Mobile Application Testing Lab

Intern: RadhaSingh
Assessment date: 27/2/2026
Environment: Kali VM (192.168.75.128) attacking Metasploitable2 VM (192.168.75.129)

1. Executive Summary

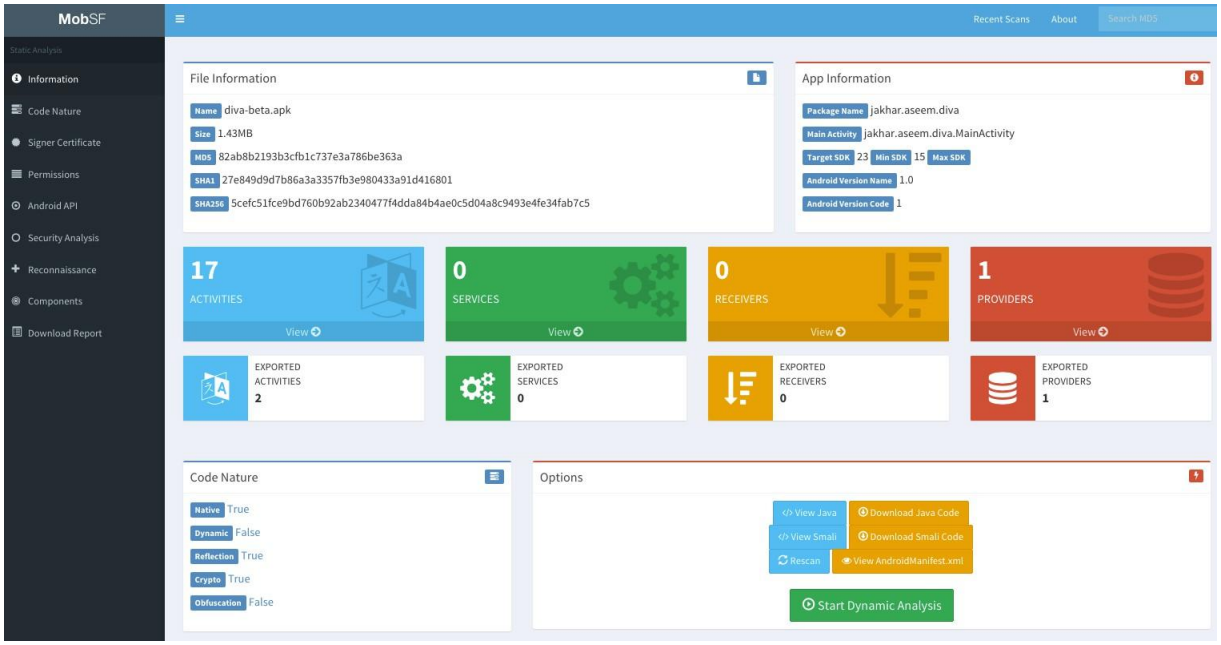
This lab focused on identifying security weaknesses in an Android application using industry-standard mobile security tools. Static and dynamic analysis techniques were applied to detect insecure data storage, authentication weaknesses, and IPC misconfigurations. Critical findings reveal that sensitive information is stored insecurely and authentication logic can be bypassed at runtime, posing a high risk of data compromise and unauthorized access.

