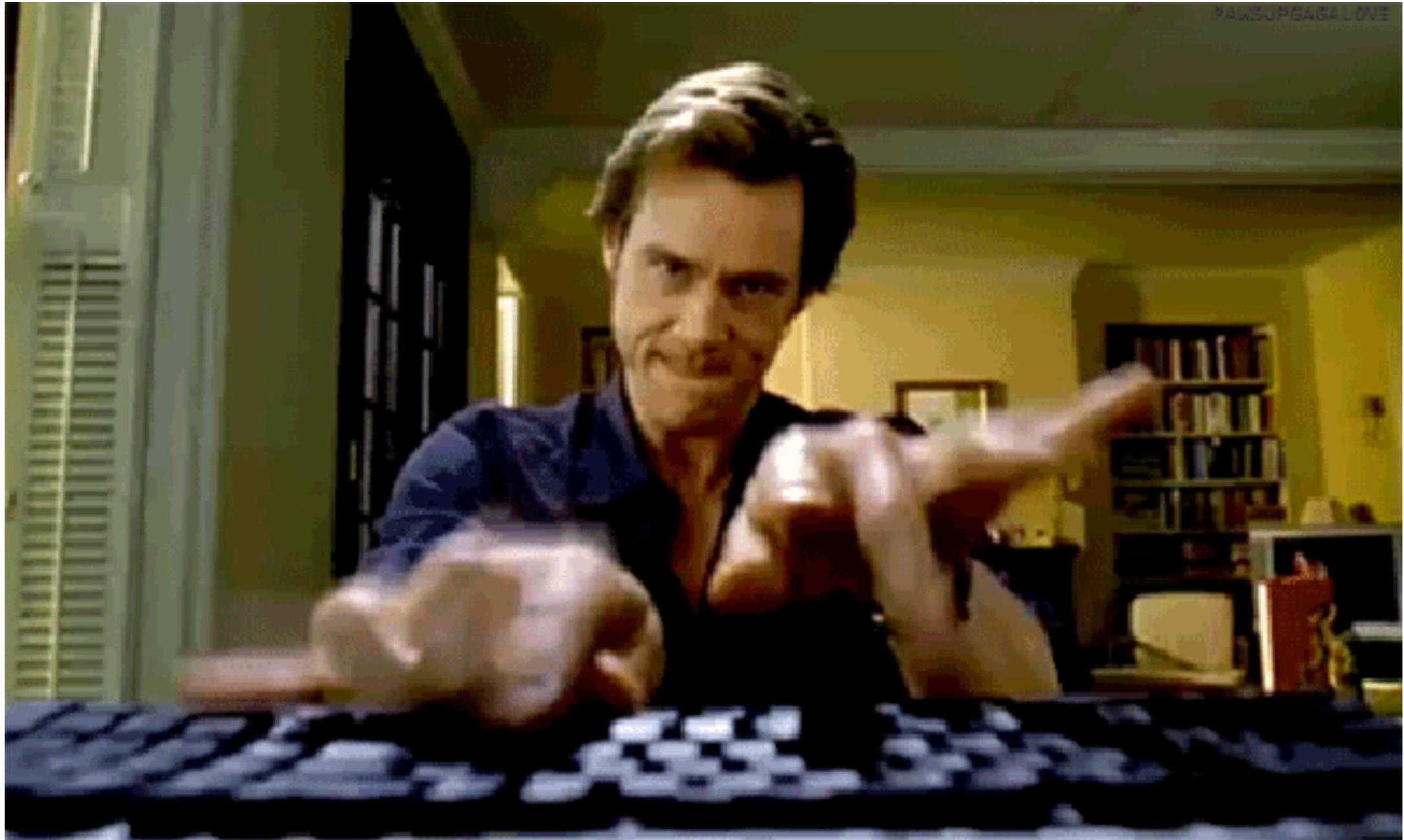


What's NNNNew in Android Security?



Android Security
Symposium 2017

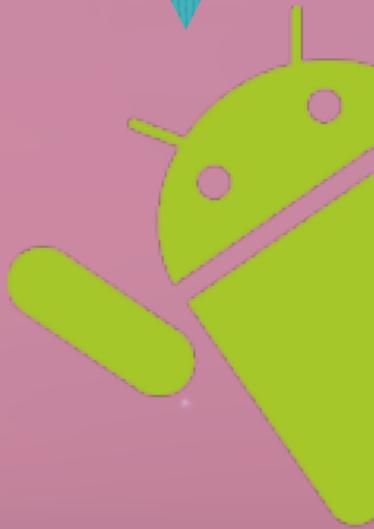
Scott Alexander-Bown
@ScottyAB



At a glance

Slides/Links

<https://goo.gl/sL5z7U>



- Direct boot
- Android Keystore (Key Attestation)
- 'Securer' networking
- Misc system and app differences
- SafetyNet

