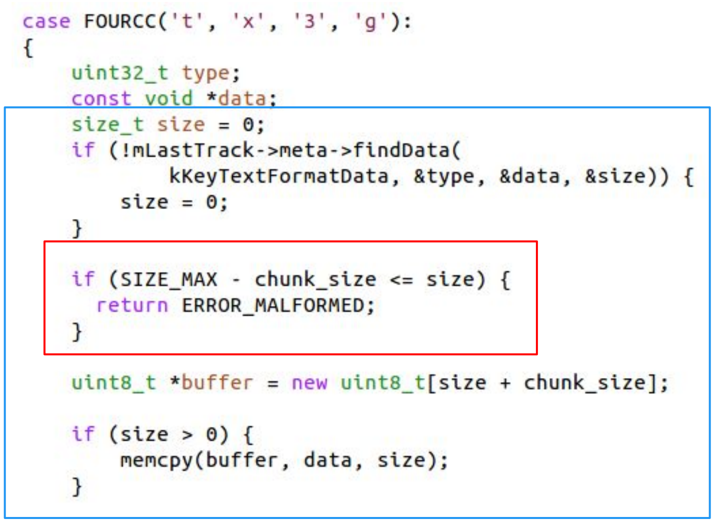


```
case FOURCC('t', 'x', '3', 'g'):
{
    uint32_t type;
    const void *data;
    size_t size = 0;
    if (!mLastTrack->meta->findData(
            kKeyTextFormatData, &type, &data, &size)) {
        size = 0;
    }

    uint8_t *buffer = new uint8_t[size + chunk_size];

    if (size > 0) {
        memcpy(buffer, data, size);
    }
```

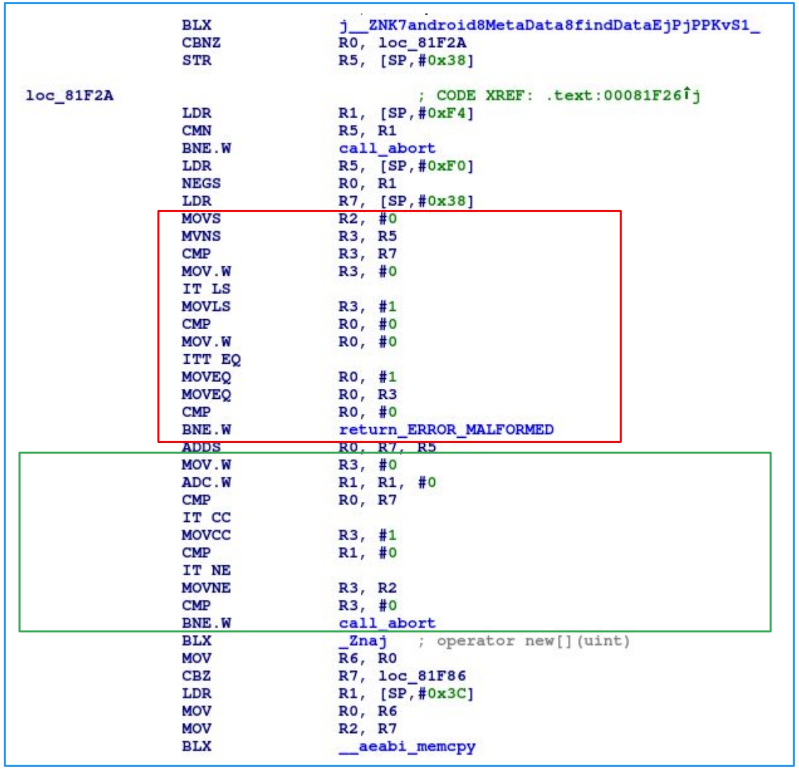


```
BLX             j__ZNK7android8MetaData8findDataEjPjPPKvS1_
CMP             R0, #1
ITE NE
STRNE           R7, [SP,#0x38]
LDREQ           R7, [SP,#0x38]
MOV             R8, R5
LDRD.W          R5, R1, [SP,#0xF0]
MOVS            R3, #0
MOVS            R2, #0
ADDS            R0, R7, R5
ADC.W           R1, R1, #0
CMP             R0, R7
IT CC
MOVCC           R3, #1
CMP             R1, #0
IT NE
MOVNE           R3, R2
CMP             R3, #0
BNE.W           call_abort
BLX             _Znaj   ; operator new[](uint)
MOV             R6, R0
CBZ             R7, loc_81F62
LDR             R1, [SP,#0x3C]
MOV             R0, R6
MOV             R2, R7
BLX             __aeabi_memcpy
```

