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

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.

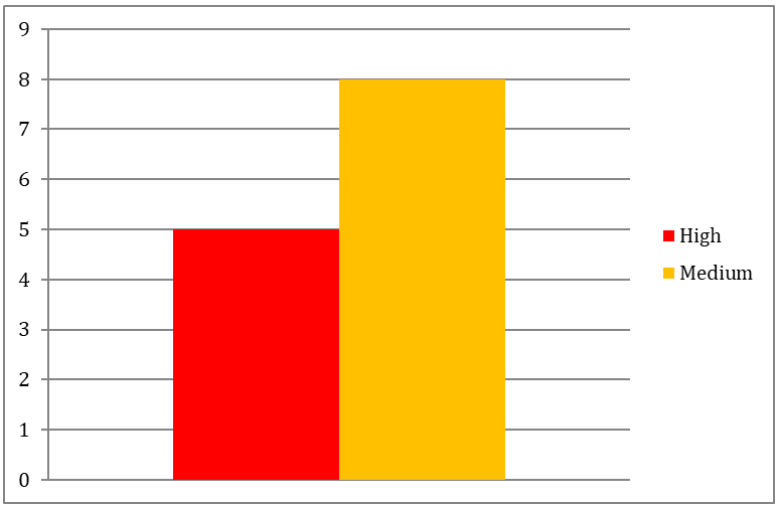
Mobile Application Penetration Testing

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]

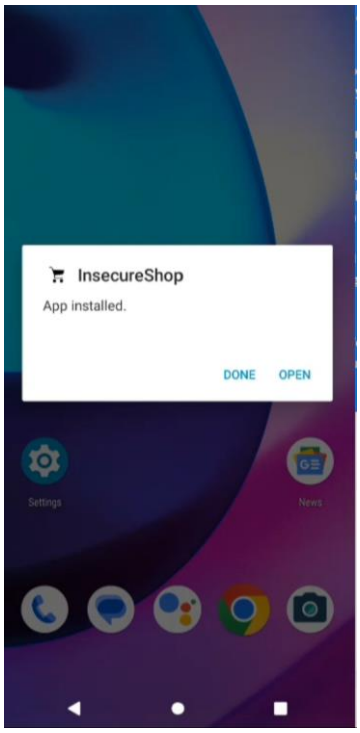
Since we have been working on some projects using Visual Code Studio, we already have a physical android device which is Lenovo K14 that we will be using during this project in order to simulate the apk file.

Task#2 : Set up the MPT environment.

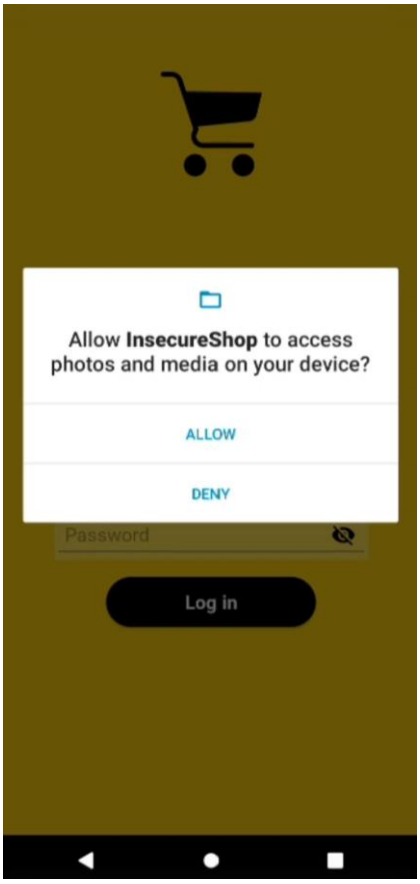
1. **Emulator**

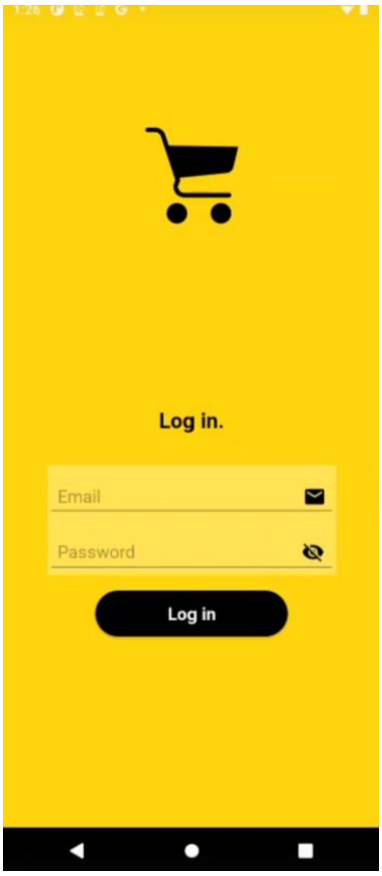
1.1 Android device which is Lenovo K14 as our simulator.

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.

