Lecture #1

Securing Mobile Applications and IoT

FALL 2025

Mobile Security: Threats, Defenses, and Modern Realities

- An Overview
- Questions
- Semester Plan

Before We Begin: Who's in the Room?

- What is your general programming comfort level?
  - **Beginner**: I'm new to programming or have only taken introductory courses.
  - **Intermediate**: I'm comfortable with programming concepts and have built a few small projects.
  - **Advanced**: I have significant experience and have worked on large or complex software projects.

Mobile Development Background

- Have you ever built a mobile application?
  - **Never**: This is my first time exploring mobile development.
  - **A Little**: I've followed tutorials or built simple sample apps (e.g., a "To-Do" list).
  - **Yes**: I have designed and built one or more complete mobile applications from scratch.

Familiar Technologies

- Which mobile technologies have you used or are you interested in?
  - **Cross-Platform**:
    - JavaScript / TypeScript (React Native)
    - Dart (Flutter)
  - **Native**:
    - Kotlin / Java (Android)
    - Swift / Objective-C (iOS)
  - **Other / None of the above**

Introduction to Mobile Security

- A Quick Story:
  - Have you ever received a text like this?
    - "Your package has a delivery issue. Please click here to update your shipping details.
- This is a common scam called "Smishing," and it's one of the simplest and most effective ways attackers target us through our phones.

Why Mobile is Different

- The Unique Challenges of Mobile
  - Constant Connectivity: Always on (Wi-Fi, 5G/LTE), creating a constant pathway for potential attacks.
  - Personal Data Consolidation: One device holds your photos, messages, financial apps, health data, and more.
  - Diverse App Ecosystems: Openness (Google Play) vs. a "walled garden" (Apple App Store) create different risk profiles.
  - Physical Portability: High risk of being lost or stolen.

Key Security Terminology

- The Language of Security

  - **Asset**: Anything of value that needs to be protected. (e.g., your personal data, access to your bank account)
  - **Vulnerability**: A weakness in a system that can be exploited. (e.g., an outdated OS, a weak password)
  - **Threat**: A potential event or attacker that could harm an asset. (e.g., a thief, a piece of malware)
  - **Attack Vector**: The path or means by which a threat gains access to exploit a vulnerability. (e.g., a phishing email, a malicious QR code)
  - **Attack Surface**: The sum of all possible attack vectors; all the potential entry points for an attack.
  - **Risk**: The likelihood of a threat exploiting a vulnerability and the resulting impact.

The Modern "Attack Surface"

- More Than Just a Phone
  - Your phone is a key that unlocks your entire digital world.
    - Cloud Services (iCloud, Google Drive)
    - Corporate Networks (BYOD - Bring Your Own Device)
    - IoT Devices (Smart Home, Connected Car)

Core Security Principles (The CIA Triad)

- The Goal: A Secure Digital Safe
  - The fundamental goal of all security is to protect three things, known as the **CIA Triad**.
    - **C**onfidentiality (The Secret)
    - **I**ntegrity (The Unchanged)
    - **A**vailability (The Accessible)

Confidentiality

- **Confidentiality (The Secret)**
- Keeping your data private and unreadable to unauthorized parties.
- Mobile Examples:
  - Biometrics: Face ID / Fingerprint Scanners
  - Encryption: End-to-end encryption in apps like Signal or WhatsApp.

Integrity

- **Integrity (The Unchanged)**
- Ensuring data has not been tampered with or altered.

- Mobile Example:
  - App Store Verification: When you download an app, your phone's OS checks its digital signature. This guarantees it's the authentic version from the developer and not a malicious copy injected with malware.

Availability

- **Availability (The Accessible)**
- Making sure you can access your data and services when you need them.

- Mobile Example:
  - Denial-of-Service (DoS) Protection: A mobile banking app needs to be protected from attacks that could flood its servers with traffic, preventing legitimate users from logging in to check their balance or make a payment.

The Threat Landscape

A Taxonomy of Mobile Threats

- **How They Attack**

  - **Malware:** Malicious Software
  - **Phishing:** Social Engineering
  - **Network Attacks:** Exploiting Connections
  - **Physical Theft & Loss:** The Oldest Threat

Threat: Malware

- **Malicious Software (Malware)**
    - **Spyware:** Secretly gathers information from your phone (e.g., Pegasus spyware).
    - **Ransomware:** Locks your device or encrypts your files until a ransom is paid.
    - **Adware/Scareware:** Deceptive pop-ups and aggressive, malicious advertising that tricks you into taking an action.

