

**External Application Penetration Testing Technical Report**

**For**

**King Saud University**

## Document Info

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 . - Dr. Alia Alabdulkarim - Instructor: Nora bin madi
Author/s	- Lama Alshaya - Aljawharah Alzamil
Date Created	22/5/2023

## Table of Content

<b>1 Executive Summary</b>	<b>4</b>
1.1 Introduction	4
1.2 Scope	4
1.3 Risk Rating	5
1.3 Threat Security Level	5
1.4 Summary Table	7
1.5 Summary Graph	7
1.6 Key Findings	7
<b>2 Conclusion</b>	<b>8</b>
<b>3 Methodology</b>	<b>10</b>
Task#1: Get familiar with Android environment.[7]	10
Task#2 : Set up the MPT environment.	10
1. Emulator	10
2. Installing Jadx tool [2].	11
3. Android Debug Bridge (ADB) [9].	15
4. Installing MobSF [3][4] .	17
4.1 Install Java JDK [5].	17
4.2 Install Python [6].	20
4.3 Install OpenSSL [8].	24
4.4 Install WkHTMLtox [10].	28
4.5 Install Git[11].	33
4.6 Install Visual Studio [12]	34
5. Setup MobSF [13].	34
<b>4 Detailed Findings</b>	<b>41</b>
4.1 Limitations	41
4.2 Technical Description of Findings	41
<b>Appendix A: About the Team</b>	<b>48</b>

# 1 Executive Summary

## 1.1 Introduction

Penetration testing involves simulating an attack on a computer system, network, or application by malicious attackers, in order to assess the security of the said system or application and identify any vulnerabilities. Black box penetration testing, on the other hand, is the process of conducting a penetration test without any prior knowledge of the system being tested.

In the context of the Application Security course, we have undertaken the task of applying Mobile Penetration Testing (MPT) on an Android Package (APK) for our project. We have conducted a black box penetration testing on the **APK file** (InsecureShop.apk) by using various tools such as **MobSF**, **jadx**, **Android device**, **ADB**. These tools have been employed to find and identify the vulnerabilities present in the system.

## 1.2 Scope

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

- **jadx** which decompiles the APK and enables us to read the source code.
- **MobSF** which automatically identifies vulnerabilities.
- **Android device** which is Lenovo K14 as an simulator.
- **ADB** which allows us to communicate through the terminal with an Android device, allowing us to preview Android logs and launch activities.

Application Name	Platform	Version	Environment	Approach
PT	Android	1	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, Availability)	HIGH	MEDIUM	HIGH	CRITICAL
	MEDIUM	LOW	MEDIUM	HIGH
	LOW	INFO	LOW	MEDIUM
	LOW	MEDIUM	HIGH	
Likelihood (Popularity and Simplicity)				

**Table 1 Risk Severity**

### **1.3 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.

#### 1.4 Summary Table

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

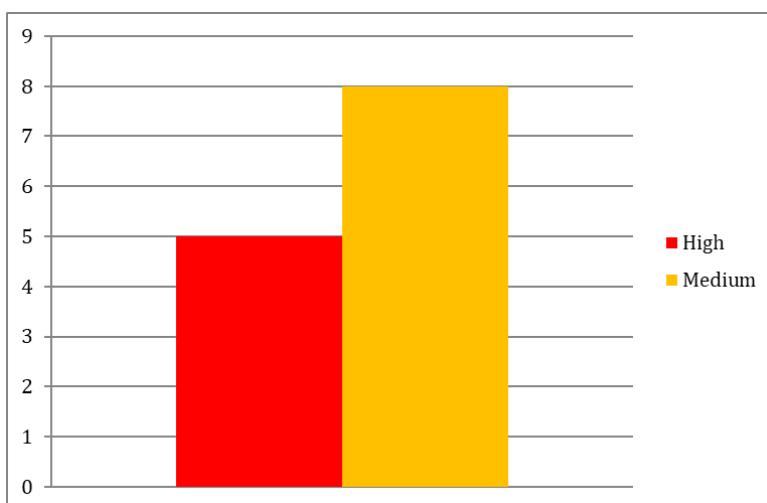
**Mobile Application Penetration Testing**

Critical	High	Medium	Low
-	5	8	-

#### 1.5 Summary Graph

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

**Figure 1 Application Penetration Testing**



#### 1.6 Key Findings

No.	Vulnerabilities Discovered	Platform	Severity Level
1	Insecure Communication	Android	High
2	Hardcoding sensitive data in the source code.	Android	Medium
3	Insecure Logging of sensitive data	Android	High
4	Insecure Data Storage	Android	Medium
5	Application is enabled for debugging.	Android	High

## 2 Conclusion

During the pentesting process, we identified five vulnerabilities. To enhance the security of the application, we recommend that you prioritize and adhere to our suggestions on how to address these vulnerabilities. By doing so, an attacker would find it difficult to exploit or benefit from these vulnerabilities.

1. **Insecure Communication:** the application lack of SSL Certificate Validation, when the certificate is invalid or malicious, it might allow an attacker to spoof a trusted entity by interfering in the communication path between the host and the client, The application may establish a connection to a fraudulent or malicious host while falsely believing that it is connecting to a trusted host.

**Our suggestions to mitigate this vulnerability:** use strong encryption algorithms to protect data confidentiality and integrity, Use transport layer security (TLS) or secure sockets layer (SSL) protocols to encrypt data in transit and protect it from interception or tampering.

2. **Hardcoding sensitive data in the source code:** hardcoding sensitive data can increase the risk of data breaches if the code is accessed by unauthorized parties, either through malicious intent or by mistake. This can result in the loss of confidential data and can lead to reputational damage.

**Our suggestions to mitigate this vulnerability:** store sensitive data such as passwords securely in a separate configuration file or a secure key store rather than hardcoding them in the source code. Use encryption or hashing algorithms to protect sensitive data.

3. **Insecure Logging of sensitive data :** storing sensitive user data in insecure logs can create an insecure avenue for attackers to obtain confidential information. In the current situation, an attacker could potentially compromise user accounts by using passwords obtained from the logs, thereby breaching both authentication and confidentiality protocols.

**Our suggestions to mitigate this vulnerability:** use secure logging frameworks, and tools that can perform the encryption. Conduct regular security assessments and penetration testing to identify and address any vulnerabilities in the application's logging practices.

4. **Insecure Data Storage:** can have a significant impact on the confidentiality, integrity, and availability of sensitive data. It can lead to data breaches, identity theft, financial loss, and other malicious activities.

**Our suggestions to mitigate this vulnerability:** Store sensitive data such as passwords, credit card details in secure databases or file systems with appropriate access controls and permissions. Use IPS help to mitigate the vulnerability of insecure data storage by detecting and blocking malicious traffic that may attempt to exploit vulnerabilities in the data storage system

5. **Application is enabled for debugging:** When an application is easily hooked to a debugger, it can be vulnerable to reverse engineering. This can allow attackers to dump a stack trace and access debugging helper classes, potentially leading to the exposure of technical or sensitive information.

**Our suggestions to mitigate this vulnerability:** Disable debugging mode to prevent access and reverse engineering.

### **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 X phases or any preparations:

**Set up our MPT Environment:**

**Preparation stage :**

***Task#1: Get familiar with Android environment.[7]***

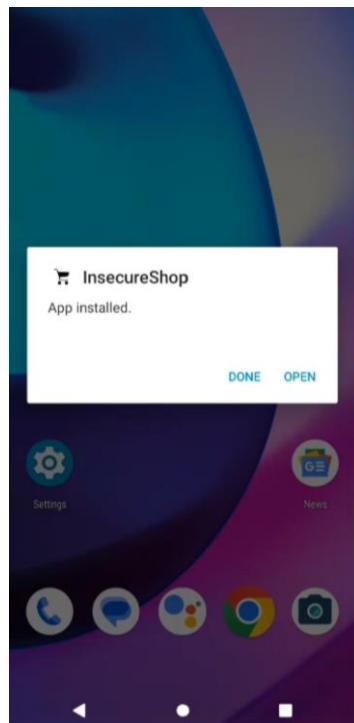
Since we have been working on some projects using Visual Code Studio, we already have a physical android device which is Lenovo K14 that we will be using during this project in order to simulate the apk file.

***Task#2 : Set up the MPT environment.***

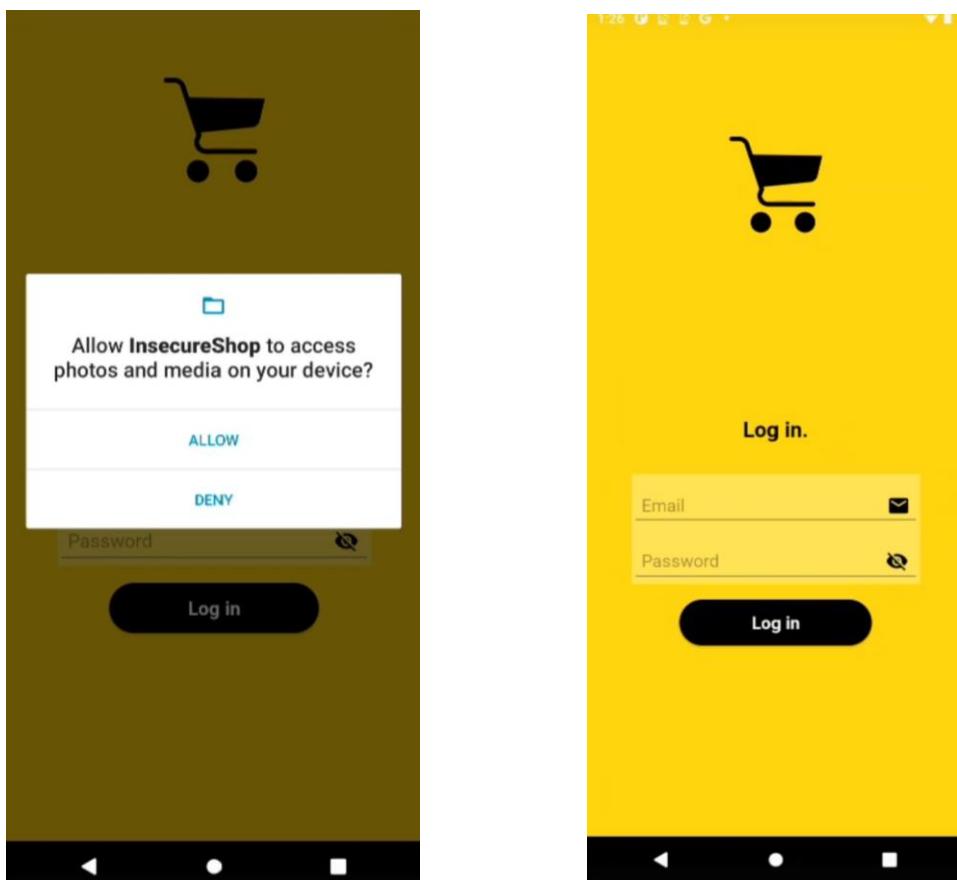
#### **1. Emulator**

1.1 Android device which is Lenovo K14 as our simulator.

1.2: We opened LMS on our device (Lenovo K14) and downloaded the InsecureShop.apk file on the device.