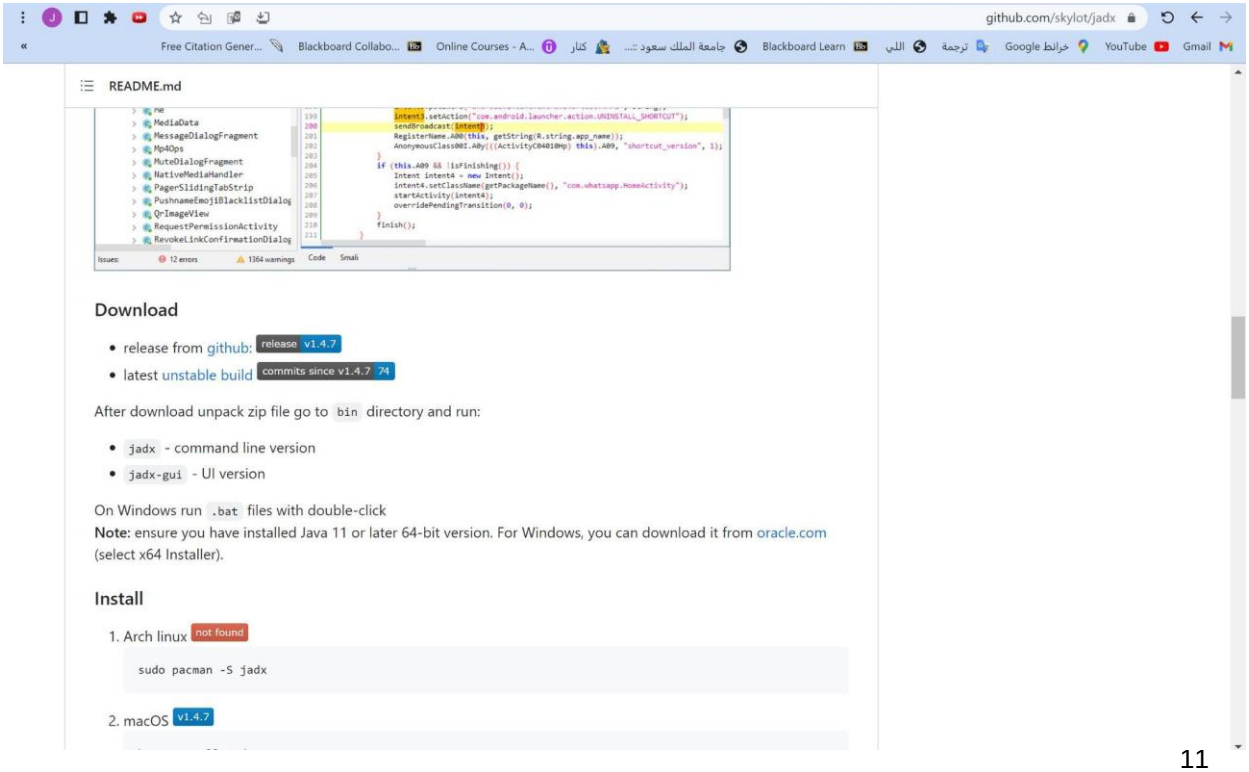




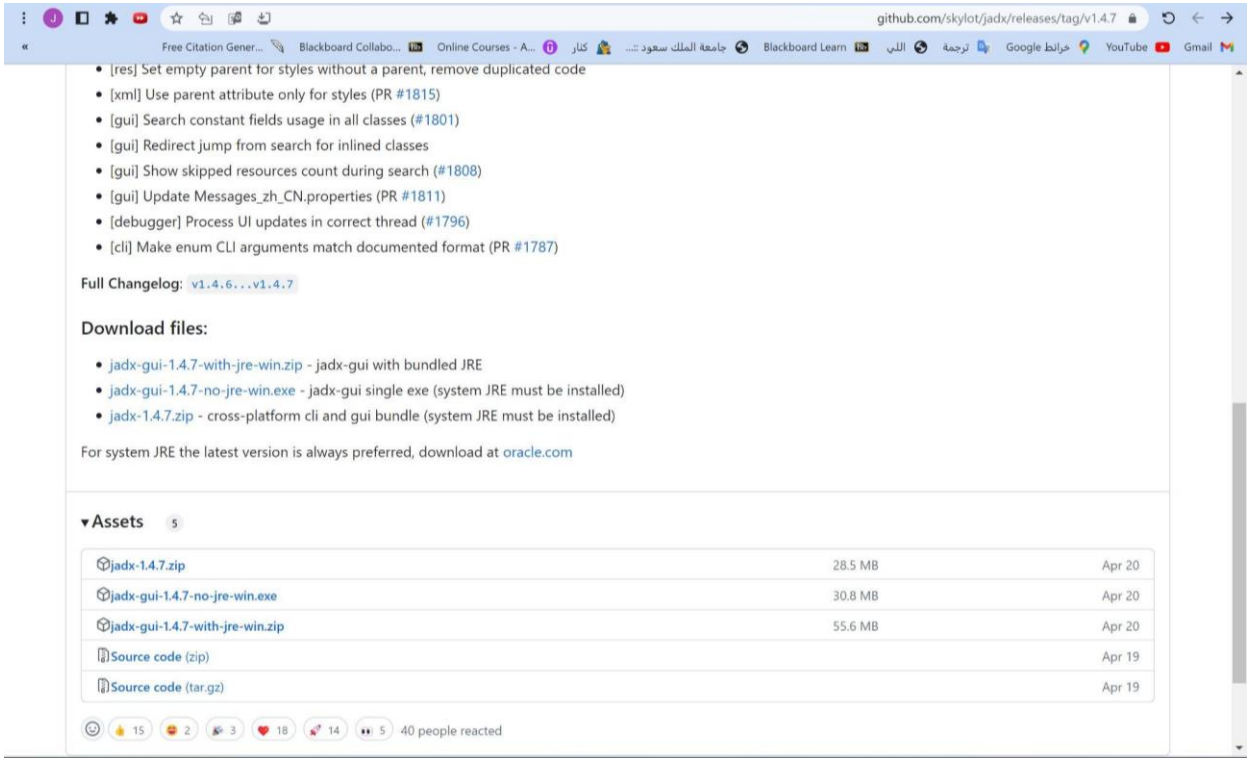
2. Installing Jadx tool [2].

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

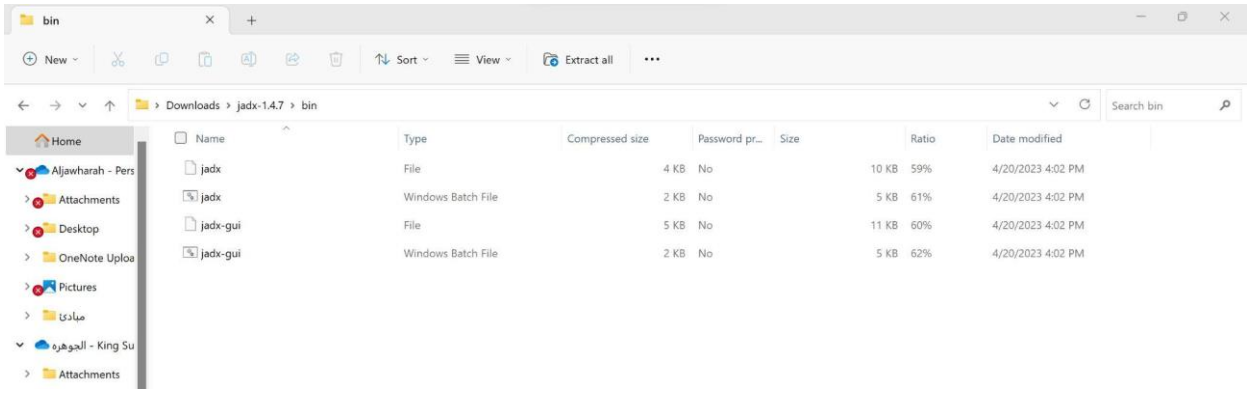
2.2 We clicked “github”.



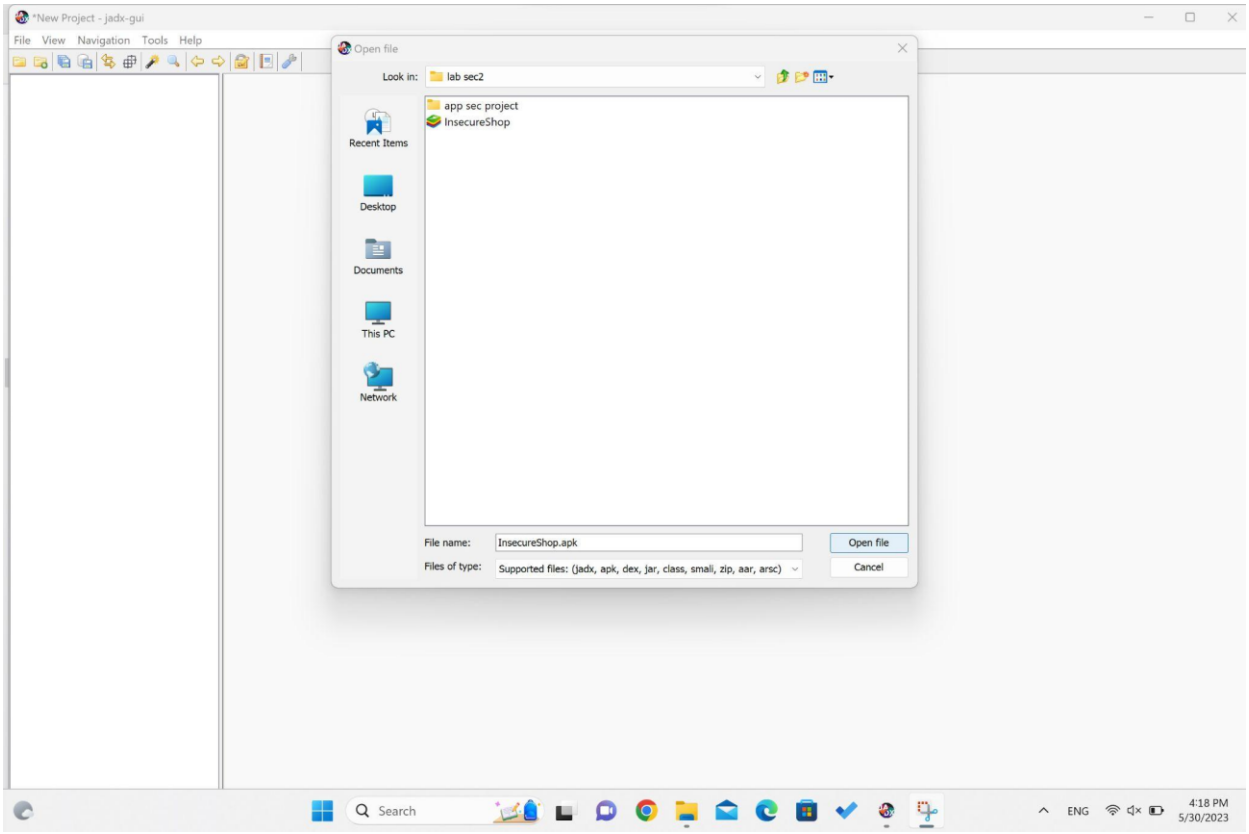
2.3 We clicked on “jadx-1.4.7.zip”. After that, it began downloading.



