External Application Penetration Testing Technical Report

For

King Saud University

Item	Description
Document Title	Mobile application penetration testing
Requestor	King Saud University | College of Computer and Information Sciences | Information Technology department | IT 371: Application security | Doctor Nourah madi
Author/s	Jumanah alDawsari – Nouf Alsadhan
Date Created	23/5/2023

Contents

1 Executive Summary

1.1 *Introduction*

In this project we implemented black box pentesting using some tools such as mobSF, jadx, and ADB.
Blackbox pen testing is security testing where the tester simulates an external attacker with no prior knowledge of the system being tested in order to identify potential vulnerabilities.

1.2 *Scope*

The specific scope of this project includes performing the Application Penetration test for the specified duration on the below mentioned applications.

We used Samsung android phone as emulator.

Jadx : we used this tool to decompile the APK and read the source code.

MobSF: we used this tool to find the vulnerabilities.

ADB: we used this tool to communicate with the Android phone through terminal and to find emulator logs.

Application Name	Platform	Version	Environment	Approach
InsecureShop	Android	1.0	Windows	Black box Penetration testing

1.3 *Risk Rating*

The risk rating for the issues and their impact on the operation of the organization is explained in the table 1 below. The overall risk rating reported will be based on vulnerability identification with its potential to be exploited by adversaries.

In general, the following factors were considered to arrive at the risk rating for vulnerability:

- Technical Impact: The extent to which an attacker may gain access to a system and the severity of it on the application. This metric will take the security triad CIA (Confidentiality, Integrity and Availability) values into account.
- Likelihood: This metric will take the Popularity and Simplicity of an exploit into consideration.
    - Popularity describes the existing or potential frequency of exploitation of the vulnerability.
    - Simplicity is the amount of effort required to exploit the vulnerability.

Overall Risk Severity				
Technical Impact (Confidentiality, Integrity,	HIGH	MEDIUM	HIGH	CRITICAL
	MEDIUM	LOW	MEDIUM	HIGH

Availability)	LOW	INFO	LOW	MEDIUM
		LOW	MEDIUM	HIGH
		Likelihood (Popularity and Simplicity)		

Table 1 Risk Severity

1.4 *Threat Security Level*

Vulnerabilities are categorized as **Critical**, **High**, **Medium**, **Low** and **Informational**.

Critical: Severe Impact on the affected application. They require immediate attention and resolution. Successful exploitation may provide the attacker **access to critical data**.

High: Severe Impact on the affected application. They require immediate attention. They are relatively easy for attackers to exploit and may provide them with **full control of the affected application.**

Medium: Moderate impact on the affected application. They are often **harder to exploit** and may not provide the same access to affected application.

Low: Limited impact on the affected application. They provide information to attackers that may assist them in mounting **subsequent attacks on the affected applications.** These should also be fixed in a timely manner, but are not as urgent as the other vulnerabilities.

Informational: It exposes information that target stake holders simply need to be aware of. These are for findings that are very difficult to exploit in practice.

1.5 *Summary Table*

The table below shows the summary of vulnerabilities disclosed during the Penetration Testing.

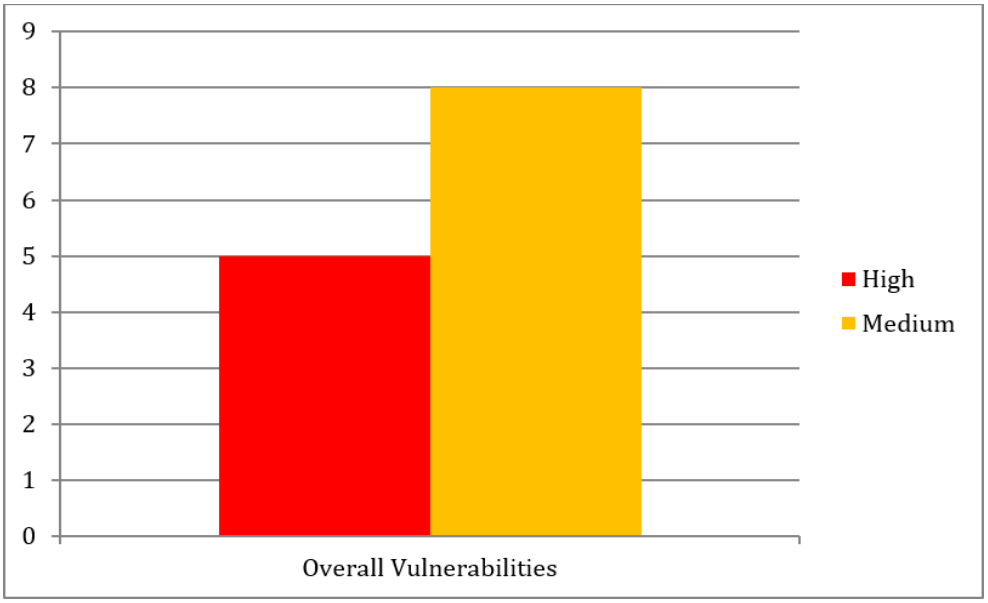
Mobile Application Penetration Testing

Critical	High	Medium	Low
-	5	8	-

1.6 *Summary Graph*

The following bar graph highlights the total number of vulnerabilities discovered during the penetration testing.