Stagefright after patch v1, sanitized



```
case FOURCC('t', 'x', '3', 'g'):
{
    uint32_t type;
    const void *data;
    size_t size = 0;
    if (!mLastTrack->meta->findData(
            kKeyTextFormatData, &type, &data, &size)) {
        size = 0;
    }

    if (SIZE_MAX - chunk_size <= size) {
      return ERROR_MALFORMED;
    }

    uint8_t *buffer = new uint8_t[size + chunk_size];

    if (size > 0) {
        memcpy(buffer, data, size);
    }
```



```
                BLX             j__ZNK7android8MetaData8findDataEjPjPPKvS1_
                CBNZ            R0, loc_81F2A
                STR             R5, [SP,#0x38]

loc_81F2A                                       ; CODE XREF: .text:00081F26↑j
                LDR             R1, [SP,#0xF4]
                CMN             R5, R1
                BNE.W           call_abort
                LDR             R5, [SP,#0xF0]
                NEGS            R0, R1
                LDR             R7, [SP,#0x38]
                MOVS            R2, #0
                MVNS            R3, R5
                CMP             R3, R7
                MOV.W           R3, #0
                IT LS
                MOVLS           R3, #1
                CMP             R0, #0
                MOV.W           R0, #0
                ITT EQ
                MOVEQ           R0, #1
                MOVEQ           R0, R3
                CMP             R0, #0
                BNE.W           return_ERROR_MALFORMED
                ADDS            R0, R7, R5
                MOV.W           R3, #0
                ADC.W           R1, R1, #0
                CMP             R0, R7
                IT CC
                MOVCC           R3, #1
                CMP             R1, #0
                IT NE
                MOVNE           R3, R2
                CMP             R3, #0
                BNE.W           call_abort
                BLX             _Znaj   ; operator new[](uint)
                MOV             R6, R0
                CBZ             R7, loc_81F86
                LDR             R1, [SP,#0x3C]
                MOV             R0, R6
                MOV             R2, R7
                BLX             __aeabi_memcpy
```

libstagefright with UBSan

- In Summary:
  - UBSan with original patch: no integer overflow, stops exploit!
  - UBSan with no patch: no integer overflow, stops exploit!

