# MANICODE

SECURE CODING EDUCATION

**JIMMY MESTA** Secure Coding Instructor *www.manicode.com*

MANICODE

SECURE CODING EDUCATION

# Securing Vintage Blackberry and Windows Phones


**JIMMY MESTA** Secure Coding Instructor *www.manicode.com*

MANICODE

SECURE CODING EDUCATION

# Mobile is Eating the World

# ***But is it Secure?***

**JIMMY MESTA** Secure Coding Instructor *www.manicode.com*

# A little background dirt…

@jimmesta

- CTO @ Manicode Security
- 10 years of penetration testing, teaching, and building security programs
- OWASP AppSec California organizer and Santa Barbara chapter founder
- Conference speaker
- Been on both sides of the InfoSec fence
- Lives in The Cloud


**WARNING**: Please do not attempt to hack any computer system without legal permission to do so. Unauthorized computer hacking is illegal and can be punishable by a range of penalties including loss of job, monetary fines and possible imprisonment.

**ALSO:** The *Free and Open Source Software* presented in these materials are examples of good secure development techniques. You may have unknown legal, licensing or technical issues when making use of *Free and Open Source Software*. You should consult your company's policy on the use of *Free and Open Source Software* before making use of any software referenced in this material.

OWASP Mobile Top Ten (Wall of Shame)

iOS and Android Architecture

iOS and Android Application Anatomy

iOS and Android Attack Surface











# Mobile Attack Surface

Device

Network

Backend

# Mobile Attack Surface

## Device

**System Configuration**
- Jailbroken/Rooted
- Poor Passcode
- Supply Chain Issues

**Applications**
- Injection
- Data Storage
- Secret Storage
- Poor Encryption
- Permission Issues
- Escalated Privileges
- Malware
- Third-Party Libs

**Mobile Browser**
- Classic App Vulns
- Man-in-the-Middle

**Phone/SMS**
- SMShing
- 2FA Attacks

## Network

- DNS Spoofing
- Rogue Access Point
- SSL Strip
- Network Sniffing
- Insecure Network
- Session Hi-Jacking
- Nation State Attacks

## Backend

**Web Server / API**
- Classic App Vulns
- Brute Force
- Vulnerable Libraries
- Server Misconfigurations
- XSS
- CSRF
- Access Control
- Insecure API
- Cloud Misconfigurations
- Broken Authentication

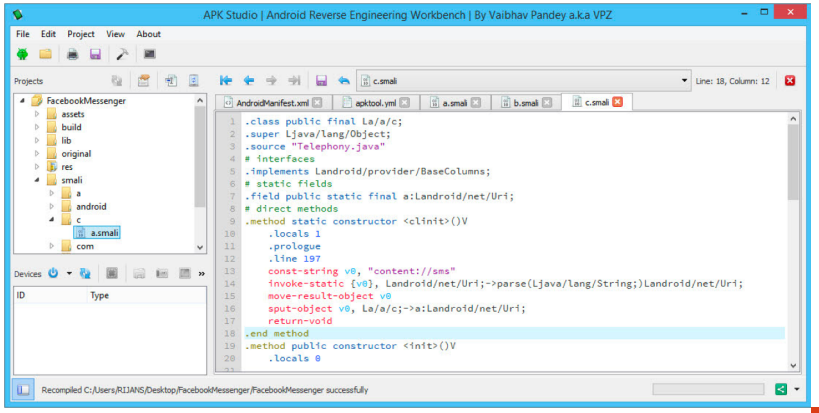
**DB**
- SQLi
- Unauthorized Data Access
- Encryption

# Mobile Attack Surface

# I Can Haz Your Traffic


# I Can Haz Your Device


# I Can Haz Your Code



# OWASP Mobile Top Ten

# Mobile Top Ten 2016

| | |
|---|---|
| **M1** | Improper Platform Usage |
| **M2** | Insecure Data Storage |
| **M3** | Insecure Communication |
| **M4** | Insecure Authentication |
| **M5** | Insufficient Cryptography |
| **M6** | Insecure Authorization |
| **M7** | Client Code Quality |
| **M8** | Code Tampering |
| **M9** | Reverse Engineering |
| **M10** | Extraneous Functionality |

# M1 Improper Platform Usage

Covers the misuse or lack of a **platform feature security control** contained within the mobile OS. Issues range from using Local Storage instead of Keychain for sensitive data to misconfigured Android intents.

# M1 Improper Platform Usage


Eavesdropper: The Mobile Vulnerability Exposing Millions of Conversations

# M1 Improper Platform Usage

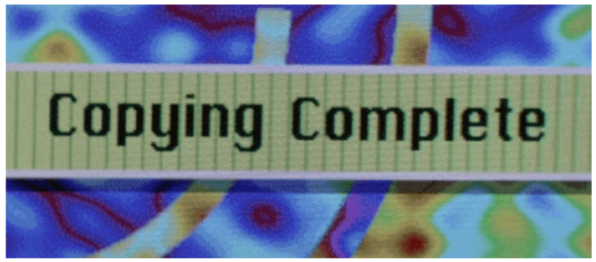
```
mysql> select * from secrets where package = 'com.[redacted]android';
+----------+-----------------------+--------------------------------------------------------------------------------------------------
| JobId    | package               | secret
+----------+-----------------------+--------------------------------------------------------------------------------------------------
| 49164528 | com.[redacted].android |     <string name="hl_aws_key">AKIA[redacted]</string>
| 49164528 | com.[redacted].android |     <string name="hl_aws_secret">Da/64uM3HH+SsXei04ZSF+/zhXvLeLyJrNSE7mRSd+3YSgcN/[redacted]</strin
```

# M1 Improper Platform Usage

## Hole In WhatsApp For Android Lets Hackers Steal Your Conversations

Jordan Crook @jordanrcrook / 5 years ago

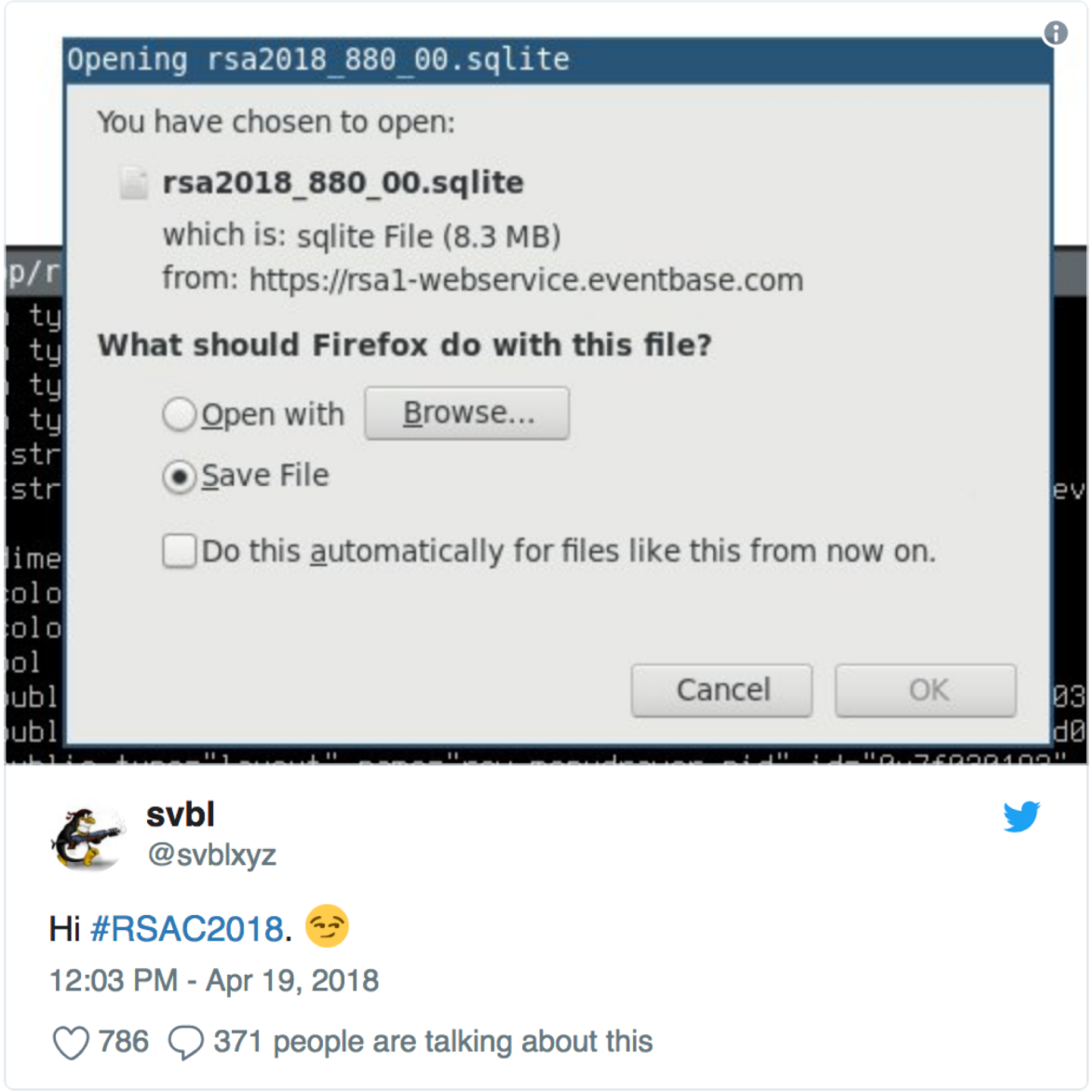
Comment




WhatsApp for Android stores conversations on the phone's SD card, which is accessible by many other apps on the phone as long as the user gives those apps the permissions they ask for…

# M2 Insecure Data Storage

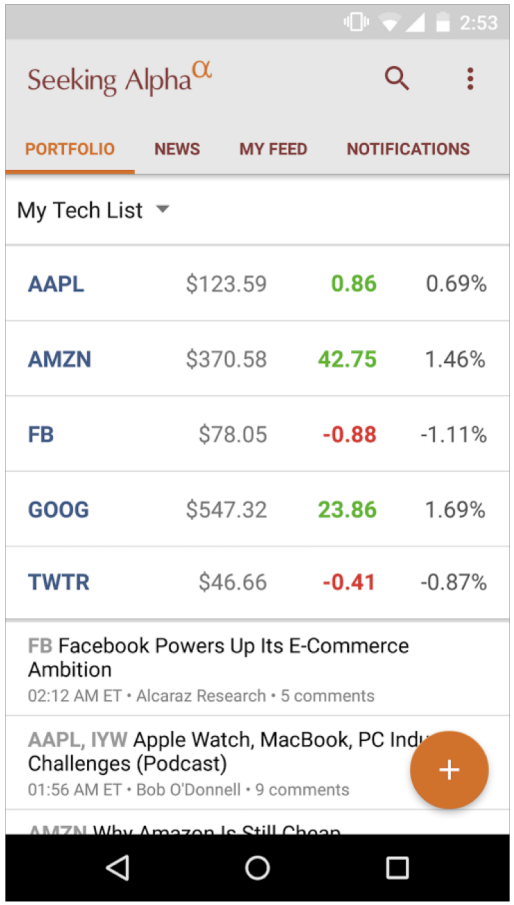
Occurs when vulnerabilities **expose or leak data in an unintended manner**. This may include log files, databases, cloud synced storage, and manifest files.



# M3 Insecure Communication

This category applies to **poor or non-existent encryption mechanisms** for data in motion.

# M3 Insecure Communication




In July 2016 researchers at Rapid 7 discovered the Seeking Alpha financial news app leaking usernames, passwords, stock selections and HTTP cookies in plaintext.

Source: Rapid 7 Blog

# M3 Insecure Communication

## Here's Why Equifax Yanked Its Apps From Apple And Google Last Week

A security researcher discovered a shocking vulnerability: "They quite frankly didn't know what they were doing."

## M3 Insecure Communication

"He found shocking results: Though Equifax's app used the secure HTTPS protocol to authenticate, **once users were in the app, it used just HTTP in a number of locations**, which makes the app vulnerable to interception. This means that any data communicated between users and Equifax is not encrypted."

# M3 Insecure Communication

Nightwatch Cybersecurity (nightwa...

| 1831 | - | 2.24 | 72nd | 22.11 | 94th |
|---|---|---|---|---|---|
| Reputation | Rank | Signal | Percentile | Impact | Percentile |

19

#281605 **RCE in TinyCards for Android**

Share:

| | | | |
|---|---|---|---|
| State | Resolved (Closed) | Severity | No Rating (---) |
| Disclosed publicly | January 4, 2018 4:11pm -0700 | Participants | |
| Reported To | Duolingo | Visibility | Public (Full) |
| Asset | com.duolingo (Android: Play Store) | | |
| CVE ID | CVE-2017-16905 | | |
| Weakness | Code Injection | | |

TinyCards loads a website via webview when starting, but that site is loaded over http then redirected to https. An MITM attack that controls either the network or the DNS, can inject their own web content into the webview. You can confirm this by using an MITM proxy to capture the traffic. Included you will find a screenshot from the proxy and the Java application file that has the incorrect URL.

# M4 Insecure Authentication

Applies to **weaknesses around identifying a user** and maintaining the integrity of that user's identity throughout a session.

# M4 Insecure Authentication

Jahrek (jahrek)

| 453 | - | 4.18 | 89th | 33.89 | 99th |
|---|---|---|---|---|---|
| Reputation | Rank | Signal | Percentile | Impact | Percentile |

19 #138101 **Weak user aunthentication on mobile application - I just broken userKey secret password**

Share:

| | | | |
|---|---|---|---|
| State | Resolved (Closed) | Severity | No Rating (---) |
| Disclosed publicly | July 27, 2016 10:28am -0700 | Participants | |
| Reported To | Pornhub | Visibility | Public (Limited) |
| Type | Authentication | | |
| Bounty | $5,000 | | |

Collapse

SUMMARY BY PORNHUB

The researcher discovered a hard coded authentication bypass on the mobile app.

