Best practices for protecting on flutter

by Alvaro Vasconcelos

Overview

When developing applications with flutter, we search productivity, but when it comes to applications with sensitive data, we must be aware of security.

Now i will show you the most common problems found on the market by OWASP and how to avoid them in the flutter.

Real world attacks examples

Top 10 mobile risks - OWASP

M1 - Improper Platform Usage
M2 - Insecure Data Storage
M3 - Insecure Communication
M4 - Insecure Authentication
M5 - Insufficient Cryptography
M6 - Insecure Authorization
M7 - Client Code Quality
M8 - Code Tampering
M9 - Reverse Engineering
M10 - Extraneous Functionality

M1 - Citrix Worx apps
M2 - Tinder
M3 - Misafe smart watches
M4 - Grab Android app
M5 - Ola app
M6 - Viper smart start
M7 - WhatsApp
M8 - Pokemon GO
M9 - Everyone
M10 - Wifi File Transfer

Real world case - Citrix Worx apps

M1 - Improper platform usage

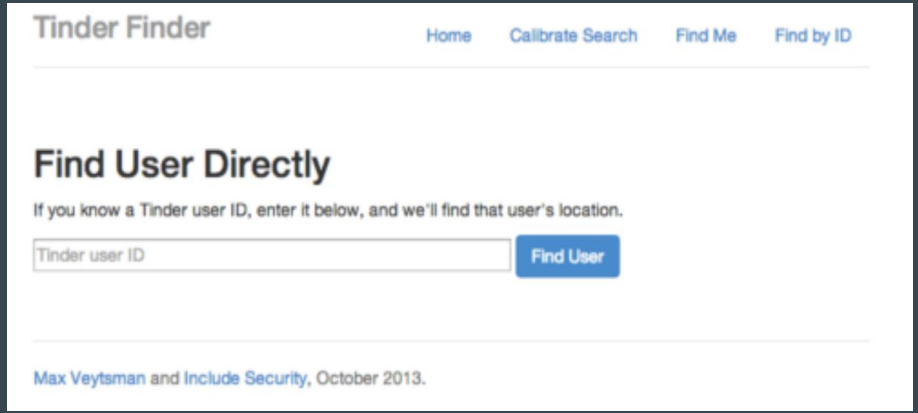
The problem like to be that the secret that was retrieved by passing Touch ID was stored incorrectly. The app assumed that the user was correctly authenticated when the authentication process was canceled and app restarted.

https://support.citrix.com/article/CTX214006

Real world case - Tinder

M2 Insecure data storage

Tinder introduced a feature that showed people logged on near you. Problem: the exact location of every person near you was sent to the device.



https://www.abine.com/blog/2014/tinder-app-vulnerability/

Real world case - Misafe smart watches

M3 Insecure communication

Communication was not encrypted and not correctly authenticated.

https://nakedsecurity.sophos.com/2018/11/16/hacking-misafes-smartwatches-for-kids-is-childs-play/

Real world case - Misafe smart watches

M4 Insecure authentication

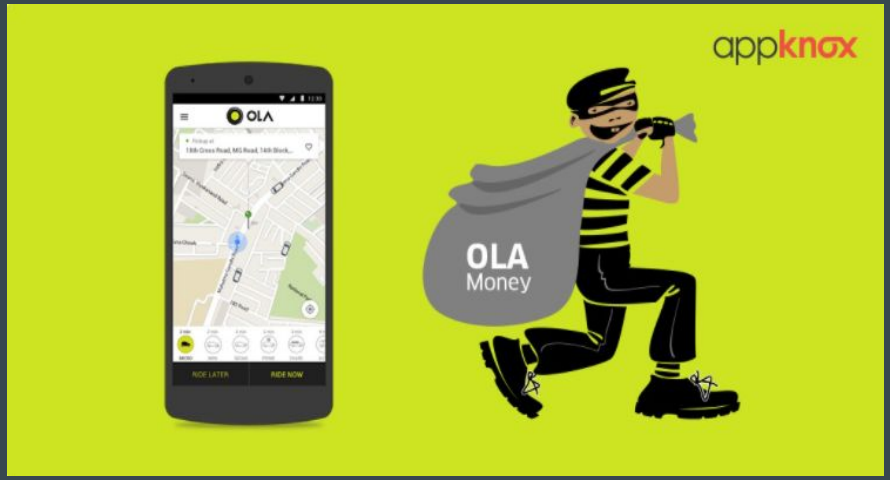
The security researcher was able to bypass 2FA by brute forcing 4 digit code. There was no limit of how many times the sent 4 digit code could be entered. Problem: gain access to account with information on rides, payment methods, orders.

https://hackerone.com/reports/202425

Real world case - Ola app

M5 Insufficient cryptography

Appknox scanned the Ola app and discovered major weaknesses in how cryptographic keys were used. They discovered that the cryptographic key used was “PRODKEYPRODKEY12“. The same key was also used to encrypt passwords which means that users other accounts where they were reusing passwords might have been at risk as well.