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

ASLR Enhancements

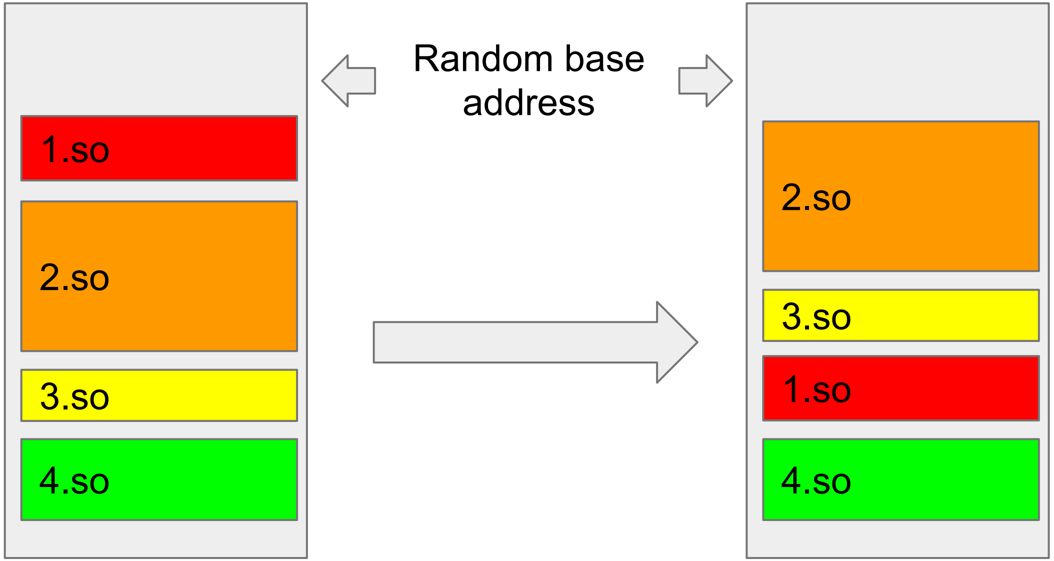
ASLR Patch #1 - Increased randomness from kernel

```
commit d07e22597d1d355829b7b18ac19afa912cf758d1
Author: Daniel Cashman <dcashman@google.com>
Date:   Thu Jan 14 15:19:53 2016 -0800

    mm: mmap: add new /proc tunable for mmap_base ASLR

[deleted]

    Concretely, the attack was against the mediaserver process, which was
    limited to respawning every 5 seconds, on an arm device.  The hard-coded
    8 bits used resulted in an average expected success rate of defeating
    the mmap ASLR after just over 10 minutes (128 tries at 5 seconds a
    piece).  With this patch, and an accompanying increase in the entropy
    value to 16 bits, the same attack would take an average expected time of
    over 45 hours (32768 tries), which makes it both less feasible and more
    likely to be noticed.
```

https://lwn.net/Articles/667790/

ASLR Patch #2 - Library Load Order Randomization

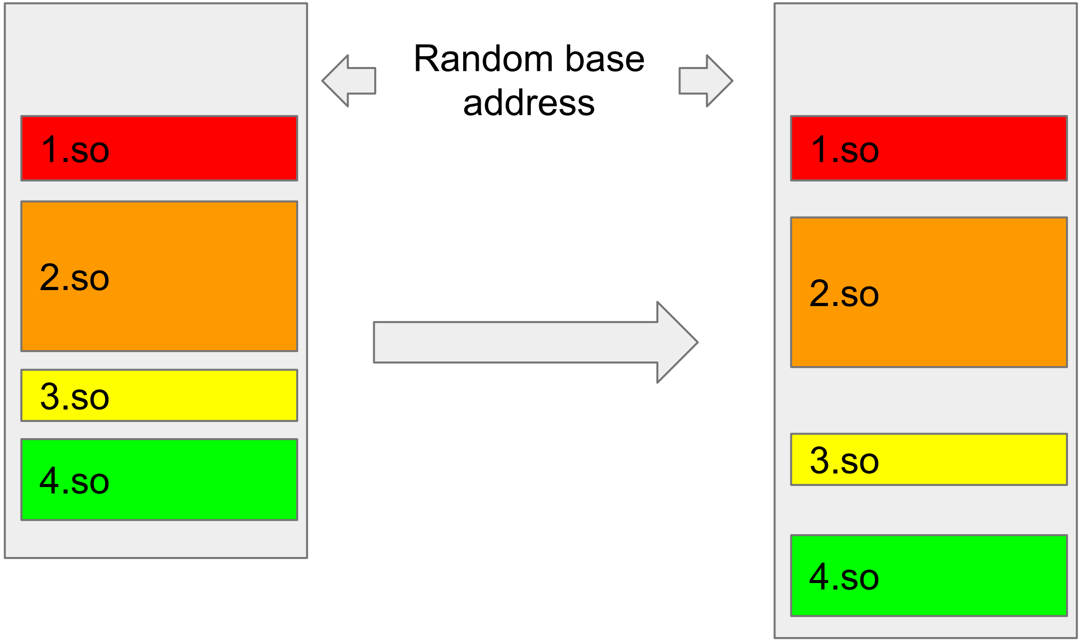
- Compliments and enhances randomized mmap base address
- Dependent shared libraries are mapped into memory in random order
- Effectiveness depends on number of shared library dependencies
- No impact on initial executable nor dynamic linker load



https://android-review.googlesource.com/178130

ASLR Patch #3 - Random gap between *.so files

- Checked in 15 days ago. :-)
  - Targeting future Android release
- Adds more gaps between shared libraries.
- Allow a lot more compact CFI shadow implementation



https://android-review.googlesource.com/248499

mediaserver: additional changes

- 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
- seccomp enforcement

```
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)

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
```

Stagefright - TL;DR

Stagefright: 7 mitigations!

