Android Application Penetration Testing Report

Internship Project 1 – Cyber Security Domain

Date: 31.08.2025

1. Introduction

The purpose of this project is to assess the security of Android applications using penetration testing techniques in a controlled lab environment. Mobile applications often manage sensitive data (credentials, banking details, tokens), which makes them a common target for attackers.

The assessment follows the OWASP Mobile Top 5 Risks (2024):

M1: Improper Platform Usage

M2: Insecure Data Storage

M3: Insecure Communication

M4: Insecure Authentication

M5: Insufficient Cryptography

The test application used is InsecureBankv2, a deliberately vulnerable Android app.

Objectives:

- Set up a mobile penetration testing lab.
- Identify, exploit, and document security flaws.
- Provide mitigation strategies to improve app security.

2. Lab Setup

Tools & Environment Used:

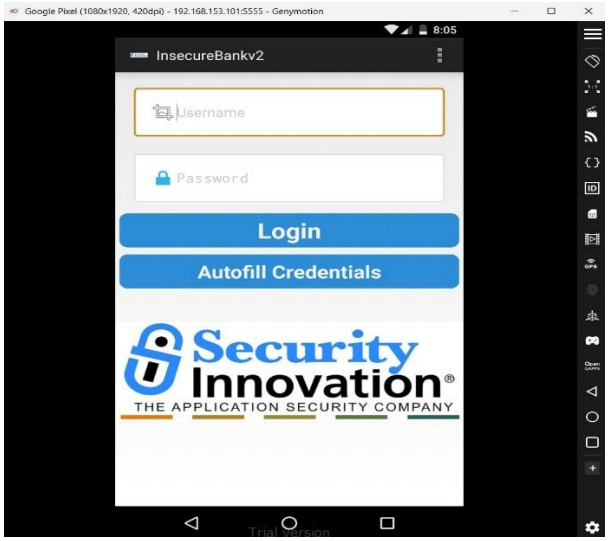
- Genymotion Emulator (Android Virtual Device)
- Kali Linux (Pentesting OS)

- Burp Suite (Intercepting proxy)
- Frida (Dynamic instrumentation)
- MobSF (Mobile Security Framework)
- Jadx / APKTool (Static analysis tools)

Steps Followed:

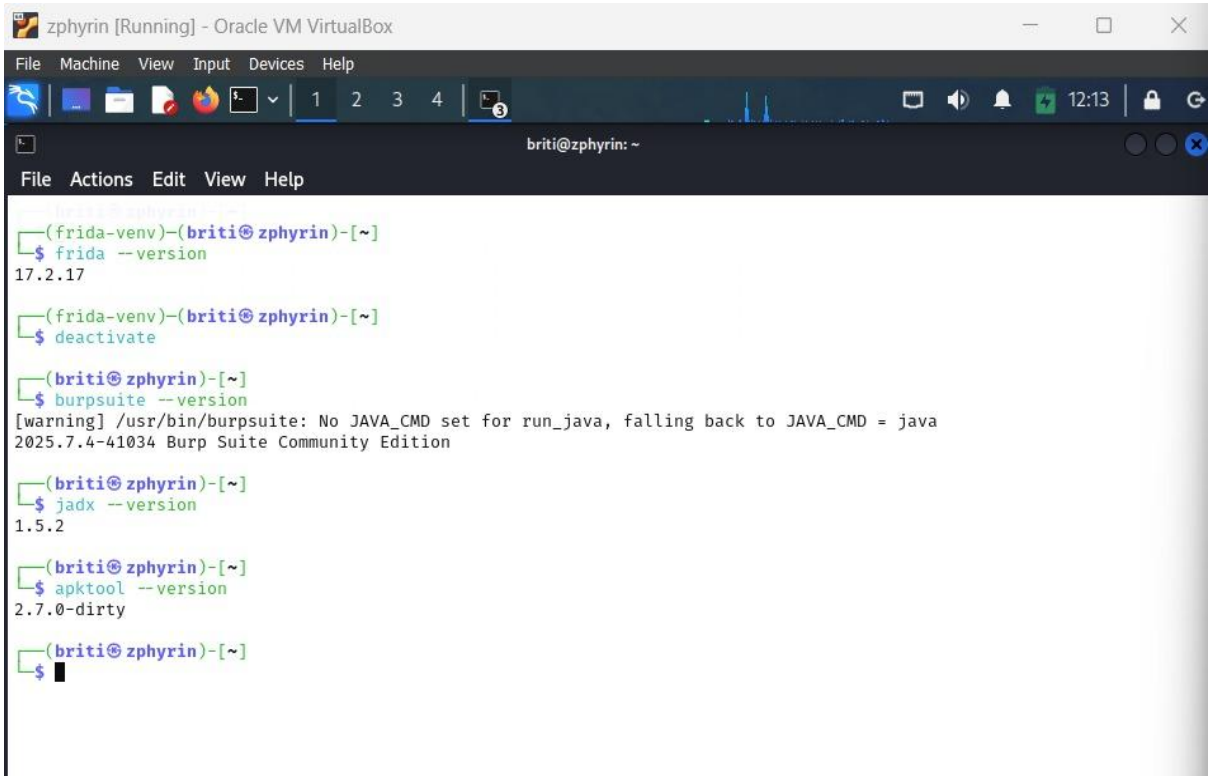
- Installed Genymotion & configured Android device.