1.2.1 Then we opened the application and there was the Login interface as shown in the picture.



## 2. Installing Jadx tool [2].

2.1 Initially, go to <https://github.com/skylot/jadx>

2.2 We clicked “github”.

github.com/skylot/jadx

README.md

```
Intent.setAction("com.android.launcher.action.UNINSTALL_SHORTCUT");
sendBroadcast(intent);
RegisterReceiver(new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (action.equals(Intent.ACTION_BOOT_COMPLETED)) {
            Intent intent4 = new Intent();
            intent4.setClassName(getApplicationContext(), "com.whatsapp.HomeActivity");
            startActivity(intent4);
            overridePendingTransition(0, 0);
        }
    }
});
```

Issues: 12 errors 1364 warnings

Download

- release from github: [release v1.4.7](#)
- latest unstable build [commits since v1.4.7 74](#)

After download unpack zip file go to bin directory and run:

- jadx - command line version
- jadx-gui - UI version

On Windows run .bat files with double-click

Note: ensure you have installed Java 11 or later 64-bit version. For Windows, you can download it from [oracle.com](#) (select x64 Installer).

Install

- Arch linux [not found](#)
- sudo pacman -S jadx
- macOS [v1.4.7](#)

2.3 We clicked on “jadx-1.4.7.zip”. After that, it began downloading.

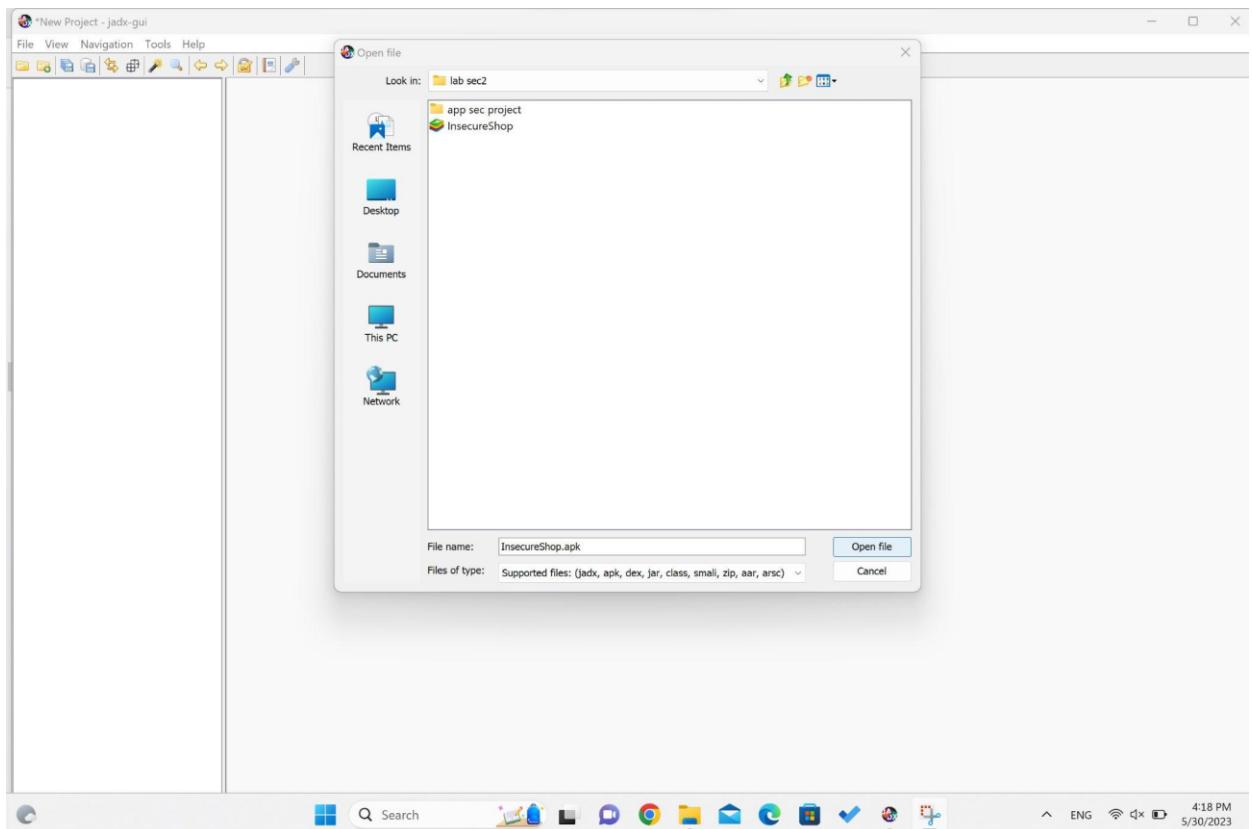
The screenshot shows a GitHub release page for the jadx-1.4.7 version. It includes a changelog with various commits, download links for different file types, and a list of assets including source code in zip and tar.gz formats. The assets section shows five items with their names, sizes, and upload dates.

Name	Size	Date
jadx-1.4.7.zip	28.5 MB	Apr 20
jadx-gui-1.4.7-no-jre-win.exe	30.8 MB	Apr 20
jadx-gui-1.4.7-with-jre-win.zip	55.6 MB	Apr 20
Source code (zip)		Apr 19
Source code (tar.gz)		Apr 19

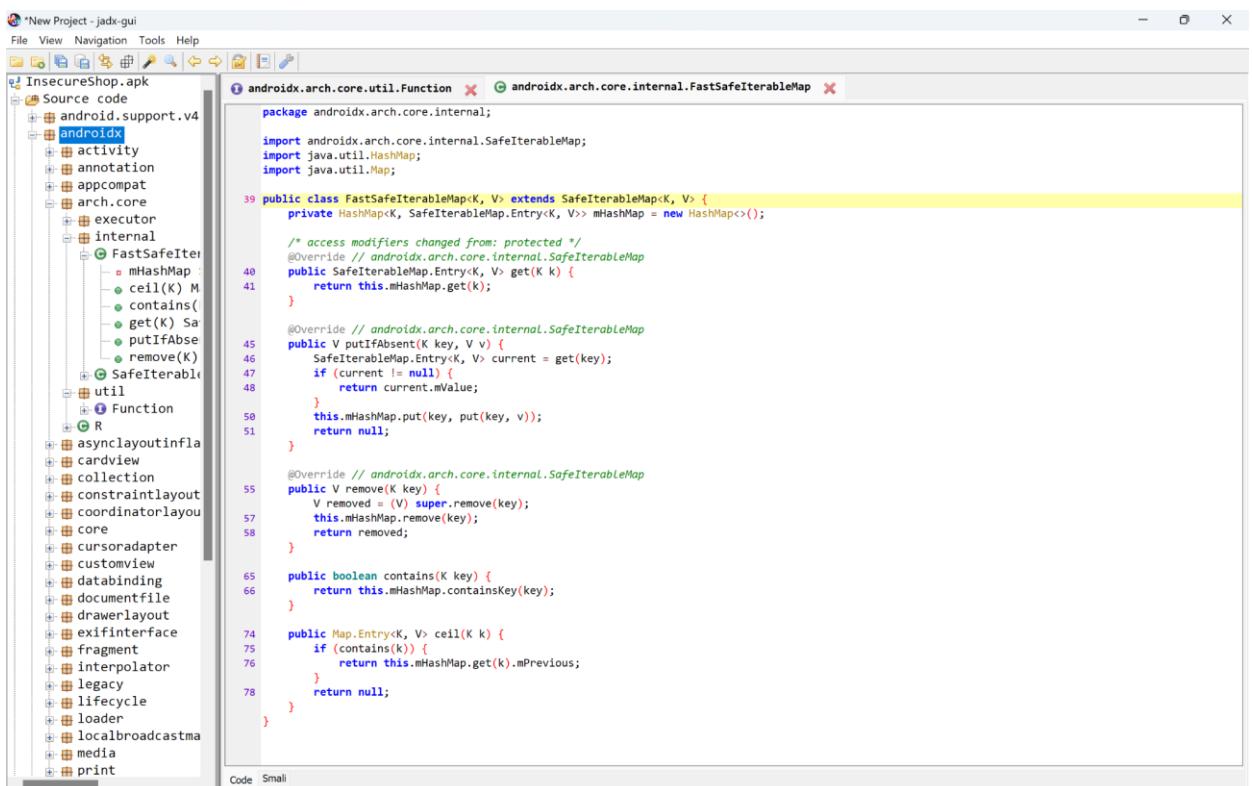
2.4 Below is an illustration of what we find inside the jadx when we open it. and then we clicked on “jadx-gui”.

The screenshot shows a file explorer window displaying the contents of the jadx-1.4.7\bin folder. Inside, there are four files: jadx, jadx.bat, jadx-gui, and jadx-gui.bat. The file jadx-gui is selected.

2.5 Then the window will open. and we chose “insecureShop” file. then clicked “Open file”.

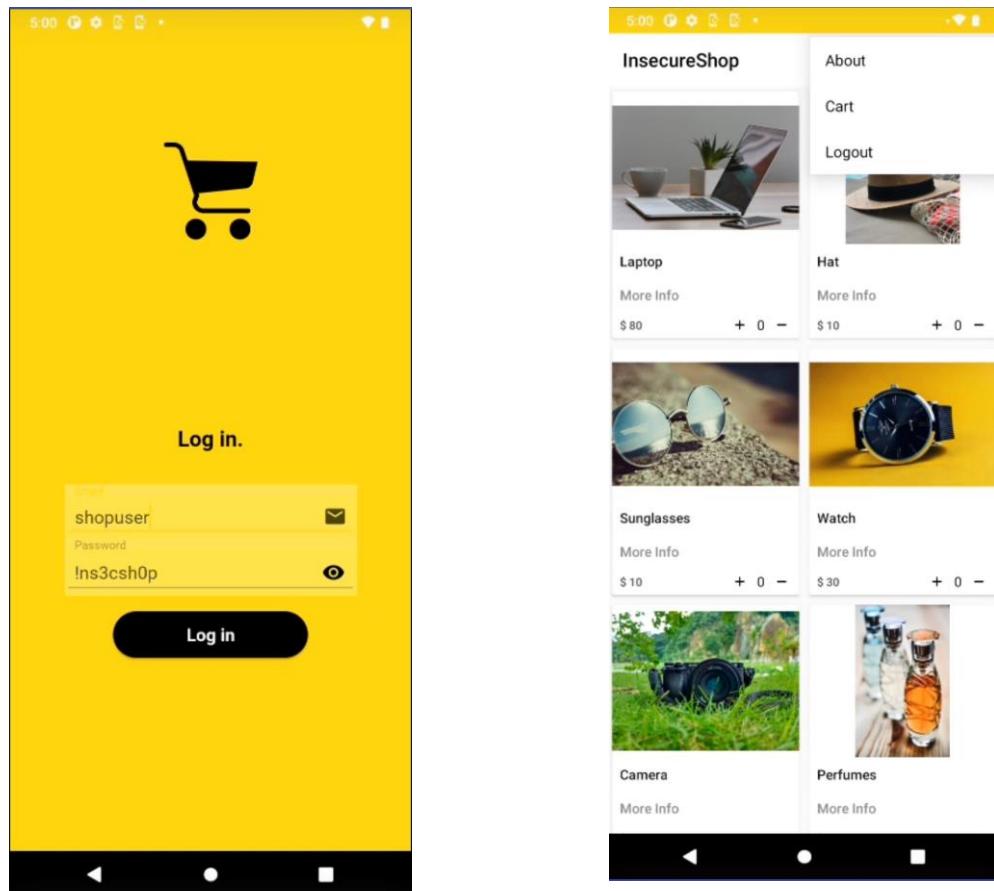


2.6 After selecting the file, jadx will decompile it and display all the Java packages and files in a "tree style view" on the left side of the application.



2.7 After that we opened this file and we found Hardcoded sensitive data(userName, password) which helped us to log in to the application.