@ScottyAB

Terms

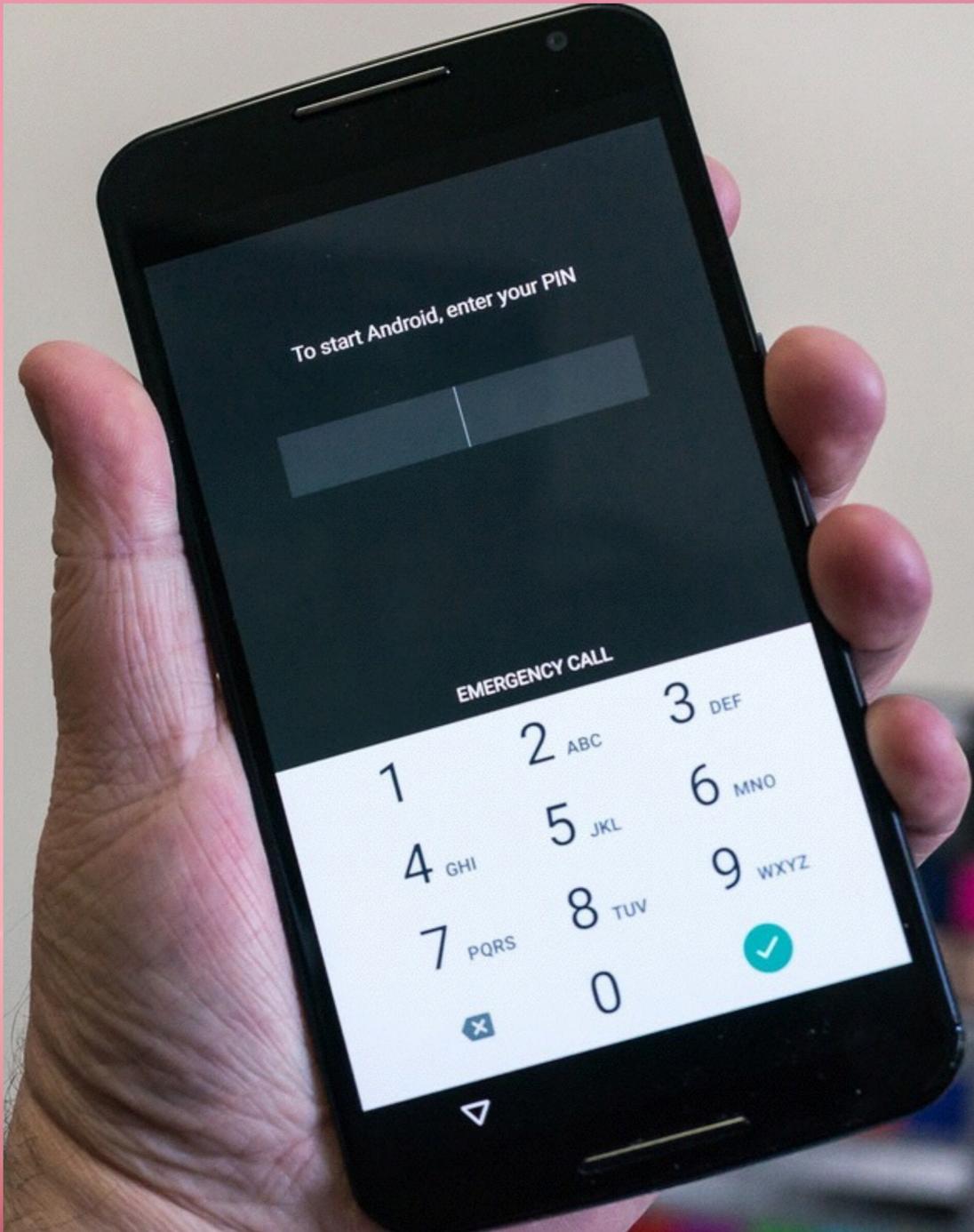
6.0 - M - API 23 - Marshmallow

7.0 - N - API 24 - Nougat

7.1 - N (MR1) - API 25 - Nougat

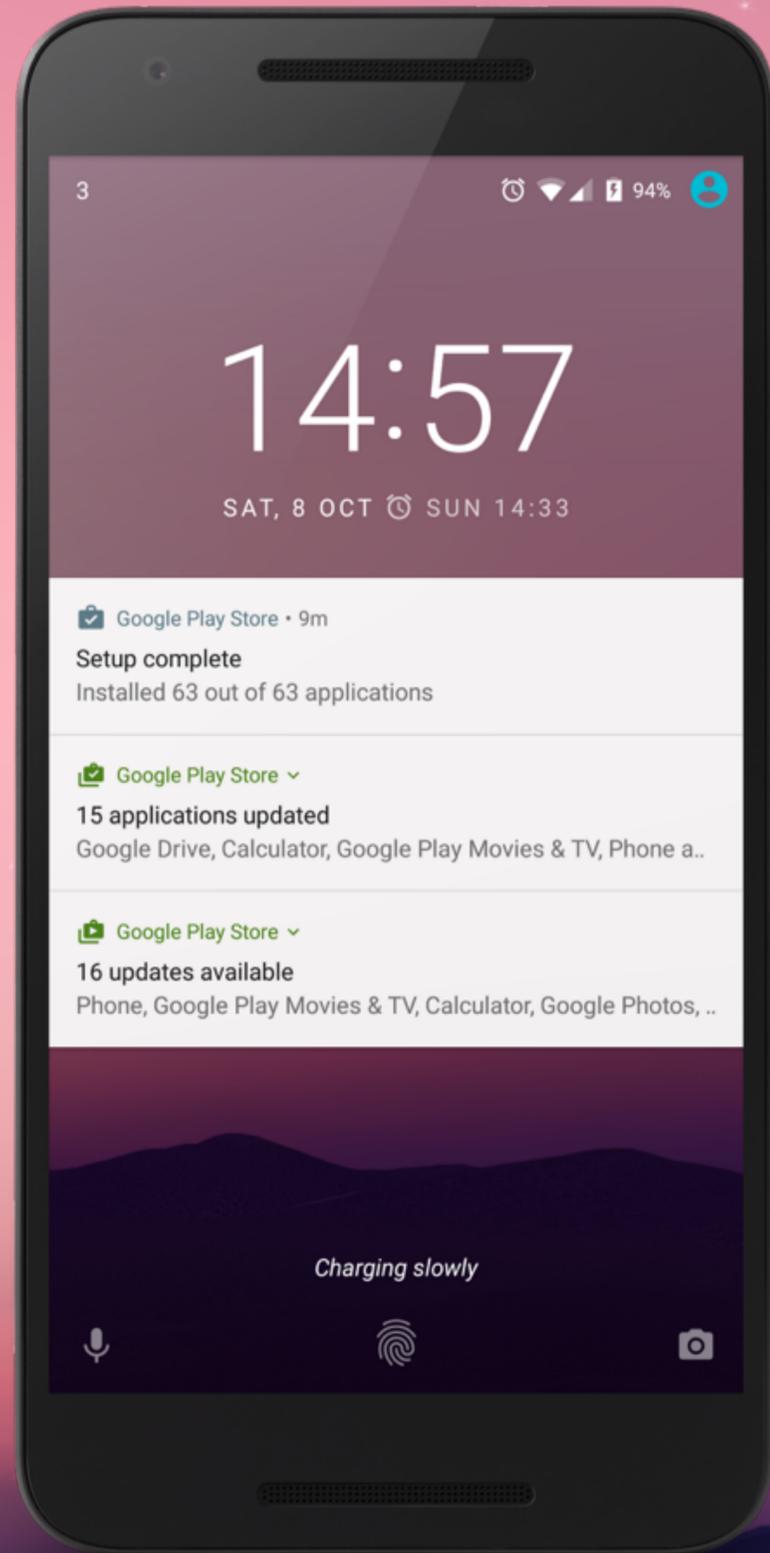


Direct Boot



Booting encrypted device pre-7.0

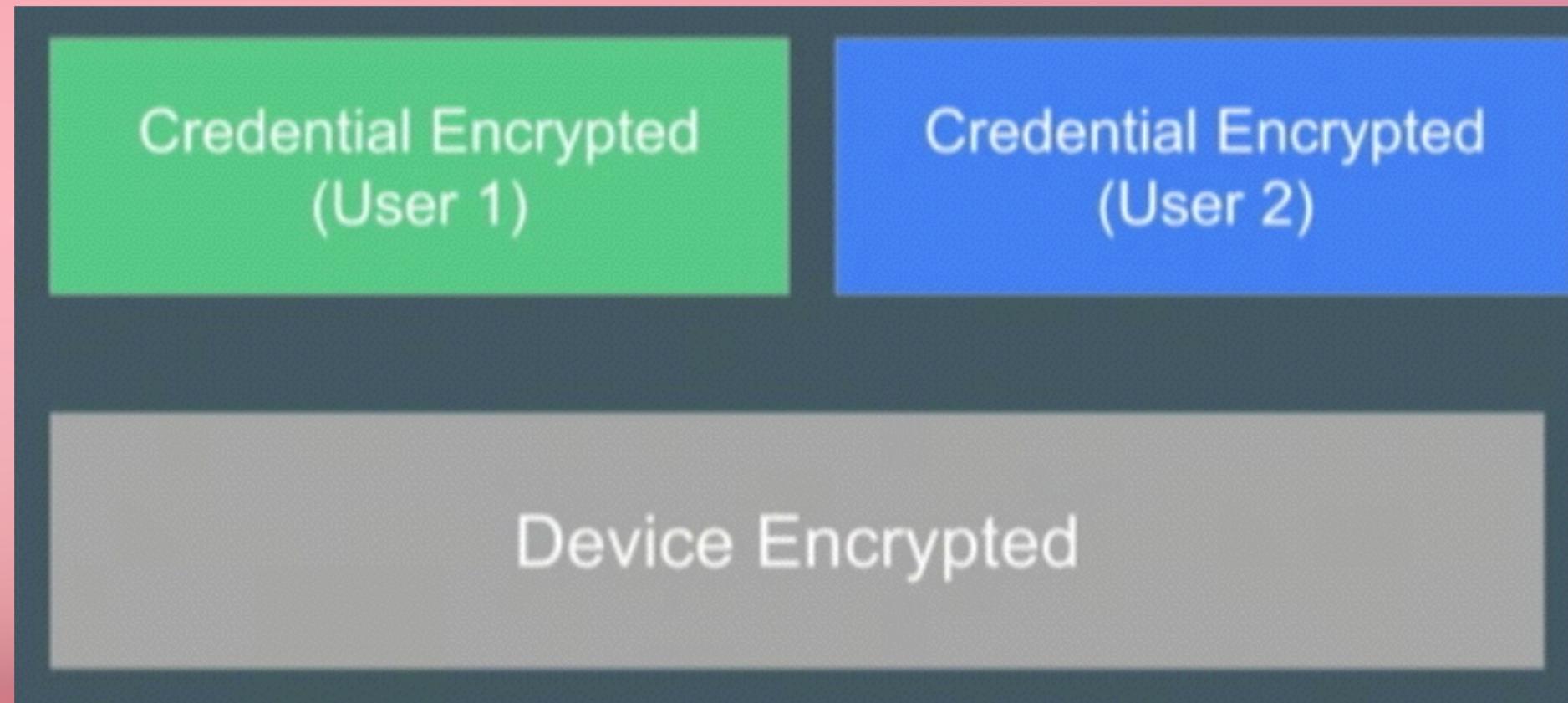
- Boot halted for pin/password
- Device encrypted with same key
- Android used block-level encryption



Direct Boot mode

- Boot direct to lock screen
- Calls, SMS & Alarms work
- And your app too!

