iOS Penetration Testing

Brown bag session

Niels Hofmans, 15/11/2023

Table of contents

Requirements

- macOS device (preferably Apple Silicon)
- USB to Lightning cable (official adapter only)
- iOS device
- Xcode (+ developer tools)
- Free Apple Developer Account
- Homebrew (https://brew.sh)
- Objection (frida + Python wrapper, frida-gadget)
- iOS-deploy (https://github.com/ios-control/ios-deploy
- mobsf (https://github.com/MobSF/Mobile-Security-Framework-MobSF)

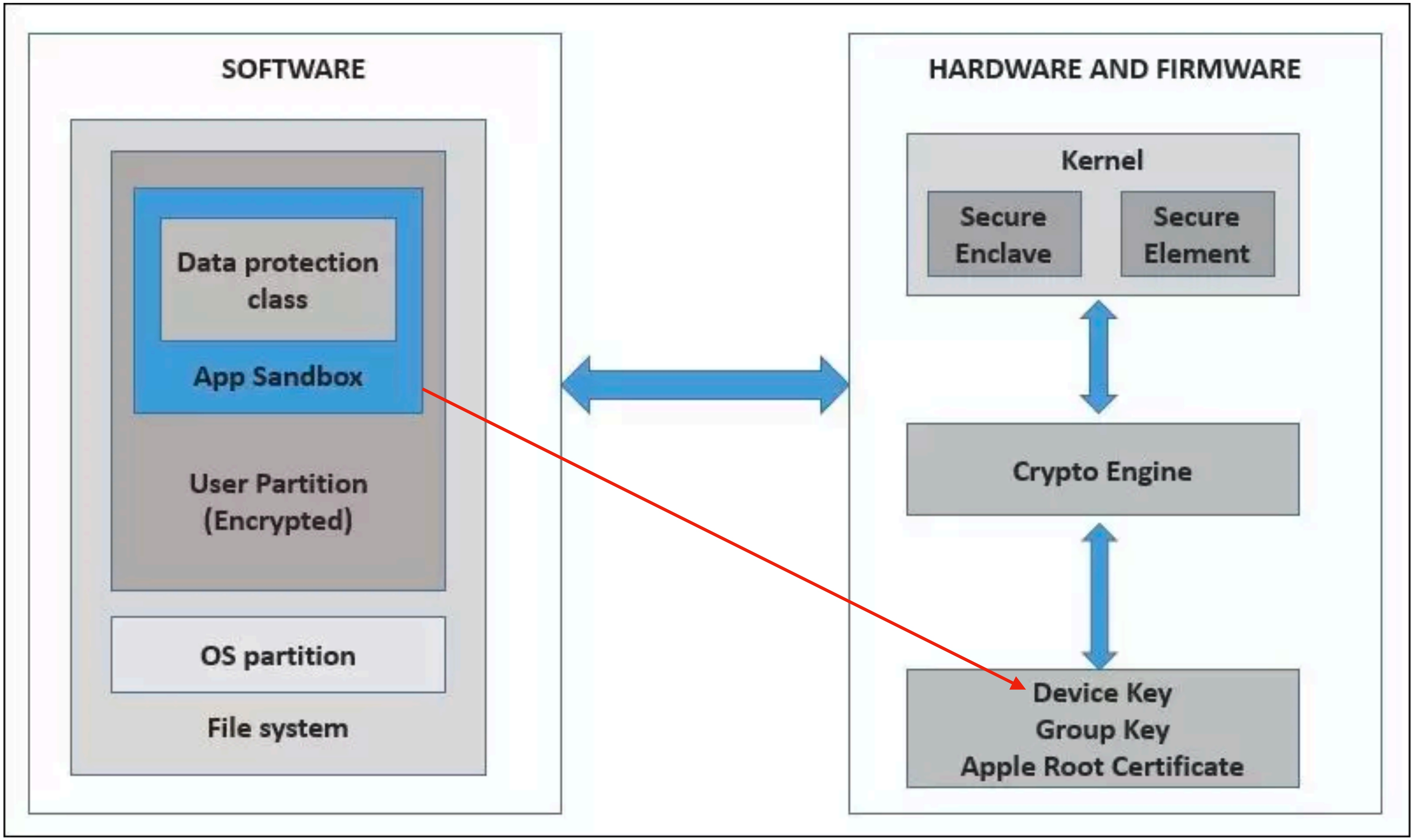
Architecture

Introduction

- macOS evolved from NeXT, which has roots in BSD (like linux)
- Apple OSs share a common kernel
  - iOS, macOS, tvOS, watchOS, iPadOS, podOS
  - open-source at https://github.com/apple/darwin-xnu
- Kernel or Framework CVE usually results in OS patches to all
- Portion of Frameworks are shared across OSs (Keychain, …)
- This enables;
  - running iOS apps on Apple Silicon (emulate)
  - building for iOS and macOS (Catalyst)

Architecture

Security

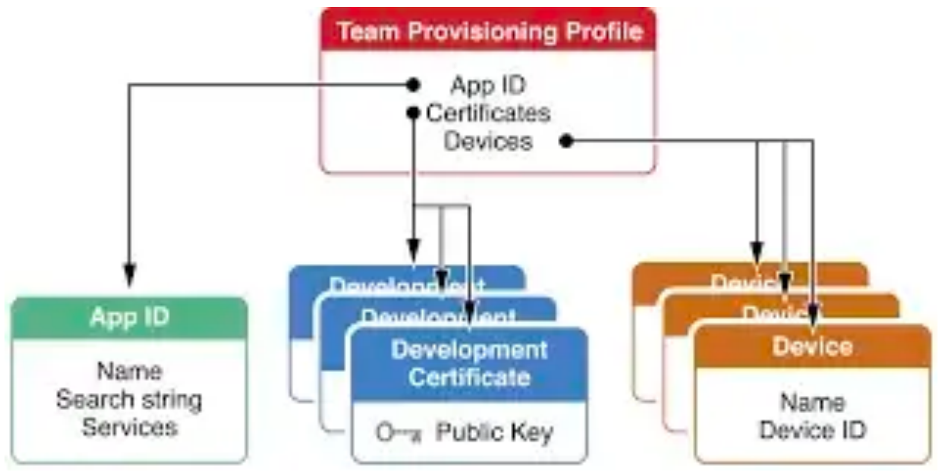


https://itzone.com.vn/en/article/ios-security-architecture/

Architecture

App delivery

- Apps are encrypted specific to device (FairPlay)
- Apps should be installed from
  - App Store: strict app review process (automated + manual)
- Unless;
  - TestFlight: ad-hoc app publishing via enrollment links
  - Developer account certificate has been approved on device (public key in keychain)
  - Provisioning profile in app matches approved account certificate and device UUID



Architecture

App Sandbox

- Runs app under mobile user (not root)