# M4 Insecure Authentication

## Teen-monitoring app TeenSafe leaks thousands of user IDs and passwords

*The data was stored in plaintext*

By Thuy Ong | @ThuyOng | May 21, 2018, 5:44am EDT

# M4 Insecure Authentication

Celery Flower | Dashboard | Tasks | Broker | Monitor

Show 10 entries

| Name | UUID | State | args |
|---|---|---|---|
| worker.tasks.detect_2fa_task | 0ddb6dd1-78a2-40b0-8529-90a72f0a3da3 | SUCCESS | ({'user_id': '1098848', 'password': '[redacted]', 'apple_id': '[redacted]@yahoo.com'},) |
| worker.tasks.detect_2fa_task | fe0cf612-afcb-4f49-88a7-0d4b94f28570 | SUCCESS | ({'user_id': '3725419', 'password': '[redacted]', 'apple_id': '[redacted]@icloud.com'},) |
| worker.tasks.detect_2fa_task | d7e2f813-1a9f-40f6-8338-1f4283296b39 | SUCCESS | ({'user_id': '3725419', 'password': '[redacted]', 'apple_id': '[redacted]@icloud.com'},) |
| worker.tasks.detect_2fa_task | 3f4fa1d3-0be5-4cd6-a723-cf4909527ac9 | SUCCESS | ({'user_id': '2829522', 'password': '[redacted]', 'apple_id': '[redacted]@icloud.com'},) |
| worker.tasks.detect_2fa_task | 424ab922-5632-4400-9141-d993d32cd3d3 | SUCCESS | ({'user_id': '1667970', 'password': '[redacted]', 'apple_id': '[redacted]@yahoo.com'},) |

# M5 Insecure Cryptography

The mechanism used to **encrypt and decrypt sensitive data is flawed** and may allow an adversary to access the data.

# M6 Insecure Authorization

Refers to the failure of a mobile applications ability to **properly enforce identity and access permissions**.

clarckowen_

| 128 | - | -1.55 | 46th | 19.17 | 92nd |
|---|---|---|---|---|---|
| Reputation | Rank | Signal | Percentile | Impact | Percentile |

41

#175490

# Able to Login deactivated staff account in shopify app mobile

Share:

| | | | |
|---|---|---|---|
| State | Resolved (Closed) | Severity | No Rating (---) |
| Disclosed publicly | November 29, 2016 5:32am -0800 | Participants | |
| Reported To | Shopify | Visibility | Public (Full) |
| Type | Privilege Escalation | | |
| Bounty | $2,000 | | |

Collapse

SUMMARY BY CLARCKOWEN_

Authentication Bypass occured to <any-store>.shopify.com for Deactivated Staff Acounts in Mobile Application

TIMELINE

clarckowen_ submitted a report to **Shopify**. Oct 12th (4 months ago)

Hi Shopify,

Deactivated staff account is able to login in shopify mobile app.

**STEPS**

1. Login your owner account
2. Go to Staff Accounts and deactivate your staff account
3. Login to your staff account in your shopify mobile app

**As you can see you were able to login even the staff account was deactivated by the account owner**

1 attachment:
**F127573:** shopifymobile.png

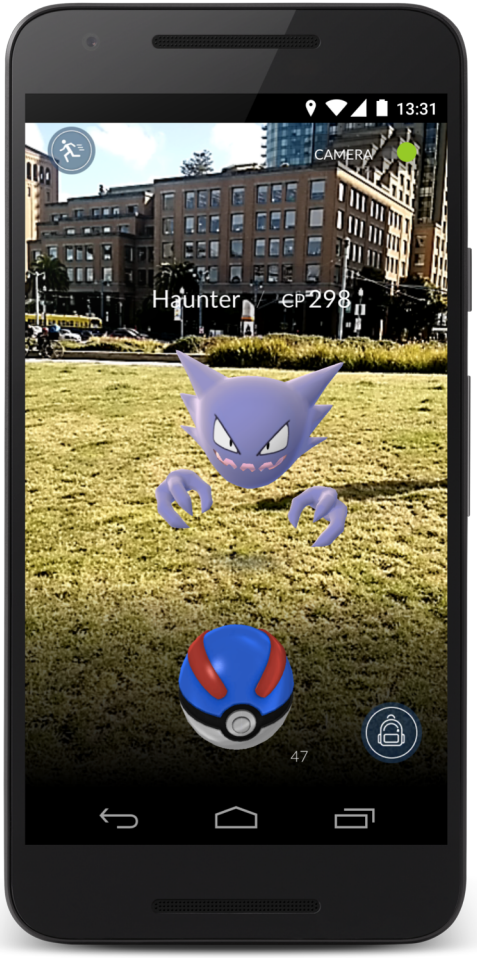
# M7 Client Code Quality

Risks that occur from code-level vulnerabilities executing on the mobile device. Common vulnerabilities include buffer overflows or format string issues.

# M8 Code Tampering

Malicious modification of a mobile applications codebase which allows attackers to abuse the applications functionality or even publish an entirely new malicious version of the application.

# M8 Code Tampering




- PokemonGO was cloned maliciously soon after release
- Found in Google Play store
- App ran in the background and generated fake add clicks

# M8 Code Tampering

# M9 Reverse Engineering

The analysis of a executable binary in its delivered state to determine the applications source code, proprietary algorithms, libraries, and more.

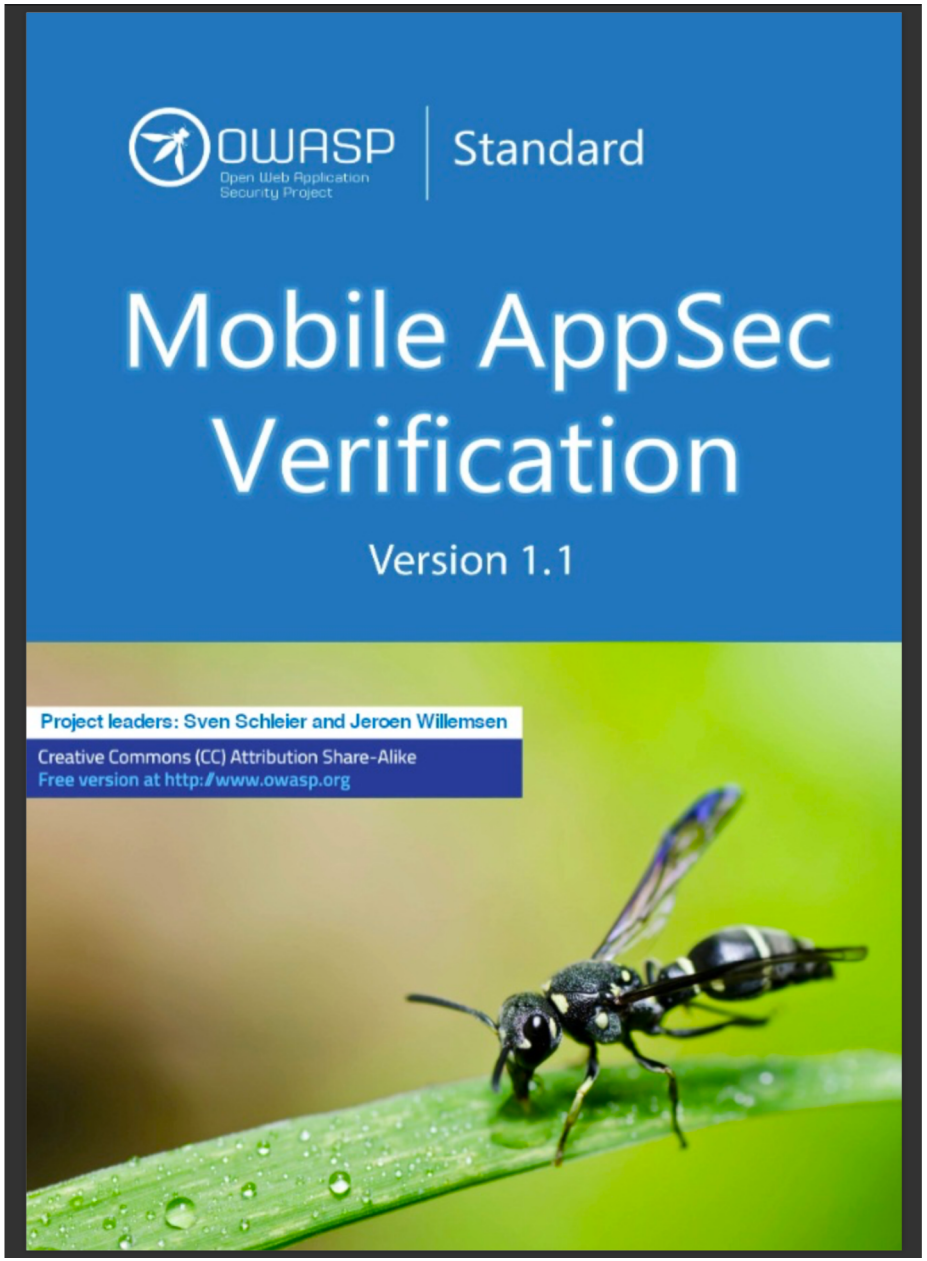
# M9 Reverse Engineering

- Popular dating app, Tinder, was reverse engineered to receive premium services for free.
- While the majority of everyday users do not have the skillset to do this, it is still a problem. Especially for expensive subscription applications.

Source: Forbes

# M10 Extraneous Functionality

Occurs when an attacker discovers features or security controls that were not intended to be released into a production environment.



# iOS Architecture

iOS

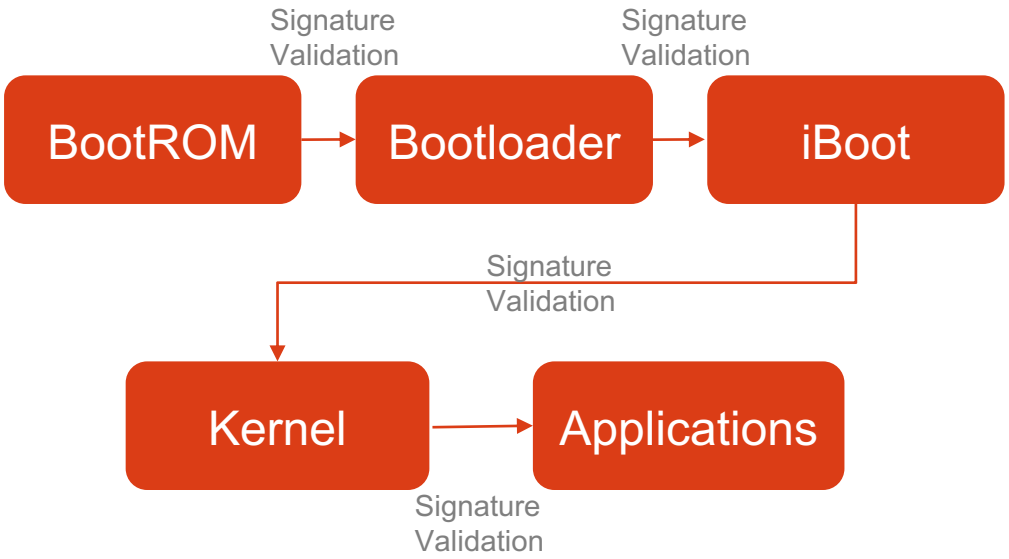
# Apple iOS

- Mobile OS built to run on a variety of Apple devices
- End-to-end ownership model for hardware and software
- Mobile Operator software is not permitted
- Closed-source and proprietary licensing
- Developers build applications using either Objective-C or Swift programming languages
- Lacks removable storage

iOS

# iOS Operating System Security

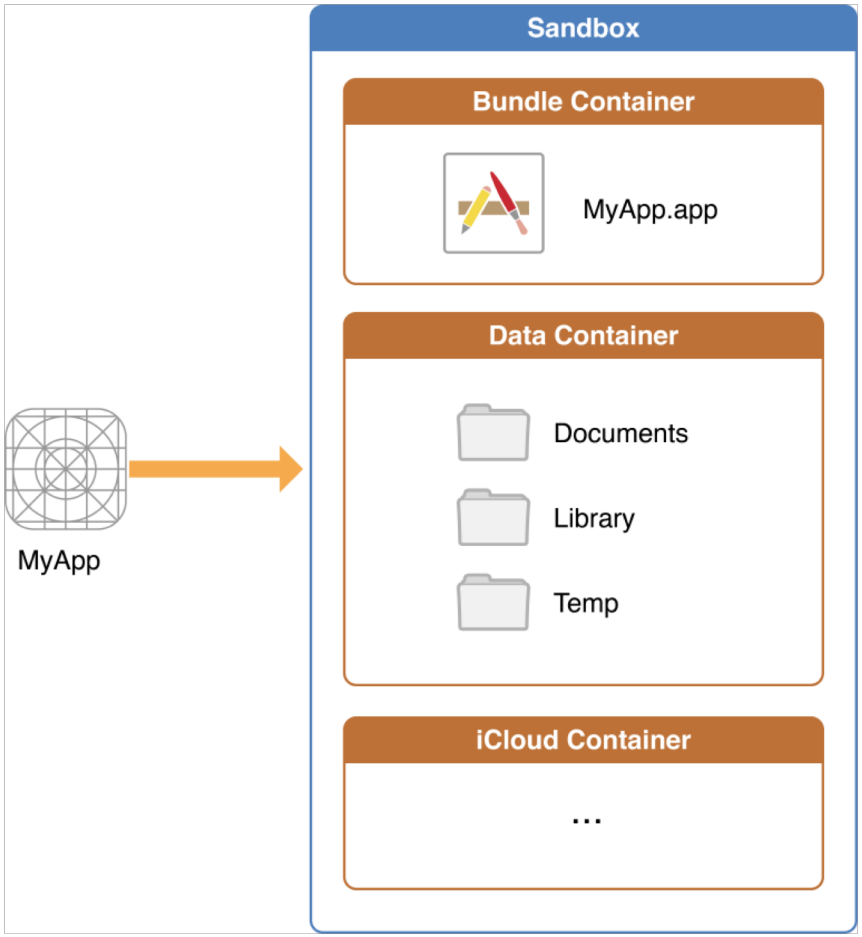
- Apple takes a number of precautions to protect against the misuse of software on their devices
- Use of code signing and signature validation from boot to application execution
- Jailbreaking takes advantage of a flaw in one of these steps



iOS

# Data Execution Protection & ASLR

- Data Execution Protection (DEP) where memory is either writable or executable, but never both
- Address Space Layout Randomization (ASLR) allows executables and libraries on iOS to be compiled with randomized memory addresses at startup
- Combination of the two can mitigate certain classes of vulnerabilities but does not mean absolute security




iOS sandboxing only allows each app to access its own files, preferences, and network resources through sanctioned iOS APIs

iOS

# iOS App Transport Security

- App Transport Security (ATS) now requires all apps use HTTPS with TLS 1.2 for network transport through NSURLSession and NSURLConnections