2.8 We wrote the username and password information in the application and entered successfully.



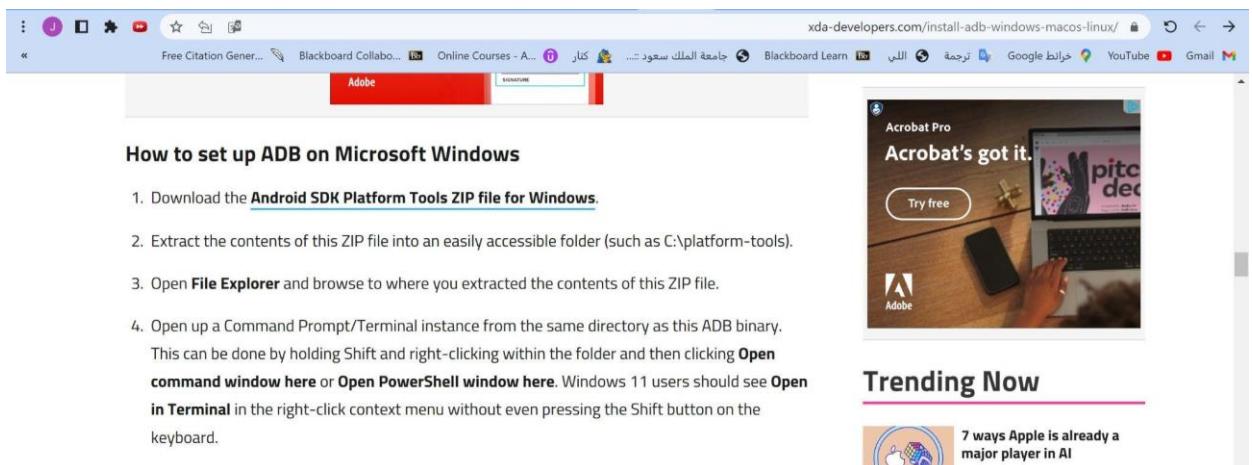
### 3. Android Debug Bridge (ADB) [9].

3.1 The Android Debug Bridge (ADB) is a command-line tool used to communicate with our Lenovo through a terminal.

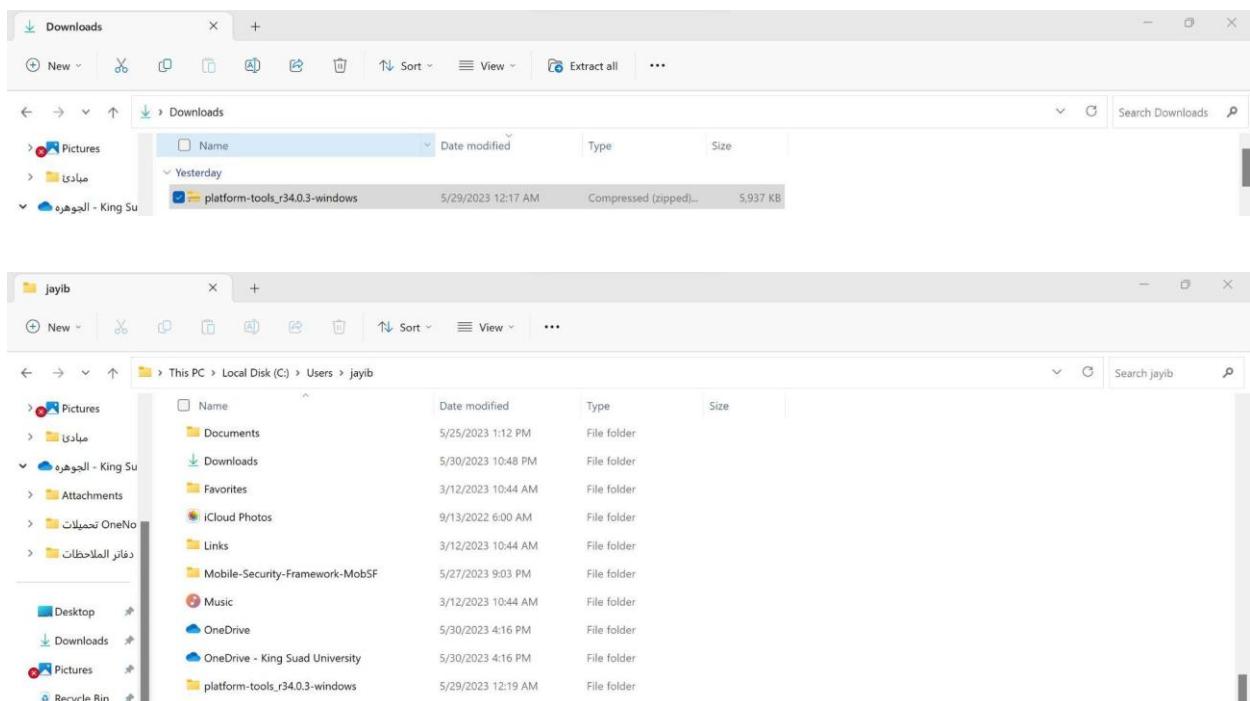
3.2 The following steps were followed to communicate with our lenovo mobile (Lenovo K14) through a terminal:

3.2.1 Initially, go to <https://www.xda-developers.com/install-adb-windows-macos-linux/>

3.2.2 We clicked on “Android SDK Platform Tools ZIP file for Windows”. After that, it began downloading.



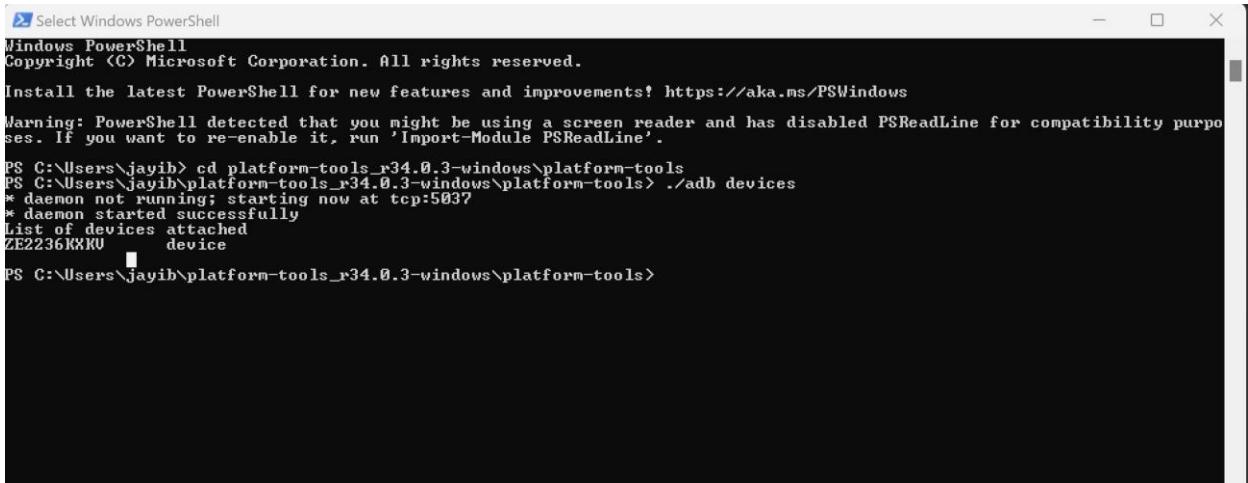
3.2.3 We extracted the zip file.



### 3.3 We opened the command line.

3.3.1 Using the command line, we executed the `./adb devices` command to see if the devices were communicated.

3.3.2 Our Lenovo device's serial number (ZE2236KXKU) has been displayed. This indicates that the communication is successful.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

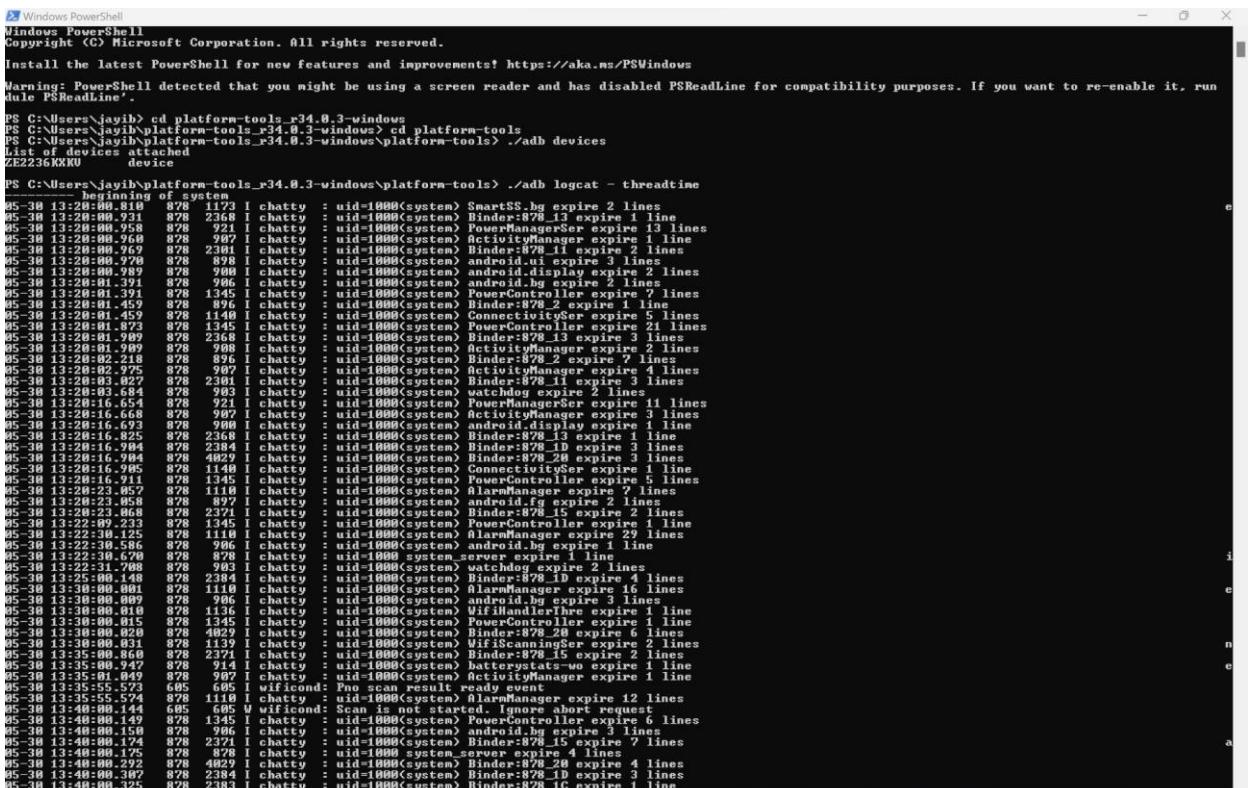
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\jayib> cd platform-tools_r34.0.3-windows\platform-tools
PS C:\Users\jayib\platform-tools_r34.0.3-windows\platform-tools> ./adb devices
* daemon not running; starting now at tcp:5037
* daemon started successfully
List of devices attached
ZE2236KXKU   device