- App resides in Container through chroot
  - /private/var/containers/Bundle/Application
  - No hardware access
  - Restricted file, network, socket, IPC, memory
  - Modified mmap/mprotect syscalls
  - ASLR/XN

Architecture

App contents

- .ipa is the iOS app extension (ZIP)
- Payload/
  - Foobar.app/
    - Info.plist $\longrightarrow$ App Manifest
    - embedded.mobileprovision $\longrightarrow$ Provisioning profile
    - Frameworks/
    - *.storyboardc
    - Foobar $\longrightarrow$ Executable (FairPlay encrypted)

Architecture

Static app analysis

- A lot of information is already extractable
- Use the mobsf web interface (try readonly on https://mobsf.live)
- Finding examples;
  - ATS (TLS trust) (Info.plist)
  - Embedded URLs/secrets (binary)
  - Missing build hardening (binary)
  - Insecure APIs
  - …

Architecture

Controlling iOS

- Connect iPhone to Mac
- Trust device via Finder (previously iTunes) (passcode)
- Instrumentable via;
  - Xcode (compile & deploy app, more later)
  - Finder (copy apps, full/partial backup, copy app data)
  - Console.app (device logs)
  - iOS-deploy
  - Objection/frida

Jailbreaking

Intro

- App Sandbox & others make it impossible to interact with App
- One could install root CA via MDM Profile and MiTM (cumbersome)
- If only we could disable those controls…

- Jailbreaking is exploiting a security vulnerability to
  - Get system root
  - Install sideloaded app store (e.g. Cydia)
  - Modify the OS, apps and more (e.g. Tweaks)
- Big community
- Most don’t patch vulnerability after jailbreaking (!)