  Exceptions for transmission of bulk encrypted streaming media

| | | |
|---|---|---|
| App Transport Security Settings | Dictionary | (1 item) |
| Allow Arbitrary Loads | Boolean | YES |

Don't try this at home…

iOS

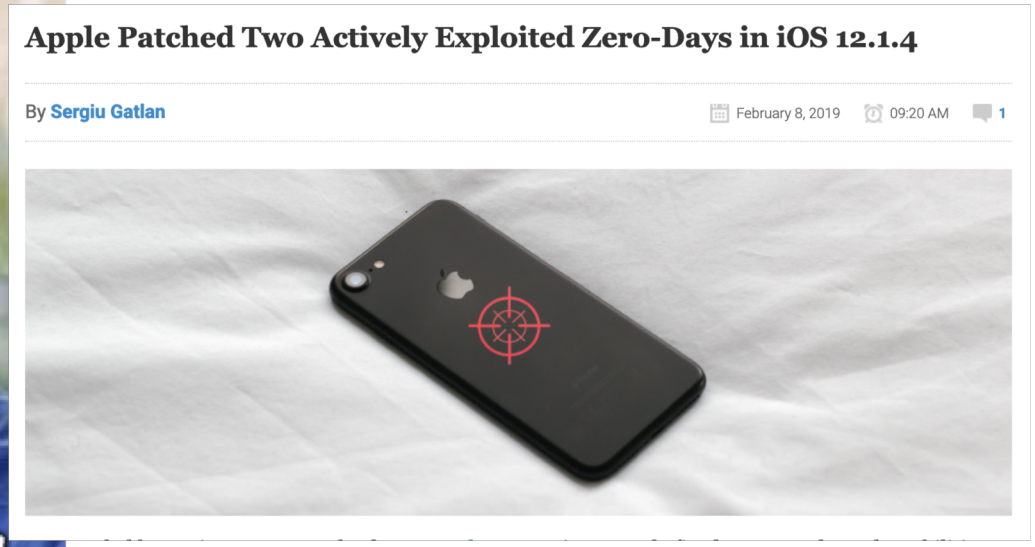
# iOS Application Signing and Distribution

- Developers build, sign, and submit applications for validation by Apple through Xcode using a Apple-issued developer certificate
- Apple reviews and verifies all submitted applications in-house and rejects those which do not follow Apple policies
- Once an application is approved, it is signed using an Apple private key and distributed to the App Store
- All applications must have valid signatures before installation
- We have some assurance that an app downloaded on a non-jailbroken device is mostly free of malware

# iOS

## iOS Security Issues – Still a Thing!

https://support.apple.com/en-al/HT209106

**Apple Patched Two Actively Exploited Zero-Days in iOS 12.1.4**

By Sergiu Gatlan February 8, 2019 09:20 AM 1

iOS

# iOS Permissions and Privileges

- Sandboxing prevents apps from interacting other than via permitted APIs
- iOS exposes minimal privilege notices to users

  Contacts

  Reminders

  Calendar

  Bluetooth

  Microphone

  ...
- Many other privileges are available to developers via APIs

iOS

# iOS Inter-Process Communication

IPC is what allows applications to communicate with each other

In order to protect users and the integrity of the platform, Apple has a limited number of ways to perform IPC

- URL Schemes
- Universal Links
- Pasteboards
- App Extensions
- ...

iOS

# iOS IPC (URL Schemes)

- URL Schemes allow one app to be opened by another app through the use of a custom, registered URL (manicode://search?course=toaster security)
- Defined by the developer in the info.plist file
- Apple does reserve some system schemes that can't be used by third-party apps

| | | |
|---|---|---|
| CFBundleURLTypes | Array | (1 item) |
| Item 0 | Dictionary | (3 items) |
| CFBundleTypeRole | String | Editor |
| CFBundleURLName | String | com.facebook |
| CFBundleURLSchemes | Array | (9 items) |
| Item 0 | String | fbauth2 |
| Item 1 | String | fbauth |
| Item 2 | String | fb |
| Item 3 | String | fblogin |
| Item 4 | String | fbapi20130214 |
| Item 5 | String | fbapi20130410 |
| Item 6 | String | fbapi20130702 |
| Item 7 | String | fbapi20131010 |
| Item 8 | String | fbapi20131219 |

iOS

# iOS IPC (URL Schemes)

- Apple does not enforce the unique naming meaning that two completely different apps may use an identical URL Scheme
- The security company, FireEye observed over 28 App Store apps all registering the URL scheme “fb://” of which 16 did not belong to Facebook
- A published malicious app that registers an identical URL scheme in hopes of intercepting a legitimate request to that app

iOS Masque Attack: Bypassing Apple’s Prompt
https://www.youtube.com/watch?v=Q1d70kCy6VQ

Guillaume K. Ross (gepeto42)

122 Reputation

- Rank

0

#28500 **iOS App can establish Facetime calls without user's permission**

Share:

| | | | |
|---|---|---|---|
| State | Resolved (Closed) | Severity | No Rating (---) |
| Disclosed publicly | April 27, 2015 6:03am -0700 | Participants | |
| Reported To | Twitter | Visibility | Public (Full) |
| Weakness | Cross-Site Request Forgery (CSRF) | | |
| Bounty | $420 | | |

Collapse

TIMELINE

gepeto42 submitted a report to **Twitter**. Sep 18th (4 years ago)

When URL Schemes for local applications are inserted in an inline frame, the web view launches them automatically.

**Example###:**

```
<html>
<header><title>Facetime Audio URL Scheme Test</title></header>
<body>
<iframe src="facetime-audio://guillaume@binaryfactory.ca"></iframe>
</body>
</html>
```

iOS

# iOS IPC (Universal Links)

- Universal links address the shortcoming introduced by URL Schemes
- Can not be claimed by other apps because they use standard HTTPS links to your own domain
- A file is verified on your web server to make sure you are the owner
- One URL for your website and app

Search the Manicode Archives!

*https://www.manicode.com/search?toaster security*

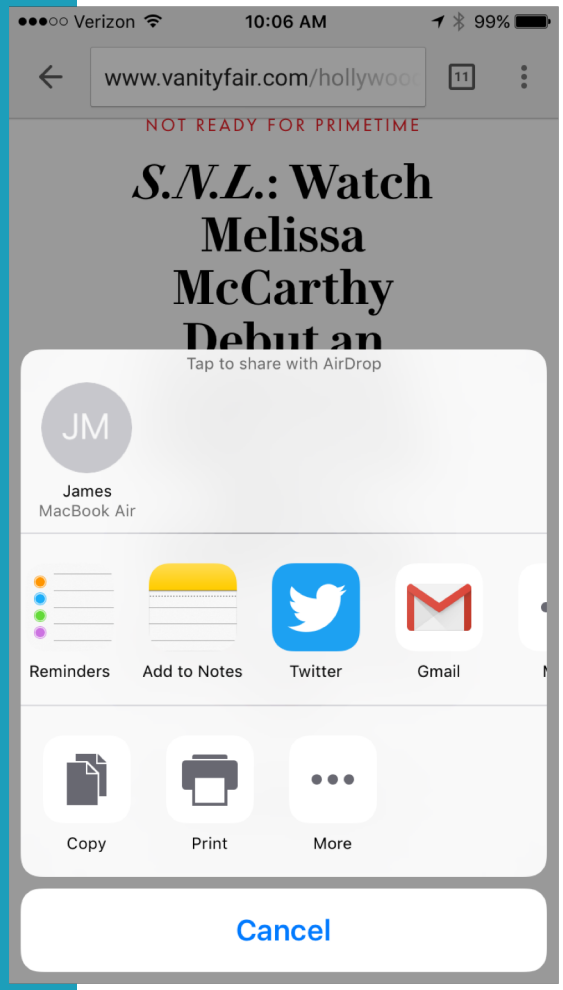
iOS

# iOS IPC (Pasteboards)

- The `UIPasteboard` can be considered a crude form of IPC
- An example would be transferring a user's data from a free version of an app to the paid version - not a great idea!
- The general pasteboard is shared among all applications installed on the device making it a particularly bad place to store private data
- iOS 10 introduced Pasteboard Handoff which allows pasteboard data to be shared between devices

iOS

# iOS IPC (App Extensions)




- Application Extensions allow developers to present data to other applications and share data through your app
- More secure alternative to custom URL Schemes
- Extensions are not invoked from within an app directly - the user must select the action from within the application

iOS

# iOS IPC (App Extensions)

**Share:** Allow data to be sent to your app via Share buttons

**Action:** Reads or manipulates data to be returned to the host app

**Photo:** Image editing and filter options

**Document Provider:** Send or receive document content between applications

**Today:** New widget in the Today view of the built in notification Center

**Keyboards:** Custom keyboard replacements for built-in iOS keyboards

iOS

# iOS IPC (App Extensions)

Share and Action extensions offer data filtering techniques using *NSExtensionActivationRule* in the apps Info.plist

- NSExtensionActivationSupportAttachmentsWithMaxCount
- NSExtensionActivationSupportsFileWithMaxCount
- NSExtensionActivationSupportsWebURLWithMaxCount
- NSExtensionActivationSupportsText
- …

# Attacking iOS Applications

iOS

## iOS Attack Methodology

- Very little “security testing” can be carried out on an iOS app these days without using a jailbroken application
- Source code review for internal teams is recommended as well as dynamic assessment

iOS

# iOS Attack Methodology - Recon

- Reconnaissance helps us understand the target better and start mapping our attack surface
- We want to look for things like:
  - Domain Names
  - Hardcoded Credentials
  - Administrative Backends
  - Server Configuration

iOS

## iOS Jailbreaking

- The sole purpose of jailbreaking iOS devices is to disable protections Apple puts in place
- A string of vulnerabilities and exploits working together to eventually give the operator root
- Tethered, semi-tethered, untethered, etc.
- Becoming very complicated

iOS

## iOS Static Analysis

- The preferred way to analyze an application from an attackers perspective
- Usually means having access to original Xcode project files
- No reliable decompilers on the market (unlike Android)
- Without source, we need to resort to reverse engineering via Assembly code

iOS

## iOS Dynamic Analysis

- With access to a jailbroken device, but not Xcode source, we can perform dynamic analysis of an application
- Dynamic analysis consists of instrumenting the runtime in order to inject your own code into the application

FRIDA

needleAgent

MOBSF

iOS

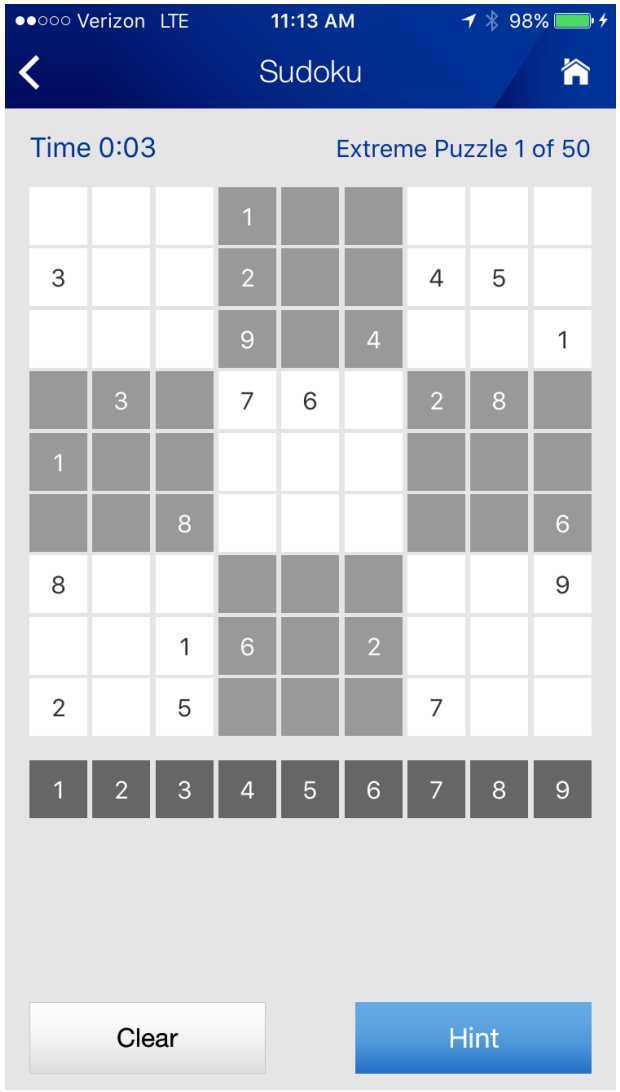
# iOS Dynamic Analysis (Non-Rooted)

- We can still modify the runtime of an app on a non-jailbroken device
- Objection handles runtime exploration and repackaging

# iOS – Insecure Data Storage

iOS

# Insecure Data Storage SQLite



iOS

# Insecure Data Storage SQLite

| App - com.united.UnitedCustomerFacingIPh... | | | |
|---|---|---|---|
| Library | | | |
| Documents | | | |
| ensighten.plist | PLIST | 125 B | Dec 11, 2016, 12:09 PM |
| ensightenQueue.plist | PLIST | 1 kB | Dec 11, 2016, 12:09 PM |
| ensightenQ.plist | PLIST | 406 B | Dec 11, 2016, 12:09 PM |
| UnitediPhoneCoreData.sqlite | SQLITE | 647 kB | Dec 11, 2016, 12:09 PM |

