Secure iOS Development

OWASP February meeting 2024

David Fekke Feb. 2024

About me

David Fekke

- fek.io/blog
- youtube.com/c/polyglotengineer
- github.com/davidfekke
- @jaxnode @polyglotengine1

How secure is the iPhone?

- Syed Rizwan Farook’s iPhone 5C
- FBI gave the iPhone to NSA to unlock
- FBI Requested Apple create an OS for breaking the encryption, Apple refused
- FBI found an Israeli security company to bypass the lock
- The iPhone uses SSD memory for non-volatile storage
- iPhone users usually have iCloud accounts
- iCloud backups are now encrypted

OWASP Recommendations

MASVS

- Mobile Application Security Verification Standard
- Document describes the different levels of mobile application security
- Divided up into STORAGE, CRYPTO, AUTH, NETWORK, PLATFORM, CODE and RESILENCE
- Assumes you are following secure coding standards

MASVS

STORAGE

MASVS-STORAGE-1	The app securely stores sensitive data
MASVS-STORAGE-2	The app prevents leakage of sensitive data

MASVS

CRYPTO

MASVS-CRYPTO-1	The app employs current strong cryptography and uses it according to industry best practices
MASVS-CRYPTO-2	The app performs key management according to industry best practices

MASVS

AUTH

MASVS-AUTH-1	The app uses secure authentication and authorization protocols and follows the relevant best practices
MASVS-AUTH-2	The app performs local authentication securely according to the platform best practices
MASVS-AUTH-3	The app secures sensitive operations with additional authentication

MASVS

NETWORK

MASVS-NETWORK-1	The app secures all network traffic according to the current best practices
MASVS-NETWORK-2	The app performs identity pinning for all remote endpoints under the developer’s control

MASVS

PLATFORM

MASVS-PLATFORM-1	The app uses IPC mechanisms securely
MASVS-PLATFORM-2	The app uses WebViews securely
MASVS-PLATFORM-3	The app uses the user interface securely

MASVS

CODE

MASVS-CODE-1	The app requires an up-to-date platform version
MASVS-CODE-2	The app has a mechanism for enforcing app updates
MASVS-CODE-3	The app only uses software components without known vulnerabilities
MASVS-CODE-4	The app validates and sanitizes all untrusted inputs

MASVS

RESILIENCE

MASVS-RESILIENCE-1	The app validates the integrity of the platform
MASVS-RESILIENCE-2	The app implements anti-tampering mechanisms
MASVS-RESILIENCE-3	The app implements anti-static analysis mechanisms
MASVS-RESILIENCE-4	The app implements anti-dynamic analysis techniques

Vectors

Why Target mobile

- Bad guys are always looking for the path of least resistance
- Operating systems and web clients have known vectors of attack
- Hackers will look for the easiest path

Types of Native apps

More than you think

- iPhone OS 1.0 was only web (HTML5)
- Native app first allowed in iPhoneOS 2.0
- Native apps using CocoaTouch
- Cordova/PhoneGap
- React Native
- NativeScript/.NET Maui/Xamarin

Programming Languages

On Apple platforms

- JavaScript thru JavaScriptCore WebKit
- HTML/CSS thru WebViews (WKWebView, SFWebView)
- Swift
- Objective-C
- C and C++

Objective-C

Language details

- Created in 1980s by Brad Cox and Tom Love
- Mixture of C and SmallTalk and Very dynamic language
- Most objects are immutable by default
- All objects use a *pointer
- Memory management baked into the Clang compiler, Automatic Reference Counting or ARC
- Described as built for reverse engineering

Objective-C++

Language details

- Objective-C and C++ interoperability added to language by Steve Naroff at Next in early 90s
- Objective-C files end in *.m, and Objective-C++ end in *.mm
- Objective-C code work in unison with C++ code
- Objective-C++ code generally used as a wrapper around C++ libraries

Swift

Language details

- Modern Strictly typed language and object oriented
- Allows for opaque types where type can be inferred
- Also uses the LLVM compiler
- Interoperates with C and Objective-C
- Recent support for C++
- Also uses ARC for memory management

iOS/iPadOS Platform

- NextStep OS derivative
- NextStep based on BSD Unix
- Uses a Microkernel
- Sockets based networking
- Apps are run in a sandbox