- Deployed the vulnerable app (DVIA / InsecureBankv2) on emulator.



- Configured Burp Suite with emulator for HTTPS interception.
- Installed and configured MobSF for static/dynamic analysis.

3.Set up Jadx and APKTool for code review.



4.Vulnerability Testing (OWASP Top 5)

- **M1: Improper Platform Usage:**

Definition: Improper use of Android platform features or permissions, e.g., misuse of exported components, insecure permissions, or improper intent handling.

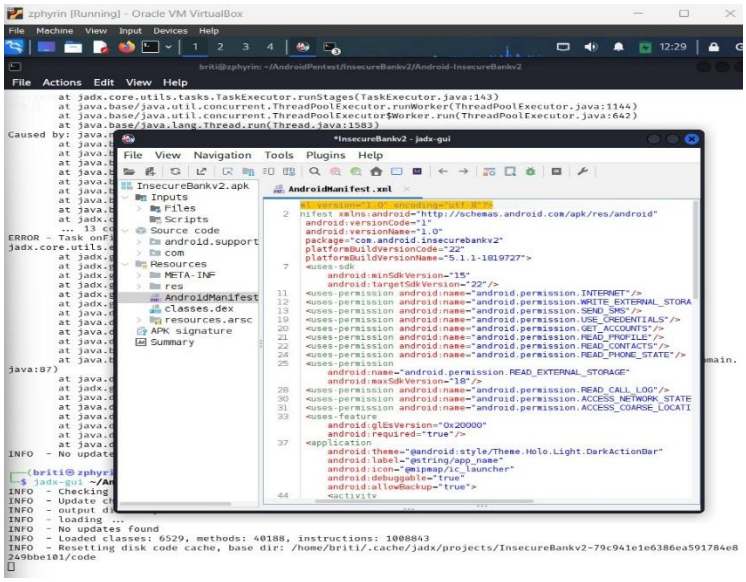
Steps Taken: Decompiled APK in Jadx, reviewed AndroidManifest.xml.

