All this text wasn’t written by me (Tiago a.k.a TigaxMT).

All of it was transcribed by me from HackerOne video lessons.

All credits and thanks go to them.

Mobile Hacking Crash Course

DON'T PANIC

Mobile testing requires a lot of setup, but it's actually quite easy. In fact, if you're comfortable with testing web apps, you'll be right at home.

Many of the biggest mobile app bugs are really web bugs wearing a disguise.

WHY MOBILE?

You might think that a lot of people are out there testing testing a mobile apps we all rely on every day, but there are almost none when you compare to the web.

With mobile devices being a bigger and bigger part of our lives, this is a serious problem.

TYPES OF APPS

Pure native apps are those that don't make use of web views at all. These include most games and simple apps that don't connect out to web services.
This is, by far, the least common type of app.
It takes the most effort to build and also the most effort to test.

Hybrid apps generally use a combination of both native views and web views. This is the most common type of application, with functionality able to be implemented in whatever means is best for development.

Web wrapper apps are thin web views around a mobile-friendly web application.
These allow the most code to be shared between existing web apps and new mobile apps in development. They are painless to test, as you're typically talking about standard web vulnerabilities.

HYBRID FRAMEWORKS

There are a number of hybrid app frameworks that complicate testing a bit, but simplify development of cross-platform mobile apps. Xamarin allows app development in C#. PhoneGap, Cordova, Titanium, and others allow app development in JavaScript.

LANGUAGES TO KNOW

iOS:
- Objective-C
- Swif

Android:
- Java

Both:
- JavaScript

GETTING START

TARGET SELECTION

- Apps that use a large number of web views or simply wrap a web app
- Apps that expose a lot of different functionality when talking to servers
- Games with leaderboards

GET SOURCE

If the original source code isn't available, follow the relevant video guide to get decompiled source for the target.
This isn't always an option and it's rarely perfect, but having some source is better than none.

SETUP A PROXY

If you're using Burp Proxy, make sure that the proxy is listening all interfaces. By default, it only listens on localhost. This is under the Proxy tab, Options, and then Proxy Listeners. Select the default listener, click edit, then select "All Interfaces".
Follow the platform guide for installing your CA certificate and bypassing certificate pinning.

STANDARD WEB BUGS

Most bugs associated with the web are possible in mobile apps. Commonly found:
- SQL Injection
- Direct object reference

- Improper authorization/authentication
- Insecure uploads

CHECK CREDENTIAL STORAGE

Both Android and iOS provide secure credential storage for saved logins, which applications should use.
If they aren't using those APIs, it's very possible they are storing them improperly.
Look for credential encryption in the source, as well as plaintext credentials on disk.

LOOK FOR INSECURE CONNECTIONS

All connections from the mobile app to various web services should be over HTTPS.

If you see plain HTTP connections in your proxy, this is a sign that critical data may go over the wire in plaintext.

LOOK FOR SECRETS

Many mobile applications contain embedded secret keys for web service access.
Developers sometimes assume that because they're distributing a binary, people won't be able to find these keys but decompilation and other techniques can allow for their discovery.

LOOK AT SESSION-KEEPING

In web apps, sessions are typically handled via cookies. In mobile apps, sessions or authentication are often handled via an extra header sent with every request. If these are sent with every single request regardless of destination, then a redirect to your server may lead to account compromise.

FIND DEBUG/DEV INTERFACES

Many mobile applications have secret development interfaces. These may allow you to change web service targets, see communications, modify sessions, and more.
While these won't generally contain bugs themselves, they can simplify testing.

LOOK AT APP DATA

Both iOS and Android have app-specific locations for data. These may contain cached credentials, transaction histories, and more.
This is a great source of low-hanging fruit in many mobile applications.

LOOK FOR INSECURE CRYPTO

Many mobile applications use hand-rolled crypto systems, generally for data at rest.
These are nearly universally broken.

Look for insecure cipher modes and known-bad algorithms especially.

CHECK FOR SCREENSHOTS

When a mobile app goes into the background, the OS will take a screenshot to show in the app switcher. This screenshot gets stored to disk and can reveal things like account numbers in sensitive applications.
If you see sensitive data when switching applications, that may be a vulnerability.