Data in Transit Protection

Data In Transit Protection

- The network is not to be trusted.
  - This has always been true but is especially for mobile devices.
  - But you already know this.
- Too much unencrypted traffic

Data In Transit Protection - Marshmallow

In order to help you accurately and easily determine if your application is making cleartext traffic in Marshmallow we added two new features.

1. Strict mode cleartext detection to help you while testing.
2. usesCleartextTraffic application manifest attribute to block accidental regressions on user devices.

Note: These are not limited to HTTP/HTTPS

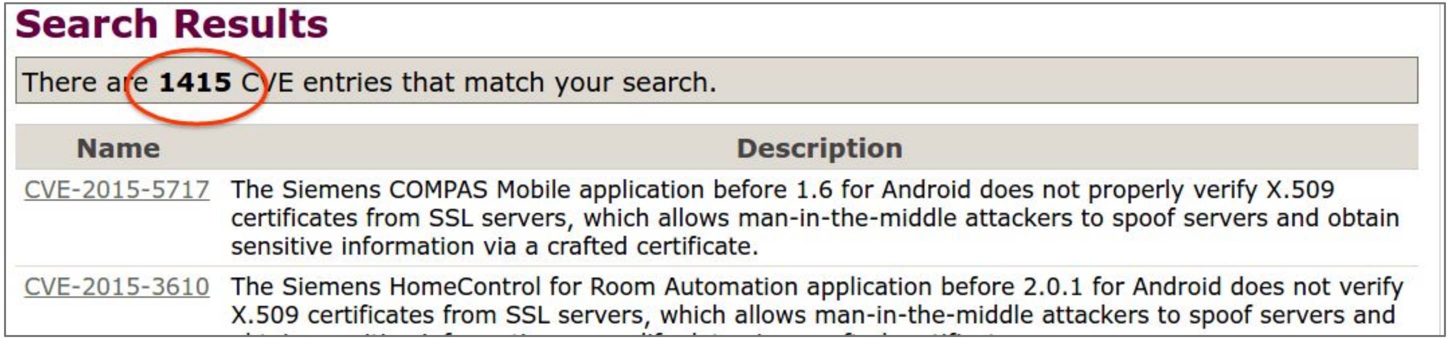
```
StrictMode.VmPolicy policy =
     new StrictMode.VmPolicy.Builder()
         .detectCleartextNetwork()
         .penaltyDeath()
         .build();
 StrictMode.setVmPolicy(policy);
```

```
<application
android:usesCleartextTraffic="false" />
```

Data In Transit Protection

- The network is not safe
  - But you already know that
- Too much unencrypted traffic
- **Too much badly encrypted traffic**

https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=android+x.509



Search Results

There are **1415** CVE entries that match your search.

Name	Description
CVE-2015-5717	The Siemens COMPAS Mobile application before 1.6 for Android does not properly verify X.509 certificates from SSL servers, which allows man-in-the-middle attackers to spoof servers and obtain sensitive information via a crafted certificate.
CVE-2015-3610	The Siemens HomeControl for Room Automation application before 2.0.1 for Android does not verify X.509 certificates from SSL servers, which allows man-in-the-middle attackers to spoof servers and

Badly Encrypted Traffic

- What causes bad encryption bugs?
    - Code testing in non-production environments
    - Third party libraries changing global state
    - Insecure code samples online
    - Connection to legacy servers