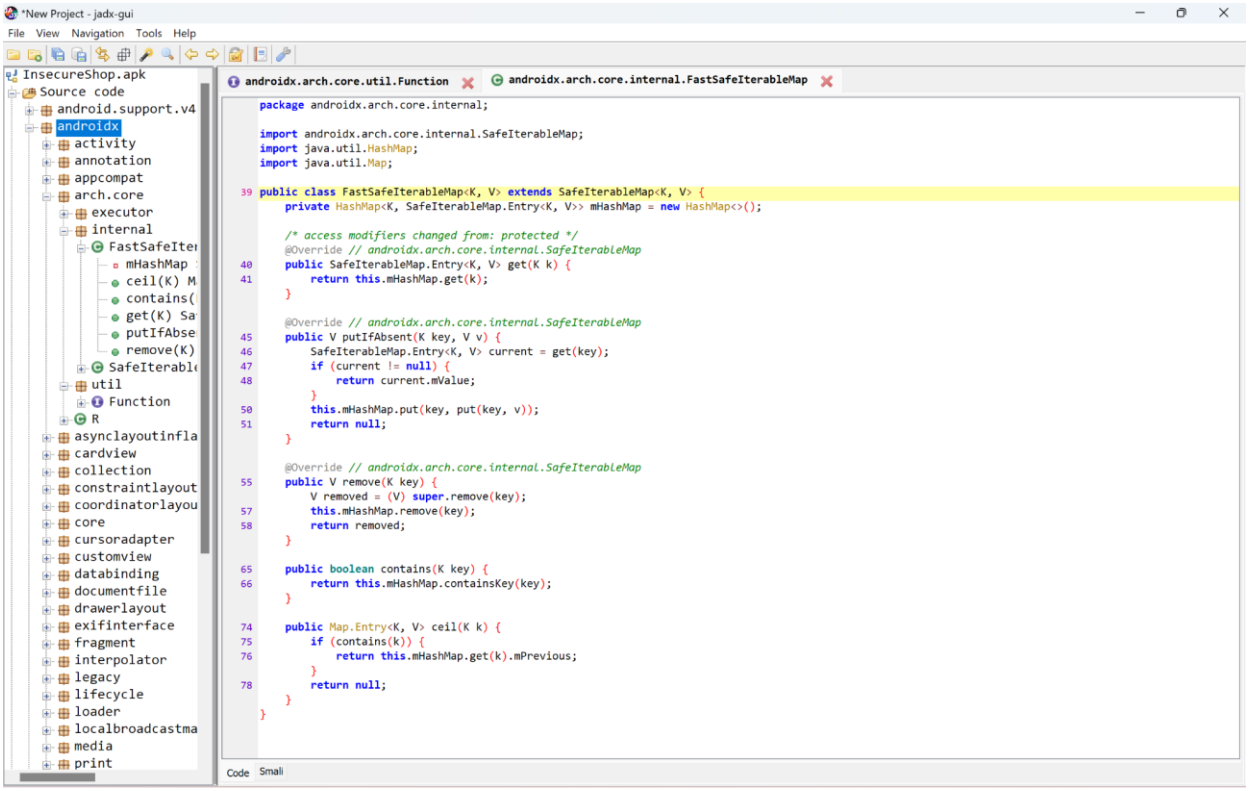
2.4 Below is an illustration of what we find inside the jadx when we open it. and then we clicked on “jadx-gui”.



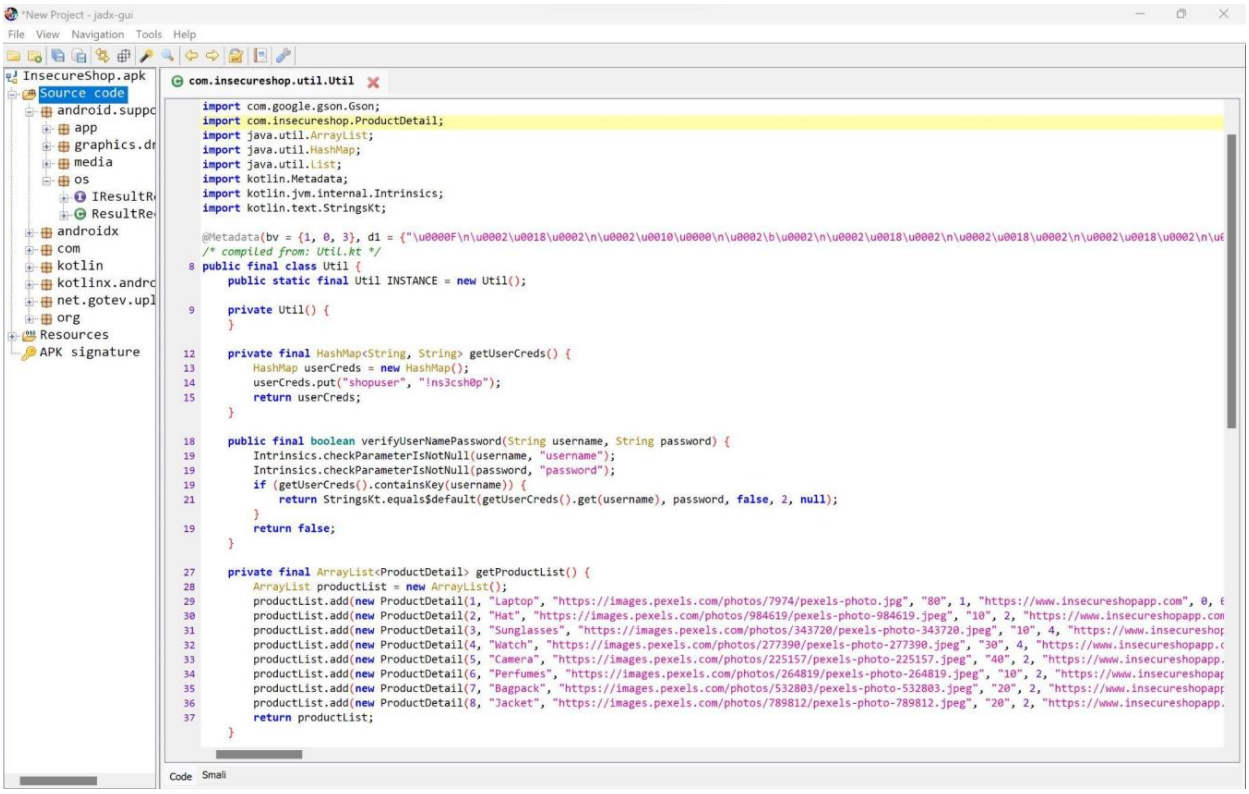
2.5 Then the window will open. and we chose “insecureShop” file. then clicked “Open file”.



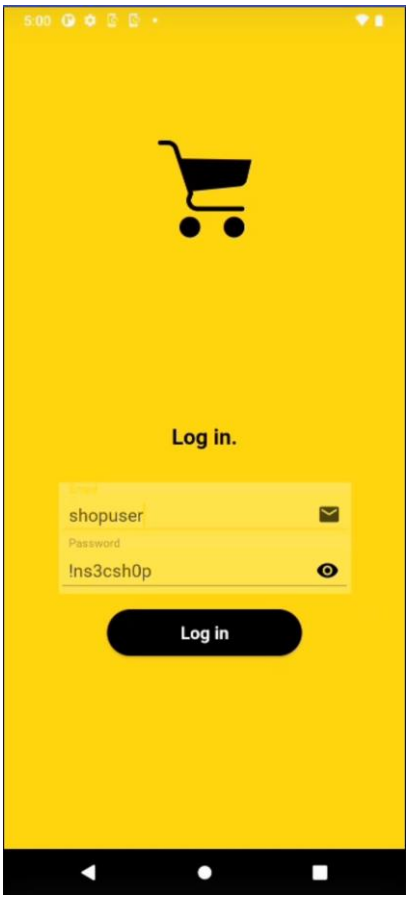
2.6 After selecting the file, jadx will decompile it and display all the Java packages and files in a "tree style view" on the left side of the application.

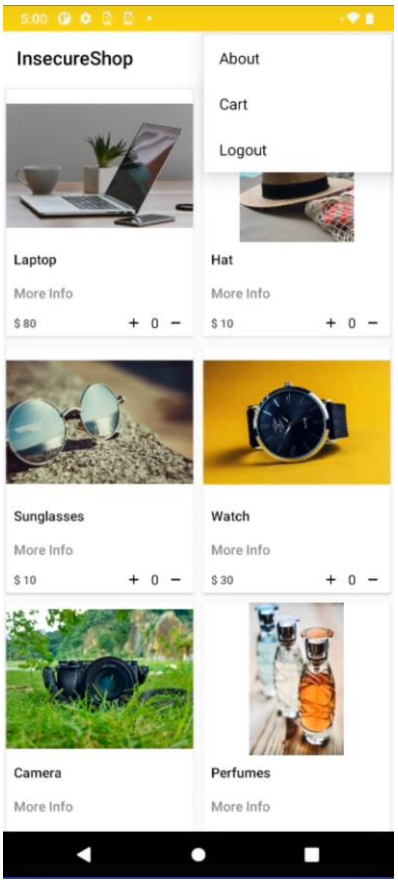


2.7 After that we opened this file and we found Hardcoded sensitive data(userName, password) which helped us to log in to the application.