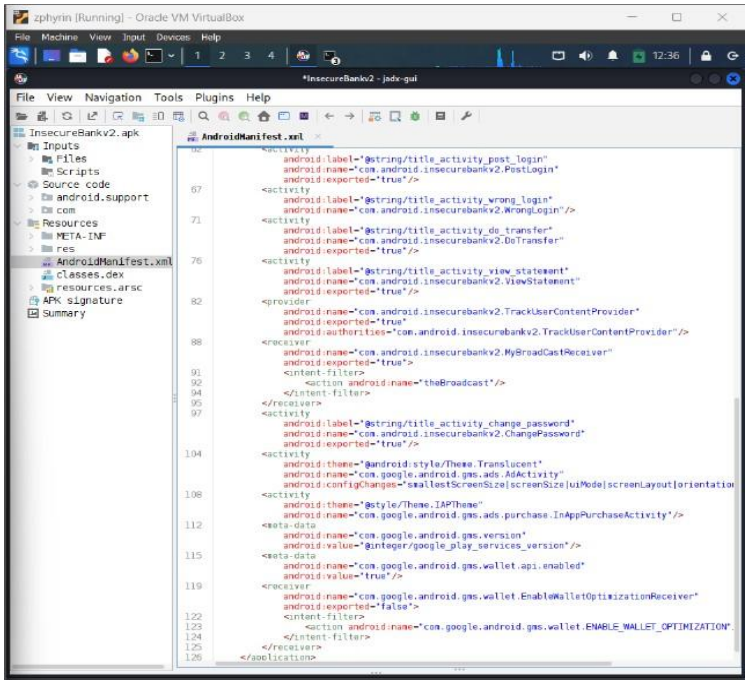
Observation: Exported activities, providers, or receivers can be accessed by other apps, which can lead to privilege escalation, data leaks, or bypassing authentication.

The app requests excessive permissions (READ_SMS, SEND_SMS, WRITE_EXTERNAL_STORAGE, READ_CALL_LOG, etc.).

Several components (Activities, Broadcast Receivers, Content Providers) are marked android:exported="true", allowing external apps to trigger them.

Evidence:





Impact:

A malicious app could interact with these exported components and misuse permissions for privilege escalation.

Tools Used: Jadx, APKTool

Remediation:

- Remove unnecessary permissions.
- Set android:exported="false" by default.
- Enforce signature-based permissions for sensitive components.

➢ **M2: Insecure Data Storage**

Definition: Sensitive data stored in plaintext on device (e.g., shared prefs, SQLite DB).

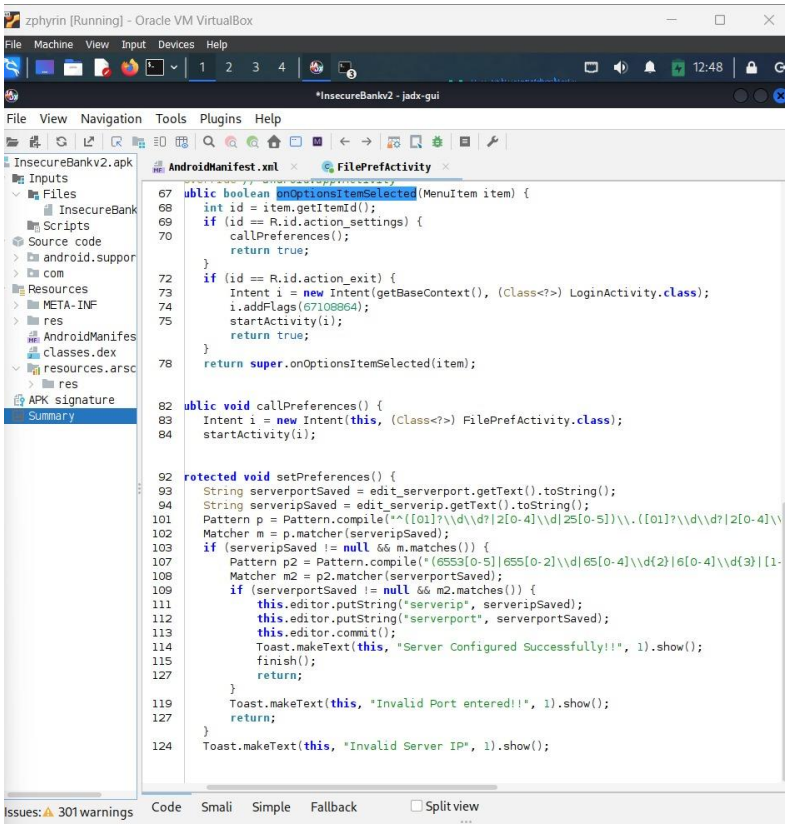
Steps Taken: Examined app's data directory using emulator shell.