# iOS

# Insecure Data Storage SQLite

Table: ZUACDSUDOKUPUZZLE

New Record

| | Z_PK | Z_ENT | Z_OPT | ZDIFFICULTY | ZNUM | ZPUZZLE | ZSOLUTION |
|---|---|---|---|---|---|---|---|
| | Filter | Filter | Filter | Filter | Filter | Filter | Filter |
| 30 | 130 | 23 | 1 | 0 | 37 | 0143000000000000006009801000900060004020090050000000... | 7143628599837512466259841373985264714621793851574386922396457188712 9... |
| 31 | 131 | 23 | 1 | 0 | 38 | 5300002790720000000090040000001805000500000000000690... | 5348162798723594616192748359261875437514329864836951273689417522975 6... |
| 32 | 132 | 23 | 1 | 0 | 39 | 0020809000106028000080004012007000000000001040500000... | 4721839569156428733685794212467351897832916455914682376593247188349 1... |
| 33 | 133 | 23 | 1 | 0 | 40 | 0040906000000000800060850204006108770000030095107 0... | 8241956736357241981796385423425619877682493159518734264863172595934 8... |
| 34 | 134 | 23 | 1 | 0 | 3 | 2580040000090560303000004000700000600000081004860 23... | 2583147967498562313619724851725493689357681424861235795942876136134 9... |
| 35 | 135 | 23 | 1 | 0 | 41 | 5180679000000008064001000000000007( | |
| 36 | 136 | 23 | 1 | 0 | 42 | 0080601002009100000700800296005203( | |
| 37 | 137 | 23 | 1 | 0 | 43 | 0040000830000049007069030056007000 | |
| 38 | 139 | 23 | 1 | 0 | 44 | 0472000900030002000000000000000000 | |
| 39 | 141 | 23 | 1 | 0 | 45 | 0020300006800010090372000800950030( | |
| 40 | 142 | 23 | 1 | 0 | 1 | 0000500080000075604200000000000001( | |
| 41 | 144 | 23 | 1 | 0 | 46 | 1060009800000000000000320700000456( | |
| 42 | 146 | 23 | 1 | 0 | 47 | 0080530000000040560058600010001420( | |
| 43 | 148 | 23 | 1 | 0 | 48 | 0000970000050012704900300002705000( | |
| 44 | 150 | 23 | 1 | 0 | 49 | 0020080077089040500060000000008693 | |
| 45 | 152 | 23 | 1 | 0 | 50 | 0530028000000000040600107900800240( | |

|< < 30 - 45 of 200 > >|

Edit database cell

Import Export Text

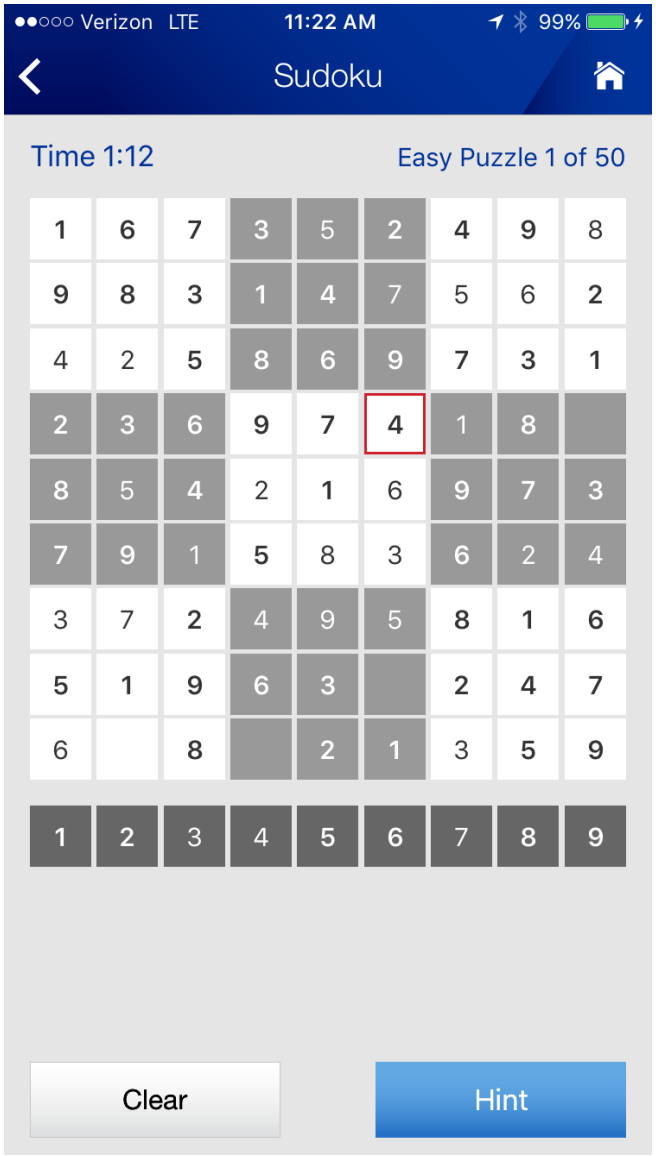
167352498
983147562
425869731
236974185
854216973
791583624
372495816
519638247
648721359

Type of data currently in cell: Text / Numeric
89 char(s)

Cancel

iOS

# Insecure Data Storage SQLite



iOS


# Insecure Data Storage Snapshots

- iOS creates snapshots of the application running as it is exited to provide more transparent transitions
- Consider the need to display sensitive data on a mobile app at all
- A few ways to handle sensitive snapshots:

UIApplicationExitsOnSuspend – Kill app entirely and not save snapshot

Change the view at the moment the app is placed into the background and ignore the fake image when app is brought back to foreground

iOS

# Insecure Data Storage Pasteboard

- iOS 10 introduced the Universal Clipboard which opens up the attack surface to other devices
- New features also introduced to limit pasteboard functionality

# iOS Pasteboard Options

## Pasteboard Item Flagged “Local Only”

```
// set pasteboard values
let aLocalOnlyStringKey = "Local key"
let aLocalOnlyStringValue = "Local value"

// Set the string in the local pasteboard
pasteboard.setItems([[aLocalOnlyStringKey: aLocalOnlyStringValue]], options: [UIPasteboardOption.localOnly : true])
```

## Pasteboard with Expiration Date

```
// date 24 hours from now
let expirationDateOfTomorrow = Date().addingTimeInterval(60*60*24)

// Add the string and mark it to expire 24 hours from now
pasteboard.setItems([[aExpiringStringKey: aExpiringStringValue]], options: [UIPasteboardOption.expirationDate:
    expirationDateOfTomorrow])
```

iOS

# Malware Custom Keyboard

**Attack:** Malicious keyboard extensions have the ability to read every keystroke that a user enters into your app

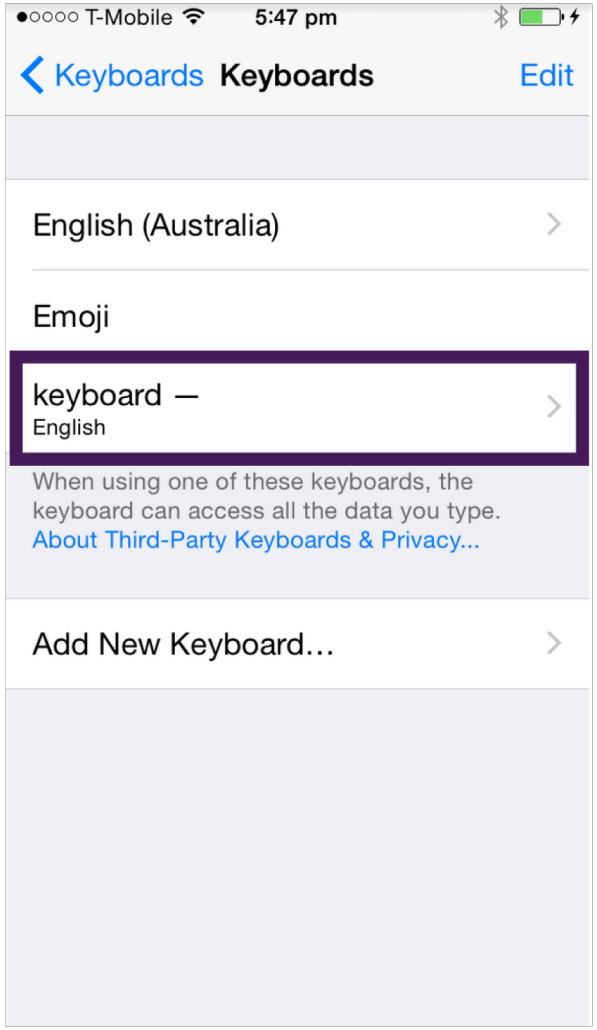
- Different levels of data exfiltration or leakage could exist (Privacy vs. Exploit)

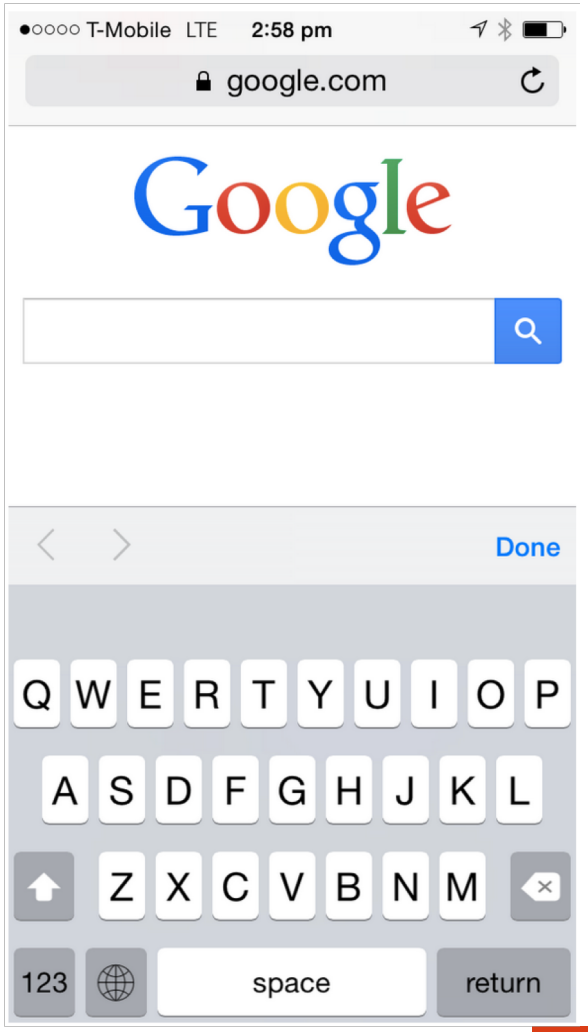
**Defense:** Consider preventing the use of third-party keyboards in your application if it collects sensitive information.

```
func application(application: UIApplication, shouldAllowExtensionPointIdentifier extensionPointIdentifier: String) -> Bool {
    if extensionPointIdentifier == UIApplicationKeyboardExtensionPointIdentifier {
        return false
    }
    return true
}
```

