Code Signing & Reverse Engineering

Pran Kishore
Lead Software Engineer @ EPAM

Goal:
Create Awareness on code signing.

It’s easy to reverse engineer.

Pran Kishore
Lead Software Engineer @ EPAM

Hash

a mathematical function that maps data of any length to a fixed-length value, called a hash value.

Why it is used

One-way
Not possible to backtrack input

Collision-resistant
Not possible to find two different inputs that map to the same output

Deterministic
for a given input it always produces the same hash value

Setup for code signing

Understand 3 things below in general

Certificates (Public and Private key pair).

Provisioning profiles

Entitlements

Peep into certificates provisioning profile

Demo

You can view the names, or identities, of your public/private key pairs used for signing your applications with the following Terminal command: security find-identity -p codesigning -v

Look into one of your certificates by using get info

Quick look your provisioning profile

Purpose of code signing

1. Ensuring the asset has not been altered
2. Ensuring the asset came from an authentic source.

Where is code signing applied

1. *While generating an IPA*
2. *While generating an Apple Wallet file*
3. *Creating a framework*

Pran Kishore
Lead Software Engineer @ EPAM

Code signing in nutshell

A + B + C = Signature

A = Asset list hash / Directory hash
B = Developer certificate
C = WWDR Certificate

If either of A / B / C is altered or Signature itself is tampered it will be detected.

Apple Wallet file generation process

Create the source files for a pass.

Create a pass type identifier.

Generate a signing certificate.

Create a digital signature for the pass.

Create the signed bundle.

Apple Wallet file generation process

Move all the files in a folder together

Zip all the files in a folder

Update the extension to pkpass

Reverse engineering Apple wallet file Live Demo

Reverse engineering an IPA file
Live Demo

1. How to look into IPA file
2. Check entitlements details codesign -d --entitlements - <path to executable>
3. Check embedded provisioning profile security cms -D -i <path to provision>
4. Check assets .car file using the tool https://github.com/bartoszj/acextract command line demo
5. Check the list of strings you have shipped in your file by using strings <app executable name inside your payload>

Any one in audience today shipping framework?

Embed xcframework directly and check it’s code signature

Demo with Lottie

Reverse engineering: it’s not as complex as it sounds.

1. Simply understand the generation process.
2. After clear understanding it’s just back tracing your steps.
3. Find the command line tools working in background of ui.
4. Check the man page of command line tool.
5. Do UI step by step via command line and you can now automate in CI CD.

Further read

https://developer.apple.com/documentation/walletpasses/building-a-pass

https://developer.apple.com/library/archive/documentation/Security/Conceptual/CodeSigningGuide/Introduction/Introduction.html

https://developer.apple.com/documentation/technotes/tn3127-inside-code-signing-requirements