Cobalt
Pen Testing as a Service

Is Your Mobile App Storing Your Company Secrets?

By Swaroop Yermalkar

Tips from pentesting 100+ mobile apps

Is your product or application a mobile app? Do you use any AWS services? Are your product security engineers working on mobile application security assessments? Looking for information about the importance of mobile app security? If you answered yes to any of these questions then this blog is for you!

In the past, I've worked on multiple mobile applications pen tests and from them I've learned a few important lessons that I want to share with you! It's important to note that this can happen with any application available in a mobile app store.

1. Hardcoded AWS Keys

Today many enterprises use AWS for their applications. We've already seen several cases where AWS keys have gotten pushed to a github public repo and accounts have been compromised. If you haven't heard about AWS keys being pushed to public repos, read about an attacker who launched various AWS services using keys found in a public github repo, What I learned when AWS (almost) billed me for $29,594 in one day.

As part of a mobile app pen test, we do reverse engineering of every binary. **As per OWASP, Reverse Engineering is a process where an attacker will typically download the targeted application from an app store and analyze it within their own local environment using a suite of different tools.**

For App Store Applications, we first decrypt the apps and then dump the code. Once we get the code, we look for hardcoded secrets, interesting methods, frameworks used, etc. Now in many applications, we observed that AWS keys were hardcoded in app. Huh! For proof of concept, we showed that this key could list a few IAM users and fetch AWS account information.

An attacker can download the app (which is publicly available), can reverse engineer, get the AWS keys and depending upon permission model for access keys, can get control to your AWS, launch unwanted AWS services which can cost you in millions.

Now as per AWS Security best practices for managing keys, **you have a mobile app. Do not embed an access key with the app, even in encrypted storage. Instead, use Amazon Cognito to manage user identity in your app.**

Reference:
https://docs.aws.amazon.com/general/latest/gr/aws-aCcess-keys-best-practices.html

2. Extraneous Functionality

As per OWASP, an attacker seeks to understand extraneous functionality within a mobile app in order to discover hidden functionality in backend systems. The attack-er will typically exploit extraneous functionality directly from their own systems without any involvement by end-users.

In many applications, we observed hardcoded QA account credentials or references to different APIs for upcoming features. In some cases, application features intended to be "hidden" from users or weren't intended to be accessed until the "official" launch, but since they were already in the app, they could be discovered and used / exploited. Using hardcoded QA account credentials or references to endpoints, we were able to access them during security testing.

Competitors can use such information to check out upcoming features or an attacker can also use this information for unauthorized access.

When you're developing mobile apps, you should always consider it as your public github repo. Anyone in the world can reverse engineer it, can dump the code, so it should be developed securely by following the best security practices. Ensure that whatever func-tionality is included in the app is well tested, regardless of whether it's intended to be used. It's also recommended to perform a security audit of your mobile app on time to time basis to find underlying vulnerabilities and train your developers to code securely and avoid such incidents.

About the Author

Swaroop Yermalkar works as core security researcher with Cobalt with a diverse skill set focused on Mobile App Pentest, Web, API and Network Pentesting. In addition, he has authored the popular book "Learning iOS Pentesting" (https://goo.gl/T8jvjJ) & open source project lead for OWASP iGoat (https://github.com/OWASP/iGoat-Swift) which is developed for mobile security. You can reach out to Swaroop at @swaroopsy.