File based encryption



← Default



Direct Boot aware <receiver />

```
<receiver
  android:name=".directboot.MyDirectBootAwareReceiver"
  android:directBootAware="true">
  <intent-filter>
    <action
android:name="android.intent.action.ACTION_LOCKED_BOOT_COMPLETED" />
    </intent-filter>
  </receiver>
```

Direct Boot aware <receiver />

```
<receiver
  android:name=".directboot.MyDirectBootAwareReceiver"
  android:directBootAware="true">
  <intent-filter>
    <action
  android:name="android.intent.action.ACTION_LOCKED_BOOT_COMPLETED" />
  </intent-filter>
</receiver>
```

Accessing device encrypted storage

```
@Override
public void onReceive(Context context, Intent intent) {

    Context directBootContext =
ContextCompat.createDeviceProtectedStorageContext(context);

    if (directBootContext != null) {
        SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(directBootContext);

        String token = sharedPreferences.getString(READ_ONLY_OAUTH_TOKEN, null);

        //do read only API lookup
        ///...
    }
}
```

Accessing device encrypted storage

```
@Override
public void onReceive(Context context, Intent intent) {

    Context directBootContext =
ContextCompat.createDeviceProtectedStorageContext(context);

    if (directBootContext != null) {
        SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(directBootContext);

        String token = sharedPreferences.getString(READ_ONLY_OAUTH_TOKEN, null);

        //do read only API lookup
        ///...
    }
}
```

@ScottyAB

Accessing device encrypted storage

```
@Override
public void onReceive(Context context, Intent intent) {

    Context directBootContext =
ContextCompat.createDeviceProtectedStorageContext(context);

    if (directBootContext != null) {
        SharedPreferences sharedPreferences =
PreferenceManager.getDefaultSharedPreferences(directBootContext);

        String token = sharedPreferences.getString(READ_ONLY_OAUTH_TOKEN, null);

        //do read only API lookup
        ///...
    }
}
```

Direct Boot, so what is it good for?

- Messaging apps, important user notifications.
- Already using a BootCompleted listener?
- Device encrypted storage for limited scope API tokens i.e Readonly