PS C:\Users\jayib\platform-tools_r34.0.3-windows\platform-tools>
```

3.3.3 After executing the command `./adb logcat - threadtime` as demonstrated in the picture below, the logs of any changes in the emulator were displayed. and When we clicked anywhere on the page in the emulator, it appeared in the command line screen with all information for that log activity.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\jayib> cd platform-tools_r34.0.3-windows\platform-tools
PS C:\Users\jayib\platform-tools_r34.0.3-windows\platform-tools> cd platform-tools
PS C:\Users\jayib\platform-tools_r34.0.3-windows\platform-tools> ./adb devices
List of devices attached
ZE2236KXKU   device

PS C:\Users\jayib\platform-tools_r34.0.3-windows\platform-tools> ./adb logcat - threadtime
beginning of system
05-30 13:20:00.921 878 2368 I chatty : uid=1000<(system) SmartSS by expire 2 lines
05-30 13:20:00.921 878 2368 I chatty : uid=1000<(system) Binder:878-13 expire 4 lines
05-30 13:20:00.958 878 921 I chatty : uid=1000<(system) PowerManagerService expire 13 lines
05-30 13:20:00.960 878 907 I chatty : uid=1000<(system) ActivityManager expire 1 line
05-30 13:20:00.960 878 2368 I chatty : uid=1000<(system) Binder:878-11 expire 1 lines
05-30 13:20:00.960 878 907 I chatty : uid=1000<(system) Binder:878-10 expire 1 lines
05-30 13:20:00.989 878 900 I chatty : uid=1000<(system) android.display expire 2 lines
05-30 13:20:01.391 878 1345 I chatty : uid=1000<(system) android.bg expire 2 lines
05-30 13:20:01.391 878 906 I chatty : uid=1000<(system) PowerController expire 7 lines
05-30 13:20:01.400 878 906 I chatty : uid=1000<(system) Binder:878-12 expire 1 lines
05-30 13:20:01.459 878 1140 I chatty : uid=1000<(system) ConnectivityService expire 5 lines
05-30 13:20:01.873 878 1345 I chatty : uid=1000<(system) PowerController expire 21 lines
05-30 13:20:01.989 878 2368 I chatty : uid=1000<(system) Binder:878-13 expire 3 lines
05-30 13:20:01.990 878 907 I chatty : uid=1000<(system) ActivityManager expire 2 lines
05-30 13:20:01.998 878 906 I chatty : uid=1000<(system) Binder:878-12 expire 1 lines
05-30 13:20:02.925 878 907 I chatty : uid=1000<(system) ActivityManager expire 4 lines
05-30 13:20:03.027 878 2301 I chatty : uid=1000<(system) Binder:878-11 expire 3 lines
05-30 13:20:03.684 878 903 I chatty : uid=1000<(system) watchdog expire 2 lines
05-30 13:20:03.684 878 906 I chatty : uid=1000<(system) PowerManagerService expire 11 lines
05-30 13:20:03.684 878 907 I chatty : uid=1000<(system) ActivityManager expire 3 lines
05-30 13:20:03.684 878 900 I chatty : uid=1000<(system) android.display expire 3 lines
05-30 13:20:03.693 878 900 I chatty : uid=1000<(system) Binder:878-13 expire 1 line
05-30 13:20:16.693 878 2368 I chatty : uid=1000<(system) Binder:878-1B expire 3 lines
05-30 13:20:16.693 878 906 I chatty : uid=1000<(system) Binder:878-1B expire 3 lines
05-30 13:20:16.995 878 1140 I chatty : uid=1000<(system) ConnectivityService expire 1 line
05-30 13:20:16.911 878 1345 I chatty : uid=1000<(system) PowerController expire 7 lines
05-30 13:20:23.852 878 1110 I chatty : uid=1000<(system) AlarmManager expire 7 lines
05-30 13:20:23.852 878 906 I chatty : uid=1000<(system) Binder:878-15 expire 2 lines
05-30 13:20:23.858 878 2371 I chatty : uid=1000<(system) Binder:878-15 expire 2 lines
05-30 13:22:09.233 878 1345 I chatty : uid=1000<(system) PowerController expire 1 line
05-30 13:22:30.125 878 1110 I chatty : uid=1000<(system) AlarmManager expire 29 lines
05-30 13:22:30.125 878 906 I chatty : uid=1000<(system) android.bg expire 1 line
05-30 13:22:30.576 878 906 I chatty : uid=1000<(system) Binder:878-1B expire 1 lines
05-30 13:22:31.298 878 903 I chatty : uid=1000<(system) watchdog expire 2 lines
05-30 13:25:00.148 878 2384 I chatty : uid=1000<(system) Binder:878-1D expire 4 lines
05-30 13:30:00.000 878 1110 I chatty : uid=1000<(system) AlarmManager expire 16 lines
05-30 13:30:00.010 878 906 I chatty : uid=1000<(system) Binder:878-1B expire 5 lines
05-30 13:30:00.010 878 1136 I chatty : uid=1000<(system) WifiHandlerService expire 1 line
05-30 13:30:00.015 878 1345 I chatty : uid=1000<(system) PowerController expire 1 line
05-30 13:30:00.020 878 4029 I chatty : uid=1000<(system) Binder:878-2B expire 6 lines
05-30 13:30:00.031 878 1139 I chatty : uid=1000<(system) WifiScanningService expire 2 lines
05-30 13:30:00.031 878 906 I chatty : uid=1000<(system) Binder:878-1B expire 1 lines
05-30 13:35:00.947 878 914 I chatty : uid=1000<(system) batterystats-ws expire 1 line
05-30 13:35:01.049 878 907 I chatty : uid=1000<(system) ActivityManager expire 1 line
05-30 13:35:55.573 605 605 I wificond: Pno icac result ready event
05-30 13:37:00.000 878 1100 I chatty : uid=1000<(system) AlarmManager expires 12 lines
05-30 13:40:00.144 605 605 U wificond: Ssl is not supported, so we skip about request
05-30 13:40:00.144 605 605 U wificond: PowerController expire 6 lines
05-30 13:40:00.150 878 906 I chatty : uid=1000<(system) android.bg expire 3 lines
05-30 13:40:00.174 878 2384 I chatty : uid=1000<(system) Binder:878-15 expire 7 lines
05-30 13:40:00.174 878 906 I chatty : uid=1000<(system) Binder:878-1B expire 1 lines
05-30 13:40:00.292 878 4029 I chatty : uid=1000<(system) Binder:878-2B expire 4 lines
05-30 13:40:00.307 878 2384 I chatty : uid=1000<(system) Binder:878-1D expire 3 lines
05-30 13:40:00.325 878 2383 I chatty : uid=1000<(system) Binder:878-1C expire 1 line
```

## 4. Installing MobSF [3][4] .

MobSF is a powerful tool in the field of mobile security that is available as an open-source software. It is capable of conducting security assessments, such as penetration testing, on various platforms including Android, Windows, and iOS.

Prior to initiating the download and launch of MobSF, we had to download certain programs such as Java, Python, Visual studio, Git, OpenSSL and WkHTMLtox.

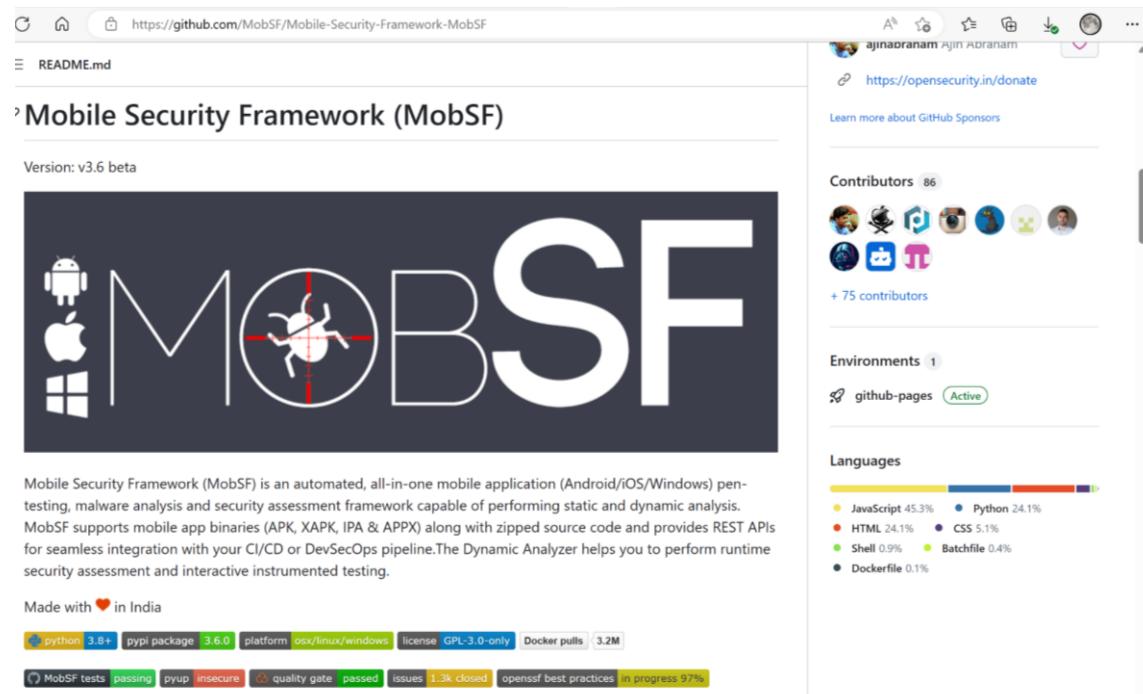
After downloading all these required tools, we started setting up MobSF.

## 5. Setup MobSF [13].

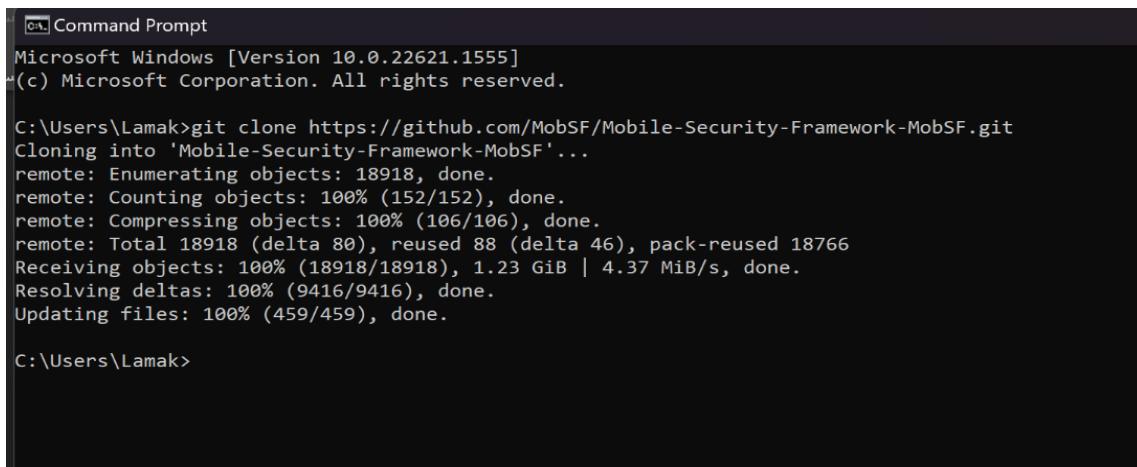
After we set up all required programs to run MobSF, finally the system is ready to install MobSF.