Figure 1 Application Penetration Testing



1.7 *Key Finding*

No.	Vulnerabilities Discovered	Platform	Severity Level
1	Debug Enabled For App [android:debuggable=true]	Android	**High**
2	Clear text traffic is Enabled For App [android:usesCleartextTraffic=true]	Android	**High**
3	Insecure WebView Implementation	Android	**High**
4	Files may contain hard coded sensitive information like usernames, password, keys etc.	Android	**Medium**
5	The App logs information	Android	**Info**

2 Conclusion

We discovered five vulnerabilities while implementing penetration testing. To enhance the security of the application, we recommend prioritizing and implementing our advice on mitigating these vulnerabilities. This will help prevent attackers from easily exploiting or leveraging the vulnerabilities to their advantage.

- **Debug Enabled For App:** Setting the android:debuggable flag to true enables an attacker to debug the application, making it easier for them to gain access to parts of the application that should be kept secure. Always make sure to set the android:debuggable flag to false when shipping your application [1].

- **Clear text traffic is Enabled For App:** When android:usesCleartextTraffic is set to true, the application will allow outgoing requests over HTTP, resulting in a potential data leakage or a Man-In-The-Middle (MITM) attack. By setting the value to false, the application will refuse the app's requests to use cleartext traffic [2].

- **Insecure WebView Implementation:** The insecure WebView implementation vulnerability refers to a situation where an app is using a WebView component to display web content, but the WebView is not configured securely. Specifically, this vulnerability occurs when the WebView is set to ignore SSL certificate errors or to accept any SSL certificate, making the app vulnerable to Man-in-the-Middle (MITM) attacks. Always use HTTPS to load web content in WebView. in addition, WebView should always validate SSL certificates presented by the server [3].

- **Files may contain hard coded sensitive information:** Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the product administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely.

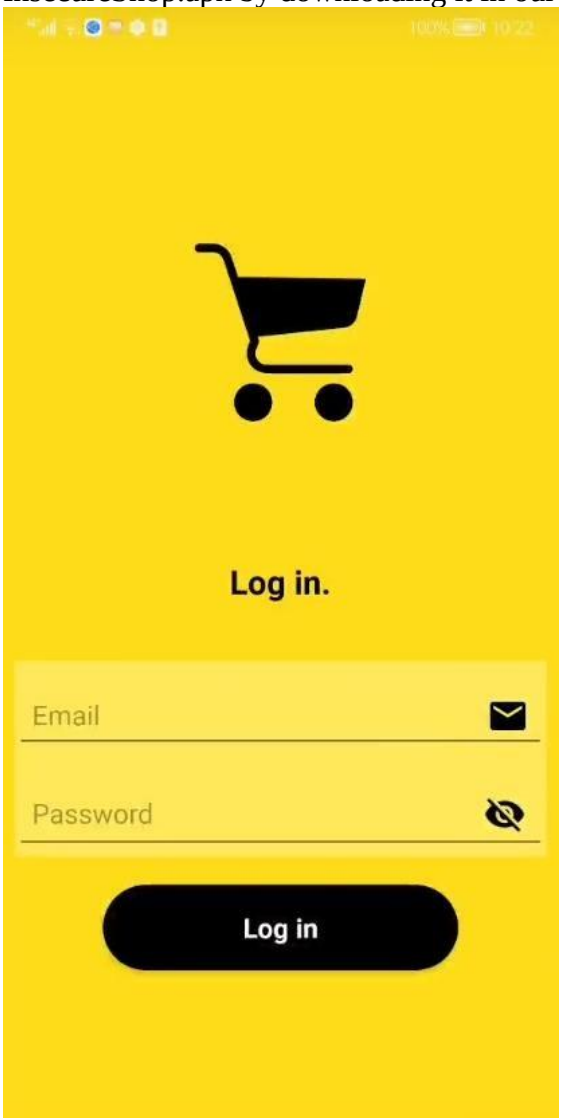
  store passwords, keys, and other credentials outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible. In Windows environments, the Encrypted File System (EFS) may provide some protection [5].

- **The App logs information:** The vulnerability reported by MOBSF, "The App logs information. Sensitive information should never be logged," means that the app is logging sensitive information, such as usernames, passwords, or other confidential data, in a way that is not secure. Logging is an important part of app development for debugging and troubleshooting, but it's essential to ensure that sensitive information is not being logged. Developers should ensure that sensitive data is not being written into logs, and if it is necessary to log such data, it should be done in a secure and encrypted manner [4].