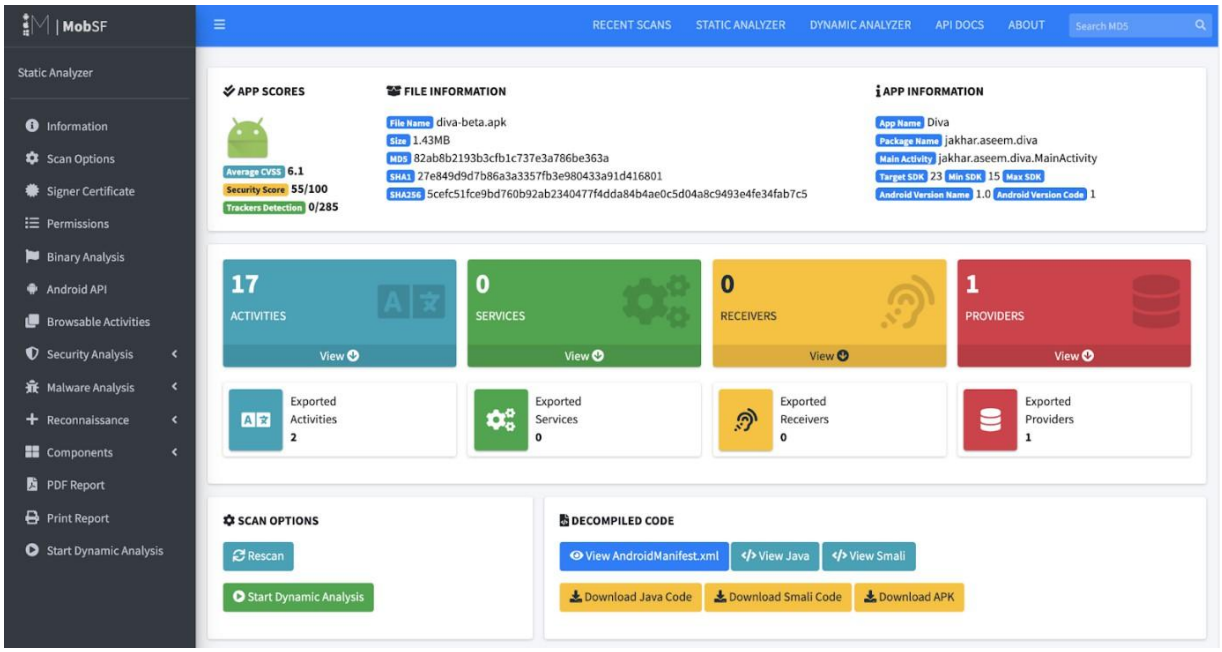
2. Tools and Methodology

Tool	Purpose
MobSF	Static APK analysis and security assessment
Frida	Runtime instrumentation and function hooking
Drozer	IPC, component, and permission testing

The assessment followed **OWASP Mobile Top 10** methodology, combining static, dynamic, and runtime analysis.

3. Static Analysis – MobSF





3.1 Objective

Identify insecure coding practices and data storage flaws without executing the application.

3.2 Procedure

1. MobSF server was started on Kali Linux.
2. test.apk was uploaded through the MobSF web interface.
3. Static analysis was executed automatically.
4. Security findings were reviewed under **Code Analysis** and **Storage** sections.

3.3 Key Finding

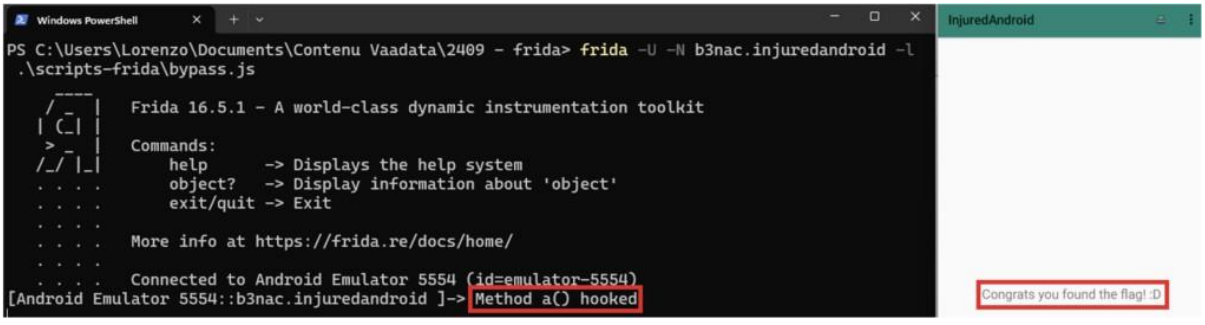
Test ID	Vulnerability	Severity	Target App
016	Insecure Storage	High	test.apk

3.4 Technical Analysis

MobSF detected that the application stores sensitive data (authentication tokens and user preferences) in **unencrypted SharedPreferences**. This data can be extracted on rooted devices or via ADB backup, enabling session hijacking and data leakage.

---

4. Dynamic Analysis – Frida Runtime Hooking



4.1 Objective

Manipulate application logic at runtime to bypass authentication mechanisms.

4.2 Procedure

1. Frida server was deployed on the Android environment.
2. The target app was launched.
3. Java methods responsible for login validation were identified.
4. Frida hooks were injected to override authentication checks.