5.1 We used this link to download MobSF .

[MobSF/Mobile-Security-Framework-MobSF: Mobile Security Framework \(MobSF\) is an automated, all-in-one mobile application \(Android/iOS/Windows\) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. \(github.com\)](https://github.com/MobSF/Mobile-Security-Framework-MobSF)



5.2 After completing the installation steps and requirements in Task#2 section 2, we opened the Command Line and wrote this command “`git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF`”, then we waited until it finished downloading.



```
Command Prompt
Microsoft Windows [Version 10.0.22621.1555]
(c) Microsoft Corporation. All rights reserved.

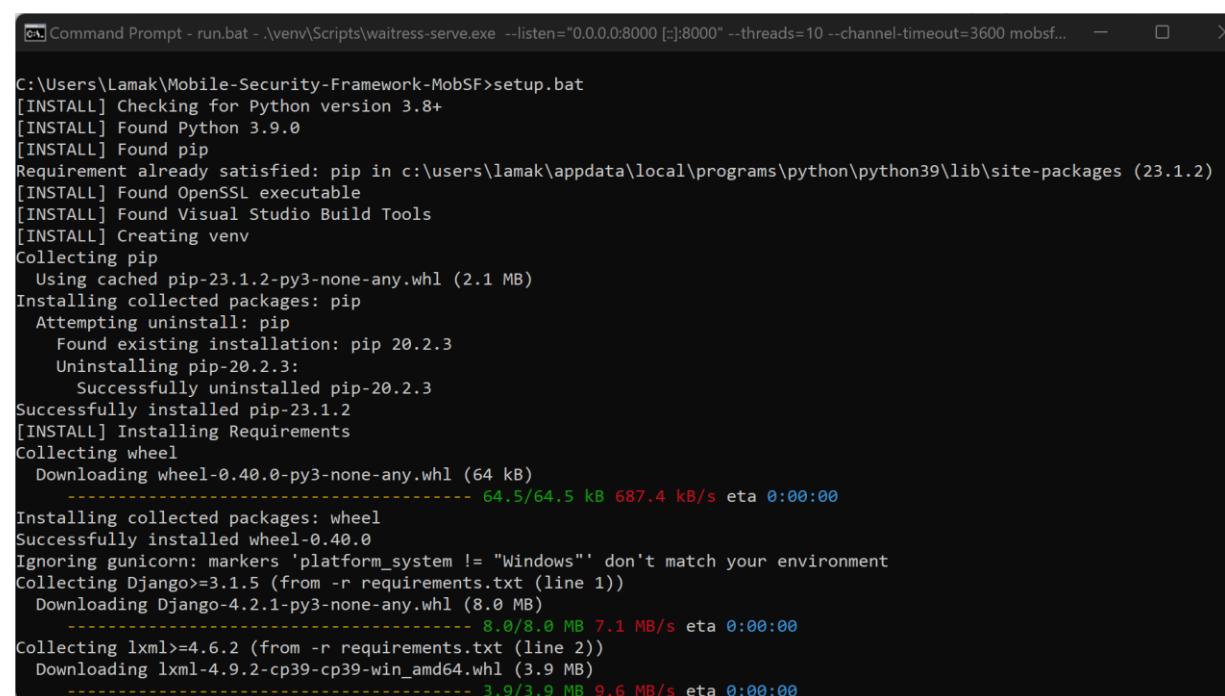
C:\Users\Lamak>git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 18918, done.
remote: Counting objects: 100% (152/152), done.
remote: Compressing objects: 100% (106/106), done.
remote: Total 18918 (delta 80), reused 88 (delta 46), pack-reused 18766
Receiving objects: 100% (18918/18918), 1.23 GiB | 4.37 MiB/s, done.
Resolving deltas: 100% (9416/9416), done.
Updating files: 100% (459/459), done.

C:\Users\Lamak>
```

5.3 Then we moved into the Mobile-Security-Framework-MobSF directory by running this command “`cd Mobile-Security-Framework-MobSF`”.

```
C:\Users\Lamak>cd Mobile-Security-Framework-MobSF
```

5.4 Then we ran this command “`setup.bat`”, then we waited until all installation of packages finished.



```
Command Prompt - run.bat - .\venv\Scripts\waitress-serve.exe --listen="0.0.0.0:8000 [::]:8000" --threads=10 --channel-timeout=3600 mobsf... - □ >

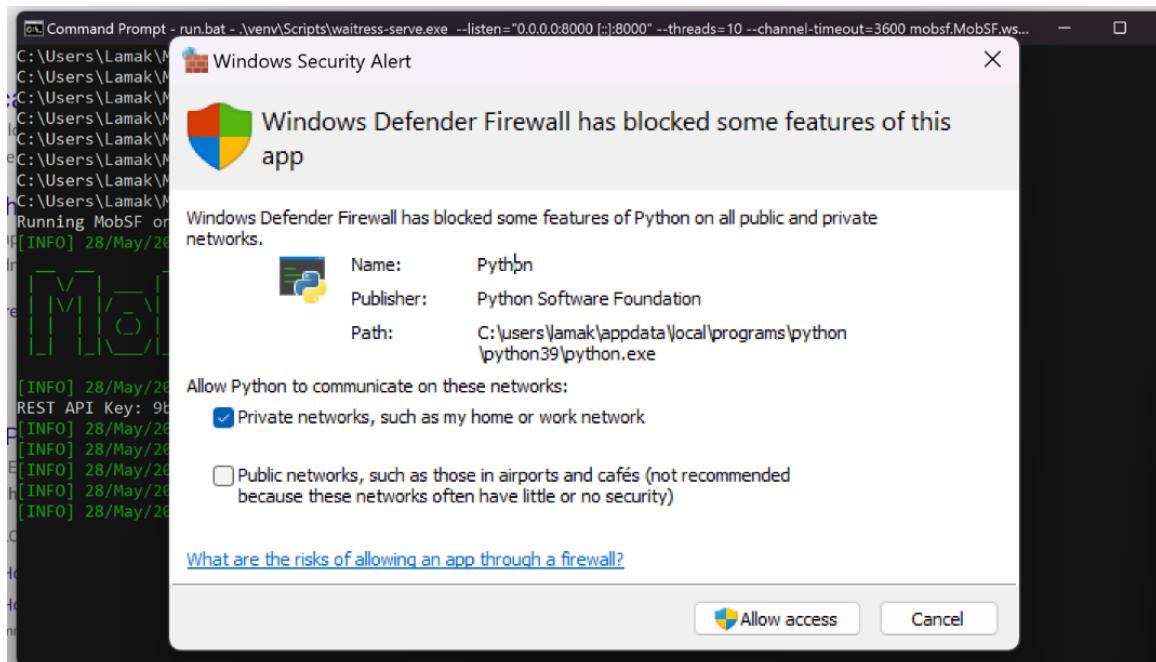
C:\Users\Lamak\Mobile-Security-Framework-MobSF>setup.bat
[INSTALL] Checking for Python version 3.8+
[INSTALL] Found Python 3.9.0
[INSTALL] Found pip
Requirement already satisfied: pip in c:\users\lamak\appdata\local\programs\python\python39\lib\site-packages (23.1.2)
[INSTALL] Found OpenSSL executable
[INSTALL] Found Visual Studio Build Tools
[INSTALL] Creating venv
Collecting pip
  Using cached pip-23.1.2-py3-none-any.whl (2.1 MB)
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 20.2.3
    Uninstalling pip-20.2.3:
      Successfully uninstalled pip-20.2.3
Successfully installed pip-23.1.2
[INSTALL] Installing Requirements
Collecting wheel
  Downloading wheel-0.40.0-py3-none-any.whl (64 kB)
    64.5/64.5 kB 687.4 kB/s eta 0:00:00
Installing collected packages: wheel
Successfully installed wheel-0.40.0
Ignoring gunicorn: markers 'platform_system != "Windows"' don't match your environment
Collecting Django>=3.1.5 (from -r requirements.txt (line 1))
  Downloading Django-4.2.1-py3-none-any.whl (8.0 MB)
    8.0/8.0 MB 7.1 MB/s eta 0:00:00
Collecting lxml>=4.6.2 (from -r requirements.txt (line 2))
  Downloading lxml-4.9.2-cp39-cp39-win_amd64.whl (3.9 MB)
    3.9/3.9 MB 9.6 MB/s eta 0:00:00
```

```
Command Prompt - run.bat - .\venv\Scripts\waitress-serve.exe --listen="0.0.0.0:8000 [::]:8000" --threads=10 --channel-timeout=3600 mobsf...
Collecting lxml>=4.6.2 (from -r requirements.txt (line 2))
  Downloading lxml-4.9.2-cp39-cp39-win_amd64.whl (3.9 MB)
    3.9/3.9 MB 9.6 MB/s eta 0:00:00
Collecting rsa>=4.7 (from -r requirements.txt (line 3))
  Downloading rsa-4.9-py3-none-any.whl (34 kB)
Collecting biplist>=1.0.3 (from -r requirements.txt (line 4))
  Downloading biplist-1.0.3.tar.gz (21 kB)
  Preparing metadata (setup.py) ... done
Collecting requests>=2.25.1 (from -r requirements.txt (line 5))
  Downloading requests-2.31.0-py3-none-any.whl (62 kB)
    62.6/62.6 kB ? eta 0:00:00
Collecting bs4>=0.0.1 (from -r requirements.txt (line 6))
  Downloading bs4-0.0.1.tar.gz (1.1 kB)
  Preparing metadata (setup.py) ... done
Collecting colorlog>=4.7.2 (from -r requirements.txt (line 7))
  Downloading colorlog-6.7.0-py2.py3-none-any.whl (11 kB)
Collecting macholib>=1.14 (from -r requirements.txt (line 8))
  Downloading macholib-1.16.2-py2.py3-none-any.whl (38 kB)
Collecting whitenoise>=5.2.0 (from -r requirements.txt (line 9))
  Downloading whitenoise-6.4.0-py3-none-any.whl (19 kB)
Collecting waitress>=1.4.4 (from -r requirements.txt (line 10))
  Downloading waitress-2.1.2-py3-none-any.whl (57 kB)
    57.7/57.7 kB ? eta 0:00:00
Collecting psutil>=5.8.0 (from -r requirements.txt (line 12))
  Downloading psutil-5.9.5-cp36-abi3-win_amd64.whl (255 kB)
    255.1/255.1 kB ? eta 0:00:00
Collecting shelljob>=0.6.2 (from -r requirements.txt (line 13))
  Downloading shelljob-0.6.3-py3-none-any.whl (9.9 kB)
Collecting asn1crypto>=1.4.0 (from -r requirements.txt (line 14))
```

```
Command Prompt - run.bat - .\venv\Scripts\waitress-serve.exe --listen="0.0.0.0:8000 [::]:8000" --threads=10 --channel-timeout=3600 mobsf...
4561186ac779ef0d4830f899a0573
  Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\50\93\f8\4f0a42a03a06626d675f13907b6982ad5ecff383530af5a900
  Building wheel for kaitaistruct (setup.py) ... done
  Created wheel for kaitaistruct: filename=kaitaistruct-0.9-py2.py3-none-any.whl size=5533 sha256=c0f27504b7476fe2ad2315bb2124fd13b9027157c33b6997a31188b78e4538bf
  Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\da\13\6a\51f1f909ab2ea9ac43db5616577860868944de1d45d2f098f
  Building wheel for pyperclip (setup.py) ... done
  Created wheel for pyperclip: filename=pyperclip-1.8.2-py3-none-any.whl size=11126 sha256=84dd4ff59b4afec5d1c23a6016503d81844665dd493674e8e34387d513335404
  Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\0c\09\9e\49e21a6840ef7955b06d47394afe0058f0378c0914e48b8b8
  Building wheel for urwid (setup.py) ... done
  Created wheel for urwid: filename=urwid-2.1.2-py3-none-any.whl size=235048 sha256=8b5399117fa7d0dbb4747d92164979c5e0655eb9307e25894da79d9212141a97
  Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\44\ec\04\2c1080c3ee4e80e76d662ac35f0594a2a86f9df12095b05cb3
Successfully built biplist bs4 http-tools libsast frida blinker kaitaistruct pyperclip urwid
Installing collected packages: typing-extensions, asgiref, sqlparse, tzdata, Django, lxml, pyasn1, rsa, biplist, charset-normalizer, idna, urllib3, certifi, requests, soupsieve, beautifulsoup4, bs4, colorama, colorlog, altgraph, macholib, whitenoise, waitress, psutil, shelljob, asn1crypto, oscrypto, distro, IP2Location, lief, blinker, Brotli, click, six, pycparser, cffi, cryptography, Werkzeug, markupsafe, Jinja2, itsdangerous, flask, hyperframe, hpack, h2, kaitaistruct, ldap3, msgpack, protobuf, pyOpenSSL, pyparsing, pyperclip, ruamel.yaml, ruamel.yaml, sortedcontainers, tornado, urwid, h11, wsproto, publicsuffix2, zstandard, pydivert, mitmproxy, http-tools, pyyaml, libsast, pdfkit, google-play-scraper, networkx, pygments, numpy, contourpy, cypher, fonttools, kiwisolver, packaging, pillow, python-dateutil, zipp, importlib-resources, matplotlib, pydot, backcall, decorator, parso, jedi, traitlets, matplotlib-inline, pickleshare, wcwidth, prompt-toolkit, executing, asttokens, pure-eval, stack-data, ipython, androguard, yara-python-dex, apkid, prettytable, tqdm, graphviz, pytz, pandas, tenacity, plotly, rzpipe, kaleido, quark-engine, frida, requests-file, filelock, tldextract, openstep-parser, svutils
```

```
[Select Command Prompt]
- Create model SuppressFindings
[INFO] 28/May/2023 00:56:54 - Checking for Update.
[INFO] 28/May/2023 00:56:55 - No updates available.
[INFO] 28/May/2023 00:56:57 -  
  
[INFO] 28/May/2023 00:56:57 - Mobile Security Framework v3.6.7 Beta
REST API Key: 9b20ec28fbf832c8ccddb4f8c90f406f76f0767baa5da1754f978dbc7a923e5
[INFO] 28/May/2023 00:56:57 - OS: Windows
[INFO] 28/May/2023 00:56:57 - Platform: Windows-10-10.0.22621-SP0
[INFO] 28/May/2023 00:56:57 - MobSF Basic Environment Check
Operations to perform:
  Apply all migrations: StaticAnalyzer, auth, contenttypes, sessions
Running migrations:
  Applying StaticAnalyzer.0001_initial... OK
  Applying contenttypes.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0001_initial... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... [INFO] 28/May/2023 00:56:59 - Checking for Update.
OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
```