3 Methodology

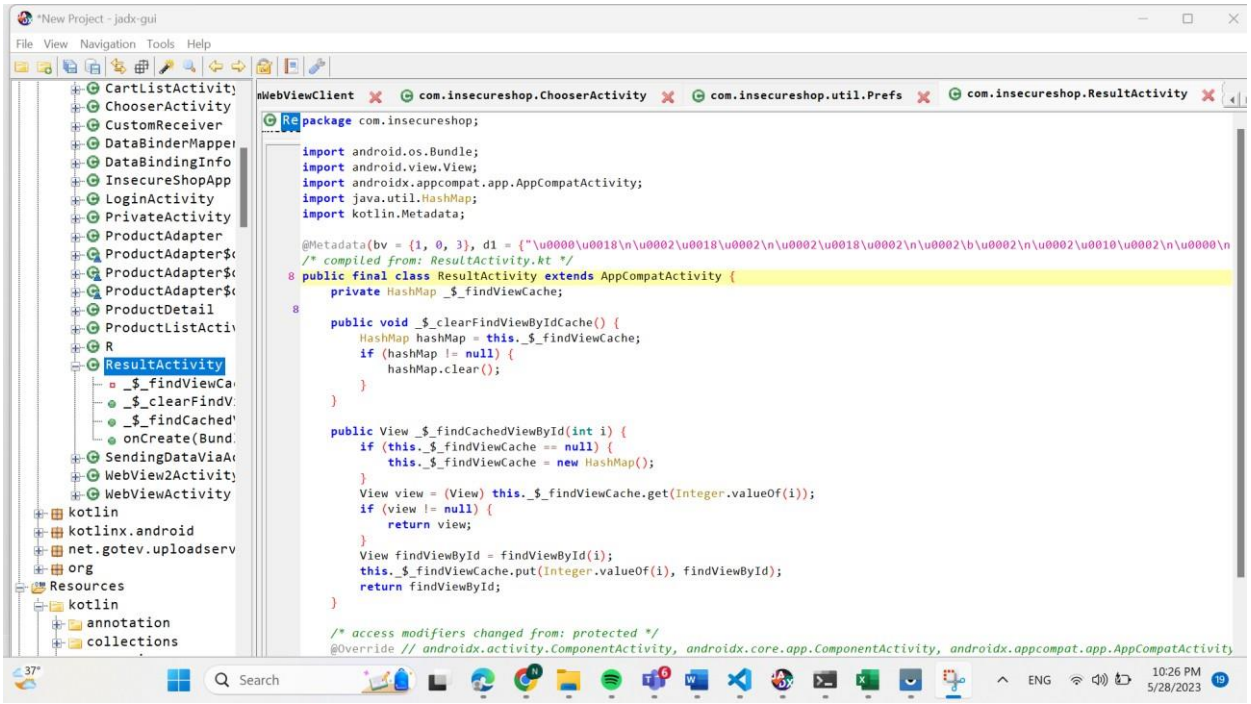
Our methodology on Mobile penetration testing is based on Mobile Open Web Application Security Project (OWASP); our assessment methodology to carry out the mobile penetration testing includes 2 phases or any preparations:

1- In order to start working on the project, one of the steps is getting a real Android phone or create an emulated one. We already have a Samsung phone that we are using to work in our graduation project, so this step is already ready and we can start working. We can run the InsecureShop.apk by downloading it in our Samsung phone and it worked correctly.



2- we had to install some tools:
2.1 Jadx[6]
Jadx is an open-source decompiler used for reverse engineering Android apps. It is designed to convert Android app binaries into a human-readable format, making it easier for developers and security researchers to analyze and understand the code. With Jadx, we can view the source code of an Android app and explore its structure, classes, and methods. This can be useful for a variety of purposes, such as debugging, modifying, or extending an app's functionality, or identifying security vulnerabilities.

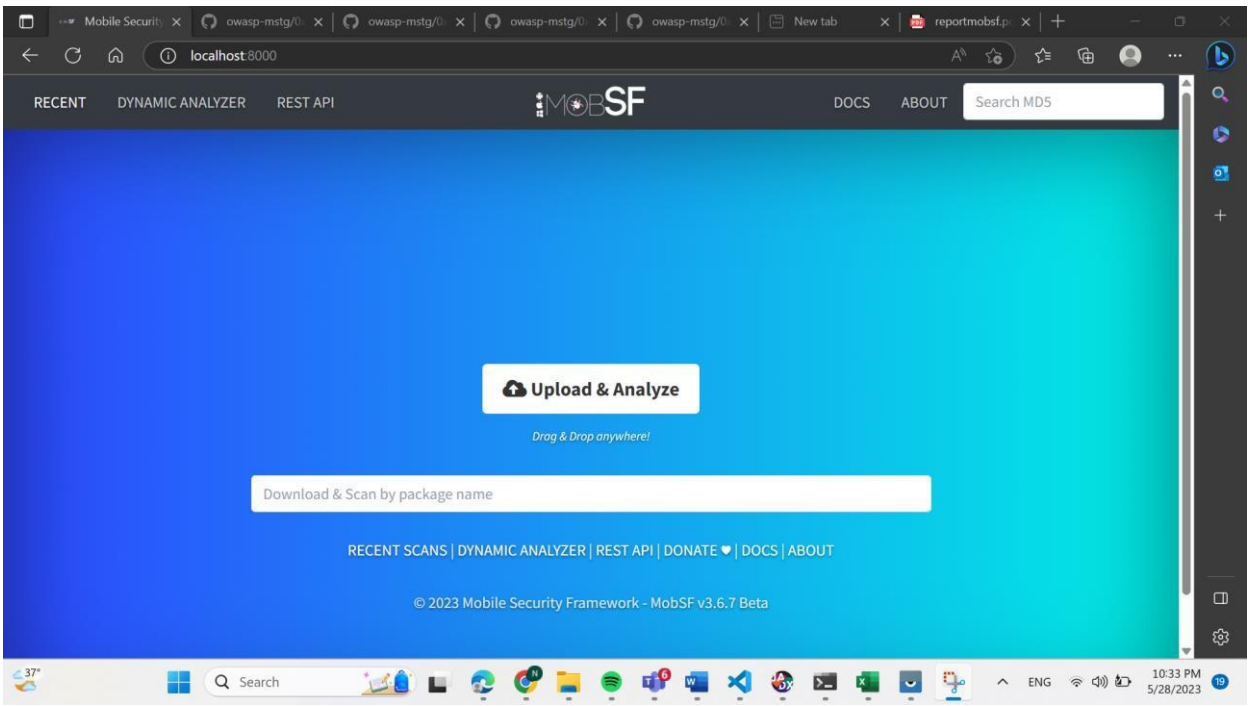
We downloaded Jadx, and opened the Jadx-GUI in order to decompile and read the source code of InsecureShop.apk.