Threat: Phishing

- **Social Engineering (Phishing)**

  - **SMS Phishing (Smishing):** Uses fake texts with malicious links.
    - "Your package has a delivery issue, click here..."
  - **QR Code Phishing (Quishing):** Malicious QR codes in public spaces (e.g., on a parking meter or restaurant menu) that lead to fake login pages.

Threat: Network Attacks

- **Exploiting Connections**

  - **Man-in-the-Middle (MitM):** An attacker on a public Wi-Fi network intercepts your traffic to steal data.
    - Analogy: A postal worker secretly opening and reading your mail before delivering it.
  - **Rogue Access Points:** Fake Wi-Fi hotspots with familiar names (e.g., "Free_Airport_Wi-Fi") set up by attackers to capture all traffic that connects to it.

Threat: Physical Theft & Loss

- **The Most Straightforward Threat**
  - If an attacker has physical possession of your device, the game changes entirely.
  - This is the first line of defense. If the device isn't locked, it's "game over."

Case Studies

- **Learning from Major Breaches**
  - Let's see how these threats manifest in the real world.
    - **Pegasus Spyware**
    - **Stagefright Android Vulnerability**

Case Study 1: The Pegasus Spyware

- **Threat:** Sophisticated Spyware (Malware)
- **Vulnerability:** Exploited "zero-click" vulnerabilities in iOS and Android. This means no user interaction was needed—no link to click, no app to download.
  - **Impact:** Attackers gained complete control of the device:
    - Access to microphone, camera, messages, location data.
    - A total violation of **Confidentiality** and **Integrity**.

Case Study 2: The Stagefright Android Vulnerability

- **Threat:** Remote Code Execution
- **Vulnerability:** A flaw in the Android OS's media processing library ("Stagefright").
  - **Impact:** An attacker could send a specially crafted MMS (multimedia message). The phone would process the malicious media file automatically, **before you even opened the message**, potentially allowing the attacker to take control of the device.
    - **Lesson:** Highlighted the critical need for timely OS updates.

Analysis and Defense

Understanding Mobile Vulnerabilities

- **Where are the Weaknesses?**
  - **OS-Level:** Flaws in Android or iOS itself. (e.g., Stagefright). This is why you MUST keep your OS updated.
  - **Application-Level:** Insecure code within an app you install.
  - **Network-Level:** Insecure data transmission over the internet.

Application-Level Vulnerabilities

- **Weaknesses Inside the App**
  - **Insecure Data Storage:** Apps storing sensitive info like passwords or tokens in plain text on the device.
  - **Excessive Permissions:** A simple calculator app asking for access to your contacts, camera, and location is a major red flag.
  - **Hardcoded Secrets:** Developers accidentally leaving API keys, passwords, or other credentials directly in the app's source code.

Network-Level Vulnerabilities

- **Weaknesses on the Network**
  - **Unencrypted Communication:** Apps that transmit sensitive data over the internet using **http** instead of **https**.
    - This makes the data readable to anyone performing a Man-in-the-Middle (MitM) attack on public Wi-Fi.

Frameworks for Mobile Risk Assessment

- **A Structured Way to Think About Risk**
  - Security can feel overwhelming. A simple risk assessment process helps you focus your efforts where they matter most.

The Four-Step Process

- **The 4-Step Risk Process**
  - **1. Identify Assets:** What are you trying to protect?
  - **2. Identify Threats & Vulnerabilities:** What could go wrong?
  - **3. Analyze Risk:** How likely is it, and how bad would it be?
  - **4. Treat the Risk:** What are you going to do about it?

Risk Example: Steps 1 & 2

- **Let's Walk Through an Example**
- **1. Identify Asset:**
  - Access to my banking app and corporate email on my phone.
- **2. Identify Threats & Vulnerabilities:**
  - **Threat:** Losing my phone or having it stolen.
  - **Vulnerability:** I currently have a weak, easy-to-guess passcode ("1234").

Risk Example: Steps 3 & 4

- **Example Continued**
- **3. Analyze Risk:**
  - **Likelihood:** Moderate. People lose phones all the time.
  - **Impact:** Severe. An attacker could access my bank account and sensitive work data.
- **4. Treat the Risk (Mitigate):**
  - Set a strong, alphanumeric passcode.
  - Enable biometric authentication (Face ID / Fingerprint).
  - Activate "Find My Device" to enable remote wipe capabilities.

Key Takeaways & Q&A

- **Key Takeaways**
  - **Mobile is a Unique Target:** Its portability, connectivity, and data density create special challenges.
  - **Threats are Diverse:** From malware and phishing to network attacks and physical theft.
  - **Defense is Layered:** Protect the OS (updates), the apps (permissions), the network (HTTPS), and the device itself (passcode).
  - **Think in Terms of Risk:** Use a simple framework (Identify, Analyze, Treat) to make smart security decisions.