Evidence: In FilePrefActivity.java, the setPreferences() method writes the values directly to SharedPreferences:

this.editor.putString("serverip", serveripSaved);

this.editor.putString("serverport", serverportSaved);

this.editor.commit();



Observation: Passwords stored in plaintext.

Impact:

An attacker with root or physical access could retrieve sensitive credentials.

Tool Used: MobSF, Jadx.

Recommendation:

Use EncryptedSharedPreferences or Android Keystore to store sensitive data.

Avoid storing sensitive configuration in plaintext.

- M3: Insecure Communication

Description:

The app makes network requests using HttpURLConnection without enforcing HTTPS. For example:

HttpURLConnection httpURLConnection = (HttpURLConnection) new URL(str).openConnection();

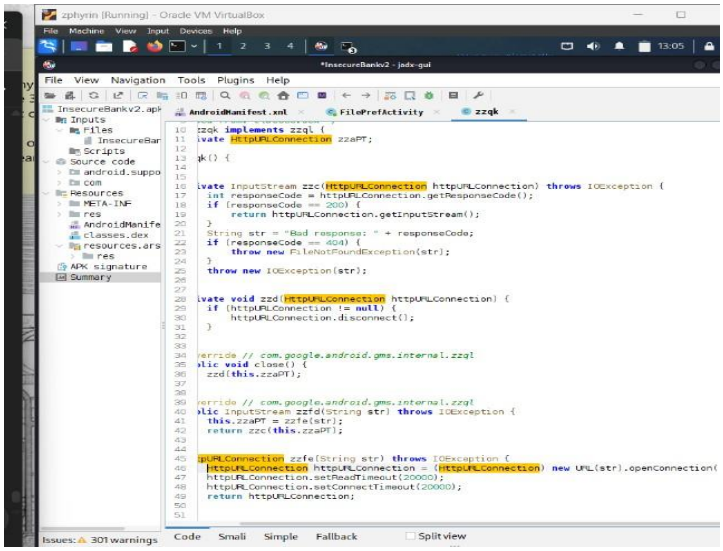
httpURLConnection.setReadTimeout(20000);

httpURLConnection.setConnectTimeout(20000);

Here, str could be any URL, including HTTP URLs.

Requests are sent in plaintext, so data can be intercepted or modified by an attacker

Steps Taken: Configured Burp Suite, intercepted login/API calls.



Impact:

Credentials, configuration, or transactions can be stolen or tampered with.

MITM (Man-in-the-Middle) attacks are possible.

Tool Used: Burp Suite, Frida.

Remediation: Enforce HTTPS, implement SSL pinning.

- M4: Insecure Authentication

Definition: Weak login/authentication that can be bypassed.

Steps Taken: Reviewed code for hardcoded credentials, tested brute force.

Evidence :

Several exported activities could allow other apps to access app functionality without authentication:

<activity android:name="com.android.insecurebankv2.PostLogin" android:exported="true"/>