Jailbreaking

To jail or not to jail

- A fully patched iPhone is useless for testing (today)
- Untethered vs semi-tethered
- https://canijailbreak.com
- https://reddit.com/r/jailbreak
- < iOS 16
- iOS 15.x (https://github.com/epeth0mus/Fugu15)
- iPhone 5s>iPhone X (A8-11, iOS 12+) (https://checkra.in)
  iPhone 5s>iPhone X (A8-11, iOS 12<x<15.7.1) (https://github.com/palera1n/palera1n)

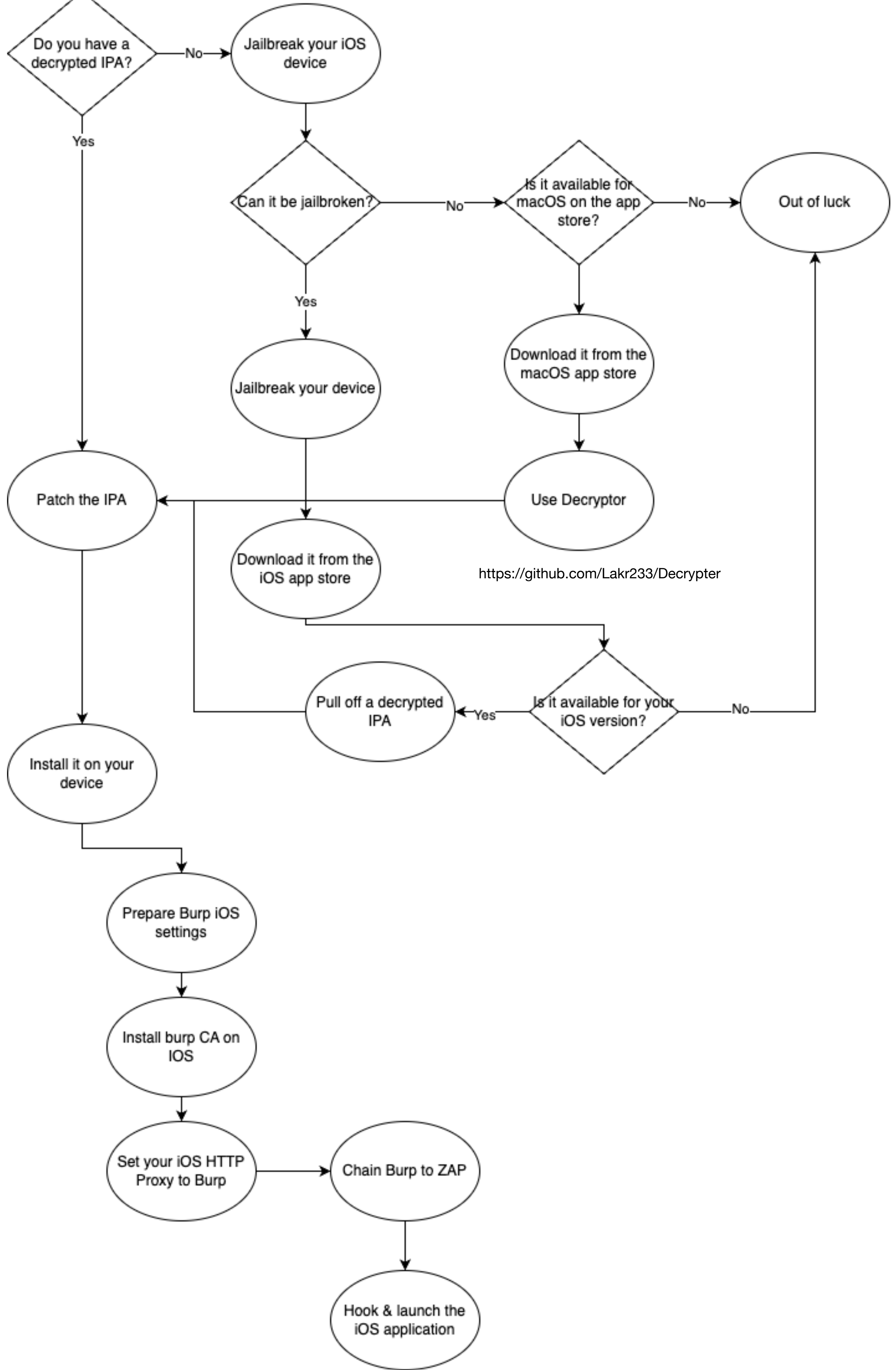
Jailbreaking

How to jailbreak

1. Get a suitable iPhone device (SDK, jailbreak)
2. Connect iPhone to Mac
3. Trust iPhone to enable USB comms
4. Execute jailbreak tool
5. Keep your iPhone charged if semi-tethered (!)
6. Install OpenSSH via Cydia (tcp/22, root/alpine)

Jailbreaking

If not…


Do you have a decrypted IPA?
No
Jailbreak your iOS device
Yes
Can it be jailbroken?
No
Is it available for macOS on the app store?
No
Out of luck
Yes
Jailbreak your device
Download it from the macOS app store
Patch the IPA
Use Decryptor
Download it from the iOS app store
https://github.com/Lakr233/Decrypter
Pull off a decrypted IPA
Yes
Is it available for your iOS version?
No
Install it on your device
Prepare Burp iOS settings
Install burp CA on IOS
Set your iOS HTTP Proxy to Burp
Chain Burp to ZAP
Hook & launch the iOS application

Patching

Frida, Objection

- Frida is a Python instrumentation toolkit for iOS & Android
- Can inject scripts, hook functions, …
- Frida-server, Frida-gadget

- Objection is a security tool wrapped around Frida
- Can inject frida-gadget into IPA and re-sign with developer account

Patching

TL;DR

1. pip3 install objection && npm install -g applesign

2. git clone https://github.com/Tyilo/insert_dylib /tmp/dylib && cd /tmp/dylib && xcodebuild && cp build/Release/insert_dylib /usr/local/bin/insert_dylib

3. Create new empty XCode project

4. Add yourself as a developer using Project Settings > Signing & Capabilities
   Select the default Developer team and Automatic Signing
   Ensure security find-identity -p codesigning -v returns a UID

5. Select your connected iOS device in the top left corner instead of an emulator
   Ensure a provisioning profile is now selected under 'Signing & Capabilities'
   Build your hello world app to generate the provisioning profile.

6. Ensure you download the frida iOS gadget (universal) to ~/.cache/frida/gadget-ios.dylib (create ~/.cache/frida directories)

