A **practical** approach to the mobile security **ecosystem** for Android

Stanford CS155

Chris Steipp
Security Partner, WhatsApp
https://www.linkedin.com/in/chrissteipp/

Meta

Agenda

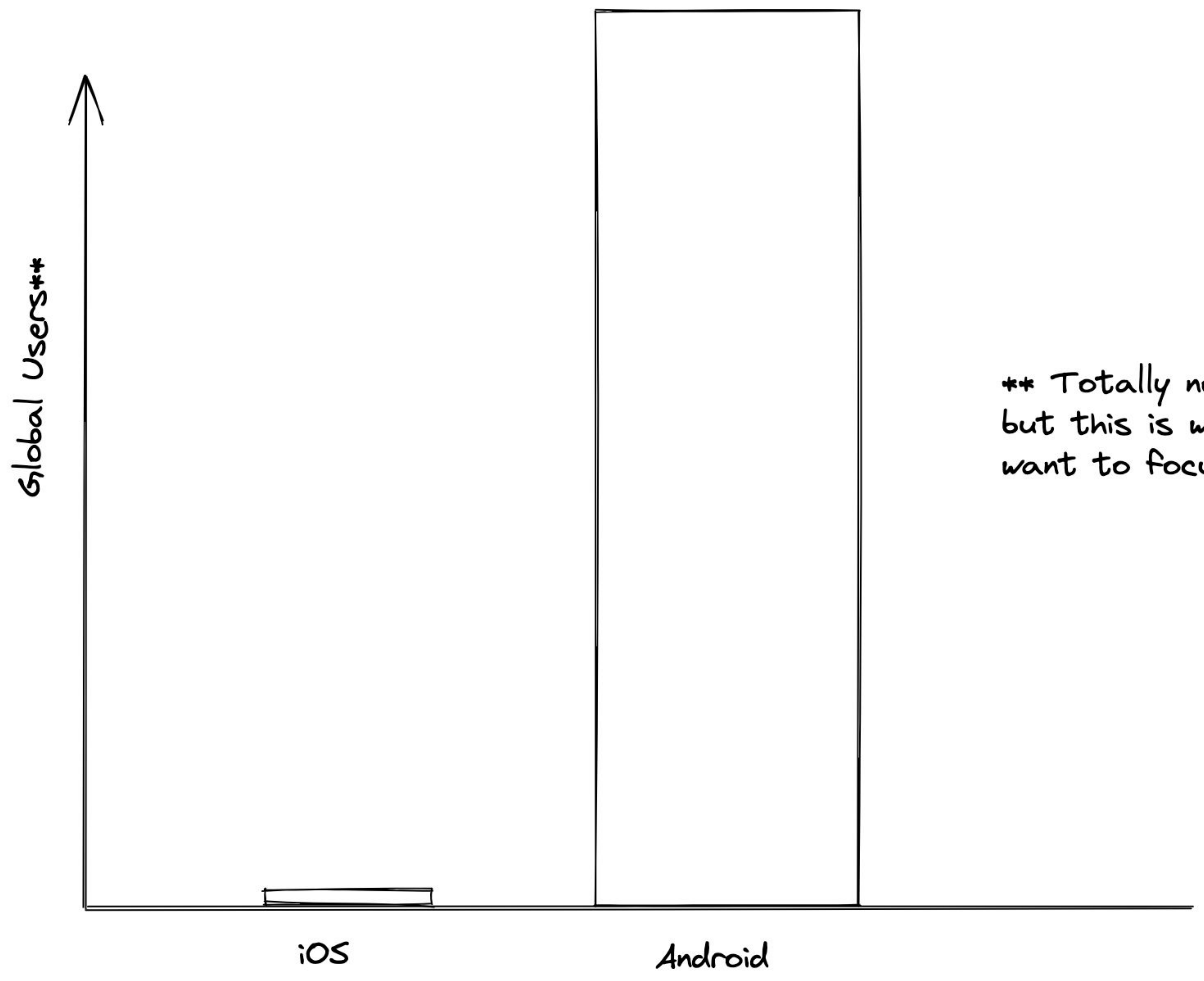
# whoami

- I live in Portland with my family and rescue dog
- Lead Info Sec for WhatsApp
- Formerly:
  - First Security Engineer at Wikimedia Foundation
  - Started AppSec program at Lyft
  - AppSec at Facebook

Who are you?