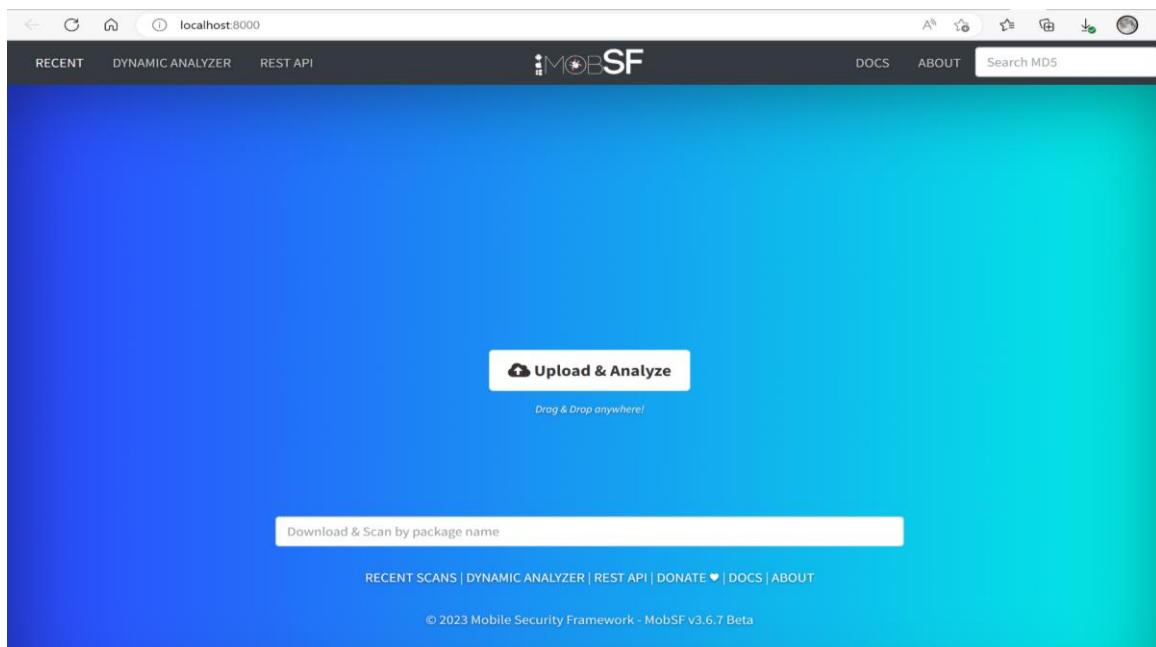
5.5 Once it finished downloading, this window appeared asking to allow access, we clicked “Allow Access”.



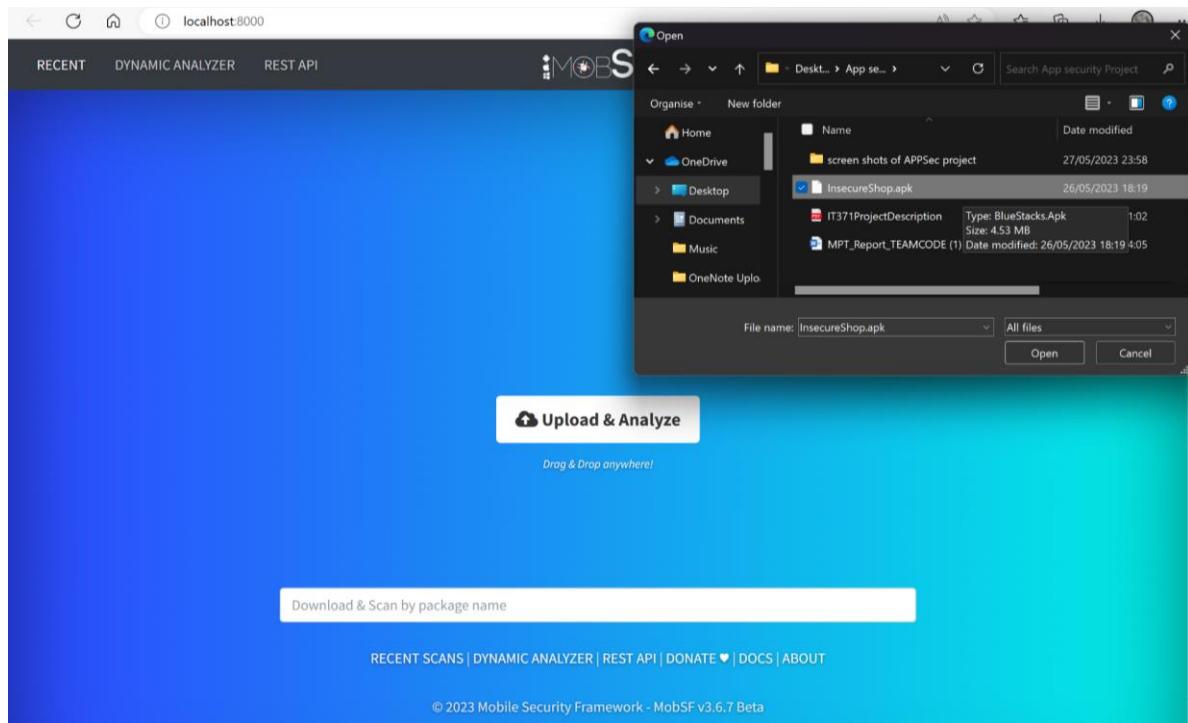
5.6 Then we run this command “run.bat”.

```
C:\Users\lamak\Mobile-Security-Framework-MobSF>run.bat
Running MobSF on "0.0.0.0:8000 [::]:8000"
```

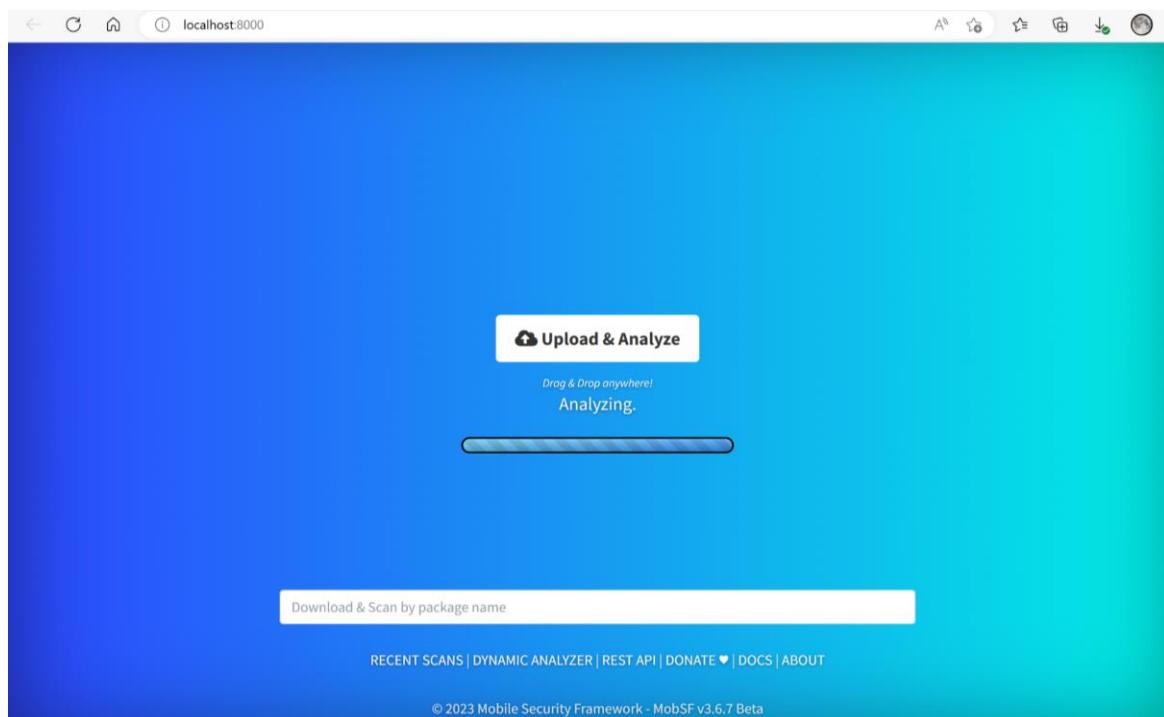
5.7 Then we opened the browser and searched “localhost:8000”. And this page appeared.



5.8 We upload the insecureshop.apk file here.



5.9 MobSF starts analyzing the apk file.



## 5.10 After analysis is completed, the analysis appears in details.

**APP SCORES**

**FILE INFORMATION**

**APP INFORMATION**

**ACTIVITIES**: 10 (View)

**SERVICES**: 1 (View)

**RECEIVERS**: 0 (View)

**PROVIDERS**: 2 (View)

**EXPORTED ACTIVITIES**: 5

**EXPORTED SERVICES**: 1

**EXPORTED RECEIVERS**: 0

**EXPORTED PROVIDERS**: 1

**SIGNER CERTIFICATE**

APK is signed  
v1 signature: True  
v2 signature: True  
v3 signature: False  
Found 1 unique certificates  
Subject: CN=Android Debug, O=Android, C=US  
Signature Algorithm: rsassa\_pkcs1v15  
Valid From: 2016-09-01 10:14:25+00:00  
Valid To: 2046-08-30 10:14:25+00:00  
Issuer: CN=Android Debug, O=Android, C=US  
Serial Number: 0x1  
Hash Algorithm: sha1  
md5: 5c935bda969c51ea7d7f5e52650f358  
sha1: c56a7946caf6923ced4cf7f4c6b0e5b0e97df26b  
sha256: d16dff509803ba1123ec7573cc18c58bde996ca05bae3efe852fb3c668cfca8  
sha512: 6bba4313106fb6844d401dc39d2a65943df9d9bd832d5c58932c3c936ab65285db35a78ace7f4fdaec28b839acd2e50a1cd0c88f2efe128268384d  
PublicKey Algorithm: rsa  
Bit Size: 1024  
Fingerprint: 782478518db2c6c714a79b619561a97ad55a8f718e05f70341a0bafffe93b22f

**APPLICATION PERMISSIONS**

PERMISSION	STATUS	INFO	DESCRIPTION

## 4 Detailed Findings

### 4.1 Limitations

During the mobile penetration testing we did, we faced some issues and challenges such as :

- 1- The installation of all the required tools was not only time-consuming but also occupied a significant amount of memory space.
- 2- MobSF required a lot of tools and prerequisites such as (python, java, wkhtmltox, etc..).
- 3- We faced issues during the setup of MobSF due to the installation of the latest version of python, and we wasted a lot of time trying to configure the problem because the error was not clear.
- 4-When we opened Jadex after downloading, a message appeared that prevented us from entering the tool "Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk". After that, we deleted the folder and downloaded it again, then it opened successfully.