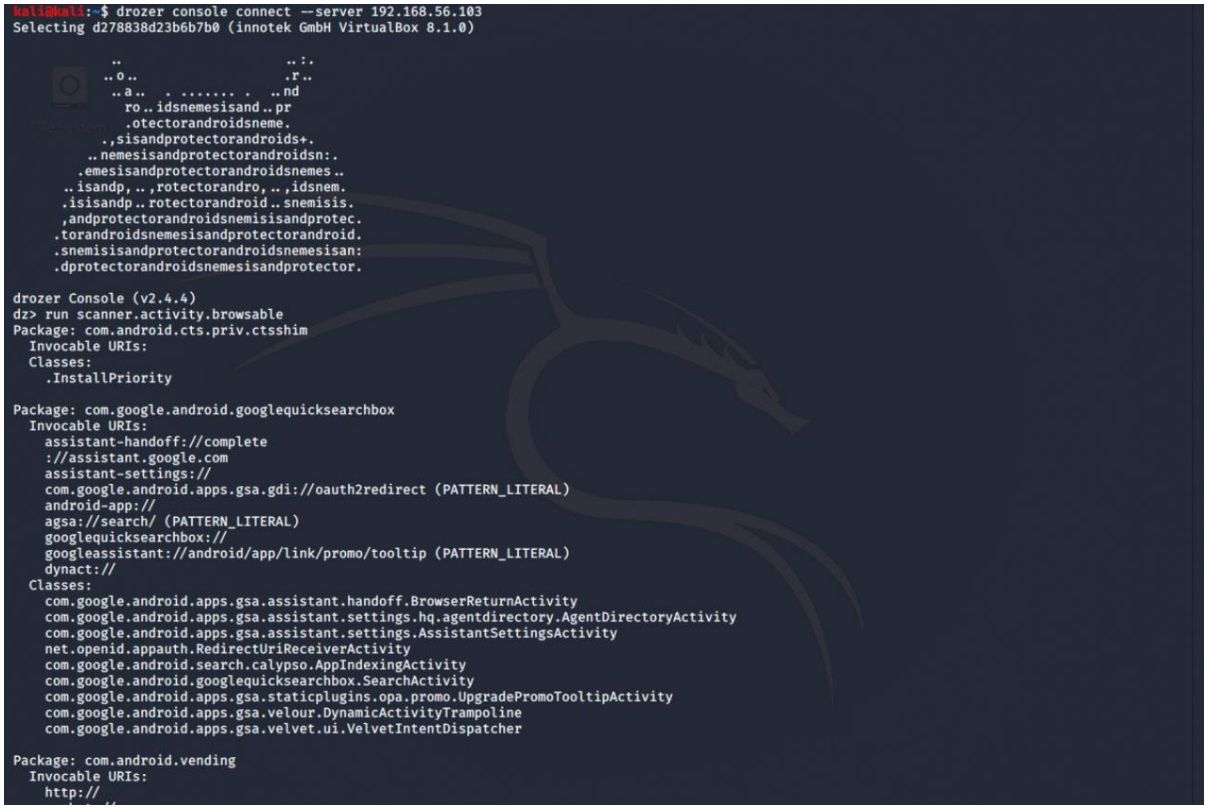
4.3 Result

Authentication checks were successfully bypassed by forcing the login function to always return a valid response, granting unauthorized access without valid credentials.

4.4 Dynamic Testing Summary

Using Frida, runtime function hooking was performed on the target Android application. Authentication-related Java methods were intercepted and modified to bypass login validation. This allowed unauthorized access without credentials, demonstrating weak client-side authentication controls and highlighting the risks of relying solely on application-side security logic.

5. IPC and Component Testing – Drozer



5.1 Objective

Evaluate exposed components and inter-process communication vulnerabilities.

5.2 Procedure

1. Drozer agent was installed on the test device.
2. Application package information was enumerated.
3. Exported activities, services, and broadcast receivers were analyzed.
4. Permission enforcement was validated.

5.3 Observations

Several components were found exported without adequate permission checks, enabling potential intent injection and unauthorized access to internal application functionality.

---

6. Findings Summary Table

Finding ID	Vulnerability	Severity	Impact
F016	Insecure Storage	High	Sensitive data extraction
F017	Authentication Bypass	High	Unauthorized access
F018	Exposed Components	Medium	Intent injection

---

7. Remediation Recommendations

1. Encrypt all sensitive data using Android Keystore APIs.
2. Move authentication logic to server-side validation.
3. Implement root and tamper detection.
4. Restrict exported components using explicit permissions.
5. Apply ProGuard/R8 to obfuscate sensitive logic.

9. Conclusion

The Mobile Application Testing Lab demonstrated how improper client-side security controls can be exploited using widely available tools. Insecure storage, weak authentication logic, and exposed components collectively increase the attack surface. Addressing these issues is essential to prevent data leakage and unauthorized access in real-world mobile applications.