# iOS

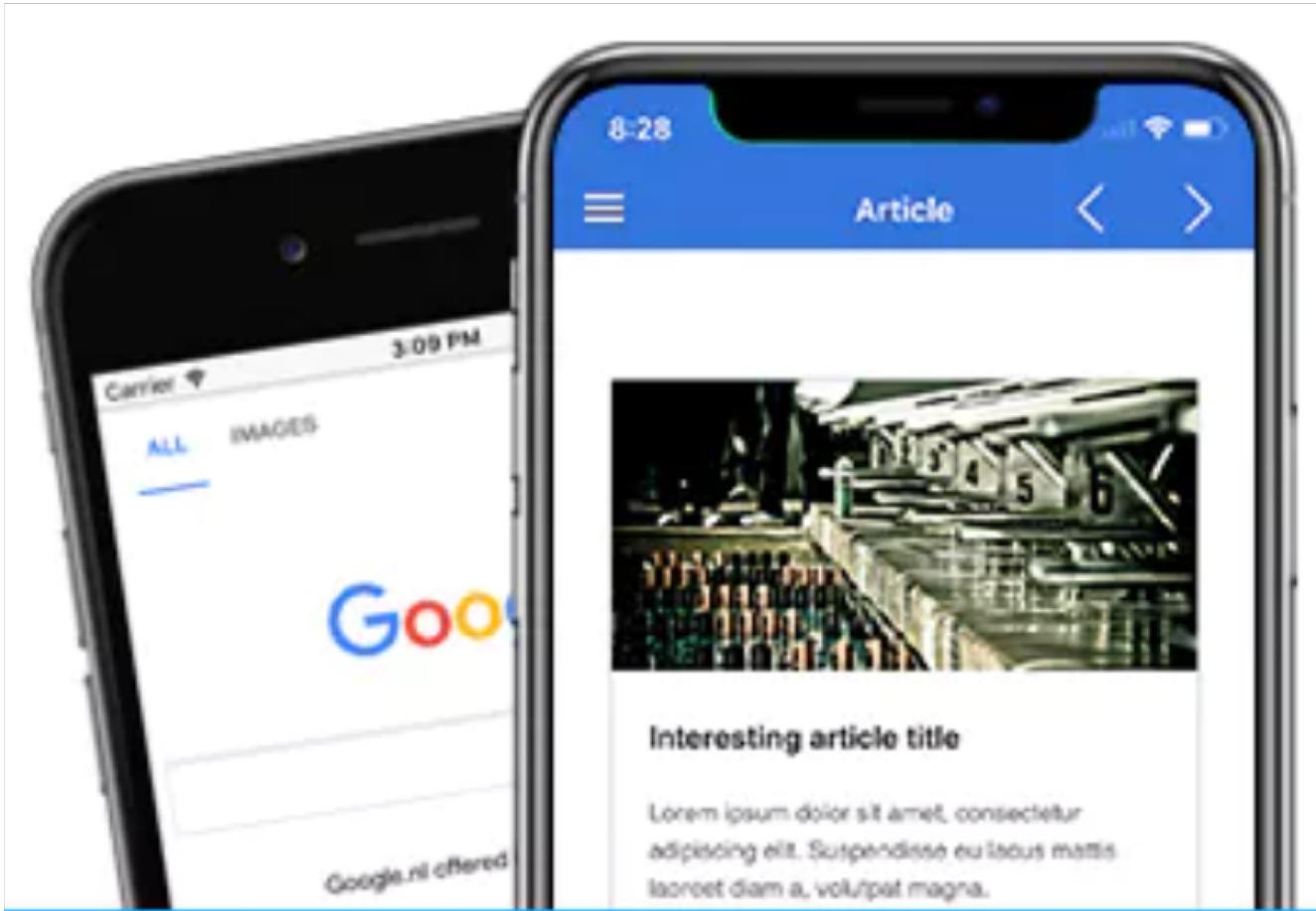
## Malware Custom Keyboard



# iOS

## iOS WebViews



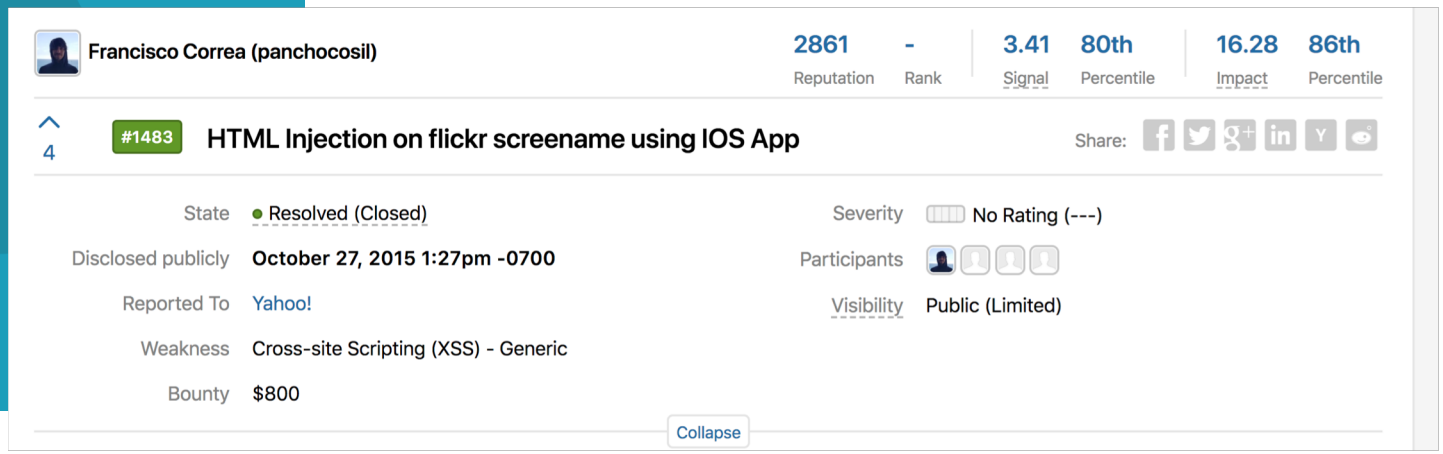
iOS

# iOS WebViews

**Attack:** XSS or other injection may be present in an iOS WebView and may lead to file or system access.

**Defense:** Ensure all WebView calls do not execute with out proper input validation.

If possible, rely on Safari or Chrome to handle WebView functionality. Turn of JavaScript if possible.



iOS

# iOS Certificate Pinning

- Aims to protect against rogue CAs, compromised CAs, and prying eyes
- Will not protect against certain reverse engineering techniques which can unpin, debug, and repackage
- Will not help if device is jailbroken / rooted
- Makes my job as a pen tester that much more difficult
- Defense in depth mechanism

# ANDROID SECURITY

The good, bad, and the ugly.

android

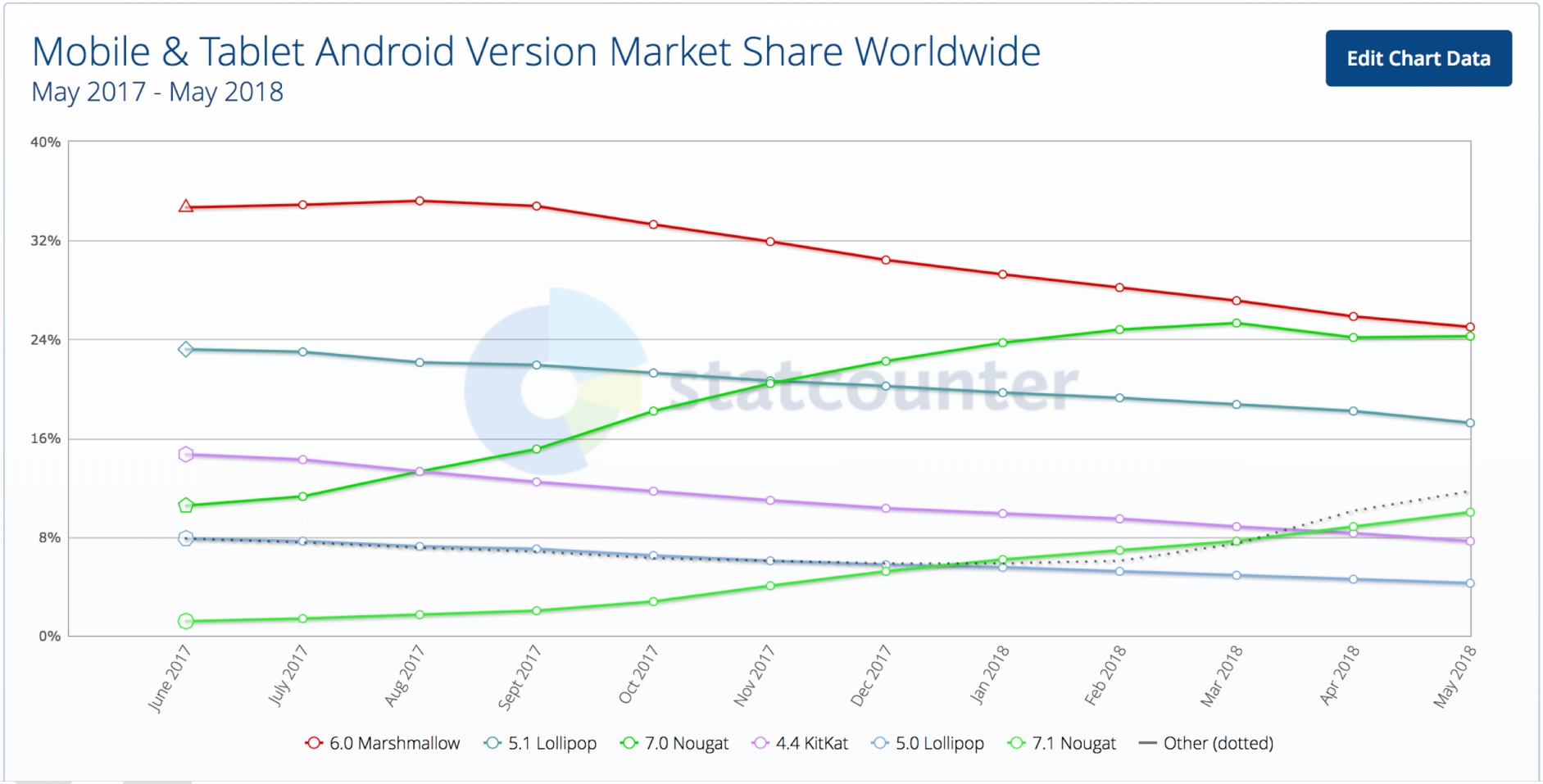
# Android

- Massively popular alternative to iOS
- Governed by the Open Handset Alliance, led by Google
- Unlike Apple, no end-to-end hardware
- Large disparity of software and hardware support

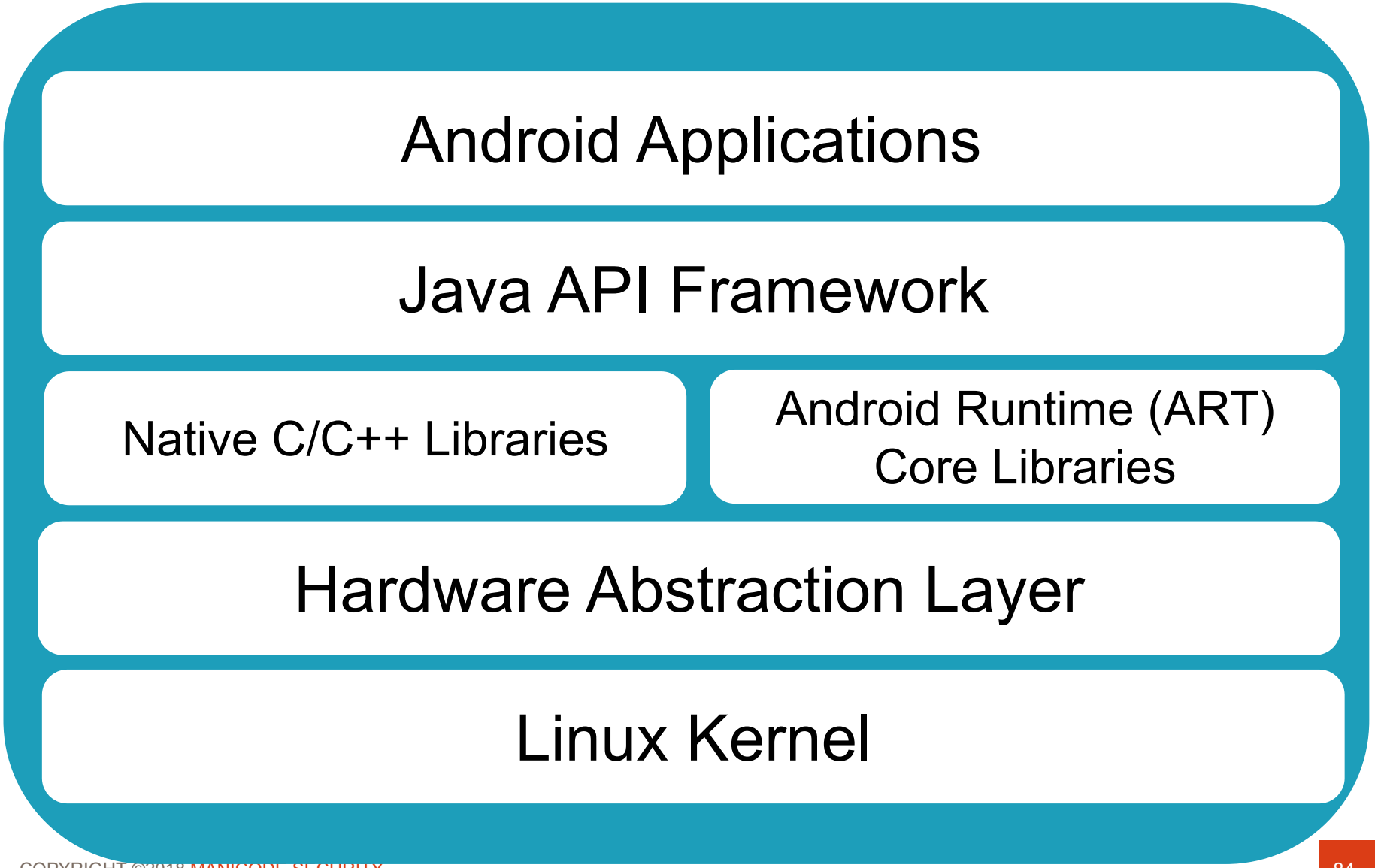
  Makes securing Android devices very challenging

| 6.0 Marshmallow | 7.0 Nougat | 5.1 Lollipop | 7.1 Nougat | 4.4 KitKat | 8.0 Oreo |
|---|---|---|---|---|---|
| 24.97% | 24.21% | 17.26% | 10.02% | 7.66% | 5.85% |

**Mobile & Tablet Android Version Market Share Worldwide - May 2018**



# Android Stack



# Android Operating System Security

- Apps written in Java (as a language not a runtime) or Kotlin and compiled to Dalvik or ART

  Sandboxing enforces a unique UID and GID at install with limited permissions

- SELinux enforced in Android 5.0 and later
- DEP and ASLR support with Android 4.0 +
- Memory protection, bounds checking string management, etc.

# **Android** Application Signing and Distribution

- Developers sign their own applications self-signed certificates

  Used to ensure that subsequent installs cannot overwrite prior applications

- This does not validate the identity of the developer
- Applications are not vetted in the same fashion as Apple apps prior to publication

# **Android** Application Signing and Distribution

- 12+ “rip-off” apps booted from the Google Play store
- QR Barcode Scanner, Compass, Flashlight, etc.
- Downloaded between 10,000 – 50,000 times
- “One of those secrets included the creation of a dex file that when executed plays a specific YouTube video and generates ad revenue for the video’s author. ”
- “It’s notable that this dex file is not embedded in the original app, but is downloaded at runtime.”

# Apps Installed On Millions Of Android Phones Tracked User Behavior To Execute A Multimillion-Dollar Ad Fraud Scheme

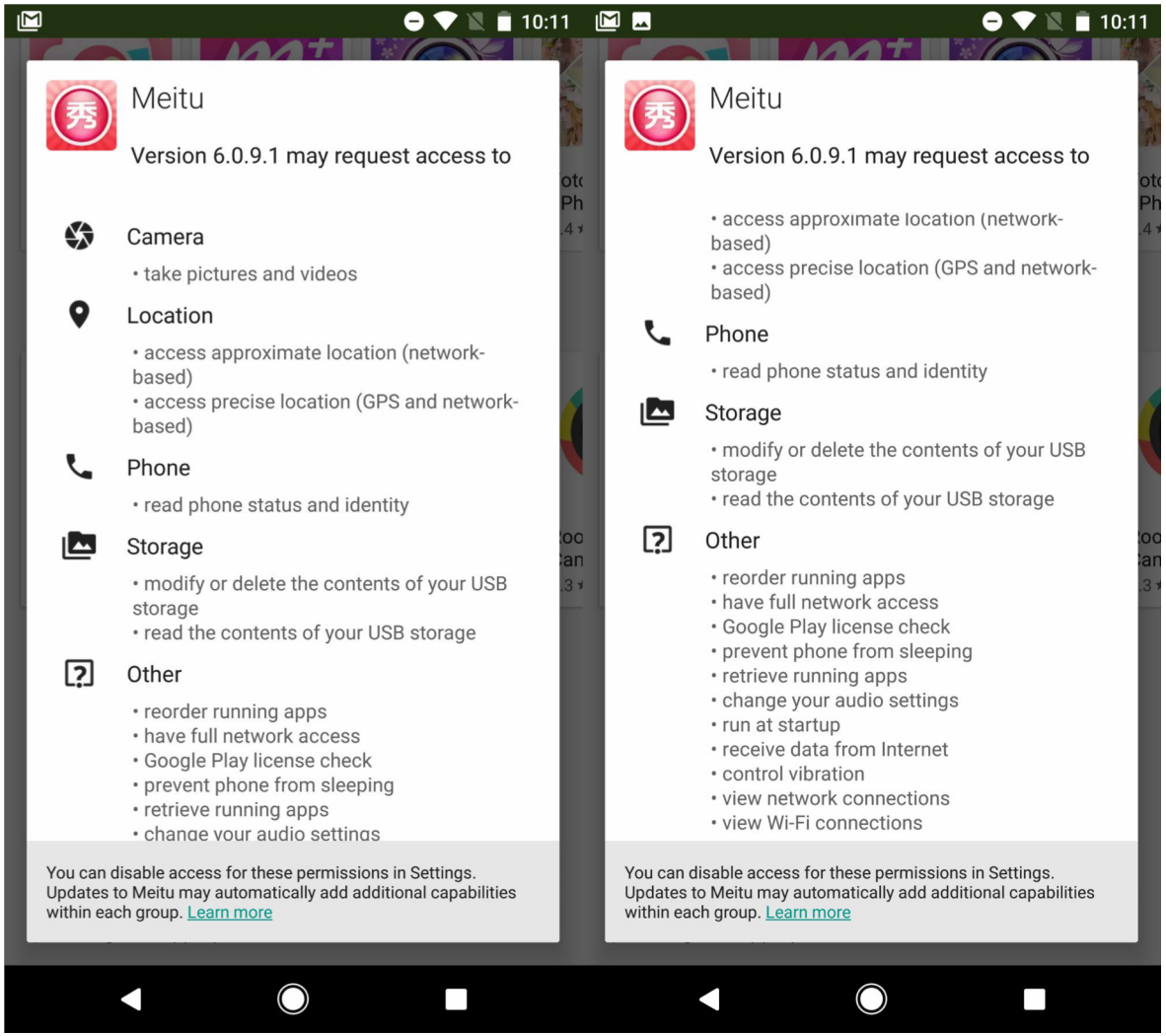
A BuzzFeed News investigation uncovered a sophisticated ad fraud scheme involving more than 125 Android apps and websites, some of which were targeted at kids.