Android Keystore

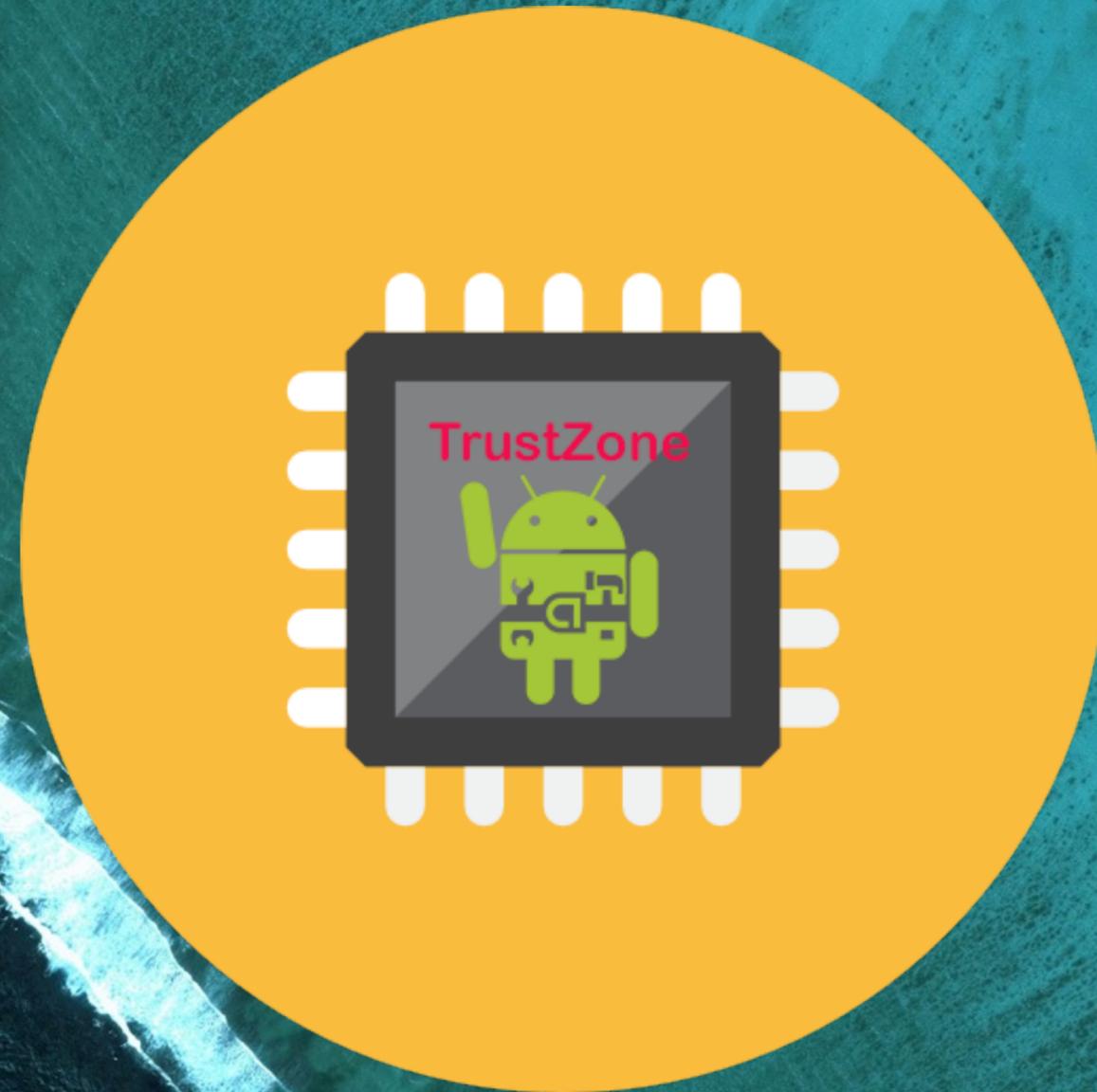


Android 4.3



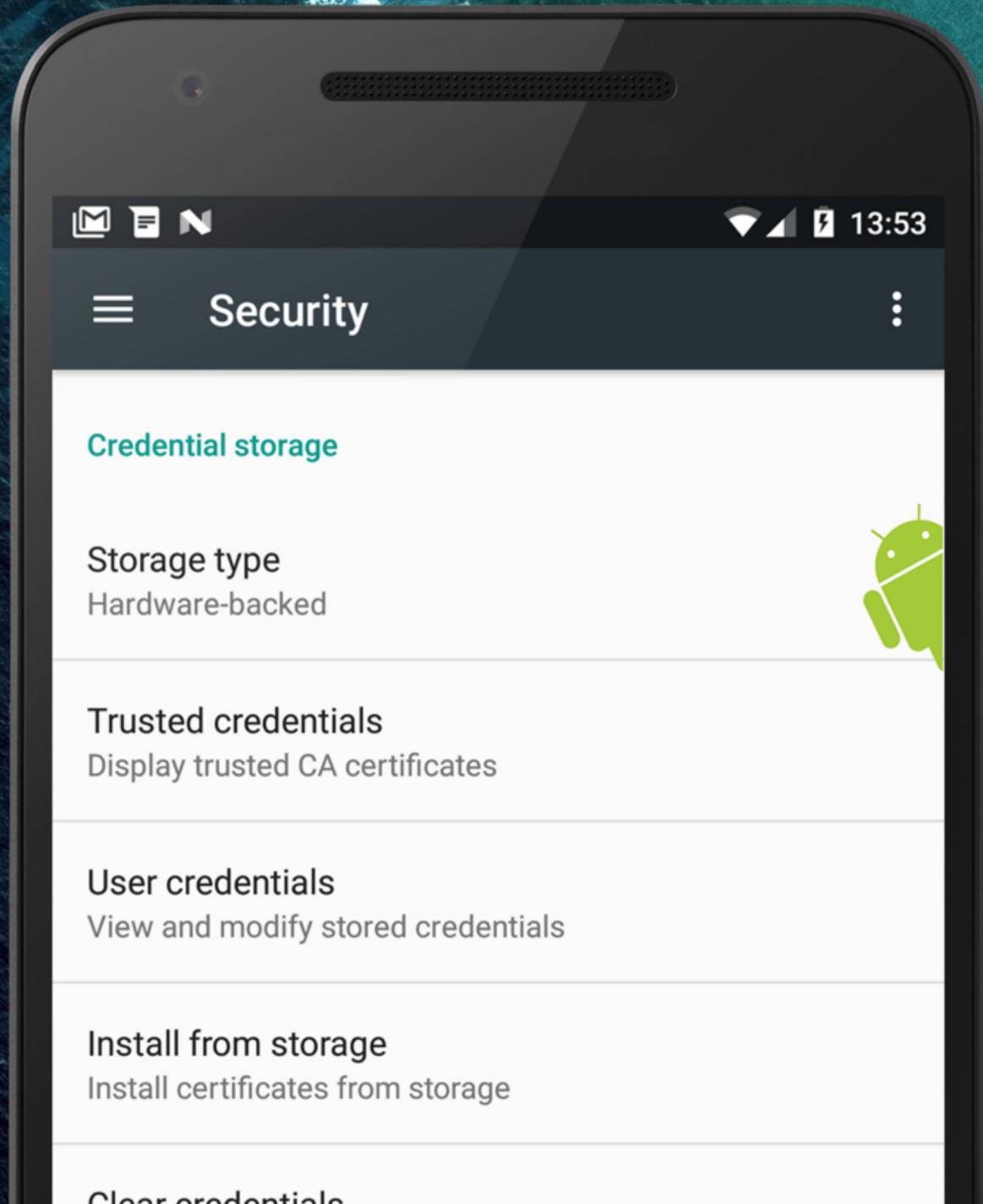
@ScottyAB

What is the KeyStore?



@ScottyAB

What's new?



- Android M introduced broader range of capabilities.
- N+ must be hardware backed (new devices)
- Time sensitive (Android M)

@ScottyAB

is the Keystore hardware backed?

```
// deprecated
```

```
KeyChain.isBoundKeyAlgorithm("RSA");
```

```
// Recommended alternative
```

```
PrivateKey key = ...; // private key from KeyChain
```

```
KeyFactory keyFactory = KeyFactory.getInstance(key.getAlgorithm(),  
"AndroidKeyStore");
```

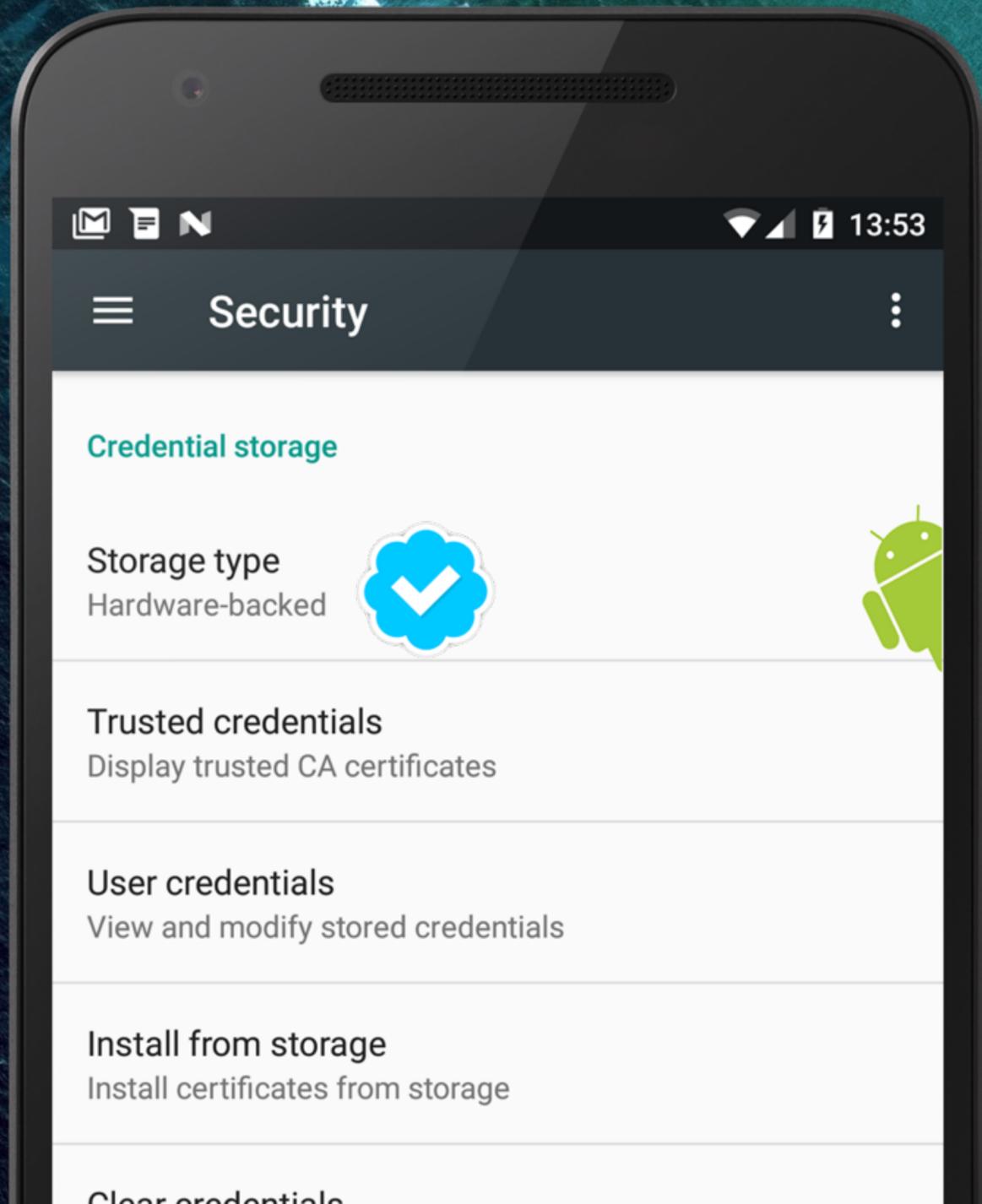
```
KeyInfo keyInfo = keyFactory.getKeySpec(key, KeyInfo.class);
```

```
if (keyInfo.isInsideSecureHardware()) {
```

```
    // The key is bound to the secure hardware of this Android
```

@ScottyAB

Key Attestation



- verify key is stored in hardware-backed keystore
- N+ (New hardware)
- Special key is baked into the firmware

Code

- `keyStore.getCertificateChain(alias)`
- Send cert chain to your server
- validate the cert chain (on your server!)

By @doriancussen



Android Security: The Forgetful Keystore

Written on February 15, 2015

Updated 13/06/2016

You've just moved in to a new house and have been given the master key for the front door. You only have one of these you know you need to keep it safe. Your really paranoid so you hire an armed guard, whose sole job is to protect this ke fact, this is all he has been trained to do and has a catchy slogan of "need to protect a key, its what I was born to do!". Y

<network-security-config>

@ScottyAB

Securer Networking

- Custom trust store / anchors
- Debug only Overrides CA
- Block non https traffic
- Limit the certs you trust

minSdkVersion=24?