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





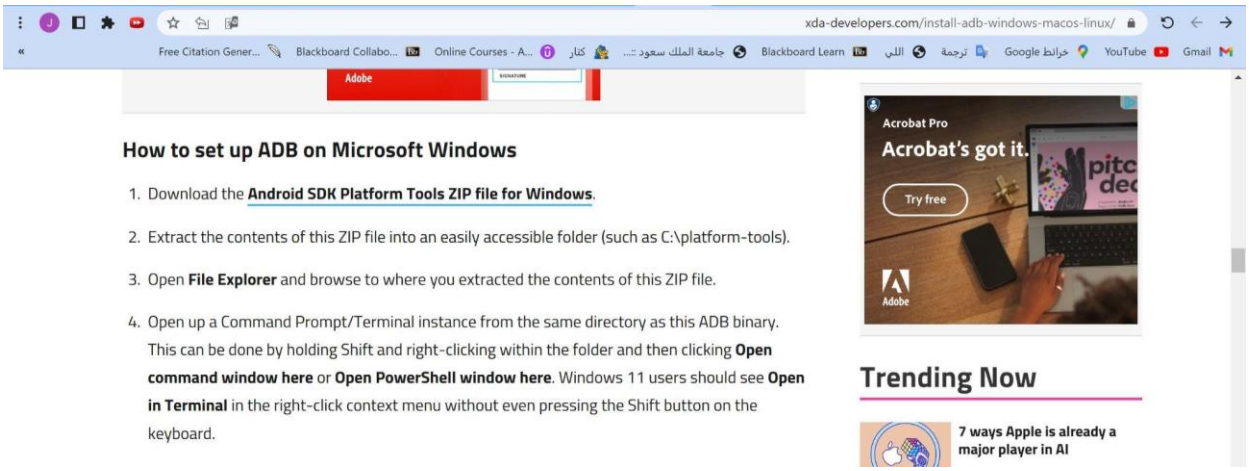
3. Android Debug Bridge (ADB) [9].

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

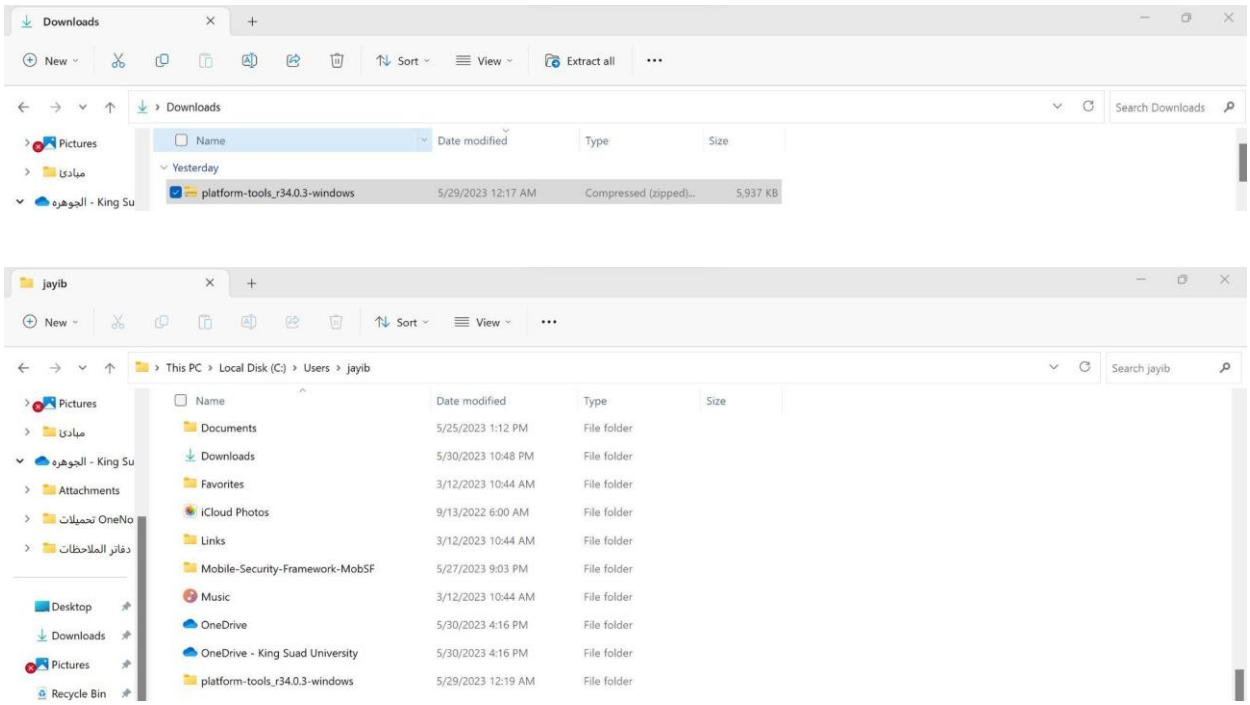
3.2 The following steps were followed to communicate with our lenovo mobile (Lenovo K14) through a terminal:

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.



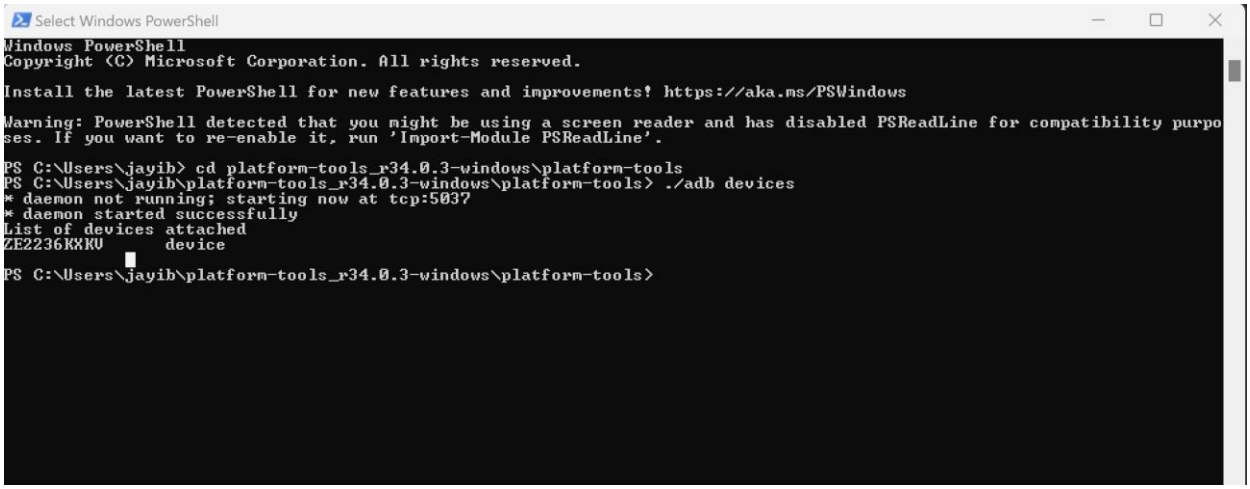
3.2.3 We extracted the zip file.



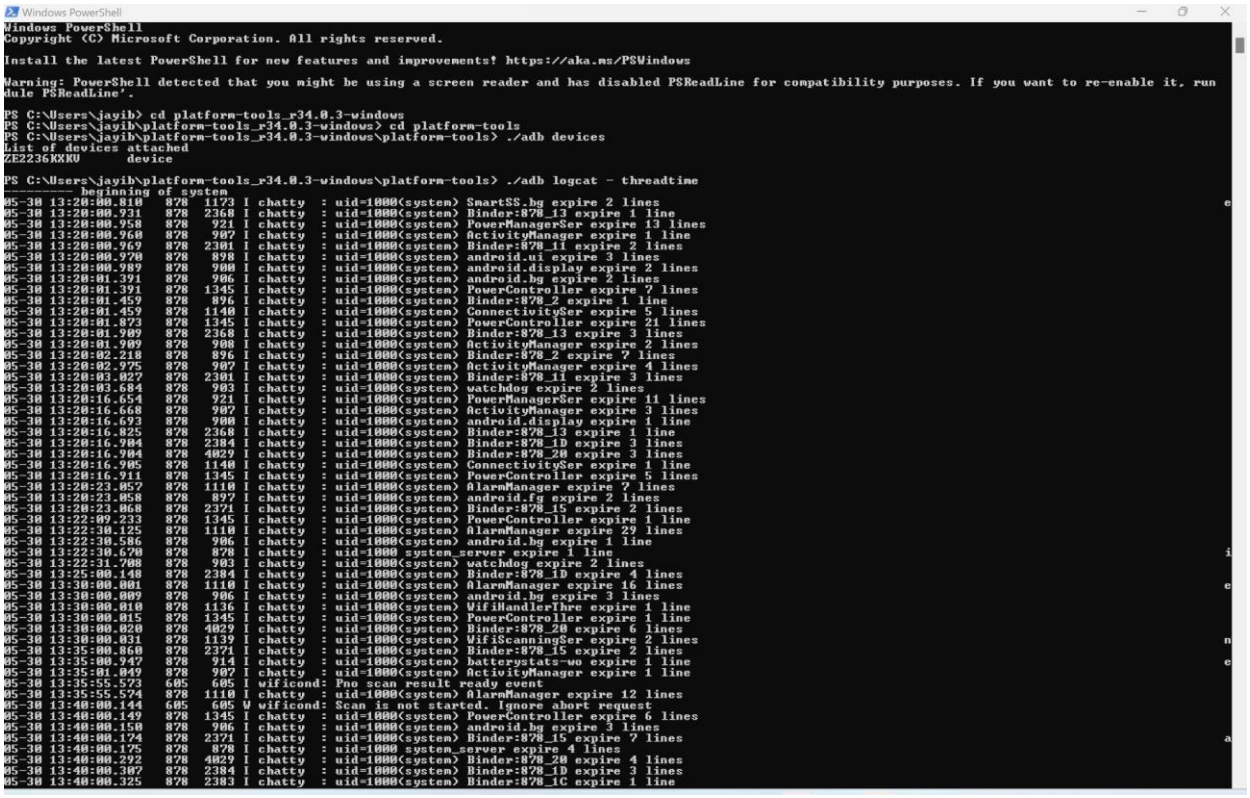
3.3 We opened the command line.

