Study Guide

Mobile App Security

Created By: Nafeel Ahmed, Teaching Assistant

Module 1: Mobile App Security

Lesson 1.1: Course Introduction

Skills Learned From This Lesson: Web App, Mobile App, Standards

- Course materials:
  - Inside the Mobile Attack Surface.
  - The Manager's guide to the OWASP Mobile Security project.
  - OWASP Mobile Security Testing Guide(MSTG).
  - OWASP Mobile App Security Verification Standards(MASVS).
- Learning Objective:
  - Prevalence of mobile app security in the news.
  - Common vulnerabilities found.
  - Difference in web & mobile app security.
- 85% of Mobile Apps have Security Vulnerabilities.
- 49% of Mobile Apps Leak Personal Data to Violate GDPR.
- 73% of Android Apps Leak Sensitive Data to System logs.
- 18% of iOS Apps Leak Sensitive Data to System Logs.
- 1 in 5 android apps use insecure HTTP instead of HTTPS.
- 1 in 7 iOS apps use insecure HTTP instead of HTTPS.
- Difference between Mobile and Web Apps:
  - OWASP has two different lists for the Mobile & WEB
  - Each of them have different vulnerabilities and issues.
- Fundamental difference of Web Apps includes:
  - Majority of code on server behind firewall and other protections.
  - Browser handles SSL/HTTPS encryption.
  - Browser Isolates data from local machines and files.
  - Web Apps are Isolated from one another.
- Fundamental difference of Mobile Apps includes:
  - Substantial code, IP logic, data on the client device.

    - Mobile apps must properly code network calls.
    - Mobile apps must properly handle local files and memory.
    - Mobile apps are capable of sharing data between one another
- To become a Mobile App Security specialists we have to learn the following:
    - Learn the methodologies.
    - Discover the tools.
    - Find an apps security baseline.
    - Rely on the industry guidelines.

Lesson 1.2: Course Introduction Quiz
Skills Learned From This Lesson: Security, Vulnerability, Standards
- 49% of mobile apps violate GDPR
- OWASP's top 3 mobile threats are as follows:
    - Improper Platform Usage.
    - Insecure Data Storage.
    - Insecure Communication.
- 1 in 7 iOS apps communicate over HTTP.

Lesson 1.3: The Mobile Landscape
Skills Learned From This Lesson: Static Testing, Dynamic testing, MobileOS
- The main mobile OS includes:
    - Android.
    - iOS.
    - iPadOS(Coming soon).
- Native languages include:
    - Java.
    - Kotlin.
    - NDK(C/C++).
    - Objective C.
    - Swift.
- Hybrid languages include:
    - Xamarin.
    - Flutter.
    - React-Native.

  - Cordova.
  - Ionic.
- Static and Dynamic testing are both needed to ensure application security.
- Static testing includes:
  - Looking for Source or Binary code.
  - No Configuration, less accuracy.
  - Coding bad practices.
- Dynamic testing includes:
  - Requires testing on a real device.
  - Real world scenario.
  - Configuration is there, more accurate.

Lesson 1.4: The Mobile Landscape Quiz

Skills Learned From This Lesson: Static Testing, Dynamic testing, MobileOS

- A real device is a requirement for dynamic testing.
- Dynamic testing should be used for networking and forensic functionality.
- The three native code languages for mobile app development are:
  - Java.
  - Kotlin.
  - NDK(C/C++).

Lesson 1.5: DeviceSec vs. AppSec

Skills Learned From This Lesson: Device Security, App Security

- Devices which install an app includes:
  - Old Device.
  - Not updated.
  - Rooted or Jailbroken.
  - Various other problems.
- Phones can have malicious apps like:
  - Fake VPNs.
  - Market Data Apps.
  - Malicious malwares.
- Biggest challenge to MobileSec is where that device connects:
  - Insecure WiFi.

- Malicious USB Chargers.
- Dangerous peripherals.

- The person who uses that device is also important, it can be anyone:
  - reverse engineers
  - Curious kids.
  - Unsuspecting users.
  - Real criminals.
- Privileged access is a big problem as it gives access to:
  - Private folders.
  - Keychain.
  - TLS bypass.
  - Process Memory.
  - Debug runtime.
  - Instrumentation.
-

Lesson 1.6: DeviceSec vs. AppSec Quiz
Skills Learned From This Lesson: Device Security, App Security

- We should avoid emulators when getting a testing device.
- A malicious app is an example of apps a user might have installed on their device.
- Difference goals, different skills and different variables are the main difference between AppSec and DeviceSec.

Lesson 1.7: Building Baselines Part 1
Skills Learned From This Lesson: Device Security, App Security

- Learning objective:
  - How do we establish security baselines?
  - OWASP as a resource.
  - How to use the MASVS.
  - The Mobile AppSec Model.
- How do we prioritize issues?
  - This issue leads us to Baseline Drivers which include:
    - Standards.
    - Data Privacy.