Badly Encrypted Traffic

Do not use these code samples!

```
HttpsURLConnection.setDefaultHostnameVerifier(new HostnameVerifier() {
    public boolean verify(String hostname, SSLSession session) { return true; }
});
```

```
SSLContext ctx = SSLContext.getInstance("TLS");
ctx.init(null, new TrustManager[] {
      new X509TrustManager() {
             public void checkClientTrusted(X509Certificate[] chain, String authType) {}
             public void checkServerTrusted(X509Certificate[] chain, String authType) {}
             public X509Certificate[] getAcceptedIssuers() { return new X509Certificate[]{}; } } }, null);
HttpsURLConnection.setDefaultSSLSocketFactory(ctx.getSocketFactory());
```

Network Security Config

- Customizing TLS through the current APIs is too error prone
- Network Security Config: Safer and easier API
- Fine grain blocking of insecure traffic in your app
- Eliminate debugging-related code in your release build
    - Connect to your development infrastructure without any code
    - Avoid writing custom code that removes security for debug builds and accidentally shipping it
- Limit the CAs you want to trust
- Easy to configure cert pinning

Network Security Config - Block insecure traffic

```
<network-security-config>
    <domain-config cleartextTrafficPermitted="false">
        <domain includeSubdomains="true">secure.example.com</domain>
    </domain-config>
</network-security-config>
```

Network Security Config - Debug only CAs

```
<network-security-config>
    <debug-overrides>
        <trust-anchors>
            <certificates src="@raw/debug_cas"/>
        </trust-anchors>
    </debug-overrides>
</network-security-config>
```