CWAC-NetSecurity by Mark Murphy



<https://github.com/commonsguy/cwac-netsecurity>

@ScottyAB

Configuring CAs for Debugging

- Self signed certs in development
- Only enabled when `android:debuggable=true`
- Safer than conditional code

Configuring CAs for Debugging

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

Manifest

```
<application
  android:icon="@mipmap/ic_launcher"
  android:label="@string/app_name"
  android:networkSecurityConfig=
"@xml/network_security_config_debug_ca" />
```

Manifest

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:networkSecurityConfig=
"@xml/network_security_config_debug_ca" />
```

User certs not trusted by default*

*Running on API 24+ and targeting API 24+



@ScottyAB

Trusting user installed certs

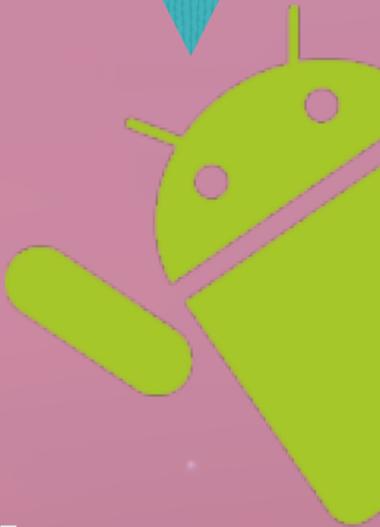
```
<network-security-config>  
  <debug-overrides>  
    <trust-anchors>  
      <certificates src="user" />  
    </trust-anchors>  
  </debug-overrides>  
</network-security-config>
```

Gist: <https://goo.gl/KN1QLp>

@ScottyAB

Pinning Certificates

Talk this p.m on
SSL pinning



- SSL pinning lets apps limit the set of certificates they accept
- Pin a hash of the SubjectPublicKeyInfo of the X.509 certificate.

@ScottyAB

SSL Pinning

```
<network-security-config>
  <domain-config>
    <domain>scottyab.com</domain>
    <pin-set expiration="2018-03-08">
      <pin digest="SHA-256">7HIpactkIAq2Y49...Y=</pin>
      <!-- backup pin -->
      <pin digest="SHA-256">fwza0LRMXouZHR...E=</pin>
    </pin-set>
  </domain-config>
</network-security-config>
```

How to get the Pin?

SSL Pin Generator

Is a simple Java base util to generate SSL pins based on a certificate's Subject Public Key Info as described on [Adam Langley's Weblog](#) (a.k.a Public Key pinning). Pins are base-64 SHA-1 [default] hashes, consistent with the format Chromium uses for [static certificates](#). See Chromium's [pinsets](#) for hostnames that are pinned in that browser.

I created this mainly to be compatible with [okhttp](#) 2.1+, but later added the option to specific which hashing algorithm can be used to make this compatible with Android's `<network-security-config>`

Usage

Warning you should ensure you run this on a trusted network

Download the jar [here](#) or clone and compile the class.

Simply pass to hostname with optionally port, and algorithm to the jar. `$ java -jar generatePins.jar <your hostname:port> algorithm`

Default

<https://goo.gl/mupcRk>

How to get the Pin?

```
$ openssl s_client -servername scottyab.com  
-connect scottyab.com:443 | openssl x509 -  
pubkey -noout | openssl rsa -pubin -outform  
der | openssl dgst -sha256 -binary | openssl  
enc -base64
```

Thanks to John Kozyrakis @ikoz

@ScottyAB

Misc

NEW

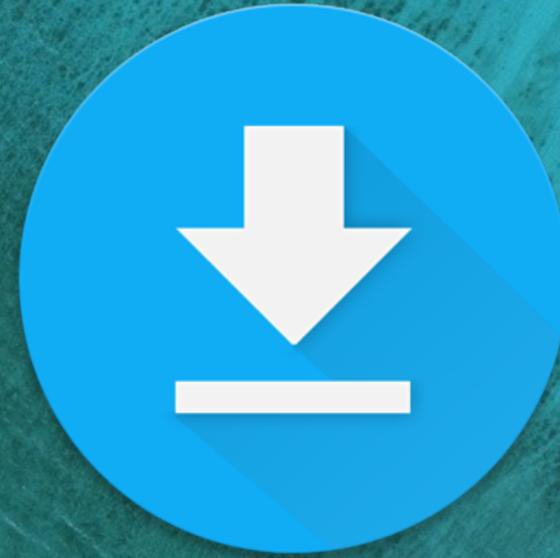
@ScottyAB

Under the hood



- The media stack and platform hardening
- Kernel hardening (with error correction)

@ScottyAB



Seamless OTA updates

@ScottyAB



00:08

TEST



Allow Tapjacking to
access your contacts?

Never ask again

DENY

ALLOW



00:10

TEST

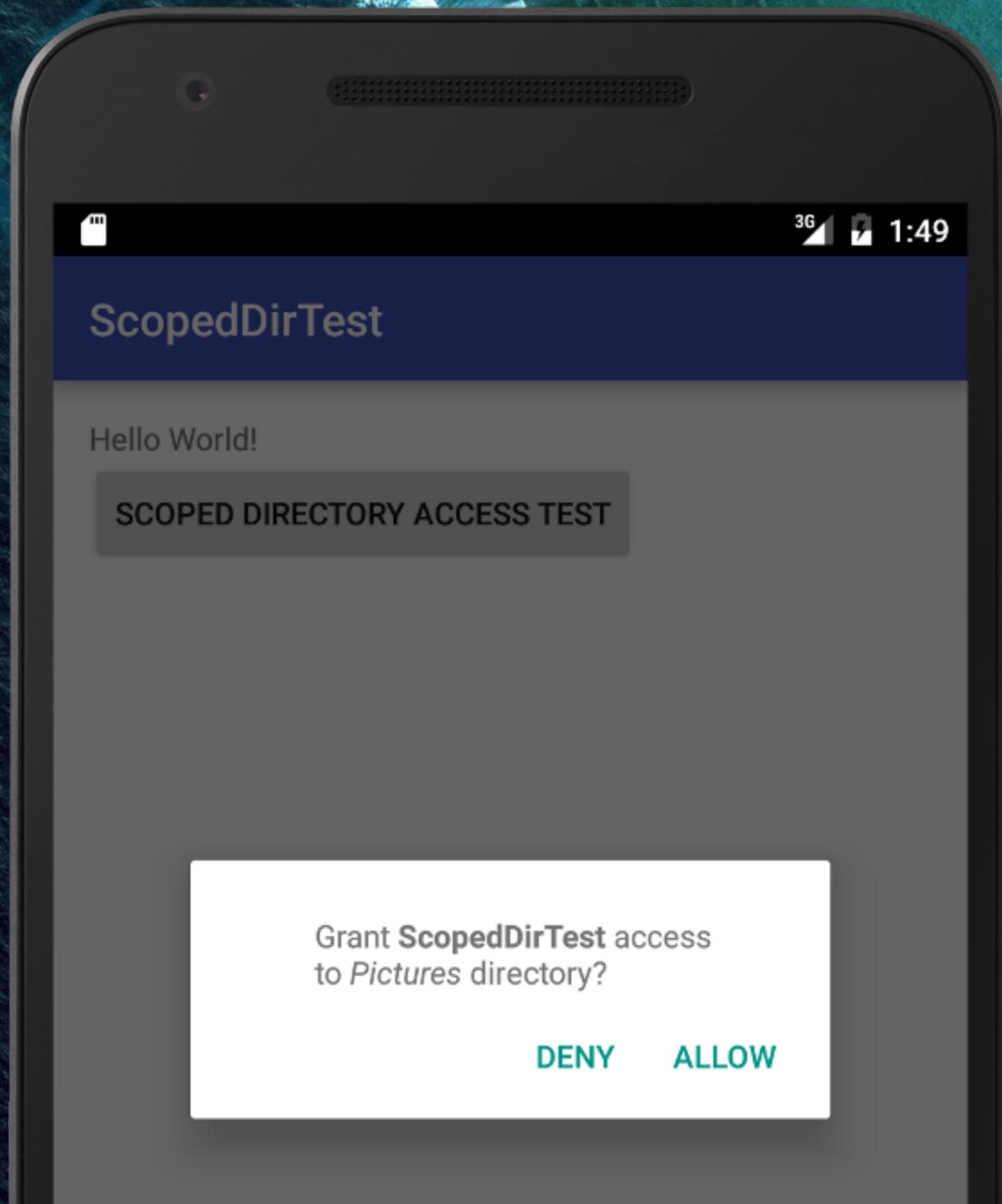
Some message covering
the permission message.

Never ask again

DENY

ALLOW

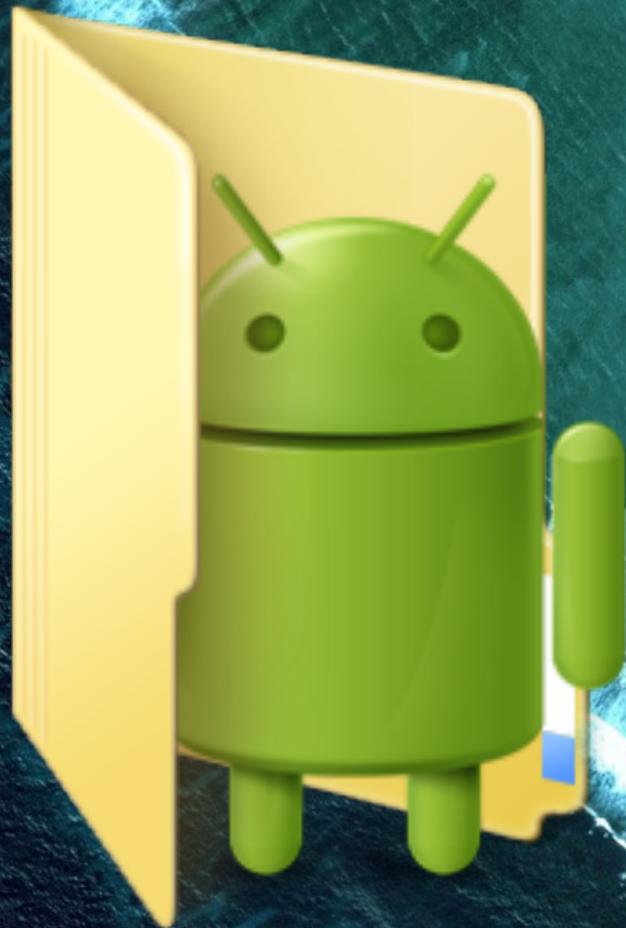
Scoped directory access



- Storage Access Framework
- `Environment.DIRECTORY_MOVIES`
- Remember to call `takePersistableUriPermission()`

@ScottyAB

App data directory



- App data directory now user only
700 permissions
- Sharing files is explicitly opt-in
- Use FileProvider (support-lib)
- content:// URI instead of file://
- Training article “Sharing Files”

@ScottyAB

APK signing schema v1



- Problems
- Deleting files
- adding files to meta-inf
- DOS app

APK signing schema v2

A green rectangular icon with the letters 'APK' in white, positioned above a white document icon with a folded top-right corner. The document icon features a green Android robot logo with the letters 'APK' written across its chest.

- Faster
- More Secure
- You're already using both?
- zipalign before (not after)

Access to Hardware Identifier