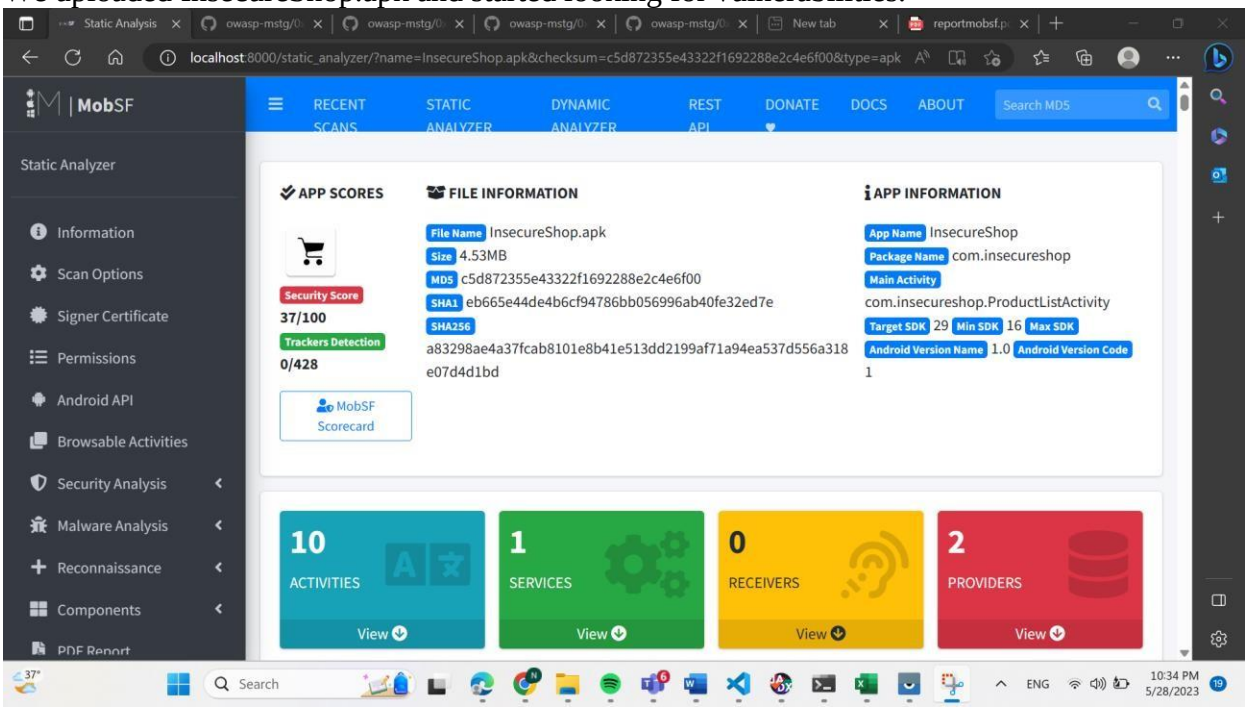
2.2 MobSF[7]

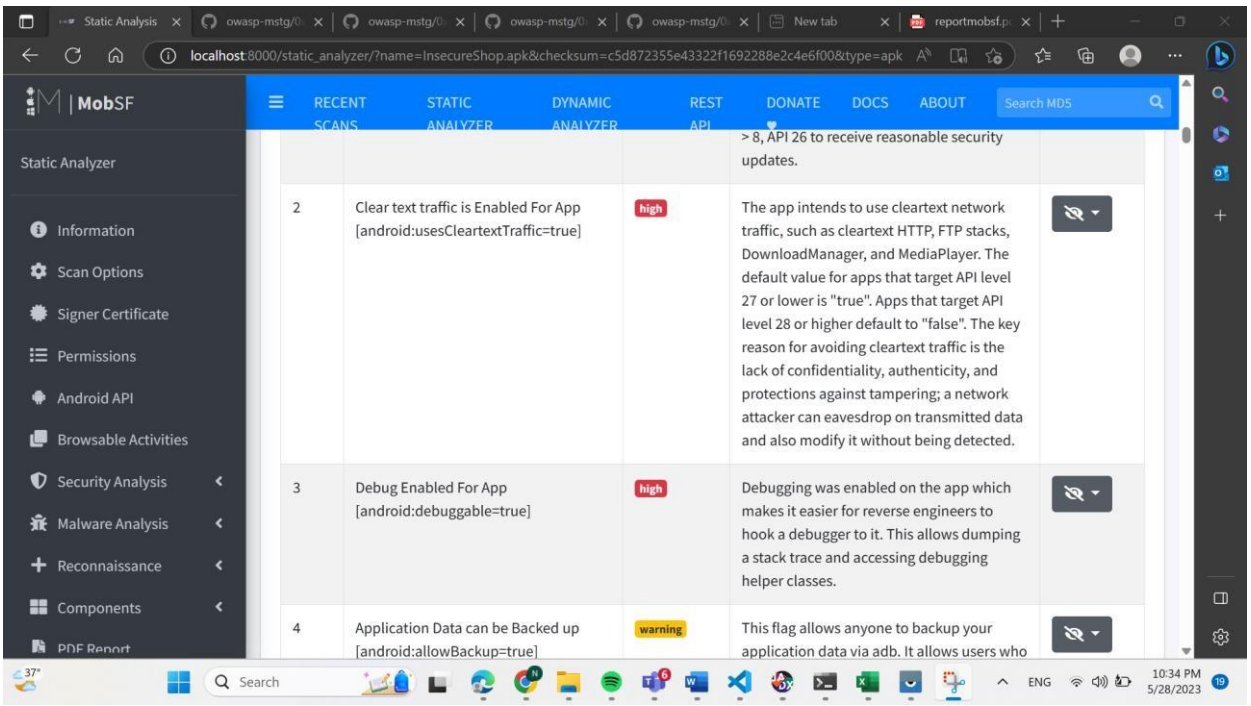
MobSF (Mobile Security Framework) is a free, open-source mobile application security testing framework used for analyzing the security of Android and IOS apps. It provides a variety of dynamic and static analysis tools to identify vulnerabilities and security issues in mobile apps, such as insecure data storage, usage of insecure libraries, and insecure network communication. MobSF can be used both as a command-line tool and as a web-based interface, and supports a range of file formats, including APK, IPA, and APPX. MobSF also integrates with other security tools and frameworks, making it a powerful and flexible solution for mobile app security testing.