Questions?

Semester Project

Assignment 1: Project Proposal & Application Blueprint

- **The Master Plan for Your Application**
  - **Goal:** Create a detailed proposal for the mobile app you will build this semester.
  - **Key Sections:**
    - **Application Concept:** What is your app? Who is it for? What value does it provide?
    - **Core Data Model:** Define your main data entities and their attributes (e.g., User, Task).
    - **User Roles & Features:** Define at least two user roles (e.g., Admin, User) and list their features.
    - **Offline & Sync Strategy:** How will the app work offline? How will it sync data later?
    - **UI/UX Mockups:** Simple wireframes for the main list and add/edit screens.
  - **Submission:** A single PDF document.
  - **Deadline:** See course schedule: `https://www.cs.ubbcluj.ro/~dan/sma/labPlan.html`

Assignment 2: Mobile Authentication & Onboarding UI/UX

- **Building the Front Door to Your App**
  - **Goal:** Design and implement a high-fidelity, **UI-only** prototype for user authentication.
  - **Key Sections:**
    - Polished Login/Signup screen.
    - UI for multiple auth methods: **Google/Apple, Email/Password, Anonymous**.
    - Mock UI states for feedback (e.g., input validation, error messages, loading spinners).
    - UI flow for **Biometric Login** (Face ID/Fingerprint) with a password fallback.
    - Obfuscate the code.
  - **Submission:** Link to your GitHub Classroom repository, containing the code.
  - **Deadline:** See course schedule: `https://www.cs.ubbcluj.ro/~dan/sma/labPlan.html`

Assignment 3: Local Database & Role-Based Features

- **Bringing Your App to Life on the Device**
  - **Goal:** Convert your UI prototype into a functional offline app with a local database.
  - **Key Sections:**
    - Integrate a **local database** (like SQLite, Realm, etc.).
    - Implement full, working authentication that saves data locally.
    - Implement your two **user roles** with their unique and shared features.
    - Securely store sensitive data (e.g., using encrypted storage).
    - Implement **biometric re-authentication** to unlock the app.
    - Use a clean architecture (e.g., MVVM, Repository Pattern).
  - **Submission:** Link to your GitHub Classroom repository, containing the code.
  - **Deadline:** See course schedule: `https://www.cs.ubbcluj.ro/~dan/sma/labPlan.html`

Assignment 4: Full-Stack Mobile Application

- **Connecting to the Cloud**
  - **Goal:** Transform your app into a full-stack solution with a remote backend.
  - **Key Sections:**
    - Connect to a **remote backend server** (REST or GraphQL).
    - Implement **remote authentication** (e.g., using JWTs).
    - **Synchronize data** between the app and the server.
    - Use the local database as a **cache for offline support**.
    - Gracefully handle network states (loading indicators, error messages).
    - Extend your architecture to manage remote and local data sources.
  - **Submission:** Link to your GitHub Classroom repository, containing the code.
  - **Deadline:** See course schedule: `https://www.cs.ubbcluj.ro/~dan/sma/labPlan.html`

Late Submission Penalties

- **Example for a due date of November 10th:**
  - Last commit by November 10th, 00:00 (GMT+2) for full credit.
  - After November 10th: 25% penalty.
  - After November 17th: 50% penalty.
  - After November 24th: 75% penalty.
  - After December 1st: 100% penalty.

Rules

- Deadlines are final! All groups share the same deadlines.
- Ensure your last commit is made before the deadline!
- A 25% penalty applies per **calendar week** for late submissions.
- To receive a grade, present results to the seminar.
- Upload sources on GitHub, but simply submitting is not enough.
- Learn to use Git and GitHub from your IDE (e.g., Android Studio, Xcode, Visual Studio Code).

Evaluation

- **Project Grade (PG)** = Project Proposal (2p) + UI (2p) + DB (3p) + Full-Stack (3p).
- **Exam Grade (EG)** - the grade from the written exam.
- **Final Grade (FG)** = (PG >= 4.5 && EG >= 5) ? (PG * 0.6 + EG * 0.4) : 0.
- To attend the normal session exam, you must have at least 75% attendance in seminars.
- If **PG** < 4.5 or attendance criteria are not met, you can only attend the written exam in the re-examination session.

Evaluation - No Exam

- Present the work in advance, everything before the **Christmas Holiday.**
- **Project Grade (PG)** = Project Proposal (2p) + UI (2p) + DB (3p) + Full-Stack (3p).
- ~~**Exam Grade (EG)** - the grade from the written exam.~~
- **Final Grade (FG)** = **PG.**