### 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.

Severity Level		Risk Severity Level of the Vulnerability								
Technical Impact	Impact level	Likelihood		Popularity and Simplicity						
		Popularity	Popularity of the Exploit	Simplicity	Simplicity to Exploit					
Observation	<b>Our observation and description about the vulnerability</b>									
Impact	<b>Describes the possible Technical Impact if the vulnerability is Exploited</b>									
Platform	<b>Describe the platform</b>									
<b>Proof of Concept</b>										
<b>Evidence to support the finding</b>										
Recommendation	<b>High Level recommendation to mitigate the vulnerability</b>									
OWASP Reference	<b>OWASP Reference</b>									
Reference	<b>Reference related to the vulnerability</b>									

#### **4.1.1 Insecure Communication**

Severity Level	High	Likelihood			Medium
Technical Impact	High	Popularity	Medium	Simplicity	Medium
Observation	While observing CustomWebViewClient.java, we found out that method <code>onReceivedSslError</code> is called when there is an SSL error while loading a webpage in the WebView. we found out that the purpose of this code is to handle SSL errors by allowing the connection to proceed, even if there is an SSL error. The method <code>handler.proceed()</code> is called to continue with the connection, bypassing the SSL error.				
Impact	This code can pose a potential security risk as it allows connections to proceed even if there are SSL errors. This can potentially allow malicious actors to intercept and modify network traffic, which can lead to security breaches.				
Platform	Android				

## Proof of Concept

3	Insecure WebView Implementation. WebView ignores SSL Certificate errors and accept any SSL Certificate. This application is vulnerable to MITM attacks	high	<b>CWE:</b> CWE-295: Improper Certificate Validation <b>OWASP Top 10:</b> M3: Insecure Communication <b>OWASP MASVS:</b> MSTG- NETWORK-3	<a href="#">com/insecureshop/util/CustomWebViewClient.java</a>
<pre>package com.insecureshop.util;  import android.net.http.SslError; import android.webkit.SslErrorHandler; import android.webkit.WebView; import android.webkit.WebViewClient; import kotlin.Metadata; /* compiled from: CustomWebViewClient.kt */ @Metadata(bv = {1, 0, 3}, d1 = {"\u00000\$\\n\u0002\u0018\u0002\\n\u0002\\u0018\u0002\\n\\u0002\\b\\u0002\\n\\u0002", "loaded from: classes.dex"}) public final class CustomWebViewClient extends WebViewClient {     @Override // android.webkit.WebViewClient     public void onReceivedSslError(WebView view, SslErrorHandler handler, SslError error) {         if (handler != null) {             handler.proceed();         }     } }</pre>				

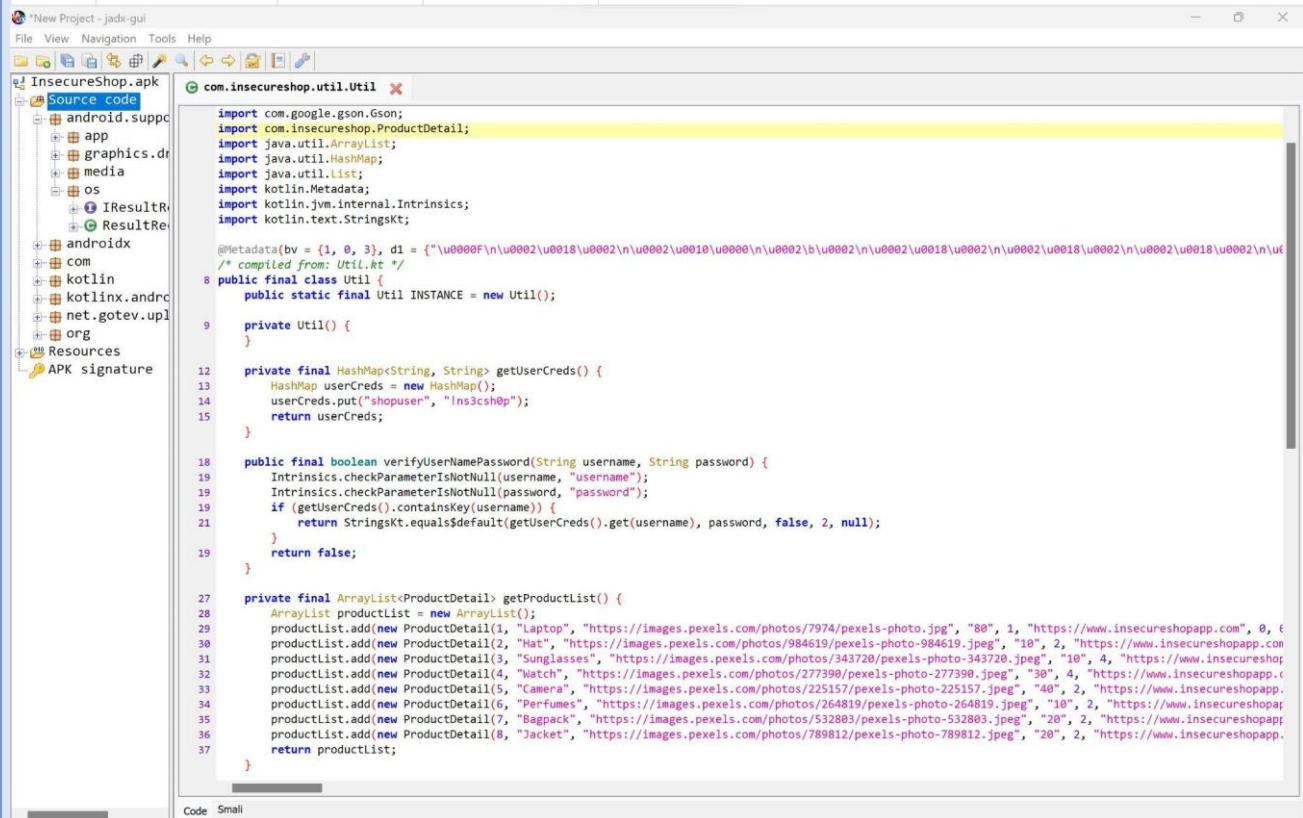
Recommendation	<ul style="list-style-type: none"><li>- Use strong encryption algorithms to protect data confidentiality and integrity.</li><li>- Use transport layer security (TLS) or secure sockets layer (SSL) protocols to encrypt data in transit and protect it from interception or tampering.</li></ul>
OWASP reference	<a href="#"><u>M3: Insecure Communication   OWASP Foundation</u></a>
Reference	<a href="#"><u>owasp-mstg/0x05g-Testing-Network-Communication.md at master · MobSF/owasp-mstg (github.com)</u></a>

#### **4.1.2 Hardcoding sensitive data in the source code.**

Severity Level	Medium				
Technical Impact	High	Likelihood		Impact	
		Popularity	High	Simplicity	Low
Observation	Upon uploading the APK file in jadx, we noticed that the password and the username were easily accessible as it were written in plain text within the source code of the. This renders the application vulnerable to attacks from any source.				
Impact	It can compromise the confidentiality and integrity of the data, it can be easily accessed by anyone who has access to the code, including developers, testers, and potential attackers.				
Platform	Android				

## Proof of Concept

2	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	warning	<b>CWE:</b> CWE-312: Cleartext Storage of Sensitive Information <b>OWASP Top 10:</b> M9: Reverse Engineering <b>OWASP MASVS:</b> MSTG-STORAGE-14	<a href="#">com/bumptech/glide/load/Option.java</a> <a href="#">com/bumptech/glide/load/engine/DataCacheKey.java</a> <a href="#">com/bumptech/glide/load/engine/EngineResource.java</a> <a href="#">com/bumptech/glide/load/engine/ResourceCacheKey.java</a> <a href="#">com/bumptech/glide/manager/RequestManagerRetriever.java</a>
---	--	---------	---	--



<b>Recommendation</b>	<ul style="list-style-type: none"><li>- Store sensitive data such as passwords securely in a separate configuration file or a secure key store rather than hardcoding them in the source code.</li><li>- Use encryption or hashing algorithms to protect sensitive data.</li></ul>
-----------------------	--

<b>OWASP reference</b>	<a href="#"><b>M9: Reverse Engineering   OWASP Foundation</b></a>
<b>Reference</b>	<p><a href="https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE">https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE</a></p> <p><a href="https://github.com/MobSF/owasp-mstg/blob/0x05d-Testing-Data-Storage.md">owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)</a></p> <p><a href="https://cwe.mitre.org/data/definitions/798.html">https://cwe.mitre.org/data/definitions/798.html</a></p>

#### 4.1.3 Insecure Logging of sensitive data.

<b>Severity Level</b>	<b>High</b>				
<b>Technical Impact</b>	High		<b>Likelihood</b>	<b>High</b>	
			<b>Popularity</b>	Medium	<b>Simplicity</b>
<b>Observation</b>	Reviewing the source code of com.insecureshop.LoginActivity shows that the username and password values that we provide are being leaked and logged as a plaintext.				
<b>Impact</b>	Logging sensitive user data can create an additional and less secure pathway for attackers to acquire confidential information. In the present scenario, an attacker may potentially gain access to user accounts by using the password shown in the logs, which would violate both authentication and confidentiality protocols.				
<b>Platform</b>	<b>Android</b>				