In order to download MobSF, we had to download Java, Python, Visual studio, Git, Open SSL and WkHTMLtox. After downloading all the previous mentioned tools, we started downloading MobSF by writing “git clone” in the command line followed by the link of the github repository, and then we wrote setup.bat and waited until it finished.
After it finished we opened localhost:8000.



We uploaded InsecureShop.apk and started looking for vulnerabilities.

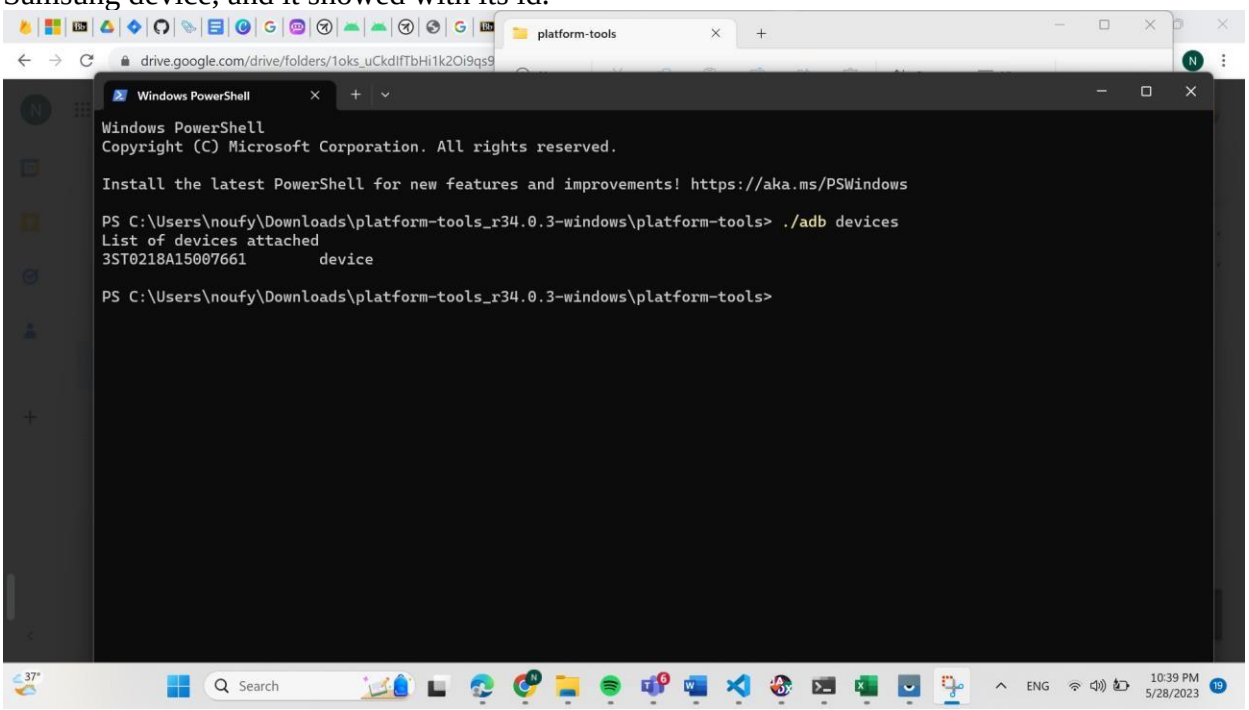




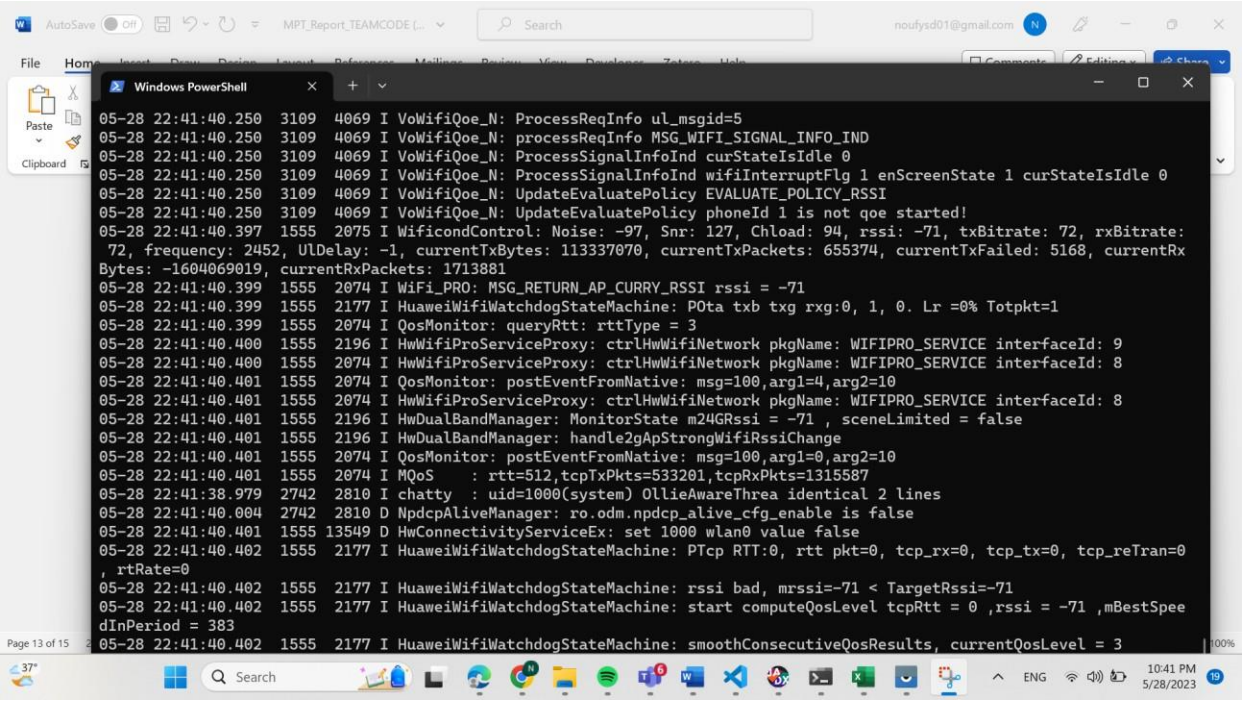
2.2 Android Debug Bridge (ADB)[8]

Android Debug Bridge (ADB) is a command-line tool that allows developers to communicate with an Android device or emulator. ADB is a part of the Android Software Development Kit (SDK) and provides a variety of features for debugging and testing Android apps, such as installing and uninstalling apps, transferring files between a device and a computer, and running shell commands on a device.

After downloading ADB and apk platform tool, we opened the command line to search for our Samsung device, and it showed with its id.