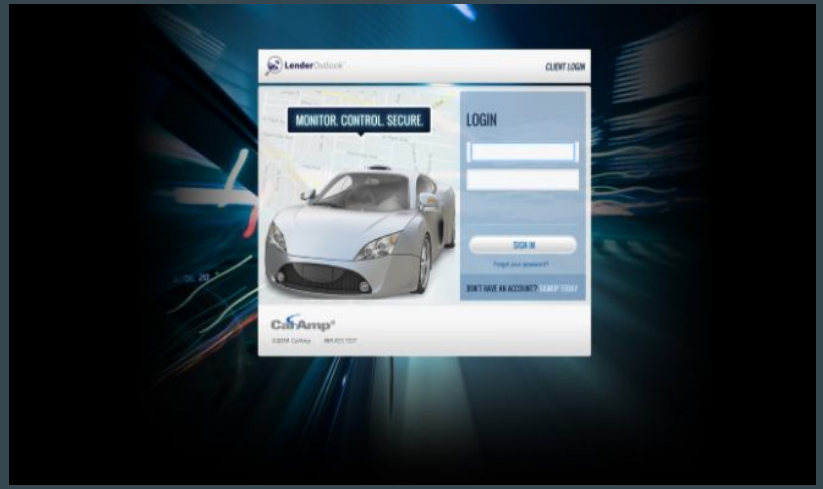


https://www.appknox.com/blog/major-bug-in-ola-app-can-make-you-either-rich-or-poor

Real world case - Viper smart start

M6 Insecure authorization

A security researcher discovered that the Viper smart start failed to correctly authorize users. After you log in to the server it was possible to change the id number of the car and gain access among other things the cars location. It was also possible to change data about the car and open the car remotely.



https://medium.com/@evstykas/remote-smart-car-hacking-with-just-a-phone-2fe7ca682162

Real world case - WhatsApp

M7 Client code quality

WhatsApp engineers found that it was possible to create a buffer overflow by sending specially crafted series of packets to WhatsApp when making a call. For this to work the call does not need to be answered and the adversary can run arbitrary code.

https://thehackernews.com/2019/05/hack-whatsapp-vulnerability.html

Real world case - Pokemon GO

M8 Code tampering

Fans reverse engineered the application, fed wrong geolocation data and time to find rare pokemon and make eggs hatch faster. A website was created that showed the location of every pokemon on a map, which changed the game dynamics quite a lot.

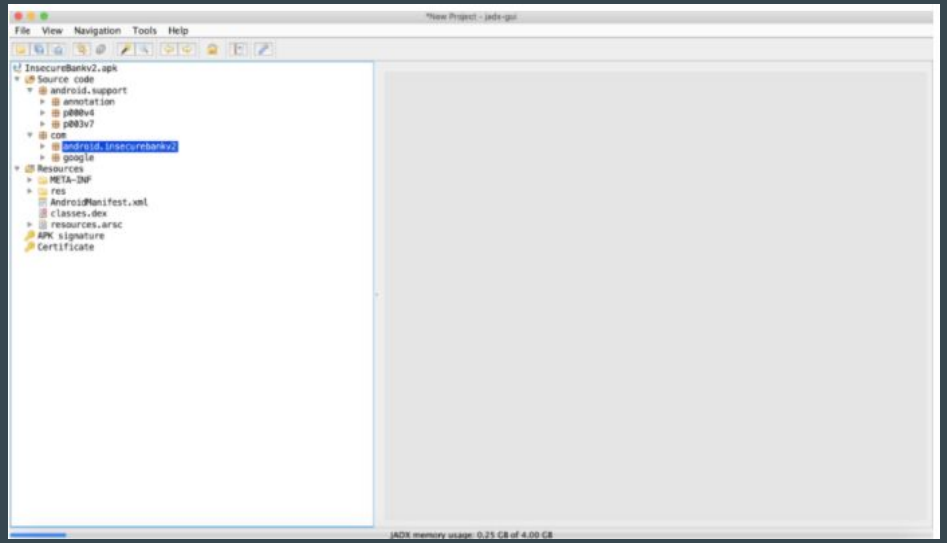
https://nordicapis.com/how-pokemon-go-fans-hacked-em-all-and-how-to-prevent-similar-reverse-engineering/

Real world case - All

M9 Reverse engineering

I decided to not include a separate example, since reverse engineering was used on most examples listed.

Reverse engineering makes it easier to exploit other vulnerabilities in the application. It can reveal information about backend, cryptographic constants and ciphers, and intellectual property.



https://www.nowsecure.com/blog/2020/02/26/what-to-look-for-when-reverse-engineering-android-apps/

Real world case - Wifi File

M10 Extraneous functionality

Wifi File Transfer App opens port on Android device to allow connections from the computer.