- InfoSec or Security Research Focus?
- Software Engineers or Computer/Data Scientists?
- Business or Product Design?

Why are we talking about Android?



** Totally not to scale, but this is why you may want to focus on Android.

The elephant in the room…

WhatsApp was exploited by NSO. Why should we listen to you?

02 Approach: Holistic Security

The security of android apps will depend as much on organizational culture and engineering practices, as it does on how you secure specific components.

Organizational Security

- Security Culture
- Security of endpoints, network, build infra
- Compliance programs
- Detection programs

Secure Development

- Engineering practices that promote security
- Gates / Checkpoints / Paved Roads / Retrospectives

Mobile App Security

- Specific practices and controls for Mobile App development

03 Risks

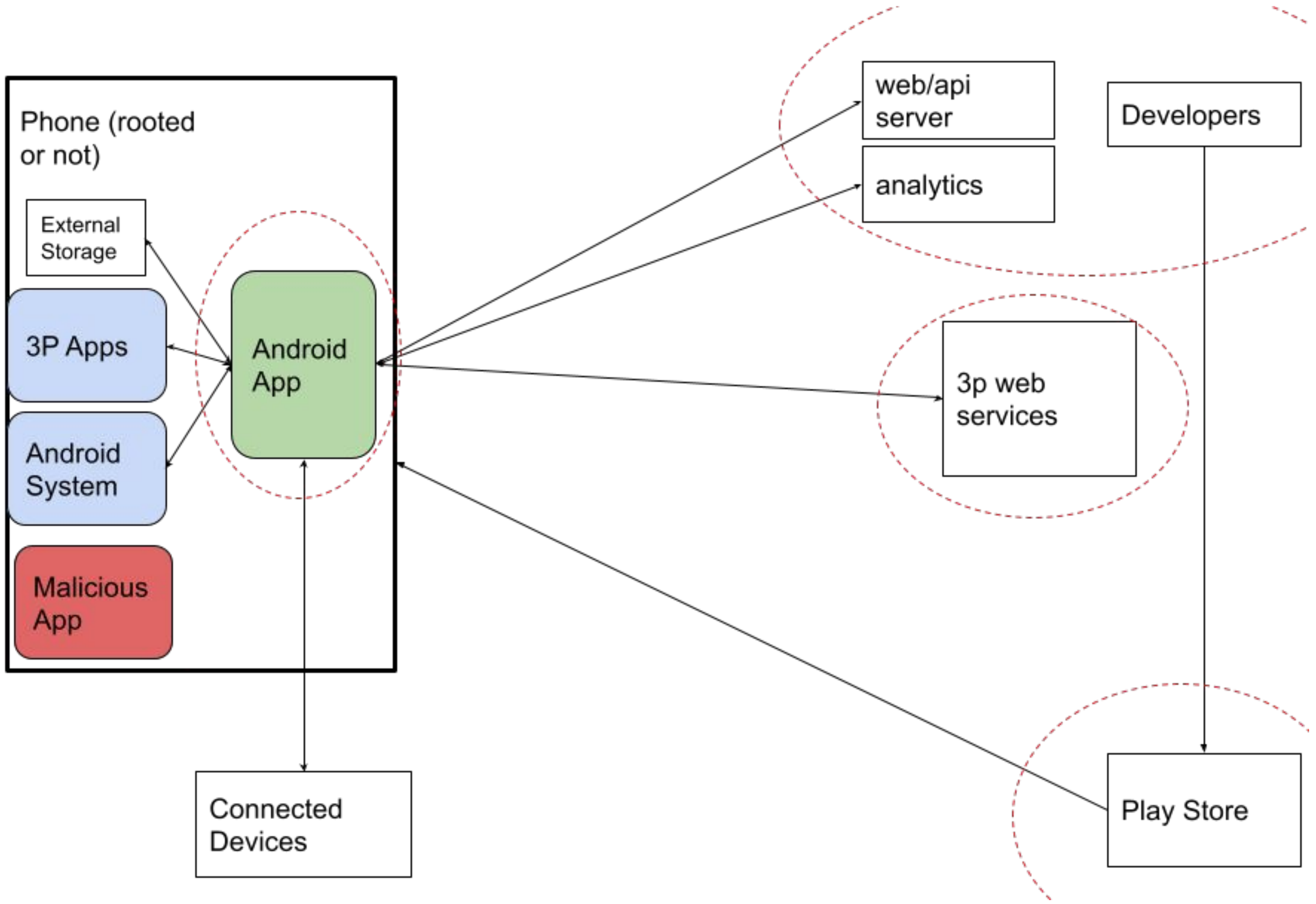
How do we know what to focus on?

