Mobile Application Testing Lab Report

Part 1: Static Analysis using MobSF

Student Name: Priyank Kumar

Lab Title: Mobile Application Testing Lab

Tool Used: Mobile Security Framework (MobSF)

Platform: Android

1. Objective

The objective of this task is to perform **static security analysis** on an Android application package (APK) using **MobSF** in order to identify vulnerabilities related to **insecure data storage.** Static analysis helps in detecting security flaws in application source code and configurations without executing the application.

2. Tool Description – MobSF

MobSF (Mobile Security Framework) is an automated, all-in-one mobile application security testing framework capable of performing static and dynamic analysis on Android and iOS applications. It detects vulnerabilities such as insecure storage, weak cryptography, hardcoded secrets, and misconfigurations.

3. Test Environment

Component	Details
Attacker Machine	Kali Linux
Analysis Tool	MobSF (Docker-based)
Target Application	test.apk (InsecureBankv2)
Analysis Type	Static Analysis

4. Procedure

Step 1: Start MobSF

MobSF was executed using Docker and accessed through a web browser at:

http://127.0.0.1:8000

Step 2: Upload APK

The vulnerable Android application **test.apk** was uploaded using the **Upload & Analyze** feature in MobSF.

Step 3: Static Analysis Execution

MobSF automatically decompiled the APK and performed static analysis, generating a detailed security report highlighting multiple vulnerabilities.

5. Vulnerability Identified: Insecure Storage

Description

The application stores sensitive information insecurely using **SharedPreferences** and local storage without encryption. This allows attackers or malicious applications to access confidential data such as usernames, passwords, or session tokens.

Evidence

MobSF identified usage of insecure storage APIs such as:

- SharedPreferences with weak access control
- Sensitive data stored in plaintext

Example code snippet identified by MobSF:

SharedPreferences prefs = getSharedPreferences("user_data", MODE_PRIVATE);

6. Vulnerability Log

Test ID	Vulnerability	Severity	Target App
016	Insecure Storage	High	test.apk

7. Impact

An attacker with access to the device or a malicious application can extract sensitive user data from local storage. This may lead to information disclosure, account compromise, and further attacks.

8. Recommendation

- Avoid storing sensitive data in plaintext
- Use encrypted storage mechanisms (e.g., EncryptedSharedPreferences)
- Implement proper access control
- Follow OWASP Mobile Security Guidelines

9. Conclusion

The static analysis conducted using MobSF successfully identified a **high-severity insecure storage vulnerability** in the target application. This demonstrates the importance of secure data storage practices in mobile application development.