We then started previewing the Android logs by running the command “adb logcat -v threadtime”, and we noticed that every click and movement in the phone it directly shows in the logs.



4 Detailed Findings

4.1 *Limitations*

Following were the limitations that we encountered:
-MobSF requires many requirements(python, jdk, etc..)
-Installing all the tools consumed a significant amount of memory space.
-Running the APK file on real phone Emulator require many steps.

4.2 *Technical Description of Findings*

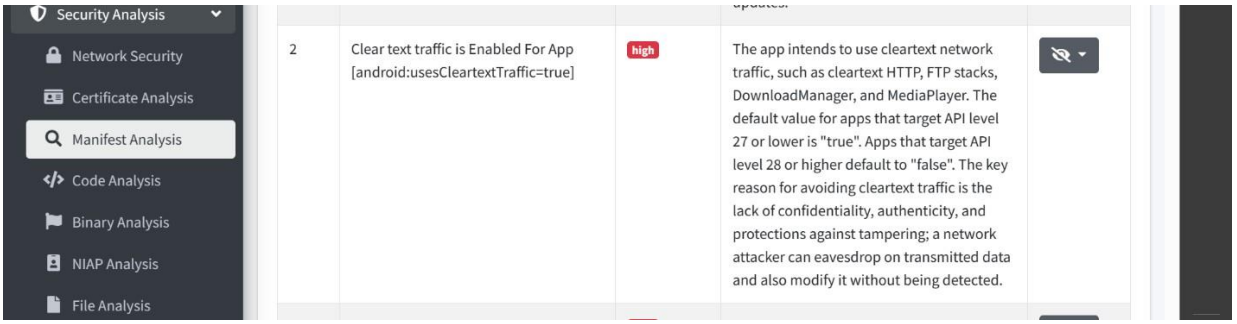
This section explains the details of the identified vulnerability along with technical impact, Proof of Concept, recommendations, and references related to the vulnerability.

4.2.1 Debug Enabled For App

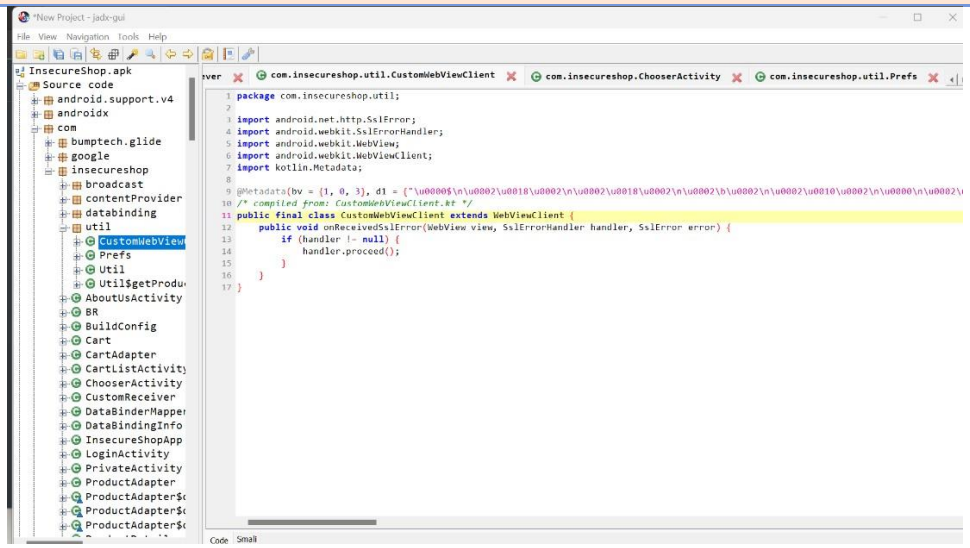
Severity Level	**High**				
Technical Impact	High	**Likelihood**		Low	
		Popularity	Low	**Simplicity**	High
Observation	We noticed Debug Enabled in the MobSF in analysis application which can pose a significant security risk.				
Impact	The debug enabled can introduce serious security vulnerabilities that could enable attackers to gain unauthorized access and access sensitive information.				
Platform	Android				