How do we know what to focus on?

Threat Modeling!

- What are you building
- What can go wrong
- What should you do about those things that can go wrong

Threat Model of an Android App



Threat Modeling with LLMs?

- Overview of Threat Modeling with LLMs

Where to find lists of risks

- OWASP Top 10 (Mobile)
- Android Security Best Practices
- CWE
- Company Top 10

OWASP Top 10 for Mobile (2024)

- M1: Improper Credential Usage
- M2: Inadequate Supply Chain Security
- M3: Insecure Authentication / Authorization
- M4: Insufficient Input/Output Validation
- M5: Insecure Communication
- M6: Inadequate Privacy Controls
- M7: Insufficient Binary Protections
- M8: Security Misconfiguration
- M9: Insecure Data Storage
- M10: Insufficient Cryptography

Android App security best practices

- Enforce secure communication
  - Intents
  - Re-authentication
  - traffic encryption
- Use WebView objects carefully
  - HTML channels
  - Javascript interface support
- Provide the right permissions
  - Intents
  - data sharing across apps
- Store data safely
  - Internal storage
  - external storage
  - shared files
  - validity of data
  - cache files
  - SharedPreferences
- Keep services and dependencies up-to-date
  - Google Play
  - app dependencies

Risks we’re going to look at in depth today

- Intents & IPC
- Web Views
- Logging Private Data
- Native Code
- Authentication & AuthZ
- Managing Features
- Dependency Management
- Post-quantum cryptography

Intents & IPC

Android apps have many components, which need to talk to each other. Android makes accidental exporting and exporting to the wrong external apps easy.