android

# Android Permissions and Privileges

- Apps use Android APIs for system interaction and data access (similar to iOS)
- Defined by developer in the AndroidManifest.xml
- User is prompted upon installation
- In Android 6.0 Marshmallow, application will not be granted any permission at installation time. Instead, application has to ask user for a permission one-by-one at runtime.
- Can be very unclear why an app would need a certain permission

# android



android

# Android Inter-Process Communication

- Android uses Intents to perform IPC as well as communication within the same application
- Unlike iOS Extensions, Intents are invoked from within the app directly
  - Activities
  - Services
  - Content Providers
  - Broadcast Receivers

# Anatomy of an Android Application

android

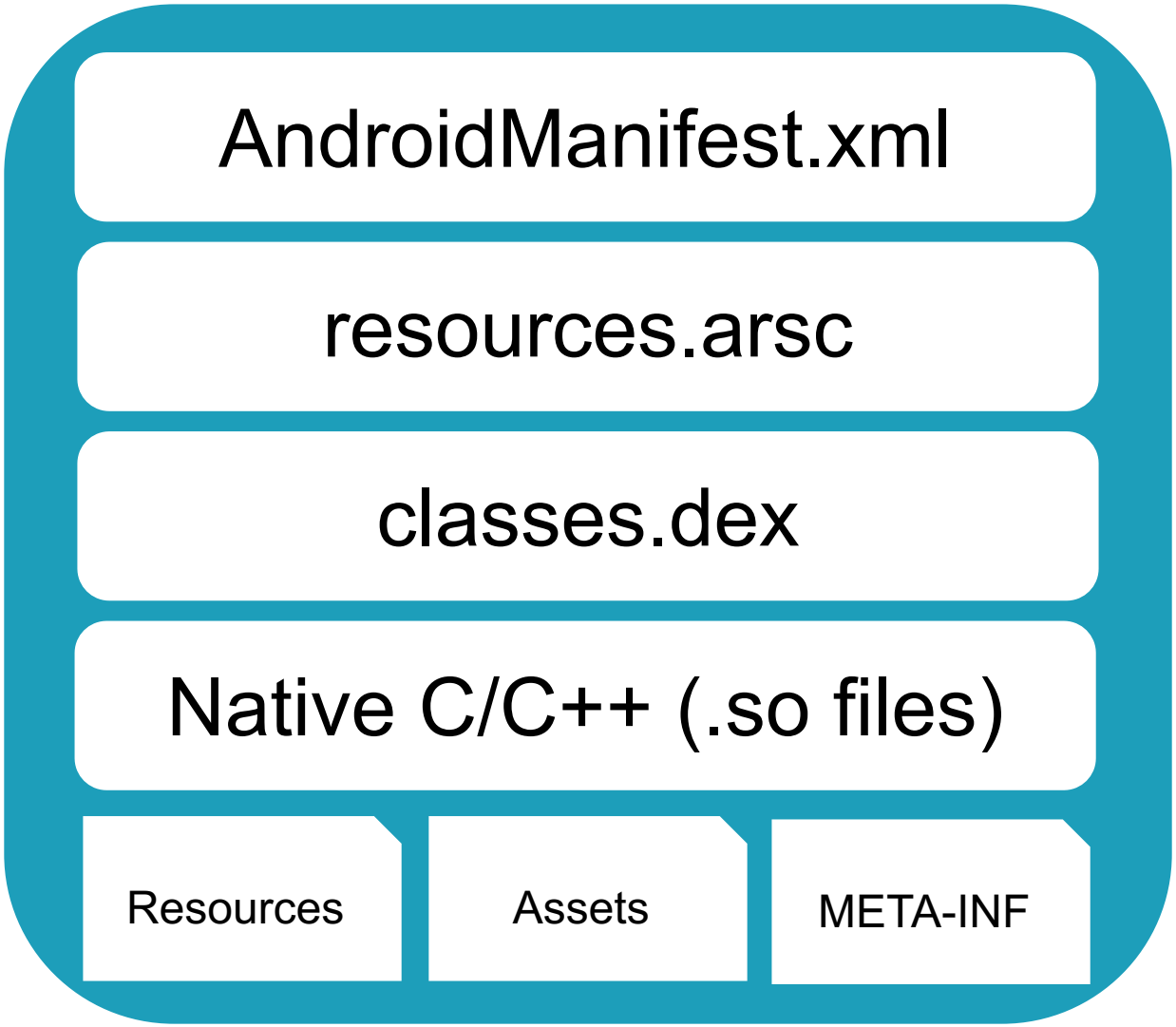
# Android Package

- Just a .zip file labeled as .apk
- The artifact that is uploaded to the Play Store or sideloaded onto a device
- Can be extracted from a device
- Always signed by the developer

*"On Google Play, application signing bridges the trust Google has with the developer and the trust the developer has with their application. Developers know their application is provided, unmodified, to the Android device; and developers can be held accountable for behavior of their application."*

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

# APK Contents



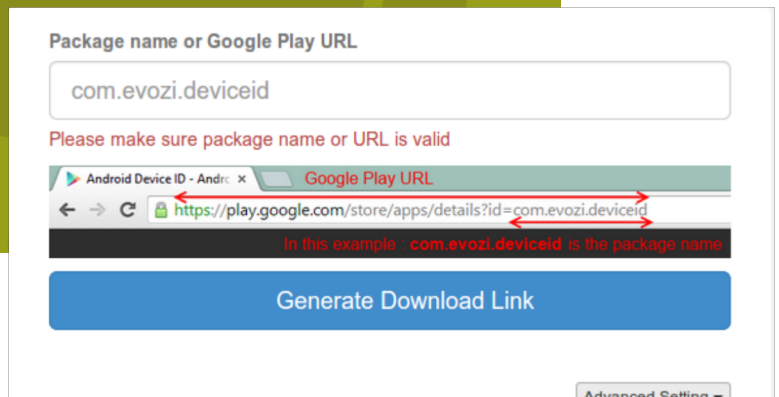
android

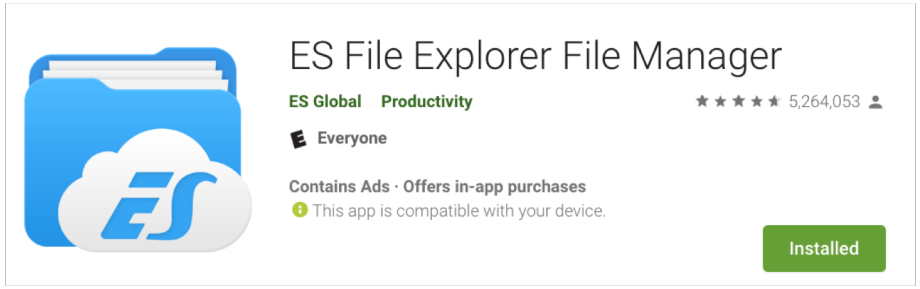
# Retrieving the APK

- googleplay-api
- Android Debug Bridge (adb)

  adb shell pm list packages

  adb pull /data/app/yourapp.apk
- APKoptik
- Sketchy Browser Extensions



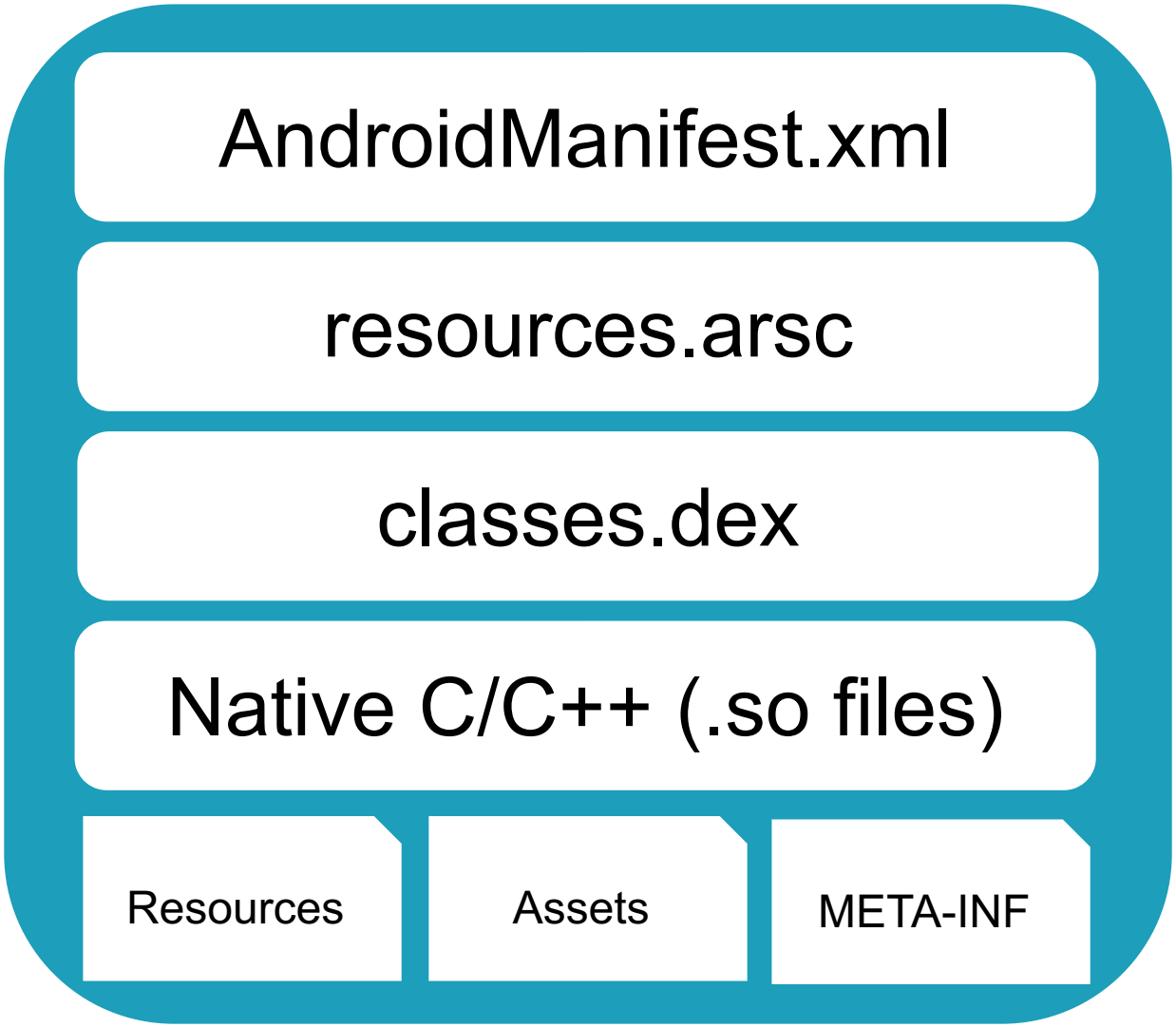
android

## Extracting the APK

- Lots of tooling out there to help us with this
- *apktool*
- *apkanalyzer*
- *aapt*


*…it's just a .zip file*

# APK Contents



android

# AndroidManifest.xml

- Where we define essential information about our application
- App Name and Unique Identifier
- Describes components of the application (activities, services, broadcast receivers, etc.)
- Declares apps permissions
- Minimum Android API version the app requires
- A curious hackers first point of interest

android

# Android Manifest Best Practices
## *Debug Mode*

- android:debuggable defines whether the app can be debugged or not
- What happens when this is set to “true” in production?