7. objection patchipa --codesign-signature 'xxx' --source app.ipa

Deployment

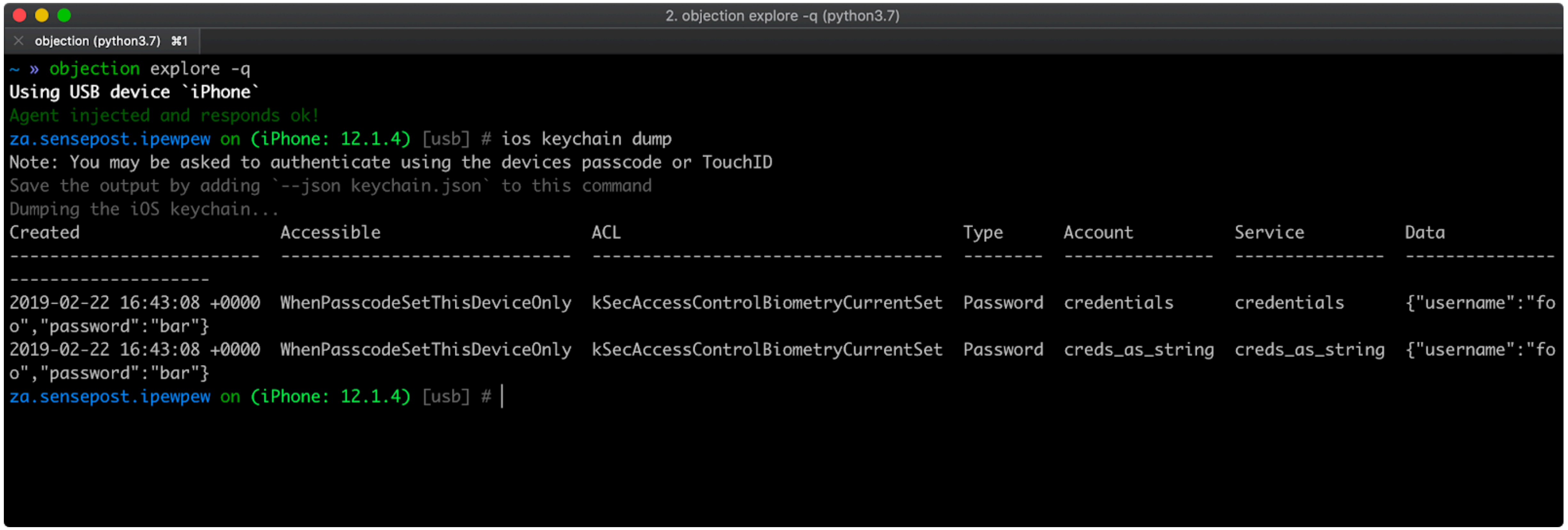
TL;DR

- npm install -g ios-deploy
- ios-deploy --bundle Foobar-patched.ipa —no-wifi —debug
- Can give issues if certain Entitlements are included in the provisioning profile
  - frida-server and hook into running app using process ID

Hooking

TL;DR

- Retrieve package identifier: frida-ps -U -ia
- objection -g <package-id> explore --startup-command 'ios sslpinning disable'



2. objection explore -q (python3.7)

objection (python3.7) ⌘1

```
~ » objection explore -q
Using USB device `iPhone`
Agent injected and responds ok!
za.sensepost.ipewpew on (iPhone: 12.1.4) [usb] # ios keychain dump
Note: You may be asked to authenticate using the devices passcode or TouchID
Save the output by adding `--json keychain.json` to this command
Dumping the iOS keychain...
Created                    Accessible                     ACL                                  Type      Account          Service          Data
-------------------------  -----------------------------  -----------------------------------  --------  ---------------  ---------------  ---------------
--------------------
2019-02-22 16:43:08 +0000  WhenPasscodeSetThisDeviceOnly  kSecAccessControlBiometryCurrentSet  Password  credentials      credentials      {"username":"fo
o","password":"bar"}
2019-02-22 16:43:08 +0000  WhenPasscodeSetThisDeviceOnly  kSecAccessControlBiometryCurrentSet  Password  creds_as_string  creds_as_string  {"username":"fo
o","password":"bar"}
za.sensepost.ipewpew on (iPhone: 12.1.4) [usb] #
```

Interception

TL;DR

- Create Burp CA (RSA/4K, 1year)
- “Import / Export CA certificate” and disable TLS 1.3
- Configure burp on 0.0.0.0 + change iOS Network/wifi http proxy settings
- Install Burp CA profile from http://burp
- Approve profile in Settings
- Enable full trust for root cert in About>Cert
- Profit! Open app again via objection and see app traffic

- Warning: Xamarin has own network stack, see nviso blog for those

Hunting

Now what?

- https://mas.owasp.org/MASTG/
- Small native pentest on native portion
  - Activities
  - Permissions
  - Receivers/Intents
  - Local storage / Keychain
- Most mobile apps pentest like a web app

Hunting

More advanced scenarios

- Sometimes you’ll need to develop custom bypasses & hooks
- Frida has a great JavaScript API: https://frida.re/docs/javascript-api/
- e.g. TouchID bypass (“binary instrumentation”)
  https://github.com/0xdea/frida-scripts/blob/master/ios-snippets/raptor_frida_ios_touchid.js
- e.g. frida bypass
  https://codeshare.frida.re/@enovella/anti-frida-bypass/
- And more on Frida codeshare
  https://codeshare.frida.re/browse

Phew!