Intents: How they're supposed to work

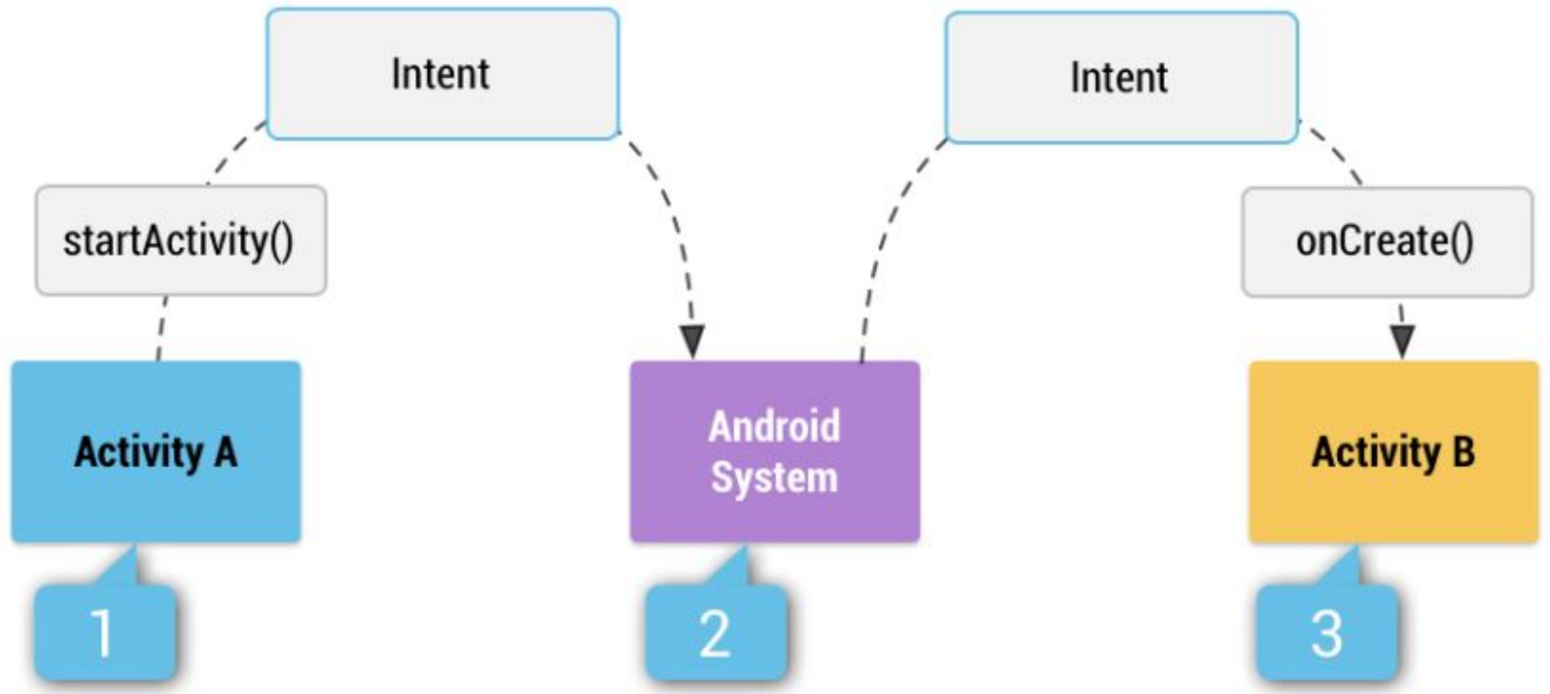


Figure 1. How an implicit intent is delivered through the system to start another activity: **[1]** *Activity A* creates an `Intent` with an action description and passes it to `startActivity()`. **[2]** The Android System searches all apps for an intent filter that matches the intent. When a match is found, **[3]** the system starts the matching activity (*Activity B*) by invoking its `onCreate()` method and passing it the `Intent`.

Deeplink Open Redirects (“Android Nesting Intents”)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   // called with myapp://?callback=app%3A%2F%2Ffoo
   Intent incomingIntent = getIntent();
   Uri u = incomingIntent.getData();
   String redirect = u.getQueryParameter("callback");
   Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(redirect));
   startActivity(intent);
}
```

Solutions:

- Check callback
- Pending Intents
- Don’t use this pattern

Deeplink Open Redirects vs Static Analysis

The Problem:

- The problem is **attacker controlled** data being **used to construct an intent**, which is then called from a privileged context
- Static Analysis can be used for taint tracking

What is Static Analysis?

Analysis of the source code looking for vulnerabilities

- Linters / minimal context
- Taint analysis from Source to Sink without a Sanitizer
- Typically have a high false positive rate

Deeplink Open Redirects (“Android Nesting Intents”)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  // called with myapp://?callback=app%3A%2F%2Ffoo
  Intent incomingIntent = getIntent();
  Uri u = incomingIntent.getData();
  String redirect = u.getQueryParameter("callback");
  Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(redirect));
  startActivity(intent);
}
```

Mariana Trench:

- Argument(0) to a class inheriting from "Landroid/content/Intent;" <- Source
- Argument(0) to method with signature "Landroid/content/Intent;\\.parseUri:.*" <- Sink

Web Views

Web views in the app have slightly different security properties than a full browser, and developers have many options for how they implement web views. Some options are more or less secure, depending on the use case.