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.



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.



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.

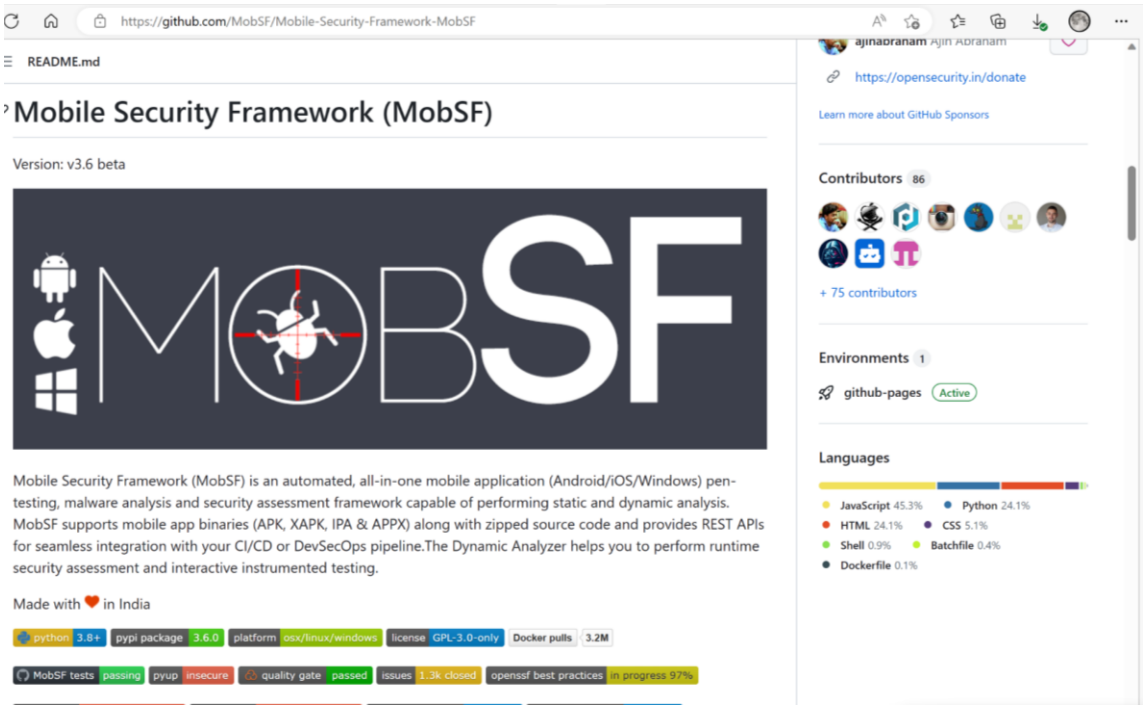
After downloading all these required tools, we started setting up MobSF.

5. Setup MobSF [13].

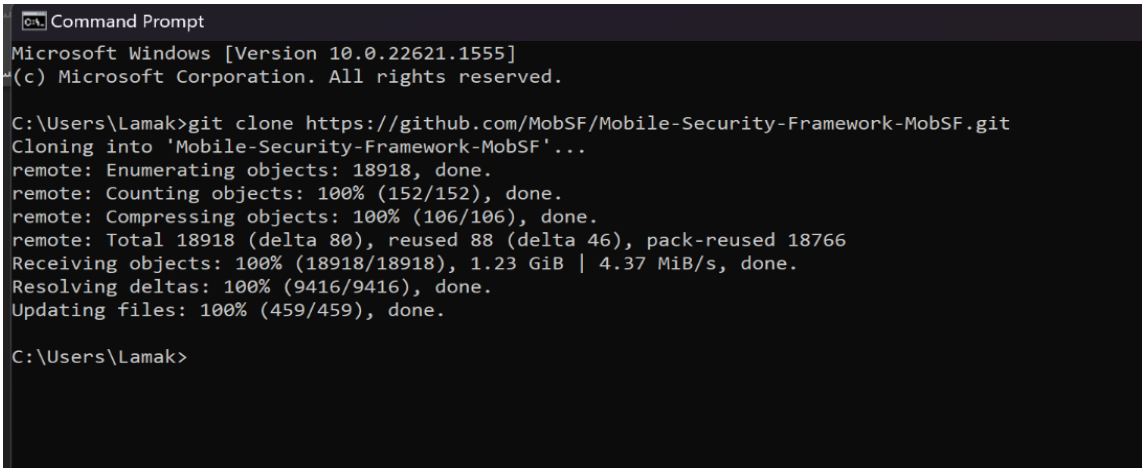
After we set up all required programs to run MobSF, finally the system is ready to install MobSF.

5.1 We used this link to download MobSF .

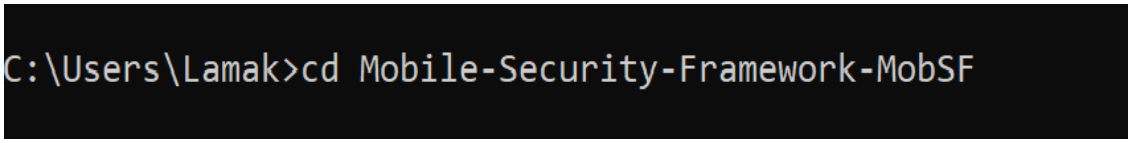
MobSF/Mobile-Security-Framework-MobSF: Mobile Security Framework (MobSF) is an automated, all-in-one mobile application (Android/iOS/Windows) pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis. (github.com)



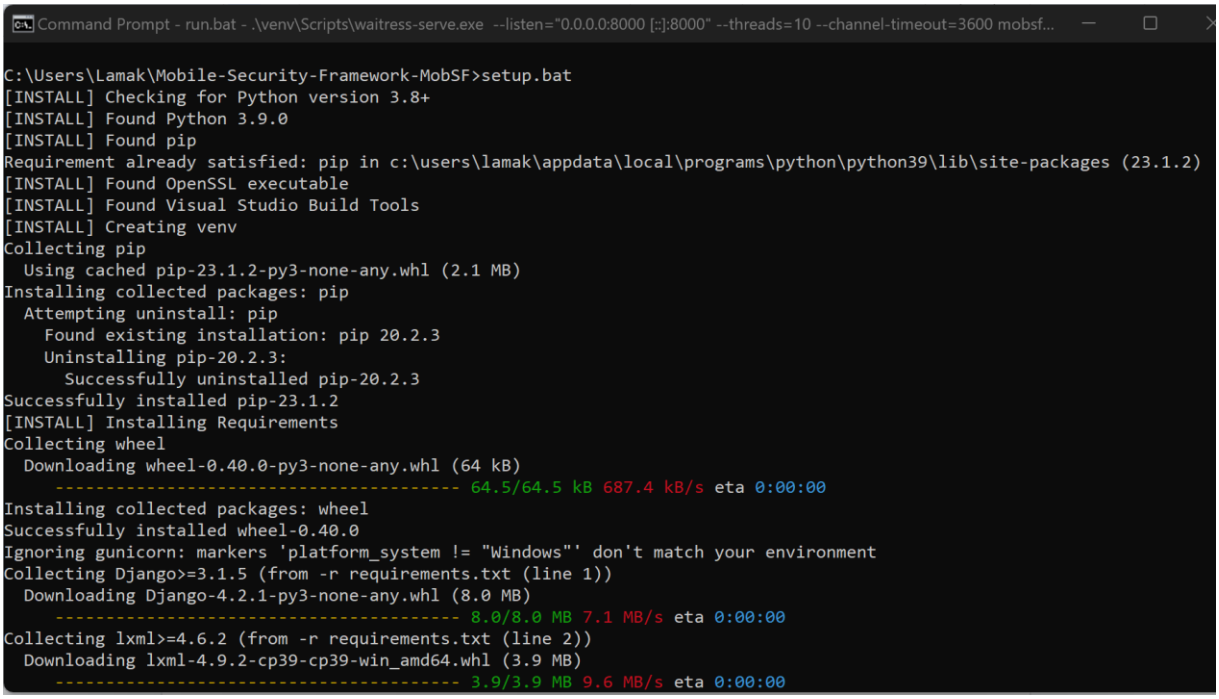
5.2 After completing the installation steps and requirements in Task#2 section 2, we opened the Command Line and wrote this command “git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF” , then we waited until it finished downloading.