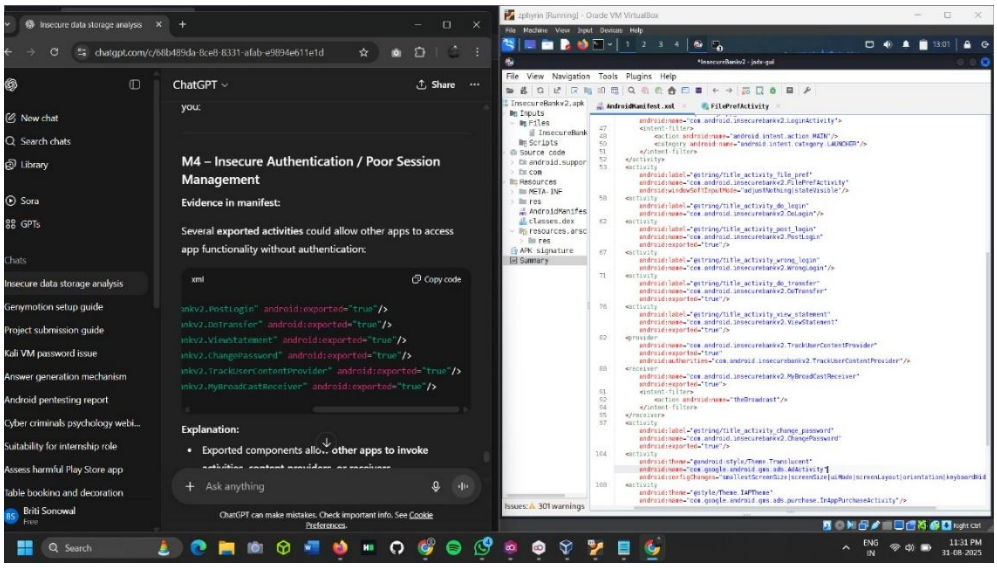
<activity android:name="com.android.insecurebankv2.DoTransfer" android:exported="true"/>

<activity android:name="com.android.insecurebankv2.ViewStatement" android:exported="true"/>

<activity android:name="com.android.insecurebankv2.ChangePassword" android:exported="true"/>

<provider android:name="com.android.insecurebankv2.TrackUserContentProvider" android:exported="true"/>

<receiver android:name="com.android.insecurebankv2.MyBroadCastReceiver" android:exported="true"/>



Observation: Hardcoded admin password found.

Impact:Attackers could brute force credentials or hijack sessions.

Tool Used: Jadx, MobSF.

Remediation: Use secure backend validation; remove hardcoded creds.