Running Javascript, XSS, and Callbacks

```
webview.executeJavaScript(getJSForCallback());

webView.addJavascriptInterface()
```

Solutions:

- "Best practices" such as,
  - *If your application doesn't directly use JavaScript within a WebView, do not call setJavaScriptEnabled()*
  - *confirm that WebView objects display only trusted content*
- Static Analysis / taint tracking
- Security Reviews

Running Javascript, XSS, and Callbacks

Shift Left:

- Developer Training (training + docs + linters)
- Frameworks

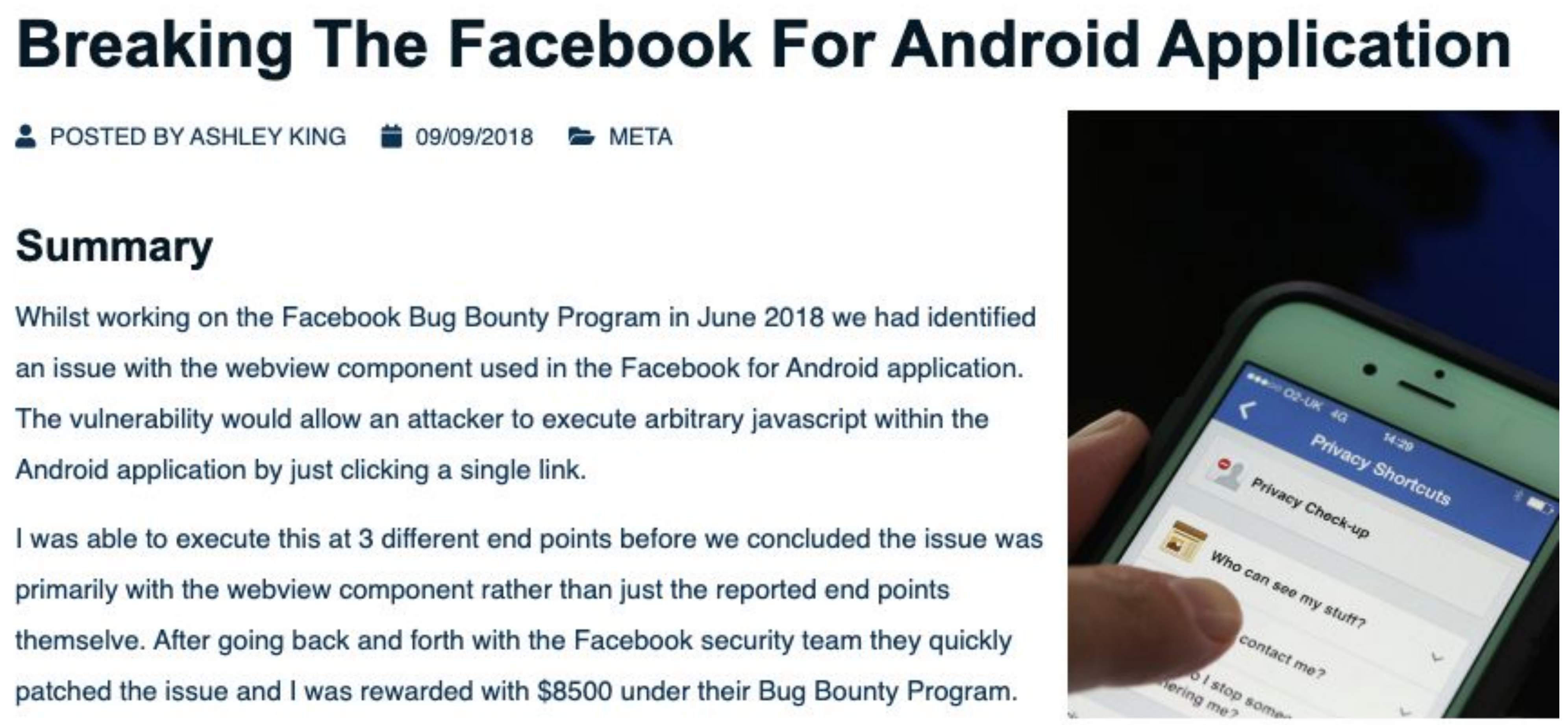
```
App:
 webview.loadUrl("javascript: var __mytoken = '" + token + "';");

Javascript:
  AndroidApp.someMethod(__mytoken, param1,...)
```

Intents + Webview for Profit

Problem:

- Intent included a url preview link
- Link wasn't properly sanitized
- Link was opened in a webview

```
adb shell am start -a
"android.intent.action.VIEW" -d
"fb://ig_lwicreate_instagram_account_full_sc
reen_ad_preview/?adPreviewUrl=javascript:c
onfirm('https://facebook.com/Ashley.King.U
K')"
```



Breaking The Facebook For Android Application

POSTED BY ASHLEY KING 09/09/2018 META

Summary

Whilst working on the Facebook Bug Bounty Program in June 2018 we had identified an issue with the webview component used in the Facebook for Android application. The vulnerability would allow an attacker to execute arbitrary javascript within the Android application by just clicking a single link.

I was able to execute this at 3 different end points before we concluded the issue was primarily with the webview component rather than just the reported end points themselve. After going back and forth with the Facebook security team they quickly patched the issue and I was rewarded with $8500 under their Bug Bounty Program.

Intents + Webview

- Benefits of a Bug Bounty programs
  - Plug: https://www.facebook.com/whitehat

Logging Private Data

When developing a product at scale, developers need visibility into errors. Your app will likely be logging app metrics, statistics, and error conditions.

- Android Vitals
- UncaughtExceptionHandler
- etc.

Logcat

Problem:

- Android <4.1 allowed apps to read log files of other apps
- Sending app logs to your server

Solutions:

- Don’t log anything sensitive

"Don’t log anything sensitive"

Logging Sensitive Data vs Runtime Analysis

- Collect logs from integration testing & QA, look for test cases’ sensitive data
- Guided fuzzers, and grep for sensitive data in the logs (WA’s FAUSTA)

Running Native Code

"Out of the 58 in-the-wild 0-days for the year, 39, or 67% were memory corruption vulnerabilities. Memory corruption vulnerabilities have been the standard for attacking software for the last few decades and it's still how attackers are having success."
- Project Zero

Running Native Code

How do we do this?

- Android NDK

What are the risks?

- All the usual memory corruption risks. E.g., CVE-2019-3568, a buffer overflow in WhatsApp's VOIP packet handling

Running Native Code

Preventions at Scale:

- Anti-exploitation compiler flags
- Fuzz Everything*
- Migrate to to safe languages**

* - Setting up JNI objects in your test harnesses is non-trivial, usually manual work

** - Developer skillset/culture, build processes, dependency management, release process, static analysis tools, dependency vulnerability detection tools

Authentication & Authorization

- Backend/API Authentication and Authorization
- How you view Identity in your Mobile App

Multiple Users Same App

Problem:

- Device sharing is common in many communities
- Each user's data is meant to be private
- We need to efficiently cache and store data, and segment the app's sandbox.

Solutions:

- Understand your product requirements and user expectations.
- Local file encryption with per-user keys

Making PINs into Keys

Problem:

- “The form factor highly encourages short passwords that are often purely based on 4-digit PINs.” (OWASP)
- Long, complex passwords are hard for users
- In WhatsApp, there are situations where users lose data if they lose their password, e.g. End-to-end Encrypted Backups.

WhatsApp's Solution:

- Fleet of HSMs that do key agreement based on a short password or PIN
  - HSM enforces brute forcing limits
  - HSM isn’t upgradable, so limits can’t be changed

Feature Management

Mobile engineering teams will often implement the ability to roll out features to only a cohort of users for A/B testing or artificially slowing rollout. The “hidden” features can be reverse engineered and enabled with tools like Frida.

Security teams often need to disable specific features without waiting for users to update their apps.

Feature Rollout

- Assume all code in your distributed app is reachable
- Code Obfuscation

Killswitches for Features

It’s far less painful to turn off a single feature than force-update many users.

Intelligent killswitches (platform-capability dependant, or operating on a regex of attacker controlled input) can be very useful.

Dependency Management

Like in any software project, if your open or closed source dependency has a security flaw, it may impact your app.