Where is the data on a iPhone

- Data can be stored in a sandboxed area on the filesystem
- Apple also has a framework called CoreData that uses SQLite
- Recently introduced SwiftData
- iCloud storage
- Most iPhone apps make HTTP requests to API servers

Filesystem

- Since the A7 chip, Apple uses a security system called Secure Enclave
- Every file is encrypted differently
- AES 256 based encryption
- Impacts of Jailbreaking

Filesystem

Working with local storage

- NSFileManager/FileManager
- UserDefaults for storing key value pairs
- Keychain for secure storage
- Path Traversal attacks
- Validate for ‘../‘ in path variables, hacker could delete whole upper directory
- When uploading a document, use NSTemporaryDirectory

URL Schemes

External URL used to open your app

- Nothing prevents another app from using the same URL Scheme as the one in your app
- Could be used to circumvent authentication
- OAuth uses returnURL
- Use Universal Links instead

Networking

- iOS and iPadOS use URL Loading System
- URLSession is used for HTTPS requests
- TLS 1.3 is the default for current iOS/iPadOS apps
- Many libraries like Alamofire are used by 3rd party developers
- Can use VPN and Secure Socket Tunnel for communication
- Implement SSL pinning

Credential Storage

- Do not store credentials on the device
- Credentials should be stored on your server
- Authenticate with your server
- JWT and past tokens can be stored using Keychain storage
- Keychain storage is a lower level API for storing keys and secrets on Apple devices
- Use this sparingly

Jailbreaking

Similar to Rooting

- The protections for the iPhone were broken the very first year the iPhone was released.
- It is possible to detect if an iPhone has been jailbroken, but not 100%
- When working with sensitive data, check to see if iPhone is jailbroken, then disable the app
- SecurityKit framework has jailbreak detection implemented
- https://github.com/vadim-a-yegorov/Jailbreak-detection-The-modern-way

```
//suspicious apps path to check
    private static var suspiciousAppsPathToCheck: [String] {
        return ["/Applications/Cydia.app",
                "/Applications/blackra1n.app",
                "/Applications/FakeCarrier.app",
                "/Applications/Icy.app",
                "/Applications/IntelliScreen.app",
                "/Applications/MxTube.app",
                "/Applications/RockApp.app",
                "/Applications/SBSettings.app",
                "/Applications/WinterBoard.app"
        ]
    }
```

Preventing JailBroken

Avoid being Hijacked

- Avoid using Objective-C
- Avoid using method names like `isJailBroken()`
- Don’t leave detection method in your AppDelegate
- Hackers will use tools like XCon to find vulnerabilities
- Try to use a library

Secure Coding

Best Practices

- Input validation
- Prevent Stack overflows and underflows
- Memory corruption
- Access control

Database

CoreData and SwiftData

- Be sure you actually want to store data locally
- Both CoreData and SwiftData use SQLite
- SQLite does not encrypt data
- SQLCypher uses AES 256 encryption
- BlackBerry also has a secure version of SQLite

Secrets

- Many SDKs and API require secrets and keys to use the service
- Many providers give orgs a JSON or PLIST file with secrets embedded
- Generally there are overrides for using these types of files
- DON’T store secrets in your source
- Use CI/CD system to replace secrets at build time

3rd Party dependencies

- Possible to introduce vulnerabilities through 3rd party libraries
- Libraries can be added through CocoaPods, SPM or Carthage
- Go through any added library and check for vulnerabilities
- Try to avoid using them unless you have to
- ShellShock security hole was in OpenSSL for 22 years
- Github checks with Dependabot

Static Analysis

- Different tools available to check for security vulnerabilities
- Veracode is one of the services you can use
- Static analysis is not a guarantee that your app will not be hacked

PEN testing

- Also known as Penetration testing
- You can hire white hat security companies to do this type of testing
- If your app deals with real PII around financial or medical information, may be legally required

Questions?