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.

iOS Quickstart

STRUCTURE OF AN iOS APP

IPA CONTAINER

iOS apps come in the IPA format, which is really a ZIP file. This can be unzipped with any standard archive tool, if you just rename to .zip .
The really interesting bits are in the Payload/<AppName>.app directory.

ENCRYPTION

If this an IPA from the App Store, it will be encrypted by default. This will prevent you from analyzing it or tampering with it in any way.
We'll cover a technique to decrypt these later.

INFO.PLIST

The Info.plist file inside the app package is used to indicate a number of properties about the application. It's typically in a binary form, but can be edited via Xcode or converted to/from an XML form using plutil utility.

iOS VS ANDROID TESTING

SIMULATOR VS EMULATOR

While most Android applications can be tested via emulation, this is not true of the vast majority of iOS apps. This is because iOS has no real emulator, but only a simulator.
Your application gets built for x86_64 and runs natively. Unless you have the source code for an app, you need a device.

NATIVE CODE

Android apps are typically compiled to Dalvik bytecode, sometimes with some native code for performance or obfuscation , this is easy to decompile. iOS apps are compiled to native machine code, which is much harder to decompile. For more than a method or two, this approach just won't work.

Note from TigaxMT:

So what is the Dalvik bytecode?

Programs for Android are commonly written in Java and compiled to bytecode for the Java virtual machine, which is then translated to Dalvik bytecode and stored in .dex (Dalvik EXecutable) and .odex (Optimized Dalvik EXecutable) files.
The compact Dalvik Executable format is designed for systems that are constrained in terms of memory and processor speed.

Source: Wikipedia

TOOLS

XCODE

This is the standard iOS development environment. Within it, you can both build applications and run them in the simulator. The ability to build small test application is very useful for mobile testing.

BFINJECT

https://github.com/BishopFox/bfinject

This lets you inject a shared library into any running app on iOS. Through this you can do effectively anything, most notably decrypting App Store apps for analysis or patching.

CYCRIPT

http://www.cycript.org/

With Cycript you can easily manipulate applications using a JavaScript variant, allowing you to call Objective-C methods and much more.

FRIDA

https://www.frida.re/

This lets you instrument applications, meaning you can hook and inspect functions that are called. This can be used for everything from watching files that are opened or monitoring network connections, to disabling certificate pinning.

CYDIA IMPACTOR

http://www.cydiaimpactor.com/

Impactor lets you easily install IPA files on non-jailbroken devices, as it will automatically sign them for installation.

HOPPER

https://www.hopperapp.com/

This is the most affordable decompiler/disassembler available for iOS applications. It isn’t perfect , but it’s great for reverse-engineering applications.

SSL KILL SWICTH 2

https://github.com/nabla-c0d3/ssl-kill-switch2

This will let you disable SSL certificate pinning for most applications on iOS. Some will still require extra work, as discussed later.

BURP SUITE MOBILE ASSISTANT

https://portswigger.net/burp/documentation/desktop/tools/mobile-assistant

This tool works with Burp to automatically configure proxying, CA certs, pinning bypasses, and more.

BURP + CA CERTIFICATE SETUP

PROXY SETTINGS

For iOS simulator, proxy settings are drawn from the system settings. If you want to split traffic so that you don’t have all of the requests from your system coming through Burp, Privoxy will let you configure what traffic goes where.

For physical devices, proxy settings are under WiFi settings. Click the ‘i’ button next to network you’re connected to, click “Manual” under proxy settings , then enter your info. Make sure that your proxy listener in Burp is set to either listen on all interfaces , or on the local network interfaces – not localhost only.

INSTALLING CA CERTIFICATE

For your device to trust Burp for SSL connections, you need to install the CA cert.
Once the proxy is enabled, go to http://burp/ in the browser and click “CA Certificate” in the top-right corner.
Accept the prompts to install and you should be able to open any HTTPS site now.

JAILBREAKING

WARNING

Never jailbreak your primary device(s). They are at a seriously increased risk of compromise and your IT people at work will hate you.

Only jailbreak devices that you would willingly unlock and hand over to a stranger.

THE PROCESS

The jailbreaking process is different for each iOS version and device, with the latest versions often being unsupported.
The easiest route is to buy a used, slightly older device and jailbreak that, keeping it on a vulnerable iOS version perpetually.

TESTING TIPS

DECRYPT IPA

To decrypt an IPA, you can use bfinject , e.g: bfinject -P AppName -L decryp

It will start a network server and tell you where you can download the decrypted IPA. From here,you can extract plaintext binaries and manipulate them at will.

DISABLE CERT PINNING

If your target application fails to connect after enabling your proxy, certificate pinning may be the problem.
Burp Mobile Assistant or SSL Kill Switch 2 will fix this for any application using the standard SSL APIs, but if an application uses OpenSSL or custom methods, you may have to patch the code or disable it via bfinject/Frida.

NON-iPAD INSTALLATION

Some applications are set up such that they are only installable on iPads. This can be changed via editing the Info.plist file and changing UIDeviceFamily to 1.
This will allow installation on any iOS device, thought the interface may be hard to use.

MEMORY CORRUPTION BUGS

Due to most iOS apps being built with Objective-C, it’s possible that you’ll run into memory corruption bugs like buffer overflows. While their discovery and exploitation is out of scope for this video/transcription, look for uses of memcpy, strcpy, and sprintf and you may be able to find a critical bug.

CUSTOM URL SCHEMES

The Info.plist file contains information on URL schemes registered by the application, under the CFBundleURLTypes key. These provide useful entry points, which you may be able to exploit – either for XSS or for some exploit against the native code.