



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

The screenshot displays the MobSF web interface with the following details:

- File Information:**
 - Name: diva-beta.apk
 - Size: 1.43MB
 - MD5: 82ab8b2193b3c9b1c737e3a786be363a
 - SHA1: 27e849d9d7b86a3a3357fb3e980433a91d416801
 - SHA256: 5cfc51fce9bd760b92ab2340477f4dda84b4ae0c5d04a8c9493e4fe34fab7c5
- App Information:**
 - Package Name: jakhar.aseem.diva
 - Main Activity: jakhar.aseem.diva.MainActivity
 - Target SDK: 23, Min SDK: 15, Max SDK: 23
 - Android Version Name: 1.0
 - Android Version Code: 1
- Summary Cards:**
 - 17 ACTIVITIES
 - 0 SERVICES
 - 0 RECEIVERS
 - 1 PROVIDERS
- Exported Elements:**
 - EXPORTED ACTIVITIES: 2
 - EXPORTED SERVICES: 0
 - EXPORTED RECEIVERS: 0
 - EXPORTED PROVIDERS: 1
- Code Nature:**
 - Native: True
 - Dynamic: False
 - Reflection: True
 - Crypto: True
 - Obfuscation: False
- Options:**
 - Buttons: View Java, Download Java Code, View Smali, Download Smali Code, Rescan, View AndroidManifest.xml, Start Dynamic Analysis



The screenshot displays the MobSF (Mobile Security Framework) web interface. The left sidebar contains navigation options: Information, Scan Options, Signer Certificate, Permissions, Binary Analysis, Android API, Browsable Activities, Security Analysis, Malware Analysis, Reconnaissance, Components, PDF Report, Print Report, and Start Dynamic Analysis. The main content area is divided into several sections:

- APP SCORES:** Shows an average CVSS score of 6.1, a security score of 55/100, and 0/285 trackers detected.
- FILE INFORMATION:** Lists file name (diva-beta.apk), size (1.43MB), MD5, SHA1, and SHA256 hashes.
- APP INFORMATION:** Provides details like App Name (Diva), Package Name (jakhar.aseem.diva), Main Activity, Target SDK, Min SDK, Max SDK, Android Version Name, and Android Version Code.
- ACTIVITIES, SERVICES, RECEIVERS, PROVIDERS:** Summary cards showing counts (17, 0, 0, 1 respectively) and links to view details.
- EXPORTED:** Summary cards for Exported Activities (2), Exported Services (0), Exported Receivers (0), and Exported Providers (1).
- SCAN OPTIONS:** Buttons for Rescan and Start Dynamic Analysis.
- DECOMPILED CODE:** Buttons to view AndroidManifest.xml, Java, and Smali code, along with download links for Java Code, Smali Code, and the APK.

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

The screenshot shows a Windows PowerShell terminal window with the following commands and output:

```
PS C:\Users\Lorenzo\Documents\Contenu Vaadata\2409 - frida> frida -U -N b3nac.injuredandroid -l .\scripts-frida\bypass.js
```

The output displays the Frida 16.5.1 interface, including a list of commands (help, object?, exit/quit) and a message indicating it is connected to an Android Emulator (5554). The final line of the output is:

```
[Android Emulator 5554::b3nac.injuredandroid ]-> Method a() hooked
```

On the right side of the terminal, a small window titled "InjuredAndroid" shows a message: "Congrats you found the flag! :D".



```
(kali@kali)-[~/Downloads]
$ frida-ps
PID  Name
-----
1697  Thunar
10709 adb
1824  agent
1614  at-spi-bus-launcher
1632  at-spi2-registryd
1525  dbus-daemon
1621  dbus-daemon
1688  dconf-service
2121  firefox-esr
2226  firefox-esr
2323  firefox-esr
2401  firefox-esr
2487  firefox-esr
2505  firefox-esr
2508  firefox-esr
2561  firefox-esr
2786  firefox-esr
1526  gnome-keyring-daemon
1967  gvfs-afc-volume-monitor
2013  gvfs-goa-volume-monitor
2003  gvfs-gphoto2-volume-monitor
1937  gvfs-mtp-volume-monitor
1891  gvfs-udisks2-volume-monitor
1653  gvfsd
1659  gvfsd-fuse
```

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.

```

root@kali:~# $ drozer console connect -server 192.168.56.103
Selecting d278838d23b67b0 (innotek GmbH VirtualBox 8.1.0)

..
..0..
..a..
ro.ids.nemesisand..pr
..ector.android.sne.
..sisandprotector.android.sne.
..nemesisandprotector.android.sne.
..emesisandprotector.android.sne.
..isandp..rotector.android..ids.nem.
..isandp..rotector.android..sne.mesis.
..andprotector.android.sne.mesisandprotector.
..tor.android.sne.mesisandprotector.android.
..sne.mesisandprotector.android.sne.mesisand.
..dprotector.android.sne.mesisandprotector.

drozer Console (v2.4.4)
dz> run scanner.activity.browsable
Package: com.android.cts.priv.ctsshim
Invokable URIs:
Classes:
..InstallPriority

Package: com.google.android.android.googlequicksearchbox
Invokable URIs:
assistant-handoff://complete
://assistant.google.com
assistant-settings://
com.google.android.apps.gsa.gdi://oauth2redirect (PATTERN_LITERAL)
android-app://
agsa://search/ (PATTERN_LITERAL)
googlequicksearchbox://
googleassistant://android/app/link/promo/tooltip (PATTERN_LITERAL)
dynact://
Classes:
com.google.android.apps.gsa.assistant.handoff.BrowserReturnActivity
com.google.android.apps.gsa.assistant.settings.hq.agentdirectory.AgentDirectoryActivity
com.google.android.apps.gsa.assistant.settings.AssistantSettingsActivity
net.openid.appauth.RedirectUriReceiverActivity
com.google.android.search.calypso.AppIndexingActivity
com.google.android.googlequicksearchbox.SearchActivity
com.google.android.apps.gsa.staticplugins.opa.promo.UpgradePromoTooltipActivity
com.google.android.apps.gsa.velour.DynamicActivityTrampoline
com.google.android.apps.gsa.velvet.ui.VelvetIntentDispatcher

Package: com.android.vending
Invokable URIs:
http://

```

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.