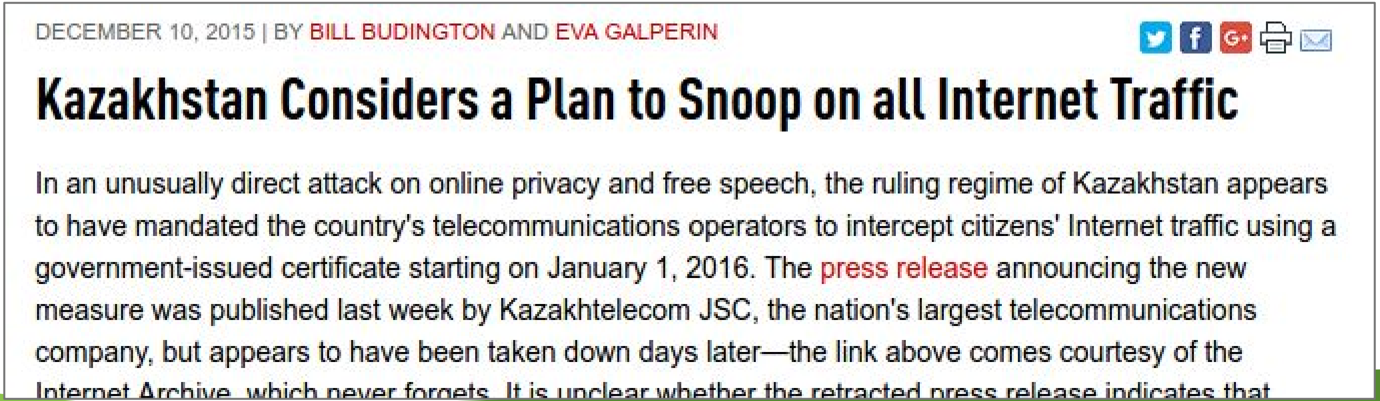
Network Security Config - Pinning

```
<network-security-config>
  <domain-config>
    <domain includeSubdomains="true">example.com</domain>
    <pin-set expiration="2018-01-01">
      <pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>
      <!-- backup pin -->
      <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

Data In Transit Protection - User Installed Certificates

- Question: How should user installed certificates be handled?
  - Opportunity to revisit old assumptions
- App files/memory/processes are protected by default
  - Why not network traffic?
- Interest from nation states

https://www.eff.org/deeplinks/2015/12/kazakhstan-considers-plan-snoop-all-internet-traffic



DECEMBER 10, 2015 | BY BILL BUDINGTON AND EVA GALPERIN

Kazakhstan Considers a Plan to Snoop on all Internet Traffic

In an unusually direct attack on online privacy and free speech, the ruling regime of Kazakhstan appears to have mandated the country's telecommunications operators to intercept citizens' Internet traffic using a government-issued certificate starting on January 1, 2016. The press release announcing the new measure was published last week by Kazakhtelecom JSC, the nation's largest telecommunications company, but appears to have been taken down days later—the link above comes courtesy of the Internet Archive, which never forgets. It is unclear whether the retracted press release indicates that

Data In Transit Protection - User Installed Certificates

- Most application developers unaware secure traffic can be intercepted
- User installable certificates not commonly used

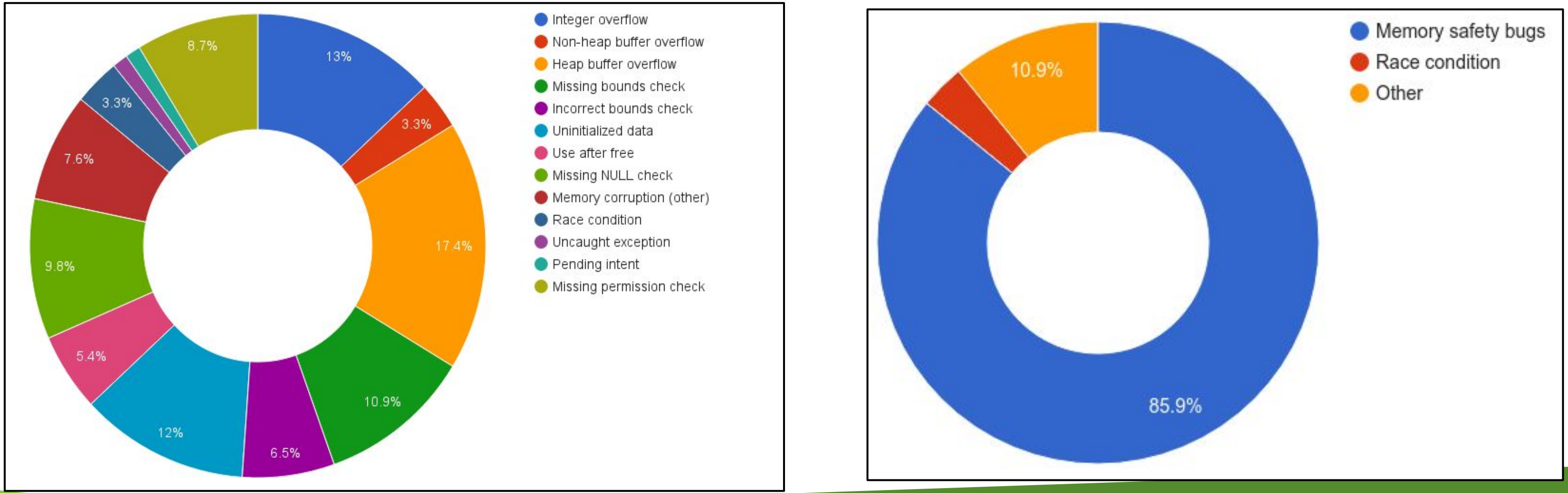
Applications targeting “Nougat” or greater no longer trust user installed certs by default.

Where do we go from here?

Languages

- **Safe by design:** As an industry, we need to move towards memory safe languages
  - This includes sacred cows such as the Linux kernel

Bug root cause for all of Android (including kernel and other components)



Invest in Defense

- **Invest in defenses:** As an industry, we need to look beyond attacks and short term solutions, and invest in architectural improvements in all these areas:
    - Exploit Mitigation
    - Exploit Containment
    - Principle of Least Privilege
    - Architectural Decomposition
    - Attack Surface Reduction
    - Safe by design APIs
    - Defense-in-depth

Black Hat Sound Bytes

Black Hat Sound Bytes

- Android has a robust, multi-layered defense designed to mitigate and contain vulnerabilities.
- Android is investing heavily in learning from vulnerabilities and applying those lessons in new releases.
- Vulnerabilities will never go away, but they can be contained and managed.

THANK YOU

Nick Kralevich

nnk@google.com