```
wifiManager.getConnectionInfo().getMacAddress()
```

```
BluetoothAdapter.getDefaultAdapter().getAddress()
```

```
D/MAC: Device:Pixel, SDK:25  
D/MAC: via WifiManager: 02:00:00:00:00:00  
D/MAC: via BluetoothAdapter: 02:00:00:00:00:00  
D/MAC: via NetworkInterface: AC:37:43: [REDACTED]
```

Permissions required by libraries.

The screenshot shows an IDE window titled "AndroidManifest.xml - mib.android.totalhealth". The main editor displays the XML content of the manifest file, with the following structure:

```
<manifest
  android:versionCode="1000000"
  android:versionName="1.0.0"
  package="com.enquos.totalhealth.scott.ide"
  xmlns:android="http://schemas.android.com/apk/res/android" >
  <uses-sdk
    android:minSdkVersion="21"
    android:targetSdkVersion="24" />
  <uses-permission
    android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission
    android:name="android.permission.BLUETOOTH" />
  <uses-permission
    android:name="android.permission.BLUETOOTH_ADMIN" />
  <uses-permission
    android:name="android.permission.INTERNET" />
  <uses-feature
    android:name="android.hardware.sensor.stepcounter"
    android:required="false" />
  <uses-feature
    android:name="android.hardware.sensor.barometer"
    android:required="false" />
  <uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE" />
  <uses-feature
    android:glEsVersion="0x00020000"
    android:required="true" />
  <uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
  <uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <android:uses-permission
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

The "Manifest Sources" panel on the right lists the following sources:

- app dev manifest
- app main manifest (this file)
- play-services-base:9.6.1 manifest
- play-services-basement:9.6.1 manifest
- play-services-maps:9.6.1 manifest
- rxandroidble:1.0.1 manifest
- leakcanary-android:1.4 manifest

The "Other Manifest Files" panel lists various library manifests that were included in the merge but did not contribute any elements:

- library-recyclerviewclicksupport manifest
- constraint-layout:1.0.0-alpha9 manifest
- animated-vector-drawable:24.2.1 manifest
- appcompat-v7:24.2.1 manifest
- cardview-v7:23.3.0 manifest
- design:24.2.1 manifest
- multidex:1.0.1 manifest
- recyclerview-v7:24.2.1 manifest
- support-compat:24.2.1 manifest
- support-core-ui:24.2.1 manifest
- support-core-utils:24.2.1 manifest
- support-fragment:24.2.1 manifest
- support-media-compat:24.2.1 manifest
- support-v4:24.2.1 manifest
- support-vector-drawable:24.2.1 manifest
- proguard-rules:1.1.6.0 manifest
- answers:1.3.8 manifest
- beta:1.2.1 manifest
- crashlytics-core:2.3.10 manifest
- crashlytics:2.6.1 manifest
- relinker:1.2.1 manifest
- Permissive:1.0.0 manifest
- Calligraphy:v2.1.1 LABEL FOR manifest
- play-services-location:9.6.1 manifest
- play-services-tasks:9.6.1 manifest
- android-maps-utils:0.4 manifest
- timber:4.1.2 manifest
- butterknife:8.1.0 manifest
- fabric:1.3.12 manifest
- fab-speed-dial:1.0.4 manifest
- rxandroid:1.2.1 manifest
- realm-android-library:1.2.0 manifest
- android.joda:2.9.3.1 manifest
- android-reactive-location:0.9 manifest

The "Merging Log" panel shows:

Added from the [play-services-maps:9.6.1](#) manifest, line 18

A white arrow points to the `android.permission.ACCESS_NETWORK_STATE` permission in the manifest file.

@ScottyAB

SafetyNet API



App info

Google Play services
Version 7.6.03 (1954927-43)

DISABLE

SafetyNetApi.attest(...)



- Read device?
- Vulnerable?
- Rooted?

<https://github.com/scottyab/safetynethelper>

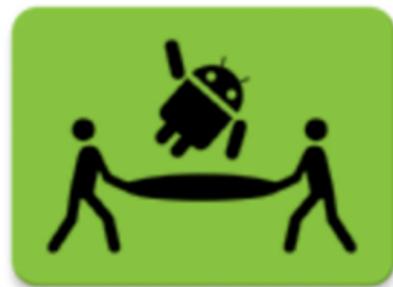
README.md

SafetyNet `attest()` Helper

SafetyNet Helper wraps the Google Play Services SafetyNet.API and verifies Safety Net API response with the [Android Device Verification API](#). The SafetyNet.API analyses the device your app is running on to test its software/hardware configuration matches that of a device that has passed the Android Compatibility Test Suite (CTS). Note this is a client only validation, it's recommended to include [server side validation](#).

Rooted devices seem to cause `ctsProfileMatch=false` .