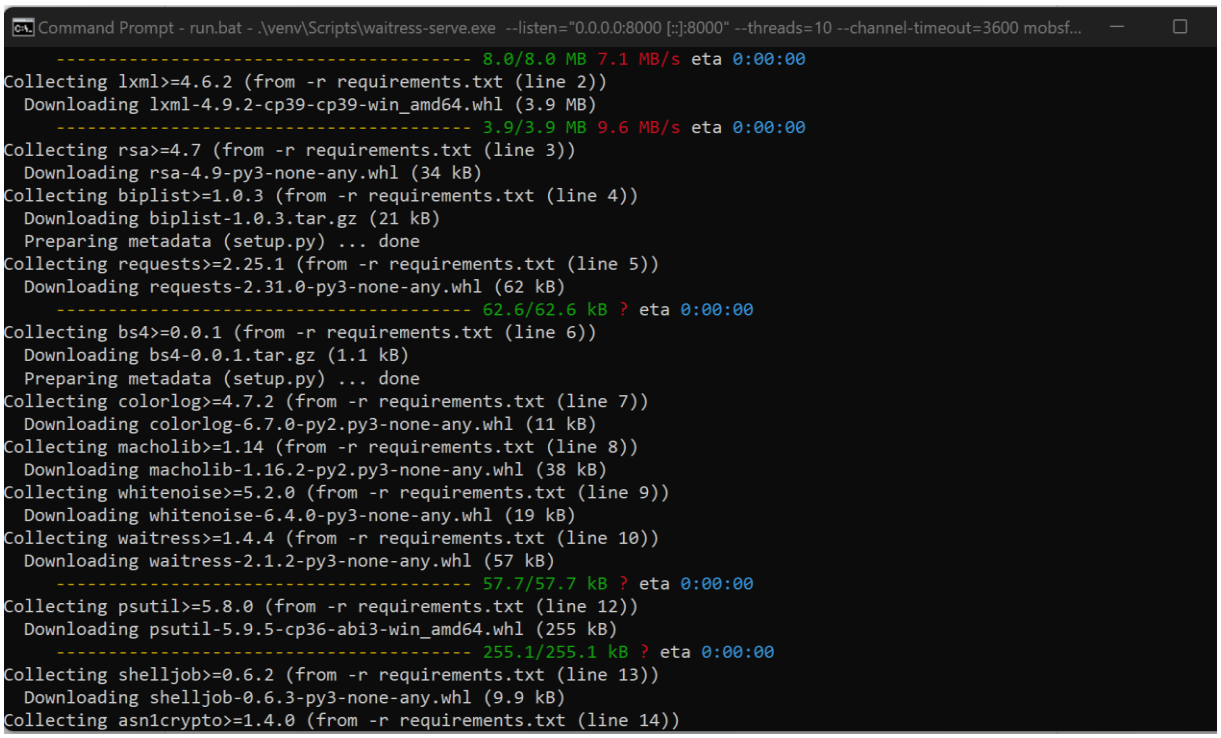


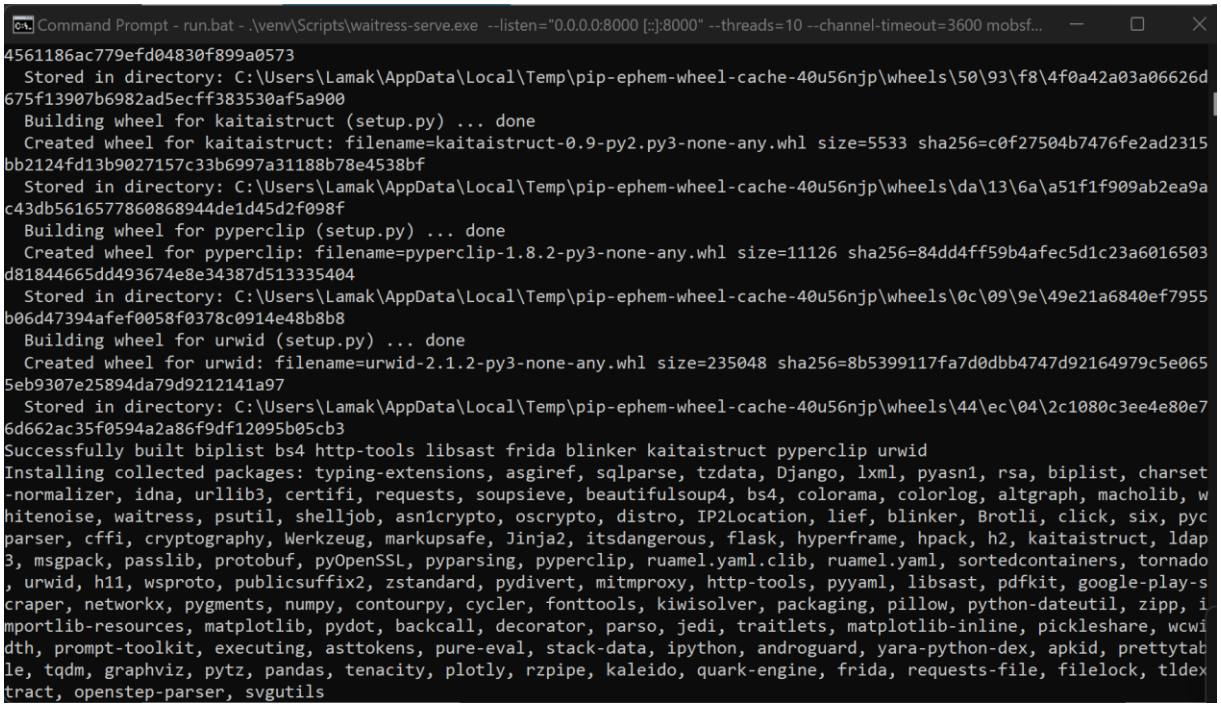
5.3 Then we moved into the Mobile-Security-Framework-MobSF directory by running this command “ cd Mobile-Security-Framework-MobSF “.