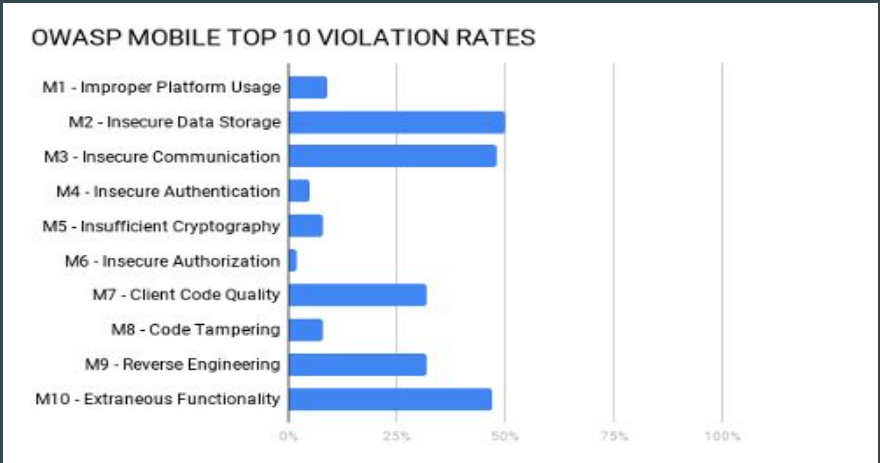
Intended use: transfer files, photos, anything stored on SD card.

Problem: there was no authentication like password, anyone could connect to device and have full access.

https://www.wired.com/2017/04/obscure-app-flaw-creates-backdoors-millions-smartphones/

Statistics

NowSecure tested apps on the App store and Google Play store and found out that 85% of apps violate at least one top 10 risk.



https://www.nowsecure.com/blog/2018/07/11/a-decade-in-how-safe-are-your-ios-and-android-apps/

Of these apps 50% have insecure data storage and almost the same number of apps use insecure communication.

How do i prevent on flutter

Improper Platform Usage, Code Tampering and Reverse Engineering	• Root detection • Obfuscation dart tools
Insecure Data Storage and Insufficient Cryptography	• Sandbox app directory • Encrypted shared preferences • Avoid the storage of any sensitive data
Insecure communication, authentication and Authorization	• Certificates by a Trusted CA • OAuth 2 on native client • Roles and Permissions only in backend.
Poor Code Quality	• Latest sdk and Dependencies up to date • Effective Dart

Prevent on flutter - M1, M8 and M9

Root detection and obfuscation

- Root detection
  - Check the device for root before running (flutter_jailbreak_detection)
- Obfuscation dart tools
  - Obfuscating Dart code on release build

Prevent on flutter - M2, M5 and M10

Sandbox directory and Encrypted shared

- Sandbox app directory
  - Sandbox app directory path using method getApplicationDocumentsDirectory(path_provider)
- Encrypted shared preferences
  - Keychain on iOS and AES encryption on Android(flutter_secure_storage)
- Avoid the storage of any sensitive data
  - Do not store sensitive user data on the device when possible

Prevent on flutter - M3, M4 and M6

Certificates, OAuth2 and Roles and permissions

- Certificates by a trusted CA
  - http client on flutter ignores the system proxy use SecurityContext with trusted roots enabled
- OAuth 2 on native client
  - Best OAuth2 implementation on flutter by the OpenID (flutter_appauth)
- Roles and permissions only in backend.

Prevent on flutter - M7

SDKs, dependencies and best practices

- Latest platforms SDK
  - Keep SDK development up to date on all platforms
- Dependencies up to date
  - Keep Dependencies up to date on all platforms
- Effective dart
  - Follow the best practices of flutter development on Effective Dart

Goodbye!

Alvaro Vasconcelos
F-TEAM

linkedin.com/in/vasconcelosdev/

Links

https://owasp.org/www-project-mobile-top-10/
https://support.citrix.com/article/CTX214006/

https://www.abine.com/blog/2014/tinder-app-vulnerability/
https://nakedsecurity.sophos.com/2018/11/16/hacking-misafes-smartwatches-for-kids-is-childs-play/
https://hackerone.com/reports/202425
https://blog.appknox.com/major-bug-in-ola-app-can-make-you-either-rich-or-poor/
https://medium.com/@evstykas/remote-smart-car-hacking-with-just-a-phone-2fe7ca682162
https://thehackernews.com/2019/05/hack-whatsapp-vulnerability.html
https://nordicapis.com/how-pokemon-go-fans-hacked-em-all-and-how-to-prevent-similar-reverse-engineering/
https://www.wired.com/2017/04/obscure-app-flaw-creates-backdoors-millions-smartphones/