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

# Android Manifest Best Practices
## *External Storage*

- Apps may request permission to write data to external storage mechanisms
- Ensure no sensitive data is being written

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

android

# Android Manifest Best Practices
## *Backups*

- Defines whether an app can automatically back itself up
- Requires debugging to be enabled on device

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

android

# Android Manifest Best Practices
## *Permissions*

- Apps must request permission to use sensitive features of the API
- Request only what your app needs to run and nothing more
- Be explicit with permission requests

```
<manifest>
    <uses-permission android:name="android.permission.SEND_SMS"/>
</manifest>
```

android

# Android Manifest Best Practices
## *Protection Level*

- Three protection levels exist:
  - *normal*
  - *dangerous*
  - *signature*
- Apply the principal of least privilege and **only request permissions that your app needs to function**.

```
<permission>
android:protectionLevel="dangerous"
</permission>
```

ACCESS_LOCATION_EXTRA_COMMANDS

ACCESS_NETWORK_STATE

ACCESS_NOTIFICATION_POLICY

ACCESS_WIFI_STATE

BLUETOOTH

BLUETOOTH_ADMIN

BROADCAST_STICKY

CHANGE_NETWORK_STATE

CHANGE_WIFI_MULTICAST_STATE

CHANGE_WIFI_STATE

DISABLE_KEYGUARD

EXPAND_STATUS_BAR

GET_PACKAGE_SIZE

INSTALL_SHORTCUT

INTERNET

KILL_BACKGROUND_PROCESSES

MANAGE_OWN_CALLS

MODIFY_AUDIO_SETTINGS

NFC

READ_SYNC_SETTINGS

READ_SYNC_STATS

RECEIVE_BOOT_COMPLETED

REORDER_TASKS

REQUEST_COMPANION_RUN_IN_BACKGROUND

REQUEST_COMPANION_USE_DATA_IN_BACKGROUND

REQUEST_DELETE_PACKAGES

REQUEST_IGNORE_BATTERY_OPTIMIZATIONS

SET_ALARM

SET_WALLPAPER

SET_WALLPAPER_HINTS

TRANSMIT_IR

USE_FINGERPRINT

VIBRATE

WAKE_LOCK

WRITE_SYNC_SETTINGS

**”Normal” Permissions do not require manual user approval and are granted to apps by default and cannot be revoked.**

| Permission Group | Permissions |
| --- | --- |
| CALENDAR | • READ_CALENDAR<br>• WRITE_CALENDAR |
| CAMERA | • CAMERA |
| CONTACTS | • READ_CONTACTS<br>• WRITE_CONTACTS<br>• GET_ACCOUNTS |
| LOCATION | • ACCESS_FINE_LOCATION<br>• ACCESS_COARSE_LOCATION |
| MICROPHONE | • RECORD_AUDIO |
| PHONE | • READ_PHONE_STATE<br>• READ_PHONE_NUMBERS<br>• CALL_PHONE<br>• ANSWER_PHONE_CALLS<br>• READ_CALL_LOG<br>• WRITE_CALL_LOG<br>• ADD_VOICEMAIL<br>• USE_SIP<br>• PROCESS_OUTGOING_CALLS |
| SENSORS | • BODY_SENSORS |
| SMS | • SEND_SMS<br>• RECEIVE_SMS<br>• READ_SMS<br>• RECEIVE_WAP_PUSH<br>• RECEIVE_MMS |
| STORAGE | • READ_EXTERNAL_STORAGE<br>• WRITE_EXTERNAL_STORAGE |

# ”Dangerous” Permissions are granted in groups and require end user approval.

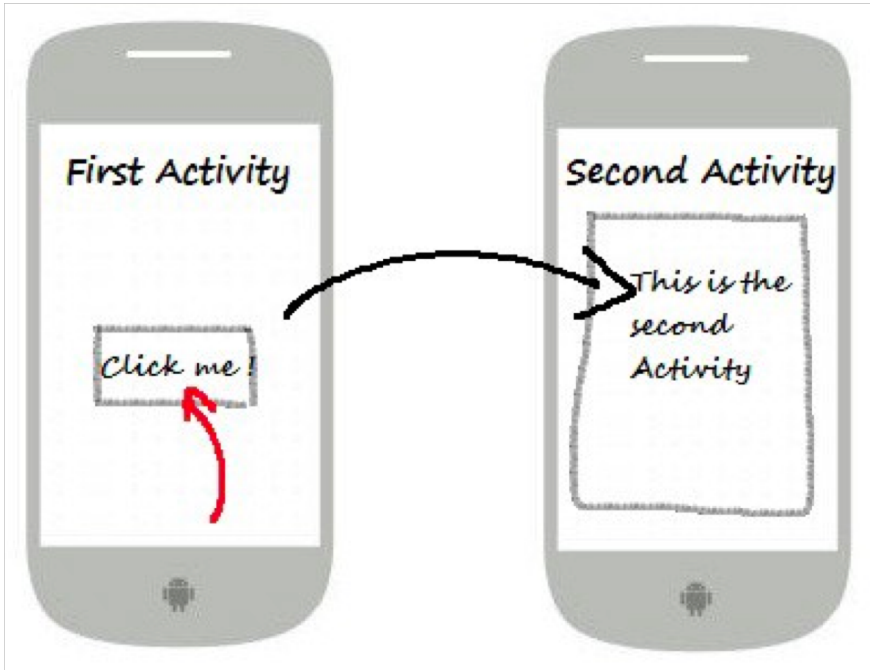
android

# Android Manifest Best Practices
## *Components*

- Each Android app consists of a collection of components that work both together, and potentially with components of other apps, in order to provide the application's functionality.
- Sounds *great* for usability but what about security?

# Android Components: Activities

Components that provide a user interface screen and correspond to activities that the user might perform



## Android Components: Services

Components that perform potentially long-running tasks that operate in the background without a user interface, e.g. downloading a file, playing music, or synchronizing email with a server.

## **Android Components:** Public or Private

A **public** component can be accessed by components in other apps, e.g. a public Service or Activity can be started by another app

android:exported=true

## Android Components: Public or Private

A **private** component can only be accessed by other components within the same app

android:exported=false

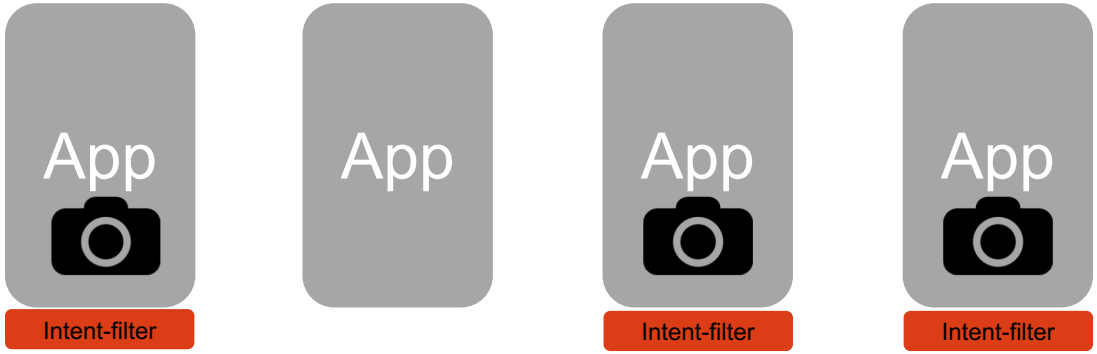
android

# Android Manifest Best Practices
## *Component Permissions*

- Always explicitly set the **android:exported** value in the AndroidManifest.xml config to have the visibility needed and nothing more.
- The default value is version-dependent and may change in the future.

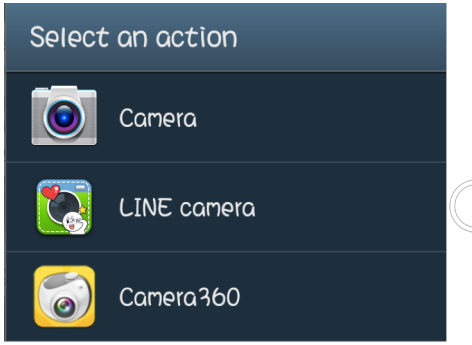
# Android Intent Overview

**Implicit Intents** inform the Android OS that it will need an app that is able to handle the intent's action when it starts.



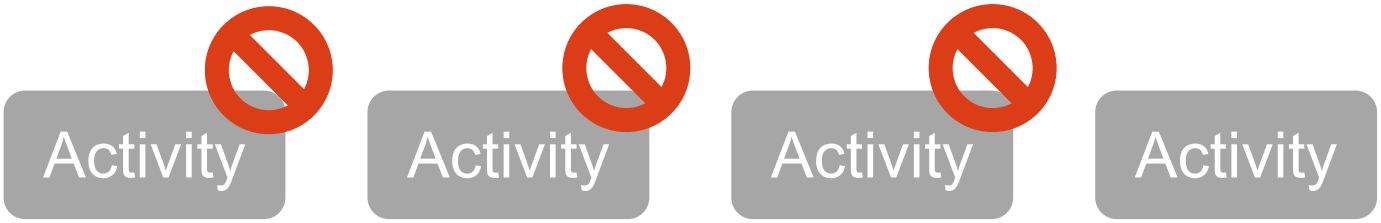

```
Intent captureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

Intent



# Android Intent Overview

**Explicit Intents** specify which component to start by fully-qualified class name



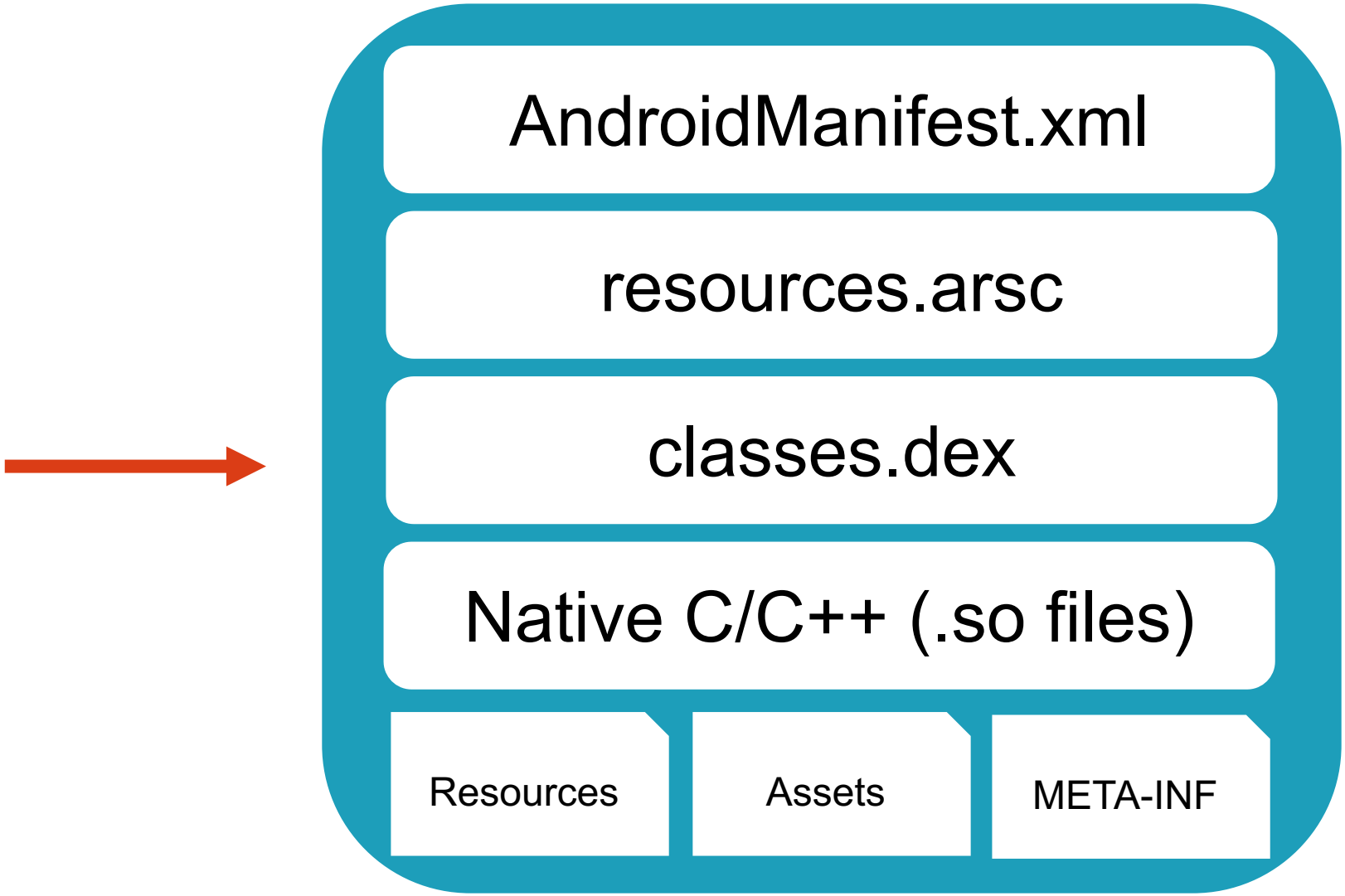