Recommend reading the developers guide to getting started with [SafetyNet](#)



play.google.com/store/apps/details?id=com.scottyab.safetynet.sample



SafetyNet Helper Sample

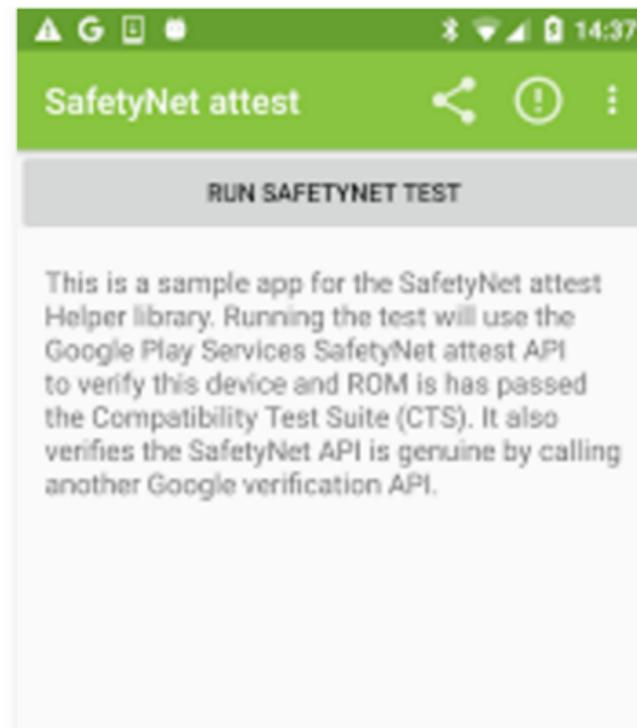
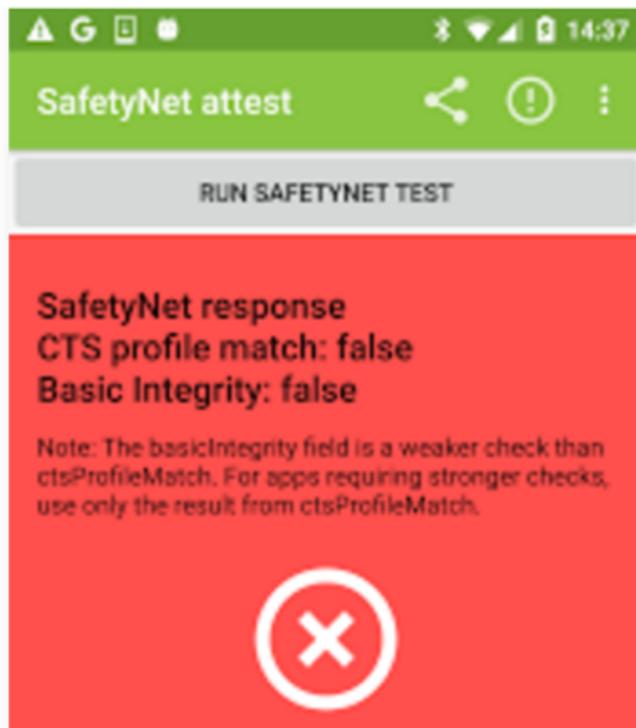
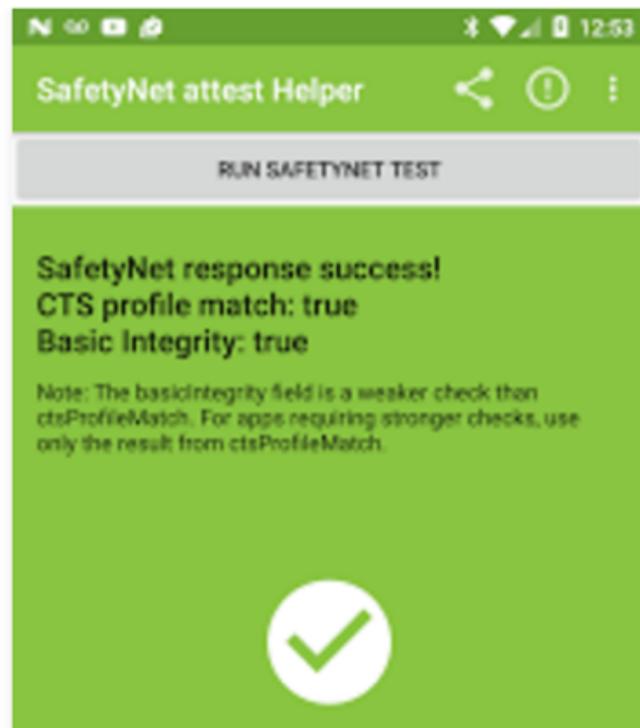
Scott Alexander-Bown Tools

★★★★★ 402

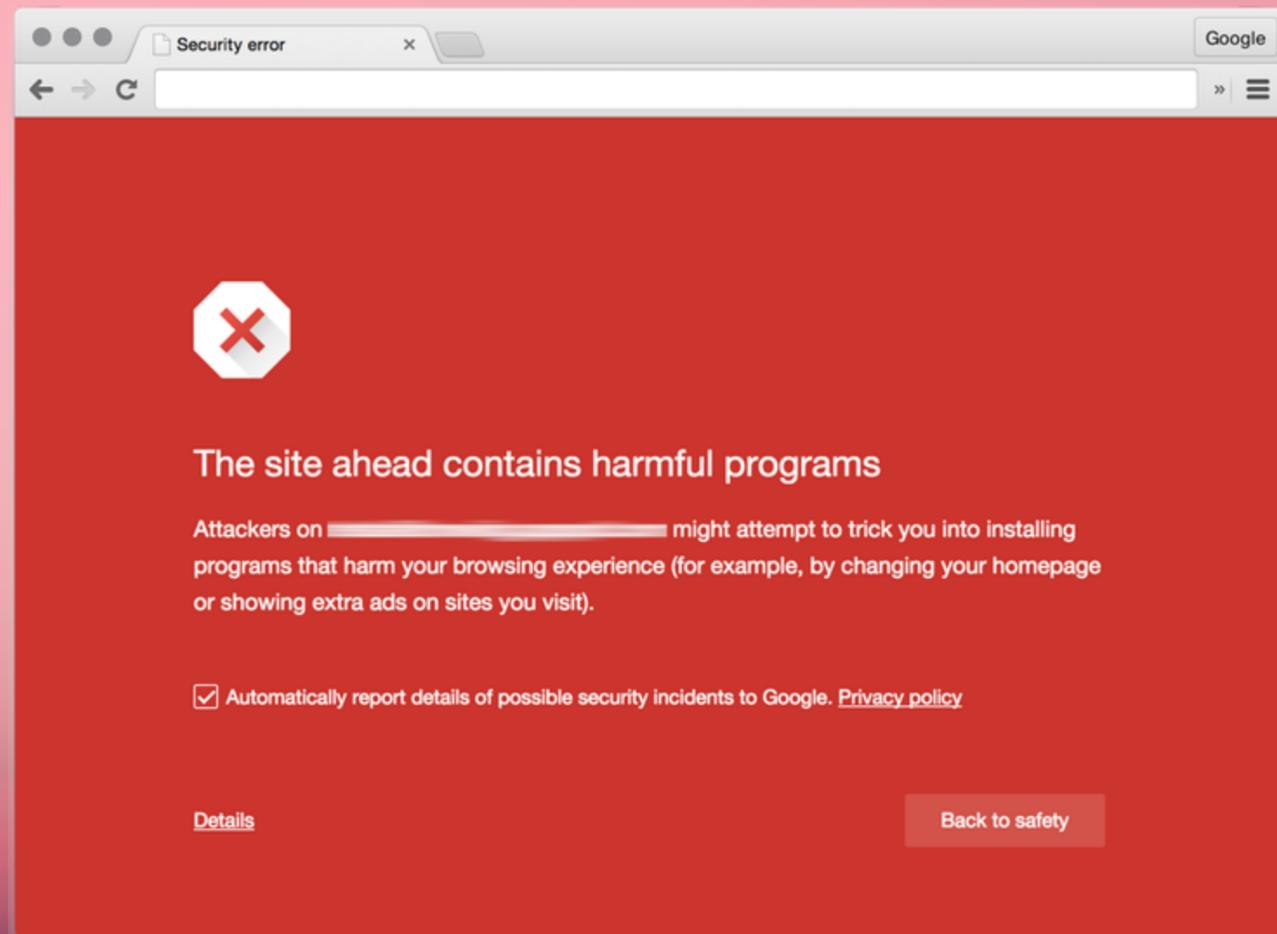
3 PEGI 3

i This app is compatible with some of your devices.