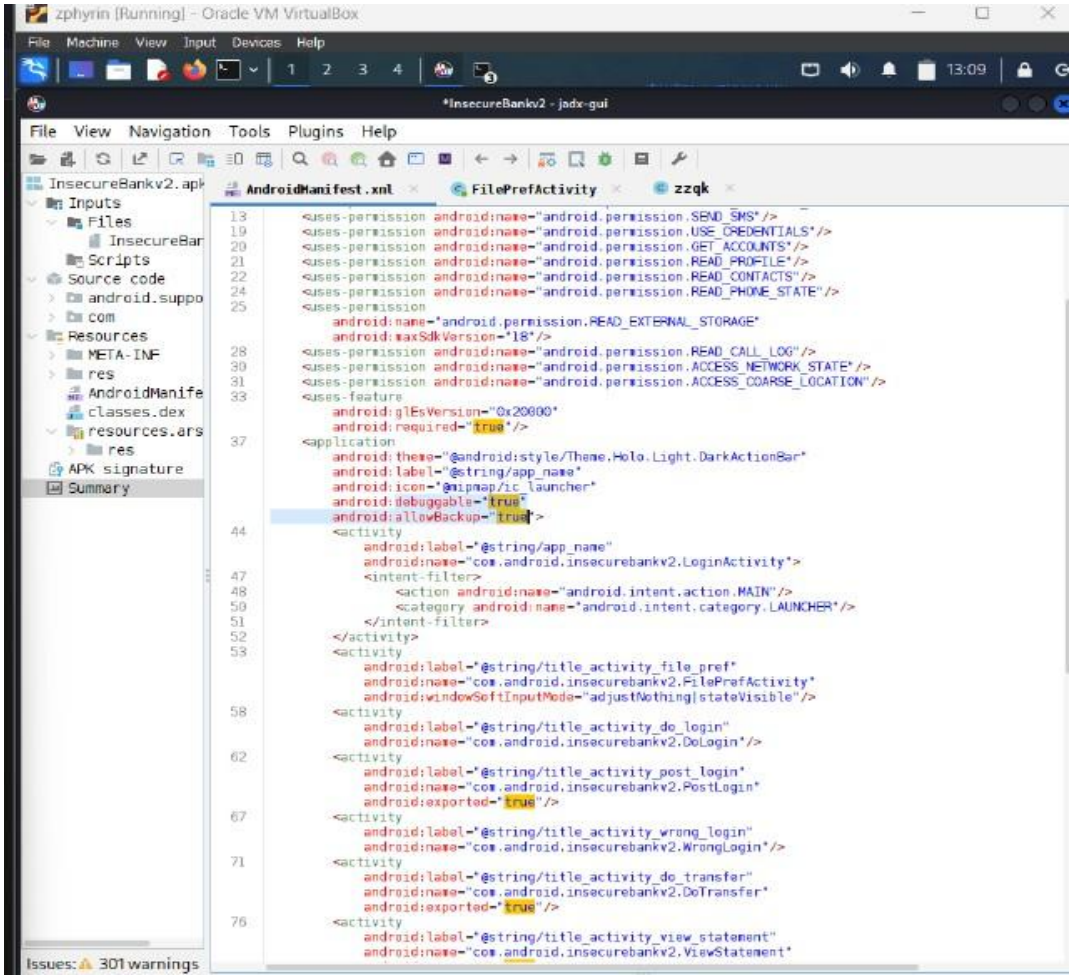
- M5: Insufficient Cryptography

Definition:

Sensitive data in the app may be exposed due to debuggable mode, exported components, and insecure logging. This allows attackers to extract credentials, transactions, or other private information.

Steps Taken: Analyzed crypto functions in code.

Evidence:



Impact:

- Data leakage can lead to identity theft or unauthorized transactions.
- Attackers can exploit debug mode or exported components to access sensitive resources.

Tool Used: Jadx, MobSF.

Recommendation:

- Set android:debuggable="false" before release.
- Limit exported components and enforce proper permissions.
- Remove all debug logs and secure sensitive data storage.

5.Summary table:

Vulnerability	Tool used	Evidence SS	Satus	Mitigation
M1: Improper Platform Usage	Jadx		Vulnerable	Restrict exported activities, enforce permissions
M2: Insecure Data Storage	MobSF		Vulnerable	Encrypt sensitive data, use Android Keystore
M3: Insecure Communication	Burp Suite		Vulnerable	Enforce HTTPS, implement SSL pinningM
M4:Insecure authentication	Jadx / MobSF		Vulnerable	Backend validation, remove hardcoded credentials
M5: Insufficient Cryptography	Jadx		Vulnerable	Use strong crypto (bcrypt, Argon2, PBKDF2)

6.Conclusion

This internship project provided **hands-on experience** in mobile application penetration testing. The testing process revealed how insecure coding practices in Android apps can lead to serious security flaws such as **data leaks, insecure authentication, and weak encryption**.

Key Learnings:

- Setting up and using a full Android pentesting lab.
- Identifying and exploiting OWASP Mobile Top 5 vulnerabilities.

- Understanding how insecure apps can compromise user security.

Limitations:

- Only OWASP Top 5 vulnerabilities tested.
- Exploits limited to lab environment.

Future Improvements:

- Extend testing to **full OWASP Mobile Top 10**.
- Automate Frida scripts for runtime manipulation.
- Perform testing on **real-world applications (with permission)**.

6. References

- OWASP Mobile Top 10: https://owasp.org/www-project-mobile-top-10/
- DVIA App: https://github.com/prateek147/DVIA
- InsecureBankv2: https://github.com/dineshshetty/Android-InsecureBankv2
- MobSF: https://github.com/MobSF/Mobile-Security-Framework-MobSF
- Jadx: https://github.com/skylot/jadx
- Burp Suite Documentation