        - Compliance.
        - Intellecutal Property.
        - BUsiness continuity.
        - Reputation.
- The Mobile Security Project was started in 2017. It includes:
    - Mobile Top 10 (Vulnerabilities).
    - Mobile App Security Verification Standards (MASVS).
    - Mobile Security Testing Guide (MSTG).
    - Mobile Security Testing Checklist.
- 8 Domains of MASVS:
    - V1: Architecture, Design and Threat Modeling Requirements.
    - V2: Data Storage and Privacy Requirements.
    - V3: Cryptography Requirements.
    - V4: Authentication and Session Management Requirements.
    - V5: Network Communication Requirements.
    - V6: Environmental Interaction Requirements.
    - V7: Code Quality and Build Setting Requirements.
    - V8: Resiliency Against Reverse Engineering Requirements.
- Difference between L1, L2 and R are as follows:
    - L1 expects standard security best practices to be met.
    - L2 expects use of Defense-in-Depth
        - Hardened against "Lost Device" scenario.
        - Certificate Pinning.
        - Multi-factor authentication.
        - In house policy for Architecture and Risk controls.
    - R - Reverse Engineering Resilience
        - Device Binding.
        - Obfuscation.
        - Anti-Tamper.
        - Not meant to compensate for poor security.

Lesson 1.8: Building Baselines Part 2

Skills Learned From This Lesson: Device Security, App Security

- There are things all apps must do, and some that makes sense to do.

- MASVS L1:
  - This is standard Security. It has following features:
    - It is minimum.
    - No compliance or regulatory needs.
    - Simple apps.
- MASVS L2:
  - This is Defense-in-Depth. IT includes following features:
    - Regulated industry data.
    - Compliance consideration.
    - Apps that perform simple tasks, but handled higly sensitive data
- MASVS L1 +R:
  - This is Standard Security + High RE Resilience. It includes following features:
    - Prioritize IP Protection
    - Prevent Malicious modification or tampering.
- MASVS L2 +R:
  - This is defense-in-Depth + High RE Resilience. It includes following features:
    - Apps that perform complex activities between users and handle high sensitive data.
    - Compliance and IP protection are key
    - Preventing malware based attacks is in your threat model.
- Determining these three things help us identify where we fall:
  - Purpose : What does the app do?
  - Data: What data does this app handle?
  - Perspective: Do we build the app or do we use it? What do we care about?

Lesson 1.9: Building Baselines Quiz
Skills Learned From This Lesson: Device Security, App Security

- The 6th domain in the Mobile App Security Verification Standard is Environmental Interaction Requirements.
- The different categories in the Mobile AppSec Model are as follows:
  - L1
  - L2
  - L1 + R, L2+3
- A hospital communication app fall under L2+R category under Mobile AppSec Model.

Lesson 1.10: Tools for Testers Part 1

Skills Learned From This Lesson: Forensic Testing, Reverse Engineering, App Security

- Learning Objectives include:
    - Common tools.
    - Network testing methodology.
    - Forensic testing methodologies.
    - Code Quality testing methodology.
- Tools of the trade include:
    - Terminal of choice.
    - Jailbroken/rooted iOS and Android devices.
    - Network interception tools.
    - Developer tools.
    - Reverse engineering tools.
    - Patience, creativity, and attention to detail.
- Testing Network Interactions:
    - Use of different MiTM environments.
        - Different types of certs.
    - Test before and after login process.
        - Be prepared to launch the proxy during different stages.
    - Exercise the entire app.
        - Third party API or other content
    - Look for sensitive data and interesting content types.
        - Less work when testing the web API.
- Testing Forensics/DATA-AT-REST:
    - We have to search for sensitive values.
        - Rainbow tables help.
        - Regex and grep.
    - Jailbroken/Rooted devices.
        - Great for testing private folders and keychain.
        - Not necessary for backups, logs and SDCard.
    - Exercise the app.
        - Different data before and after logout.
- Users, storage files like Logs are really helpful in exploring the vulnerabilities .

Lesson 1.11: Tools for Testers Part 2

Skills Learned From This Lesson: Forensic Testing, Reverse Engineering, App Security

- Testing Code Quality:
    - It should be a Block-Box testing approach.
- Reverse Engineering should include the following:
    - Zip Files -> AndroidManifext.xml, INfo.plist.
    - Disassemblers/Decompilers(Radare2, apktool, procyon).
    - Dynamic binary instrumentation(Frida).
    - Developer tool(Android Studio, Xcode).
    - Source code analysis .
    - “Strings” and “grep” are also great tools to use.
- Checklist:
    - Don’t trust the app.
    - Get test devices and the tools.
    - Read MASVS.
    - Pick a section of MSTG to learn.

Lesson 1.12: Tools for Testers Quiz

Skills Learned From This Lesson: Forensic Testing, Reverse Engineering, App Security

- The app must be exercised. This is a common that network and forensic testing share.
- Executable code to disassembled code(assembly) is the equivalent in iOS compared to Dex to Smali in Android.
- Frida and radare are open source tools useful for reverse engineering.