iOS App Security Tips and Tricks

Ankur Vekariya

Currently, security is one of the most important topics in IT industry. Users, companies are taking the subject of data security and privacy more and more seriously. This also applies to mobile applications due to their proximity to the users. Frequency of use and convenience mean that mobile Applications often store important private data.

iOS, due to its closed system and restrictions imposed by Apple, is considered one of the most secure mobile operating systems. This does not mean, however, that you can neglect security when developing an iOS application.

Potential Security risks in iOS

Data leak — By using application user mostly enters their private data and storing that data in an unsecured manner that creates risk of this data being leaked if device got unauthorized hands.

Man in middle attack — Intracting http/https requests and responses is relatively easy to do when it comes to iOS apps. Using tools like Charles Proxy, even an amateur can get to know our app requests, corresponding server responses, and manipulate network traffic by sending doctored

requests. Unfortunately, SSL is not enough to make your app secure.

How to make our app secure?

in this article, we will discuss mistakes that developers make towards app security and how to taking care while developing iOS app.

User data protection (Keychain vs. NSUserDefaults for storing sensitive data)

As per researched on multiple apps from the Store and a lot of them are doing the same mistake, storing sensitive data where they do not belong.

If you are storing sensitive data in **UserDefaults**, then you are risking your application's information.

UserDefaults get stored simply as a property list file that is located inside Preferences folder of your app. They get saved in our app without being encrypted .

Basically, by using a third party mac application like **iMazing** without even having to Jailbreak your device, you can easily view **UserDefaults** data for any app downloaded from the AppStore.

These mac apps are simply designed to allow you to explore and manage third-party application files that are on your iPhone. And you can easily explore **UserDefaults** of any app.

Examples are **Access Tokens, subscription flags, email ,password** ,etc.

All this data can be easily retrieved and altered and make damage to apps, from free usage of paid features to hacking network layer and much more.

You should always keep in mind one thing **UserDefaults** is designed only to save a small amount of data like preferences of a user inside the app,

like stuff that is completely insensitive.

Keychain API

In Order to save our apps sensitive data, we must use Security services provided by Apple.

Keychain services API helps you solve these problems by giving your app a way to store the small amount of user data in an encrypted database called the **keychain**.

here some below is sample for store and retrieve values

for save data we need to query with `kSecClass , kSecAttrAccount , kSecAttrServer , kSecValueData` as below on key

Saving data with keychain

For Get data from keychain by key which was used at save data like below

for load data from keychain

Below are example for saving string value with keychain and retrieve it:

Example: save (here “XYZ ” value stored with key “**TestString**”)

save value

Get Value (here getting stored value from “TestString”)

receiving data from keychain

In the **keychain**, you are free to save **passwords and other secrets** that the user explicitly cares about, such as credit card information or even short sensitive notes.

Few useful Keychain Wrappers:

1. **SwiftKeychainWrapper by Jason Rendel(jrendel) for Swift.** https://cocoapods.org/pods/SwiftKeychainWrapper or https://github.com/jrendel/SwiftKeychainWrapper
2. **SAMKeychain by Sam Soffes for Objective C.** https://cocoapods.org/pods/SAMKeychain
3. **Locksmith by Matthew Palmer for Swift.** (Check out the video tutorial) https://github.com/matthewpalmer/Locksmith

Download keychain sample code from here.

Enable Application Transport Security:

With the launch of iOS 9 and OS X El Capitan, Apple has introduced App Transport Security, which enforces developers to use secure network connections. This change implies that every connection the application makes must use HTTPS protocol and TLS 1.2.

In other words, our application cannot communicate with a server using a non-secure connection, such as HTTP, unless it is explicitly indicated. As

this was a breaking change, Apple provided an easy way to master this new requirement by adding exceptions or disabling it in the plist file.

Nevertheless, it is strongly recommended not to bypass this restriction, and instead, use secure connections in our apps to avoid potential (and easy) attacks.

Resources:

• *Configuring App Transport Security Exceptions in iOS 9 and OSX*

- *10.11Working with Apple's Application Transport*
- *NSAppTransportSecurity Apple's official documentation*

SSL Pinning:

Once ATS is enabled, the second step to increase our apps security consists in **enabling SSL Pinning**.

SSL Pinning is a technique that allows us to deal with an attack called *Man in the Middle*. SSL is based on the certificate's "chain of trust". When the communication starts, the client checks if the received server's SSL certificate is trusted by any **SSL Certificate Authority.**

We can use SSL Pinning to ensure that the app communicates only with the designated server itself. This is done by saving the target server's SSL certificate inside the app bundle.

SSL Pinning has a visible disadvantage, not related to the security itself: the app must be updated whenever the server's SSL key is changed, due to expiration and other reasons.

Resources:

- *How to make your iOS apps more secure with SSL Pinning*
- *Enforcing Stricter Server Trust Evaluation Apple's official*

documentation

Next time you are developing an iOS application, make sure to keep these security “must-dos” in mind. If you have any suggestions or questions, please feel free to contact me or leave a comment in the section below!