Proof of Concept	
	
Recommendation	Setting the android:debuggable flag to true enables an attacker to debug the application, making it easier for them to gain access to parts of the application that should be kept secure. Always make sure to set the android:debuggable flag to false when shipping your application [1].
OWASP Reference	https://owasp.org/www-project-mobile-top-10/2016-risks/m10-extraneous-functionality
Reference	https://developer.android.com/topic/security/risks/android-debuggable

4.2.2 Clear text traffic is Enabled For App

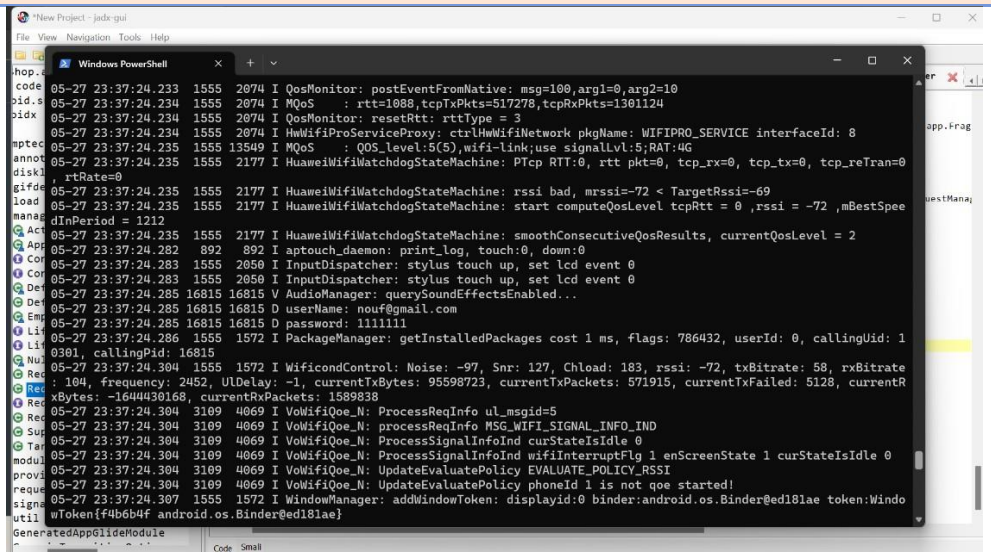
Severity Level	**High**				
Technical Impact	High	**Likelihood**		High	
		Popularity	High	**Simplicity**	High
Observation	In MobSF we open mainfest anaylsis and we observe clear text traffic which can be a serious security risk.				
Impact	It can allow attackers to intercept and read sensitive information, such as usernames, passwords, and other confidential data, that is being transmitted over the network				
Platform	Android				
Proof of Concept					
				
Recommendation	When android:usesCleartextTraffic is set to true, the application will allow outgoing requests over HTTP, resulting in a potential data leakage or a Man-In-The-Middle (MITM) attack. By setting the value to false, the application will refuse the app's requests to use cleartext traffic [2].				
OWASP Reference	https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication				
Reference	https://sensei.securecodewarrior.com/recipes/scw%3Aandroid%3Adisabled-cleartext				

4.2.3 Insecure WebView Implementation

Severity Level	**High**				
Technical Impact	High	**Likelihood**		Medium	
		Popularity	High	**Simplicity**	High
Observation	We observed through the use of jadx and MobSF that the implementation of an insecure WebView in the application which poses a significant security risk.				
Impact	The impact of an insecure WebView in an application can be significant, as it can leave the application vulnerable to various types of attacks, such as cross-site scripting (XSS), and data theft. An insecure WebView can allow attackers to execute steal user credentials, and inject malicious code				
Platform	Android				
Proof of Concept					
					
Recommendation	The insecure WebView implementation vulnerability refers to a situation where an app is using a WebView component to display web content, but the WebView is not configured securely. Specifically, this vulnerability occurs when the WebView is set to ignore SSL certificate errors or to accept any SSL certificate, making the app vulnerable to Man-in-the-Middle (MITM) attacks. Always use HTTPS to load web content in WebView. in addition, WebView should always validate SSL certificates presented by the server [3].				
OWASP Reference	https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication/				
Reference	https://developer.android.com/guide/webapps/webview#security				

4.2.4 The App logs information

Severity Level	**Info**				
Technical Impact	High	**Likelihood**		High	
		Popularity	High	**Simplicity**	High
Observation	In command prompt we used ADB command and then we observe sensitive logs after trying to login with a password and a username, the password showed in the logs same as the password entered.				
Impact	Can potentially expose sensitive information to unauthorized parties, making it easier for attackers to exploit vulnerabilities and compromise the system.				
Platform	Android				
Proof of Concept					