Installed

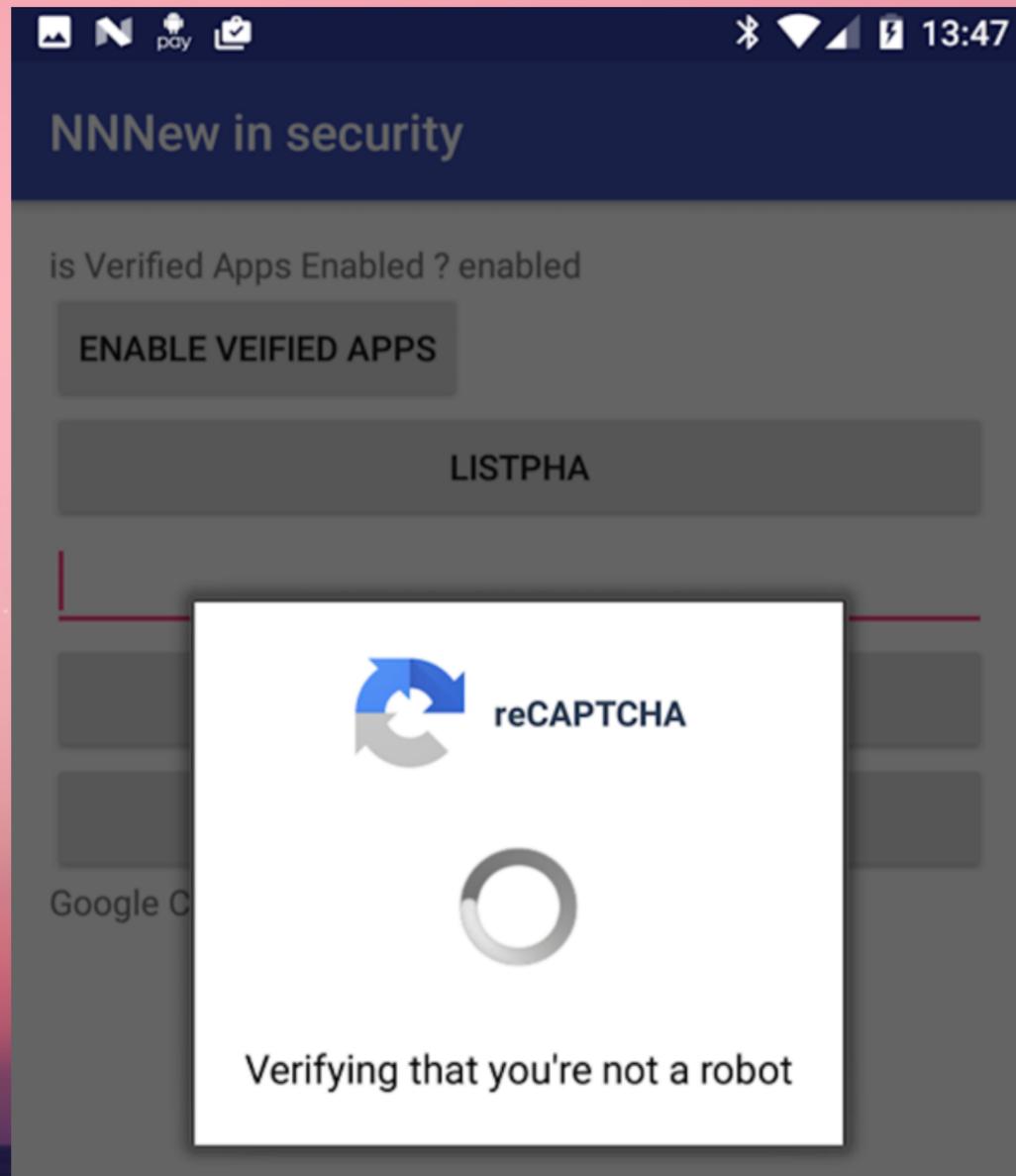


SafetyNetApi.lookupUri(...)



- Social Engineering
- Potentially Harmful Apps

SafetyNetApi - Misc



- Check Verified Apps status
- Enable Verified Apps
- List installed Potentially Harmful Apps (PHA)
- reCAPTCHA Integration

Recap

- Direct boot**
- Android Keystore (Key Attestation)**
- 'Securer' networking**
- Misc system and app differences**
- SafetyNet**

Slides/Links
<https://goo.gl/sL5z7U>

Thanks for listening

Shout outs:
@commonsguy
@ikoz
+AdrianLudwig
@doriancussen
@niallscott
@trionkidnapper
@subsymbolics

Scott Alexander-Bown
@ScottyAB
hello@scottyab.com

Hire me



Resources

- ❑ <https://www.blackhat.com/ldn-15/summit.html#what-can-you-do-to-an-apk-without-its-private-key-except-repacking>
- ❑ <https://doridori.github.io/android-security-the-forgetful-keystore/#sthash.hFHQpV3A.5WcUVfYk.dpbs>
- ❑ <http://android-developers.blogspot.co.uk/2016/09/security-enhancements-in-nougat.html>
- ❑ https://developer.android.com/about/versions/nougat/android-7.0.html#apk_signature_v2
- ❑ <https://blog.stylingandroid.com/nougat-direct-boot/>
- ❑ SafetyNet Helper library <https://github.com/scottyab/safetynethelper>
- ❑ Security patch date util - <https://gist.github.com/scottyab/77bac6600986eb6a619e07a3d0abae3f>
- ❑ *Adrian Ludwig's Google IO talk - What's new in Android Security (M &N) - <https://www.youtube.com/watch?v=XZzLjllizYs>

Training / Developer Docs

- <https://developer.android.com/training/articles/security-key-attestation.html>
- <https://developer.android.com/training/articles/scoped-directory-access.html#accessing>
- https://developer.android.com/training/articles/user-data-permissions.html#tenets_of_working_with_android_permissions
- <https://developer.android.com/training/articles/direct-boot.html>

@ScottyAB

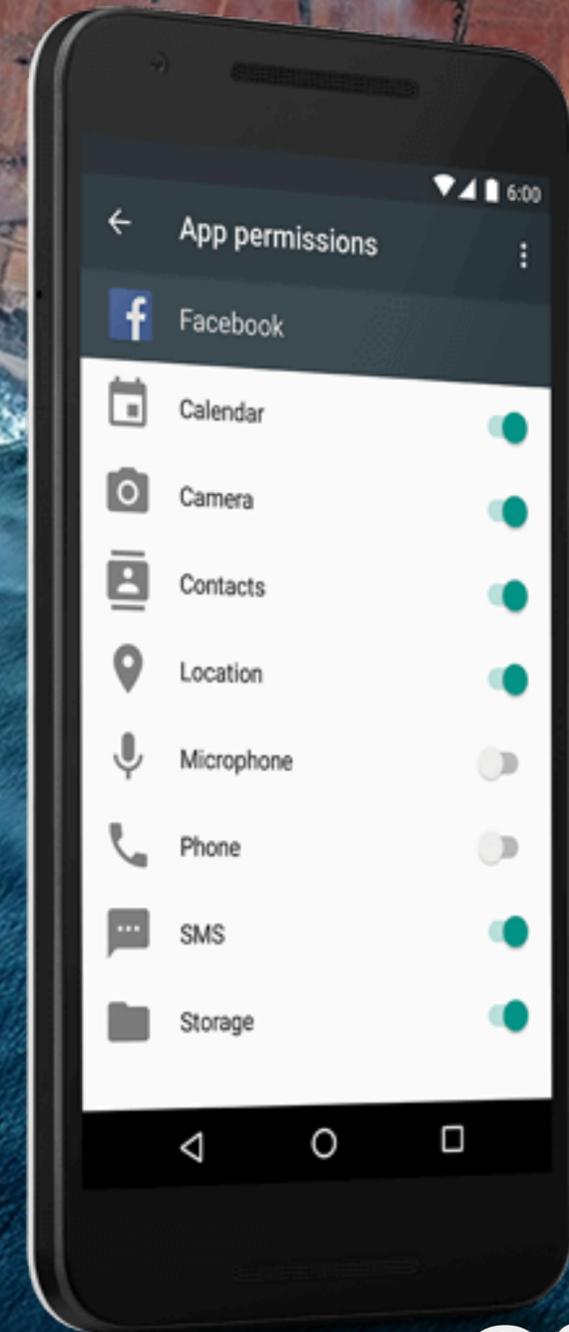
Runtime Permissions



- Permissions that users “get”
- Control on specific permissions
- Easy for users
- Updates don't require approval

Tips of Working with Android Permissions

- Only use the permissions necessary for your app to work
- Be transparent
- Make system accesses explicit
- Context, Context, Context!



@ScottyAB