```
Intent myIntent = new Intent(myContext, com.example.testapps.test1.mainActivity.class);
```

android

# Android Manifest Best Practices
## *Intents*

- Use explicit intents when possible
- If implicit intents *must* be used, require appropriate permissions of the caller, validate the origin, action, and data of the incoming intent
- If receivers must be exposed publicly, require appropriate permissions and security checks

# APK Contents



android

# Dalvik Executable

- Register-based Bytecode
- Executed by Dalvik / ART Runtime
- Compiled to native code that runs on the Android device

DEX Byte Code for boolean access$000(sg.vp.owasp_mobile.OMTG_Android.OMTG_CODING_003_Best_Practice,java.lang.String,java.lang.Strin...

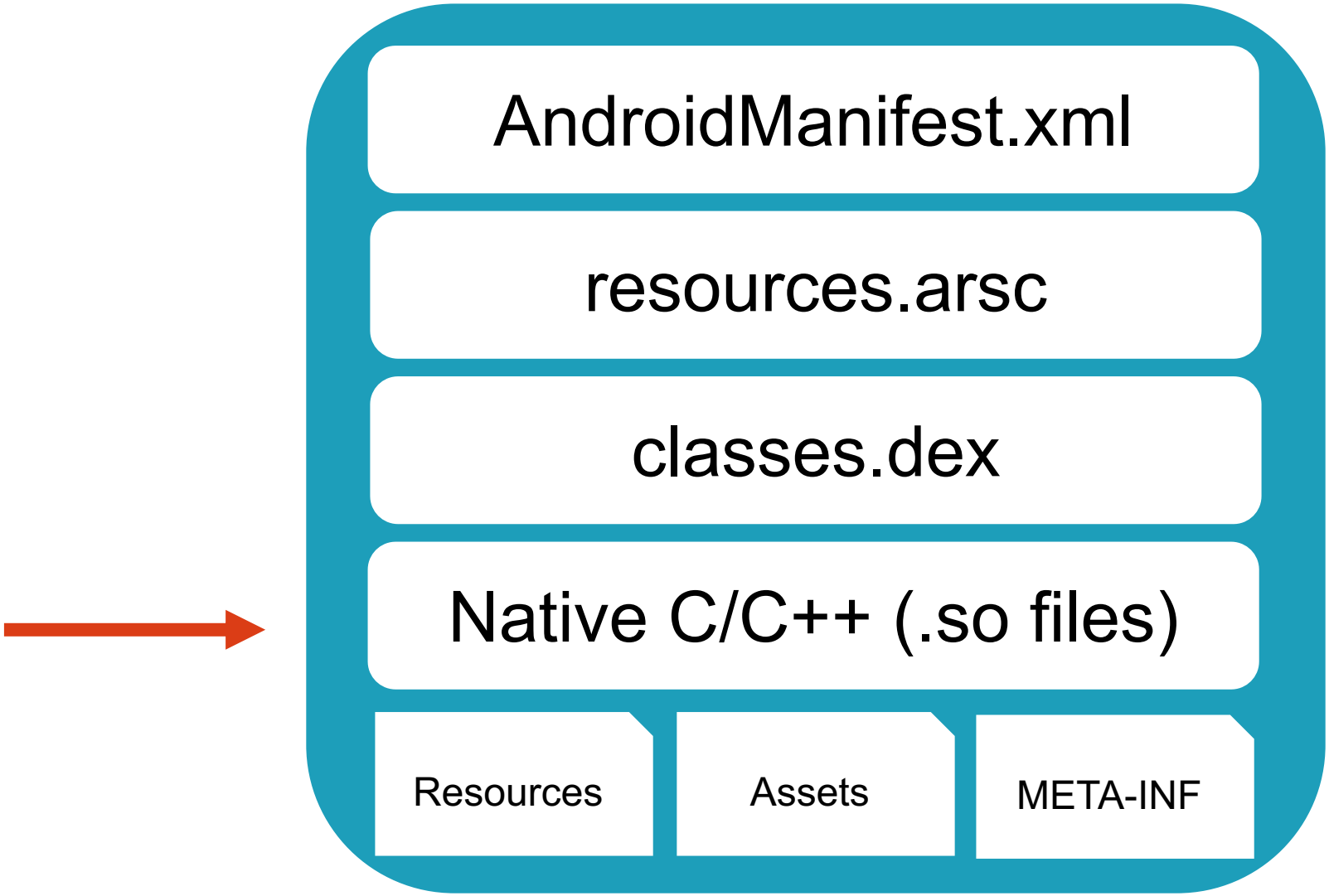
```
.method static synthetic access$000(Lsg/vp/owasp_mobile/OMTG_Android/OMTG_CODING_003_Best_Practice;Ljava/lang/String;Ljava/lang/S
    .registers 4
    .param p0, "x0"    # Lsg/vp/owasp_mobile/OMTG_Android/OMTG_CODING_003_Best_Practice;
    .param p1, "x1"    # Ljava/lang/String;
    .param p2, "x2"    # Ljava/lang/String;

    .prologue
    .line 16
    invoke-direct {p0, p1, p2}, Lsg/vp/owasp_mobile/OMTG_Android/OMTG_CODING_003_Best_Practice;->checkLogin(Ljava/lang/String;Lja

    move-result v0

    return v0
.end method
```

# APK Contents



android

## Native Code

- "Shared Object" files are compiles libraries from C or C++ source code
- Can be disassembled but you are entering a world of complexity
- Still not a safe place for hardcoded credentials or sensitive data

# Android Attack Surface

# Dex to Smali

- Smali can be thought of as “intermediate bytecode”
- Easier to read by humans than Dex
- Can still be modified and recompiled into an APK

```
.class public Lfah;
.super Lesb;
.source "SourceFile"


# instance fields
.field private a:Ljava/math/BigInteger;

.field private b:I


# direct methods
.method public constructor <init>(Ljava/math/BigInteger;Ljava/security/SecureRandom;II)V
    .locals 2

    .prologue
    .line 20
    invoke-direct {p0, p2, p3}, Lesb;-><init>(Ljava/security/SecureRandom;I)V
```

# Dex to Smali

- APKTool is popular **disassembler** for DEX
- https://ibotpeaches.github.io/Apktool
- Transforms Dalvik Executable files (DEX) to Smali Bytecode
- Able to modify Smali and rebuild back to running app

# dex2jar

- dex2jar is popular **decompiler** for DEX files
- https://github.com/pxb1988/dex2jar
- Converts Dalvik Bytecode (DEX) to Java source code (jar)
- One way operation and cannot be re-compiled to DEX

```
➜ dex2jar-2.0 sudo sh d2j-dex2jar.sh -f ~/Desktop/APKs/keeper_password_manager-11.2.3-313.apk
dex2jar /Users/jb0ss/Desktop/APKs/keeper_password_manager-11.2.3-313.apk -> ./keeper_password_manager-11.2.3-313-dex2jar.jar
Detail Error Information in File ./keeper_password_manager-11.2.3-313-error.zip
Please report this file to http://code.google.com/p/dex2jar/issues/entry if possible.
```

android

android

# Secrets Storage Woes!

Option 1: Include Secrets in strings.xml

Option 2: Include Secrets in Source Code

Option 3: Include Secrets in Source Code and do not Check in to git

Option 4: Include Secrets in Build Config

Option 5: Obfuscate with Proguard / DexGuard

Option 6: Obfuscate using moar Encryption

Option 7: Hide in Native C/C++

Option 8: Store in Keystore

Option 9: Keep Secrets on the Server

Option 10: Give Up

android

# Android Data Storage

- Data may be stored in a number of locations. Build your threat model appropriately.
- SQLite Databases
- Log Files
- Shared_prefs
- XML Data Stores or Manifest Files
- Binary data stores
- Cookie stores
- SD Card
- Cloud synced

android

# Android Data Storage

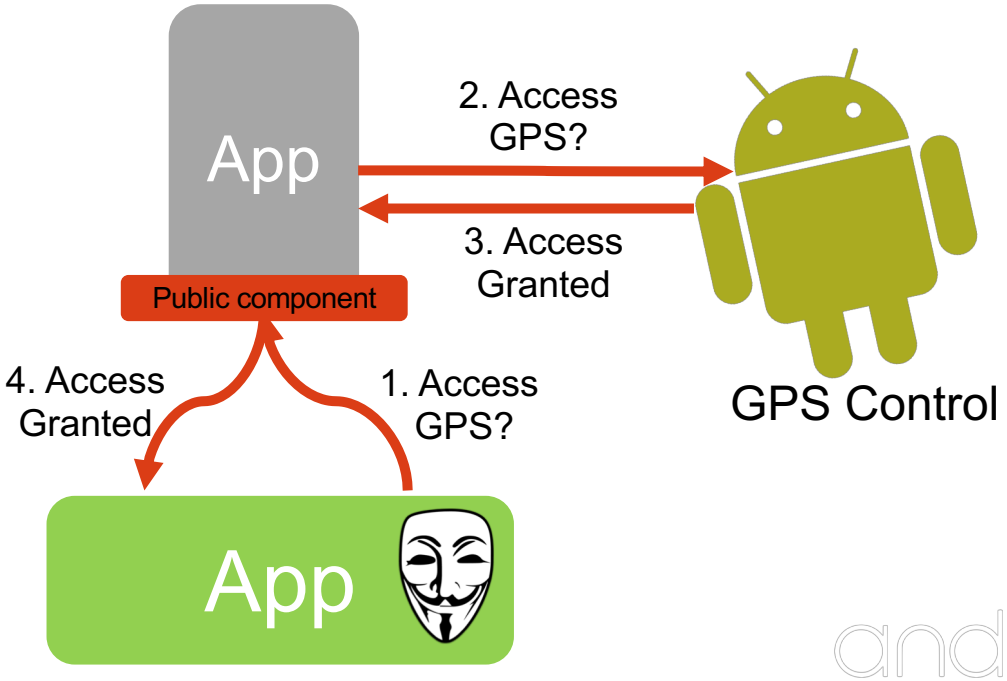
## Consider the need to store sensitive data on the device.

As a developer, you should prepare for the worst case scenario (outdated devices, physical compromise, malware, spying, etc.)

RSA SECURITY™

# Android Permission Re-delegation

- Permission re-delegation occurs when your application ***exposes*** a component that may be called by other applications
- If that API can be invoked by lesser-privileged applications, permission re-delegation occurs and these applications may abuse your application's privileges.



# Android Permission Re-delegation

For example, your application has the permission to send SMS messages (which costs the user money).

- If your application exposes a public interface that accepts some data from outside the application and uses it to construct and send an SMS message, a malicious application without the SMS privilege can leverage your application to send SMS messages
- This circumvents the permissions model and costs the user money against their will.

android

# Android Permission Re-delegation

**Attack:** A victim application exposes a component that is abused by a malicious application installed on the device

**Defense:** First, avoid exposing any data or functionality that require dangerous permissions over IPC channels unless explicitly needed. If this cannot be avoided, be sure to follow these guidelines:

- Limit who the data/functionality is exposed to
- Require the caller to have the same permission that the exposed component must have
- Validate any data received from intents to ensure that no malicious actions are taking place
- Be very explicit about the permissions your application needs to operate and do not overly permission the application

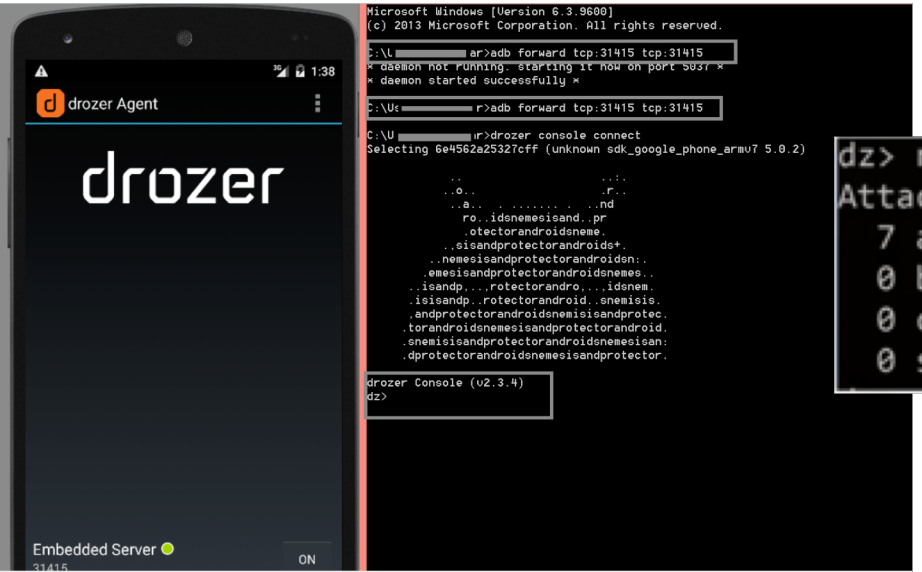
# Android Permission Re-delegation

## Vulnerable Code Snippet

```
<activity
    android:name="com.manicode.android.codeexamples.SendSMS"
    android:label="@string/title_activity_send_sms"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/smstext" />
    </intent-filter>
</activity>
```

# Android Testing IPC With Drozer

- Drozer is an Android penetration testing framework
- Allows operators to assume the role of an Android app on a device and interact with other applications as well as the Android IPC mechanism
- Server-Agent Architecture

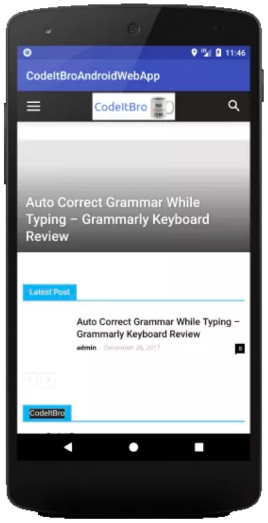



```
dz> run app.package.attacksurface  com.ovelin.guitartuna
Attack Surface:
  7 activities exported
  0 broadcast receivers exported
  0 content providers exported
  0 services exported
```

android

# **Android** Attacking WebViews

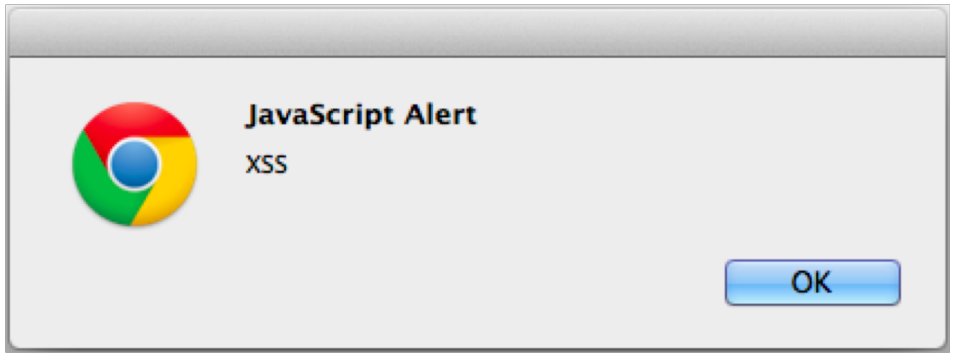
- WebViews offer many benefits for developers:
  - Reuse of existing code
  - Portability
  - Rapid patching without rolling out new app
  - Same old technologies we know and love
  - No data shared between WebView and mobile browser



android

# Android Attacking WebViews

- When misconfigured, WebViews may be vulnerable to a variety of attacks
  - Cross-Site Scripting
  - Man-in-the-Middle
  - SSL Stripping
  - Loading Malicious Links or HTML
  - CSRF via Intents



# Android Attacking WebViews

```
setAllowUniversalAccessFromFile URLs(true)
```

“Sets whether JavaScript running in the context of a file scheme URL should be allowed to access content from any origin. This includes access to content from other file scheme URLs. ”

# **Android** Protecting WebViews

When possible, ensure all WebViews explicitly disable access to files using **`setAllowFileAccess(false)`** and **`setAllowUniversalAccessFromFile URLs(false)`**

Ensure that all external external resources loaded by a *WebView* are using TLS and the app has a correct TLS configuration

# Hack Yourself.

Loads of resources out there to dive deep into mobile application security!

- OWASP GoatDroid
- OWASP Mobile Security Project
- Damn Vulnerable iOS Application (DVIA)

# It's been a pleasure.

jmesta@manicode.com