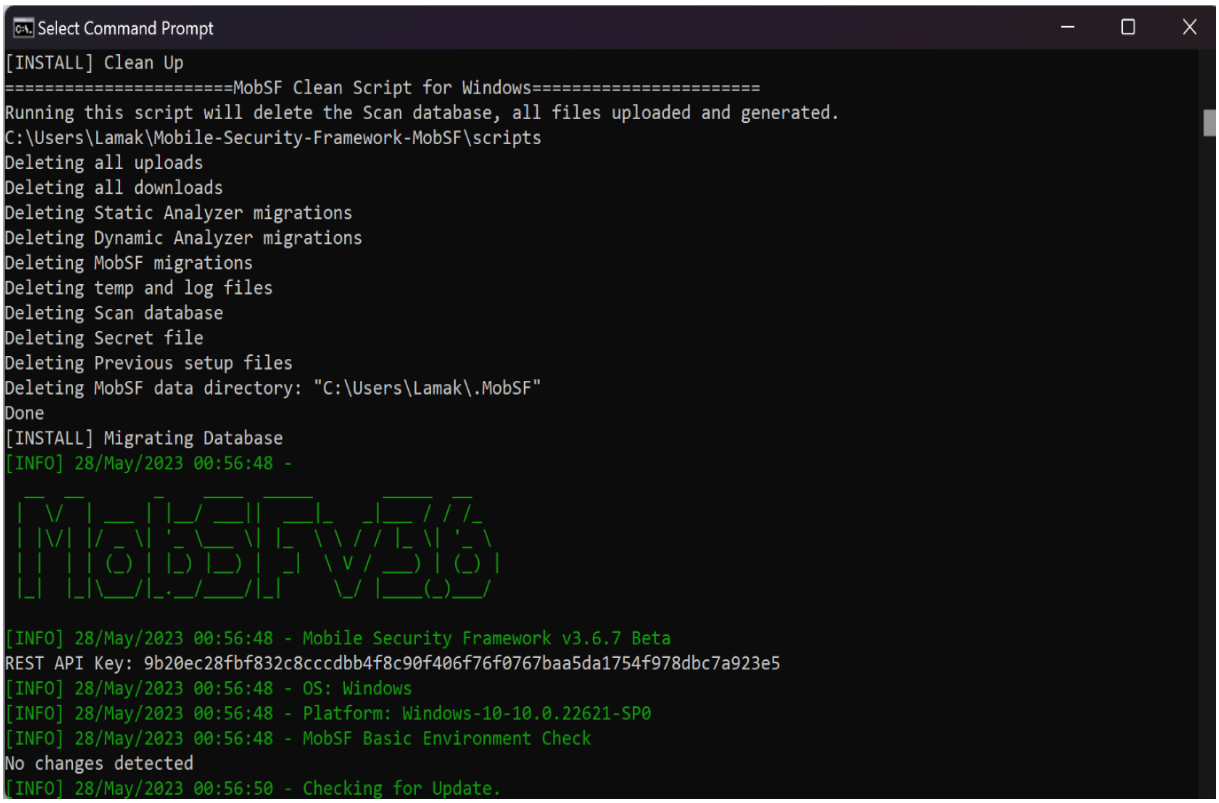


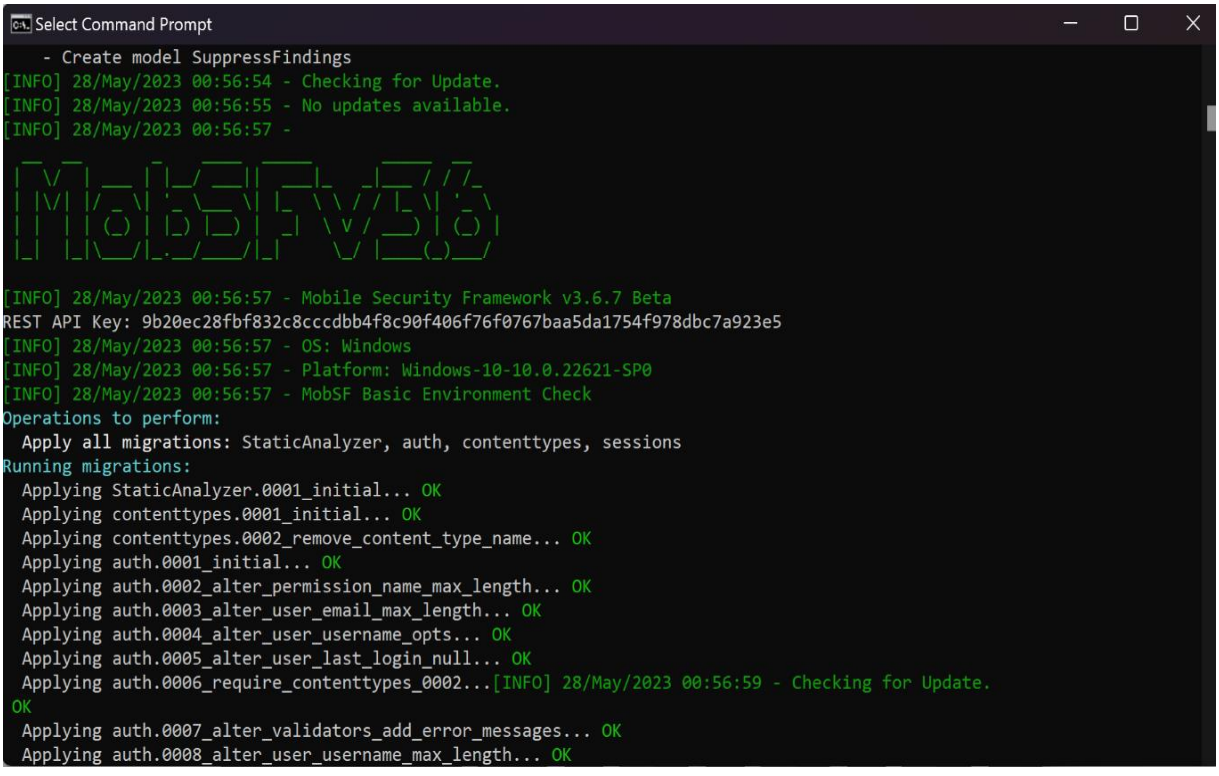
5.4 Then we ran this command “ setup.bat ”, then we waited until all installation of packages finished.



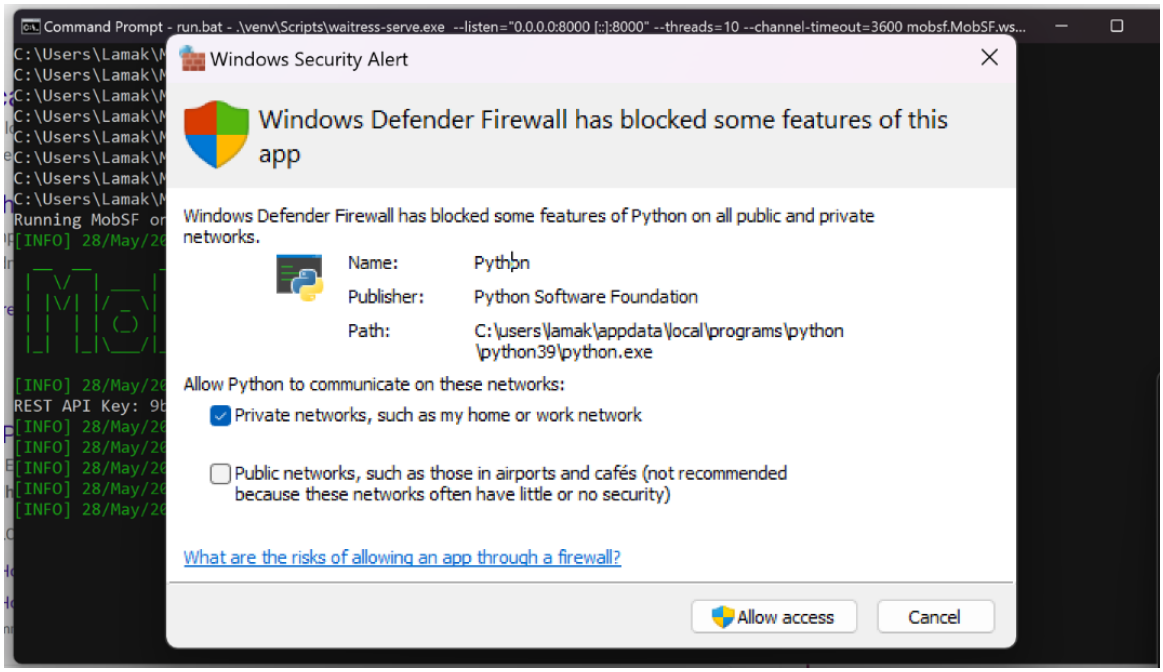

Command Prompt - run.bat - .\venv\Scripts\waitress-serve.exe --listen="0.0.0.0:8000 [::]:8000" --threads=10 --channel-timeout=3600 mobsf...
---------- 8.0/8.0 MB 7.1 MB/s eta 0:00:00
Collecting lxml>=4.6.2 (from -r requirements.txt (line 2))
Downloading lxml-4.9.2-cp39-cp39-win_amd64.whl (3.9 MB)
---------- 3.9/3.9 MB 9.6 MB/s eta 0:00:00
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...
4561186ac779efd04830f899a0573
Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\50\93\f8\4f0a42a03a06626d
675f13907b6982ad5ecff383530af5a900
Building wheel for kaitaistruct (setup.py) ... done
Created wheel for kaitaistruct: filename=kaitaistruct-0.9-py2.py3-none-any.whl size=5533 sha256=c0f27504b7476fe2ad2315
bb2124fd13b9027157c33b6997a31188b78e4538bf
Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\da\13\6a\a51f1f909ab2ea9a
c43db5616577860868944de1d45d2f098f
Building wheel for pyperclip (setup.py) ... done
Created wheel for pyperclip: filename=pyperclip-1.8.2-py3-none-any.whl size=11126 sha256=84dd4ff59b4afec5d1c23a6016503
d81844665dd493674e8e34387d513335404
Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\0c\09\9e\49e21a6840ef7955
b06d47394afef0058f0378c0914e48b8b8
Building wheel for urwid (setup.py) ... done
Created wheel for urwid: filename=urwid-2.1.2-py3-none-any.whl size=235048 sha256=8b5399117fa7d0dbb4747d92164979c5e065
5eb9307e25894da79d9212141a97
Stored in directory: C:\Users\Lamak\AppData\Local\Temp\pip-ephem-wheel-cache-40u56njp\wheels\44\ec\04\2c1080c3ee4e80e7
6d662ac35f0594a2a86f9df12095b05cb3
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, w
hitenoise, waitress, psutil, shelljob, asn1crypto, oscrypto, distro, IP2Location, lief, blinker, Brotli, click, six, pyc
parser, cffi, cryptography, Werkzeug, markupsafe, Jinja2, itsdangerous, flask, hyperframe, hpack, h2, kaitaistruct, ldap
3, msgpack, passlib, protobuf, pyOpenSSL, pyparsing, pyperclip, ruamel.yaml.clib, ruamel.yaml, sortedcontainers, tornado
, urwid, h11, wsproto, publicsuffix2, zstandard, pydivert, mitmproxy, http-tools, pyyaml, libsast, pdfkit, google-play-s
craper, networkx, pygments, numpy, contourpy, cycler, fonttools, kiwisolver, packaging, pillow, python-dateutil, zipp, i
mportlib-resources, matplotlib, pydot, backcall, decorator, parso, jedi, traitlets, matplotlib-inline, pickleshare, wcwi
dth, prompt-toolkit, executing, asttokens, pure-eval, stack-data, ipython, androguard, yara-python-dex, apkid, prettytab
le, tqdm, graphviz, pytz, pandas, tenacity, plotly, rzpipe, kaleido, quark-engine, frida, requests-file, filelock, tldex
tract, openstep-parser, svgutils

