OWASP
Open Web Application
Security Project

Mobile Apps Security & Best Practices

OWASP Mobile Top 10

Eng. Mohammad Khreesha
Twitter : @banyrock
Facebook : @khreesha

OWASP Mobile Top 10

- **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: Improper Platform Usage

- Misuse of a platform feature or failure to use platform security controls.

- Might include:
  - android intents,
  - platform permissions,
  - misuse of TouchID,
  - misuse the Keychain,
  - misuse of other security controls.

Example: Citrix Worx apps

- What happens on background.
  - User opens a managed app protected by WorxPIN such as WorxMail;
  - Worx Home is launched and prompts for Touch ID;
  - Apple’s API validates if user’s fingerprint is valid;
  - Is valid, then push Worx PIN from device’s Keychain;
  - Worx Home puts Worx PIN on prompted field;
  - Worx Home validates Worx PIN;
  - If Worx PIN is valid, then allows user launch the app.

Continue..

- Vulnerability Explained : https://vimeo.com/167255641
- Exploiting the vulnerability.
  - Consider a client with micro VPN established, e.g., previously using WorxMail client;
  - Reboot the iOS device;
  - Authenticate with device’s passcode normally;
  - Open WorxMail app;
  - When prompted for Touch ID, press the Cancel button.
  - You will be prompted to input the Worx PIN. In this step, close the app by pressing twice on the device’s Home button and sliding the app window to the top.
  - Open WorxMail again. It will no longer require authentication at this point.

M2: Insecure Data Storage

- Covers insecure data storage and unintended data leakage.
- Might include:
  - wrong keychain accessibility option,
    - (f. ex. kSecAttrAccessibleWhenUnlocked vs. kSecAttrAccessibleAlways)
  - insufficient file data protection,
    - (f. ex. NSFileProtectionNone vs NSFileProtectionComplete)
  - access to privacy resources when using this data incorrectly.

Example: Tinder

- 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.

- References :
  - https://www.youtube.com/watch?v=3E2DwdS_PvQ
  - https://techcrunch.com/2014/02/20/problem-in-tinder-dating-app-leaked-user-locations/

M3: Insecure Communication

- Might include:
  - poor handshaking/weak negotiation,
    - (f. ex. lack of certificate pinning)
  - incorrect SSL versions,
  - cleartext communication of sensitive assets,
  - HTTP instead of HTTPS.

Example: Misafe smart watches

- Reference : **https://www.pentestpartners.com/security-blog/tracking-and-snooping-on-a-million-kids/**
- Communication was not encrypted and not correctly authenticated.
- Attackers could:
  - retrieve real-time GPS coordinates of the kids' watches,
  - call the child on their watch,
  - create a covert one-way audio call, spying on the child,
  - send audio messages to the child on the watch, bypassing the approved caller list,
  - retrieve a photo of the child, plus their name, date of birth, gender, weight and height.

M4: Insecure Authentication

- Problems authenticating the end user or bad session management.

- Might include:
  - failing to identify the user at all when that should be required,
  - failure to maintain the user's identity when it is required,
  - weaknesses in session management.

Example: Grab Android app

- Reference : https://hackerone.com/reports/202425

- 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.

M5: Insufficient Cryptography

- Cryptography was attemted, but insufficient in some way.

- For example developer might have used an outdated cryptographic algorithm or written a custom vulnerable algorithm.

Example: Ola app

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

- The security researchers 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.

M6: Insecure Authorization

- Might include:
  - failures in authorization,
    - (e.g., authorization decisions in the client side, forced browsing, etc.)
  - able to execute over-privileged functionality.
  - It is distinct from authentication issues (e.g., device enrolment, user identification, etc.).

Example: Viper smart start

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

- 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.

M7: Client Code Quality

- Catch-all for code-level implementation problems in the mobile client.
- Might include:
  - buffer overflows,
  - format string vulnerabilities,
  - various other code-level mistakes where the solution is to rewrite some code that’s running on the mobile device.

Example: WhatsApp

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

- 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.

M8: Code Tampering

- Typically, an attacker will exploit code modification via malicious forms of the apps hosted in third-party app stores.

- The attacker may also trick the user into installing the app via phishing attacks.

- Might include:
  - binary patching,
  - local resource modification,
  - method hooking and swizzling,
  - dynamic memory modification.

Example: Pokemon GO

- References :
  - https://nordicapis.com/how-pokemon-go-fans-hacked-em-all-and-how-to-prevent-similar-reverse-engineering/
  - https://www.youtube.com/watch?v=uSocrLMjyLM

- 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.

M9: Reverse Engineering

- Might include analysis of the final core binary to determine its :
  - source code,
  - Libraries,
  - algorithms and other assets.

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

M10: Extraneous Functionality

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

- Might include:
  - hidden backdoor functionality,
  - other internal development security controls not intended for production environment.

Example: Wifi File Transfer

- References :
  - https://www.wired.com/2017/04/obscure-app-flaw-creates-backdoors-millions-smartphones/
  - https://www.youtube.com/watch?v=9YIE-XywpOA

- 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.

Q&A