Mobile Application Security Testing

Codemash – January 10, 2020

Hans Weisheimer

hans@ath0.io

@hweisheimer

https://github.com/hweisheimer/ios-testing

Web is: Stable

Backwards Compatibility > *

Protocols are 20+ years old

HTML5 is as old as the iPhone

Mobile is: “Evolving”

Surviving platforms are ~12 years old

New versions every year
New security features every year
iOS propagates quickly, Android not so much

iOS: Swift ABIs continue to change
Android: It’s all bytecode, no big deal

Web is: Transparent (Thanks F12)

HTTP Requests & Responses
Cookies & JS Local Storage
JS Debugging
JS Console & REPL
DOM Manipulation

Mobile is: Opaque

Imagine you're holding a device in your hands and want to know:

What's in the KeyChain? 

Is this app's API connection encrypted? 

Is the app *data* encrypted? ¯_(ツ)_/¯

Tools of Past

Many *required* a Jailbroken/Rooted device

Toolsets mostly differed between iOS and Android

New OS releases would lock things down, break tools

On iOS, Jailbreaks eventually got *really* hard

Goals: Minimal Roadblocks

- Off-the-Shelf hardware
- Rooting/Jailbreaking not required
- Free software
- Free accounts/subscriptions

Low Hanging Fruit

What can we inspect that's relatively low effort?

- Web APIs
- Compiler options
- Permissions
- Transport security settings
- Certificate pinning
- Root and Jailbreak detection
- Stored data (files, SQLite, KeyChain, SharedPrefs)

Prerequisites: Hardware

Mac of some kind (for iOS)

Physical device and USB cable

Both connected to the same WiFi

Prerequisites: iOS

Install XCode from the App Store

Run `xcode-select --install`

Preferences -> Components -> DL iOS Simulator

Register as an Apple Developer (free tier is fine)

Create a code signing certificate

Prerequisites: iOS Third-Party

Package manager such as Homebrew

```
brew install libusbmuxd libimobiledevice ios-deploy python
```

Prerequisites: Android

Install Android Studio, this will get you the SDK and ADB (Android Debug Bridge)

Java Development Kit

Third-Party: aapt, apktool

Prerequisites: Shared

Proxy – Burp Suite, ZAP, Charles, Fiddler, …

```
pip install virtualenv
virtualenv ~/venv-objection &&
   source ~/venv-objection/bin/activate
pip install frida frida-tools objection
```

Network Inspection: HTTP

Use an intercepting proxy

Quirky on emulators and simulators

Testing strategies are similar to web

Android: Must allow user certs in manifest

Network Inspection: Everything Else

iOS - Remote Virtual Interfaces
(`rvictl` command)

Attach Wireshark to the new virtual adapter

Or, sniff from a strategic point on the network

Meet your new friends

FRIDA

github.com/hweisheimer/ios-testing

Instrumenting an App

Frida modes we're interested in are:

- Injected - Using frida-server on rooted device
- Embedded, aka Frida Gadget – Packaged w/ app

We'll use Embedded mode

Patching Applications

Working from iOS source? Add dylib to project
Run on device from XCode

Working from an IPA? Use **`objection patchipa`**
Sideload with ios-deploy

An APK? Use **`objection patchapk -N`**
Allow sideload in device settings

Test the Plumbing

Run **`frida-ls-devices`** – you should see a USB device

Run the application – it will appear to hang*

Attach with **`objection explore`**

* with default config

Finding Loot: Filesystem

Find the DocumentsDirectory location:

```
env
cd [location-from-output]
```

Use **`ls`**, download interesting files to inspect:

```
file download name-on-device name-on-mac
!cat name-on-mac
```

Built-in helpers exist for things like plists

Finding Loot: SQLite DBs

Open: **`sqlite connect [file]`**

Show schema: **`.schema`**

Show tables: **`.tables`**

Query: **`sqlite execute select * from loot;`**

Finding Loot: KeyChain Entries

This is where you're likely to find the best stuff

In Objection, use:

`ios keychain dump`

May need to respond to TouchID/FaceID prompt

(OK, that part *is* magic)

Fun fact: KeyChain entries persist after the app is uninstalled

Passive Shenanigans

Simulate Jailbreak status, or bypass common checks

Bypass certificate pinning

Bypass weak biometric authentication

Monitor method calls and dump the argument values

github.com/hweisheimer/ios-testing

Demo

github.com/hweishelmer/ios-testing

Resources: Docs

Frida:

https://frida.re/docs

@fridadotre @oleavr

Objection:

https://github.com/sensepost/objection/

Wiki is good (also on Github)

@leonjza

Resources: More Tools

Passionfruit (iOS)

Slick Web UI, does a lot of what we've seen today

https://github.com/chaitin/passionfruit

MobSF

Web app, automated analysis for iOS and Android

https://github.com/MobSF/Mobile-Security-Framework-MobSF

Resources: Better Talks Than This

Kev Cody - How to Frida Good

SnowFROC 2019, mDevCamp 2019

https://slideslive.com/38916544/how-to-frida-good?locale=en

Leon Jacobs - Meticulously Modern Mobile Manipulations

DEFCON 27

https://www.youtube.com/watch?v=7LKXSYFrYAM

Resources: Better Guides Than This

OWASP Mobile Security Testing Guide

https://mobile-security.gitbook.io/mobile-security-testing-guide/

OWASP Mobile Application Security Verification Standard

https://mobile-security.gitbook.io/masvs/

github.com/hweisheimer/ios-testing

Final Thoughts

Building a lab? Let's talk device selection.

Casual testers can remain stock

Serious testers will benefit greatly from jailbreaking

Seriously, don't jailbreak a device that you use for other things

iOS: Avoid last 2 gens (XS/XR, 11)

Android: Many options. Fewer OS updates over the phone's lifetime versus Apple