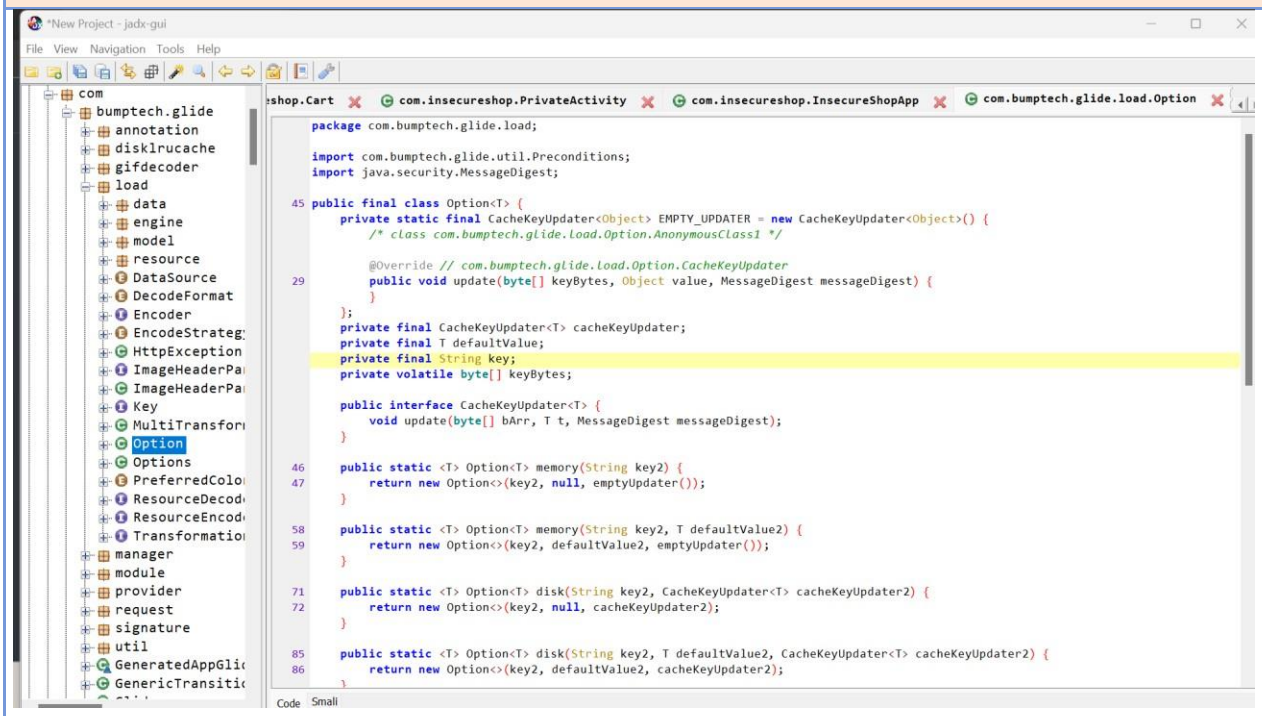


Recommendation	The vulnerability reported by MOBSF, "The App logs information. Sensitive information should never be logged," means that the app is logging sensitive information, such as usernames, passwords, or other confidential data, in a way that is not secure. Logging is an important part of app development for debugging and troubleshooting, but it's essential to ensure that sensitive information is not being logged. Developers should ensure that sensitive data is not being written into logs, and if it is necessary to log such data, it should be done in a secure and encrypted manner [4].
OWASP Reference	https://owasp.org/www-project-top-ten/2017/A10_2017-Insufficient_Logging%2526Monitoring https://owasp.org/www-project-mobile-top-10/2016-risks/m2-insecure-data-storage
Reference	https://developer.android.com/topic/security/data

4.2.5 Files may contain hard coded sensitive information

Severity Level	**Medium**				
Technical Impact	High	**Likelihood**		Medium	
		Popularity	High	**Simplicity**	High
Observation	By jadx and MobSF we noticed the key without hashing it in the code.				
Impact	Can make it easier for attackers to gain unauthorized access to the system, for example, if the password is the thing that is stored hardcoded, the attacker can easily gain access to it.				
Platform	Android				

Proof of Concept



Recommendation	Hard-coded credentials typically create a significant hole that allows an attacker to bypass the authentication that has been configured by the product administrator. This hole might be difficult for the system administrator to detect. Even if detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. store passwords, keys, and other credentials outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible. In Windows environments, the Encrypted File System (EFS) may provide some protection [5].
OWASP Reference	https://owasp.org/www-project-mobile-top-10/2014-risks/m2-insecure-data-storage
Reference	https://cwe.mitre.org/data/definitions/798.html

Appendix A: About the Team

Team Code:	
Student Name	Role
Nouf Alsadhan	Work done in her device, Methodology and multiple parts in the report.
Jumanah alDawsari	Detailed findings, conclusion and multiple parts in the report.

References :

[1] https://developer.android.com/topic/security/risks/android-debuggable

[2] https://sensei.securecodewarrior.com/recipes/scw%3Aandroid%3Adisabled-cleartext

[3] https://owasp.org/www-project-mobile-top-10/2016-risks/m3-insecure-communication

[4] https://developer.android.com/topic/security/data

[5] https://cwe.mitre.org/data/definitions/798.html

[6] https://github.com/skylot/jadx/releases/tag/v1.2.0.

[7] https://github.com/MobSF/Mobile-Security-Framework-MobSF

[8] https://www.xda-developers.com/install-adb-windows-macos-linux/