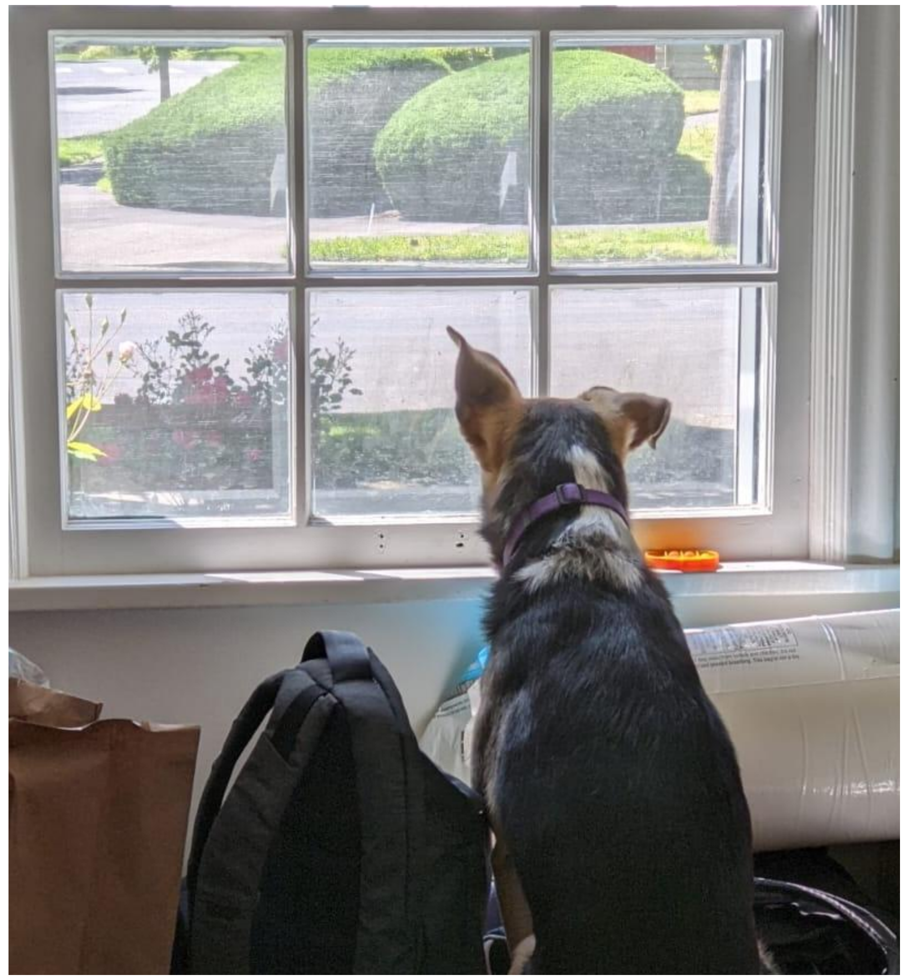
Best way to handle this?

Automation

- Software Bill of Materials (SBOM) - Know what libraries your code includes
- Automation to alert developers when libraries have known issues
- Static analysis for flow analysis

Post-Quantum Cryptography

Store now decrypt later



Solutions?

- Signal & PQXDH
- Google Chrome's new post-quantum cryptography may break TLS connections

04 Culture, Architecture, and Paved Roads

Culture

- "Culture eats strategy for breakfast" (Drucker)
- Values that include Security/Privacy

Architecture & Paved Roads

- Setting clear expectations (Paved Roads) for problems that have been solved before will free your time and most teams want to do the right thing.
- If you're in leading company/organization, you will need still need process to get help for new challenges.

05 Wrap Up and Q&A

Summary

- Your Android App's security is impacted by the entire ecosystem of your organization's security.
- This was a (very) quick overview of current threats and controls for securing Android apps. The specifics will change by the time you graduate. The threat model is slowly evolving. But hopefully I've communicated the principles so you can work these out in the future.

We're Hiring (again!)

- https://www.metacareers.com/jobs/350699871165627/

Meta

Image Credits

- (slide 3, 4, 10, 12, 43, 45) Owner
- (slide 6) Basile Morin, CC BY-SA 4.0 <https://creativecommons.org/licenses/by-sa/4.0>, via Wikimedia Commons
- (slide 19, 27) Screenshots by Owner