#### Proof of Concept

1	The App logs information. Sensitive information should never be logged.		<b>CWE:</b> CWE-532: Insertion of Sensitive Information into Log File <b>OWASP MASVS:</b> MSTG-STORAGE-3	<a href="#">com/bumptech/glide/Glide.java</a> <a href="#">com/bumptech/glide/gifdecoder/GifHeaderParser.java</a> <a href="#">com/bumptech/glide/gifdecoder/StandardGifDecoder.java</a> <a href="#">com/bumptech/glide/load/data/AssetPathFetcher.java</a> <a href="#">com/bumptech/glide/load/data/HttpUrlFetcher.java</a> <a href="#">com/bumptech/glide/load/data/LocalUriFetcher.java</a> <a href="#">com/bumptech/glide/load/data/mediastore/ThumbFetcher.java</a> <a href="#">com/bumptech/glide/load/data/mediastore/ThumbnailStreamOpener.java</a> <a href="#">com/bumptech/glide/load/engine/DecodeJob.java</a> <a href="#">com/bumptech/glide/load/engine/DecodePath.java</a> <a href="#">com/bumptech/glide/load/engine/Engine.java</a> <a href="#">com/bumptech/glide/load/engine/GlideException.java</a> <a href="#">com/bumptech/glide/load/engine/SourceGenerator.java</a>
---	---	--	---	--

```

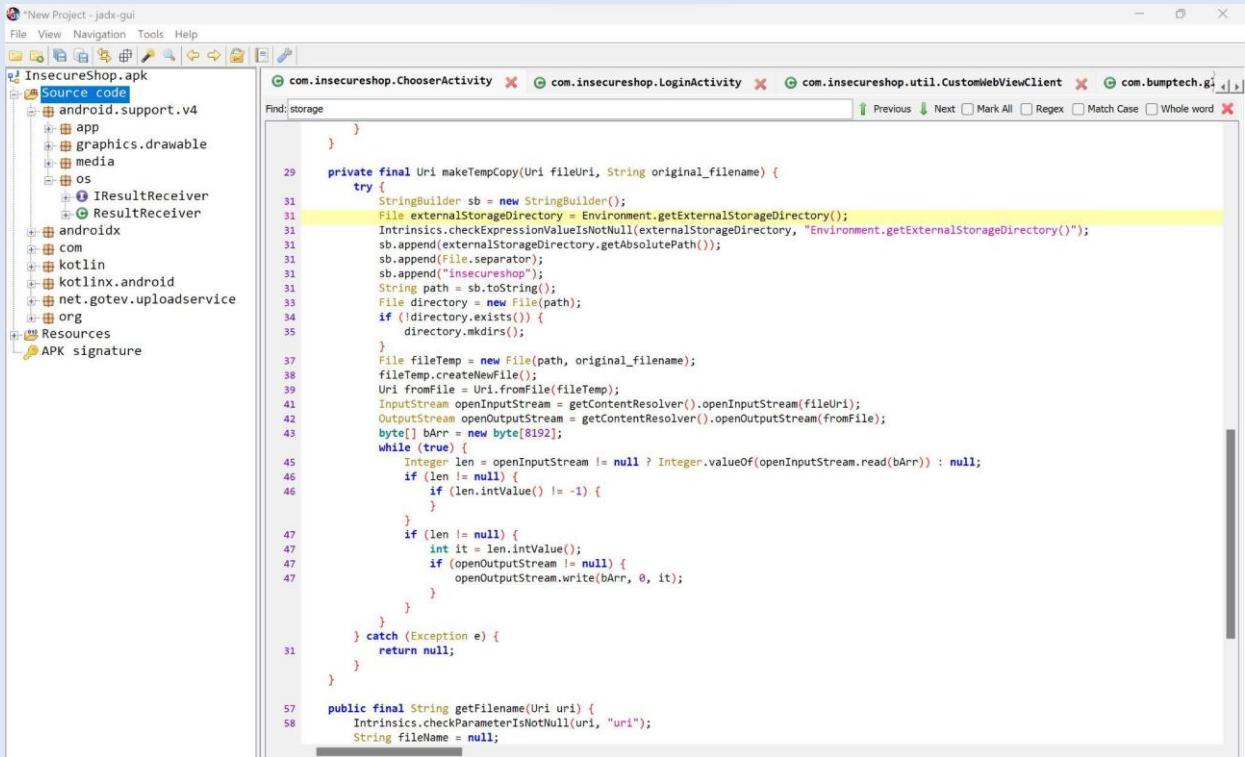
public final void onLogin(View view) {
    Intrinsics.checkNotNull(view, "view");
    ActivityLoginBinding activityLoginBinding = this.mBinding;
    if (activityLoginBinding == null) {
        Intrinsics.throwUninitializedPropertyAccessException("mBinding");
    }
    TextInputEditText editText = activityLoginBinding.edtUserName;
    Intrinsics.checkNotNull(editText, "mBinding.edtUserName");
    String username = String.valueOf(editText.getText());
    ActivityLoginBinding activityLoginBinding2 = this.mBinding;
    if (activityLoginBinding2 == null) {
        Intrinsics.throwUninitializedPropertyAccessException("mBinding");
    }
    TextInputEditText editText2 = activityLoginBinding2.edtPassword;
    Intrinsics.checkNotNull(editText2, "mBinding.edtPassword");
    String password = String.valueOf(editText2.getText());
    Log.d("username", username);
    Log.d("password", password);
    boolean auth = Util.INSTANCE.verifyUserNamePassword(username, password);
}

```

<b>Recommendation</b>	<ul style="list-style-type: none"> <li>- Use secure logging frameworks, and tools that can perform the encryption.</li> <li>- Conduct regular security assessments and penetration testing to identify and address any vulnerabilities in the application's logging practices.</li> </ul>
<b>OWASP reference</b>	<a href="#">OWASP Top Ten 2017   A10:2017-Insufficient Logging &amp; Monitoring   OWASP Foundation</a>
<b>Reference</b>	<a href="#">owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)</a>

#### 4.1.4 Insecure Data Storage.

Severity Level	Medium				
Technical Impact	High	Likelihood		Medium	
		Popularity	Medium	Simplicity	High
Observation	When observing ChooserActivity.java, we found out that method getExternalStorageDirectory() retrieves the path to the external storage directory on an Android device and stores it in the externalStorageDirectory variable.				
Impact	This code can be used to access and manipulate files stored in the external storage directory of an Android device. Insecure data storage is a vulnerability that can have serious impacts on the confidentiality, integrity, and availability of data. When data is stored in an insecure manner, it can be easily accessed or stolen by attackers, leading to data breaches, identity theft, financial loss, and				

		other malicious activities.		
Platform	Android	Proof of Concept		
<b>4</b> App can read/write to External Storage. Any App can read data written to External Storage.				
	<b>warning</b>	<b>CWE:</b> CWE-276: Incorrect Default Permissions <b>OWASP Top 10:</b> M2: Insecure Data Storage <b>OWASP MASVS:</b> MSTG-STORAGE-2		<a href="#">com/insecureshop/ChooserActivity.java</a>
 <pre> 29     private final Uri makeTempCopy(Uri fileUri, String original_filename) { 30         try { 31             StringBuilder sb = new StringBuilder(); 32             File externalStorageDirectory = Environment.getExternalStorageDirectory(); 33             Intrinsic.checkExpressionValueIsNotNull(externalStorageDirectory, "Environment.getExternalStorageDirectory()"); 34             sb.append(externalStorageDirectory.getAbsolutePath()); 35             sb.append(File.separator); 36             sb.append("insecureshop"); 37             String path = sb.toString(); 38             File directory = new File(path); 39             if (!directory.exists()) { 40                 directory.mkdirs(); 41             } 42             File fileTemp = new File(path, original_filename); 43             fileTemp.createNewFile(); 44             Uri fromFile = Uri.fromFile(fileTemp); 45             InputStream openInputStream = getContentResolver().openInputStream(fileUri); 46             OutputStream openOutputStream = getContentResolver().openOutputStream(fromFile); 47             byte[] bArr = new byte[8192]; 48             while (true) { 49                 Integer len = openInputStream != null ? Integer.valueOf(openInputStream.read(bArr)) : null; 50                 if (len != null) { 51                     if (len.intValue() != -1) { 52                         if (len != null) { 53                             int it = len.intValue(); 54                             if (openOutputStream != null) { 55                                 openOutputStream.write(bArr, 0, it); 56                             } 57                         } 58                     } 59                 } catch (Exception e) { 60                     return null; 61                 } 62             } 63         public final String getFilename(Uri uri) { 64             Intrinsic.checkNotNull(uri, "uri"); 65             String fileName = null; 66         } 67     } </pre>				
Recommendations	<ul style="list-style-type: none"> <li>- Store sensitive data such as passwords, credit card details in secure databases or file systems with appropriate access controls and permissions.</li> <li>- Use IPS help to mitigate the vulnerability of insecure data storage by detecting and blocking malicious traffic that may attempt to exploit vulnerabilities in the data storage system.</li> </ul>			
OWASP Reference	<a href="#">M2: Insecure Data Storage   OWASP Foundation</a>			
Reference	<a href="#">owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)</a> <a href="#">Unpacking Android Security: Part 2 — Insecure Data Storage   by Ed Holloway-George   ProAndroidDev</a>			

#### 4.1.5 Application is enabled for debugging.

<b>Severity Level</b>	<b>High</b>						
<b>Technical Impact</b>	High	<b>Likelihood</b>		<b>MEDIUM</b>			
		<b>Popularity</b>	<b>MEDIUM</b>	<b>Simplicity</b>	<b>LOW</b>		
<b>Observation</b>	While observing the 'AndroidManifest.xml' file, we found out that many permissions and important configurations are written in this file. One of the important things noticed is that "android:debuggable" is set to true, which poses a security risk.						
<b>Impact</b>	Debuggable is set to true during development, it enables developers to identify and fix issues in the code more easily, but leaving an app in debug mode when deploying it to production can pose a security risk as it can allow attackers to gain access to sensitive information and potentially exploit vulnerabilities in the code. Also attackers can perform reverse engineering on the application.						
<b>Platform</b>	<b>Android</b>						
<b>Proof of Concept</b>							
3	Debug Enabled For App [android:debuggable=true]	high	Debugging was enabled on the app which makes it easier for reverse engineers to hook a debugger to it. This allows dumping a stack trace and accessing debugging helper classes.				
<pre>&lt;?xml version="1.0" encoding="utf-8"?&gt; &lt;manifest android:versionCode="1" android:versionName="1.0" android:compileSdkVersion="29" android:compileSdkVersionCodename="10" package="com.insecureshop" platformBuildVersionCode="29" platformBuildVersionName="1.0" xmlns:android="http://schemas.android.com/apk/res/android"&gt;     &lt;uses-sdk android:minSdkVersion="16" android:targetSdkVersion="29" /&gt;     &lt;uses-permission android:name="android.permission.INTERNET" /&gt;     &lt;uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" /&gt;     &lt;uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" /&gt;     &lt;uses-permission android:name="android.permission.READ_CONTACTS" /&gt;     &lt;permission android:name="com.insecureshop.permission.READ" /&gt;     &lt;uses-permission android:name="android.permission.WAKE_LOCK" /&gt;     &lt;application android:theme="@style/AppTheme" android:label="@string/app_name" android:icon="@mipmap/ic_launcher" android:name="com.insecureshop.InsecureShopApp" android:debuggable="true" /&gt; </pre> <p style="background-color: #e0f2e0; padding: 5px; text-align: center;"><b>android:debuggable="true"</b></p>							
<b>Recommendation</b>	Disable debugging mode to prevent access and reverse engineering.						
<b>OWASP Reference</b>	<a href="#">M9: Reverse Engineering   OWASP Foundation</a>						
<b>Reference</b>	<a href="#">android:debuggable   Android Developers</a> <a href="https://cwe.mitre.org/data/definitions/489.html">https://cwe.mitre.org/data/definitions/489.html</a> <a href="https://docs.fluidattacks.com/criteria/vulnerabilities/058/">https://docs.fluidattacks.com/criteria/vulnerabilities/058/</a>						

## Appendix A: About the Team

Team Code:		
Student Name	Serial Number	Role
Lama Alshaya	7	(python, OpenSSL, wkhtmltox, java JDK, MobSF) installation and setup. Documentation.
Aljawharah Alzamil	6	jadx and adb setup and installation, Emulator.

## References :

- [1] Downloads Archive - Genymotion – Android Emulator for app testing
- [2] skylot/jadx: Dex to Java decompiler (github.com)
- [3] MobSF Installation on Windows [Updated] - YouTube
- [4] MobSF/Mobile-Security-Framework-MobSF: Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. (github.com)
- [5] Home | Adoptium
- [6] Welcome to Python.org
- [7] How to Set Up and Run an Android Simulator - YouTube
- [8] Win32/Win64 OpenSSL Installer for Windows - Shining Light Productions (slproweb.com)
- [9] <https://www.xda-developers.com/install-adb-windows-macos-linux/>.
- [10] wkhtmltopdf
- [11] Git - Downloads (git-scm.com)
- [12] Thank you for downloading Visual Studio - Visual Studio (microsoft.com)
- [13] MobSF/Mobile-Security-Framework-MobSF: Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. (github.com)