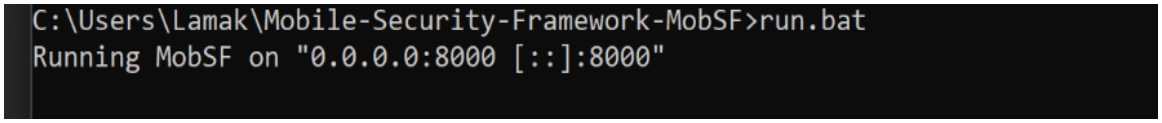

Select Command Prompt
[INSTALL] Clean Up
=====================MobSF Clean Script for Windows=====================
Running this script will delete the Scan database, all files uploaded and generated.
C:\Users\Lamak\Mobile-Security-Framework-MobSF\scripts
Deleting all uploads
Deleting all downloads
Deleting Static Analyzer migrations
Deleting Dynamic Analyzer migrations
Deleting MobSF migrations
Deleting temp and log files
Deleting Scan database
Deleting Secret file
Deleting Previous setup files
Deleting MobSF data directory: "C:\Users\Lamak\.MobSF"
Done
[INSTALL] Migrating Database
[INFO] 28/May/2023 00:56:48 -
[INFO] 28/May/2023 00:56:48 - Mobile Security Framework v3.6.7 Beta
REST API Key: 9b20ec28fbf832c8cccdbb4f8c90f406f76f0767baa5da1754f978dbc7a923e5
[INFO] 28/May/2023 00:56:48 - OS: Windows
[INFO] 28/May/2023 00:56:48 - Platform: Windows-10-10.0.22621-SP0
[INFO] 28/May/2023 00:56:48 - MobSF Basic Environment Check
No changes detected
[INFO] 28/May/2023 00:56:50 - Checking for Update.


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: 9b20ec28fbf832c8cccdbb4f8c90f406f76f0767baa5da1754f978dbc7a923e5
[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

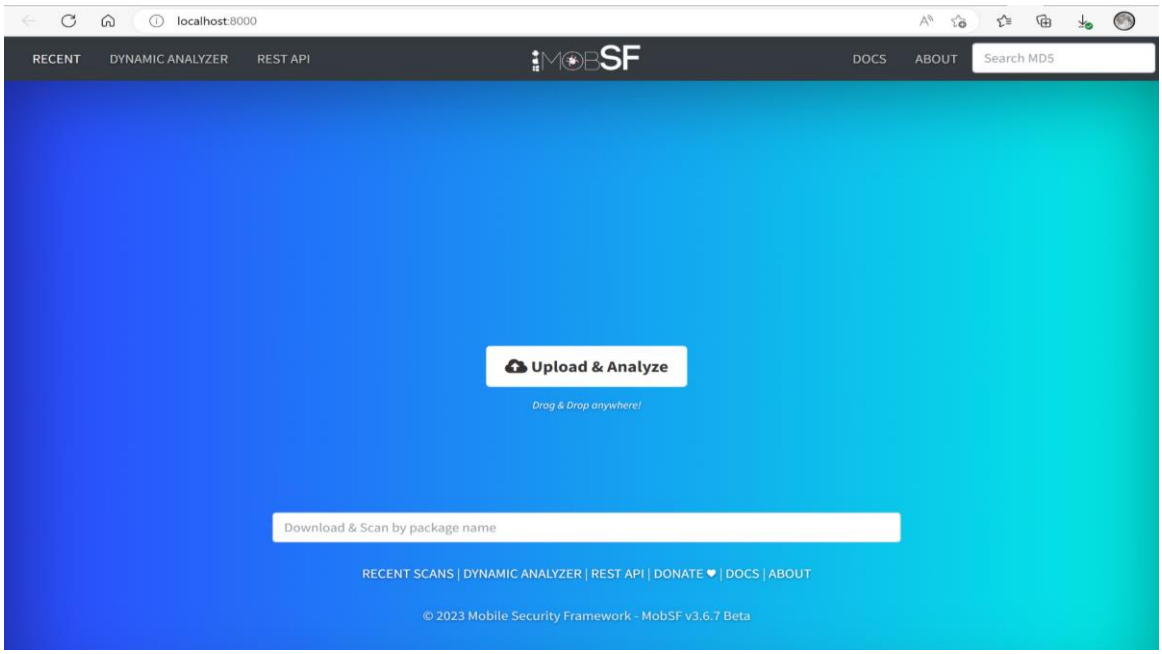
5.5 Once it finished downloading, this window appeared asking to allow access, we clicked “Allow Access”.



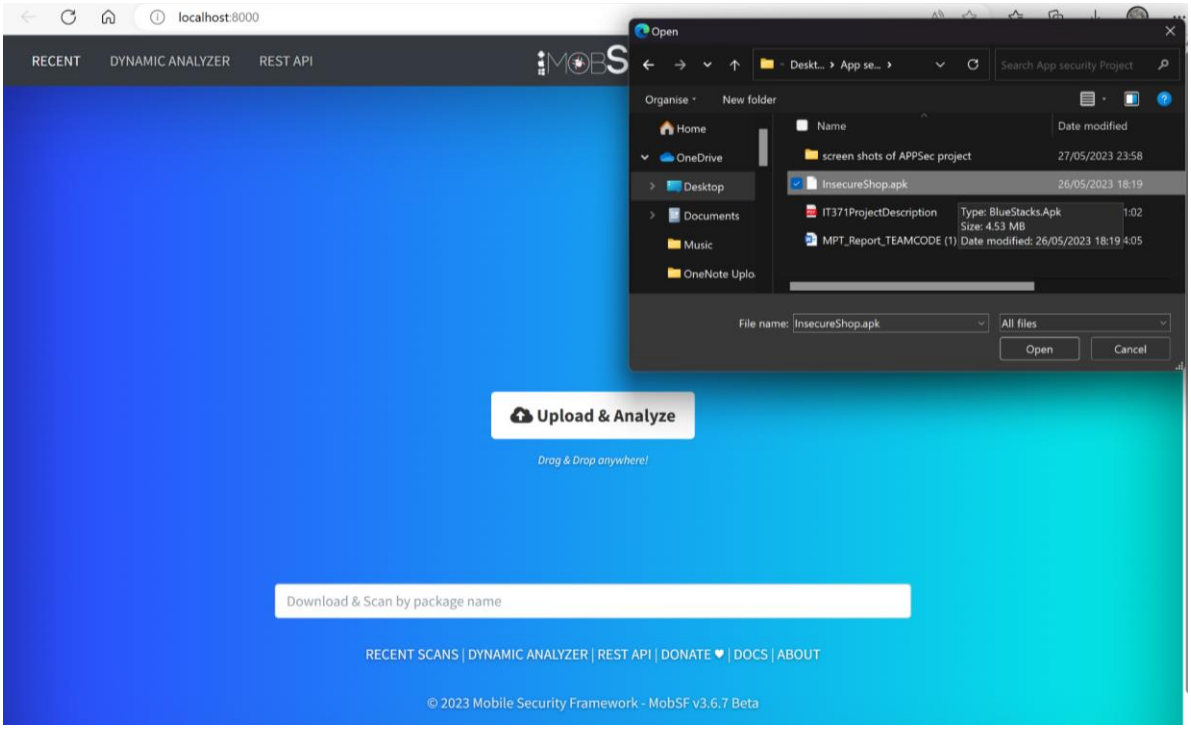
5.6 Then we run this command “run.bat”.



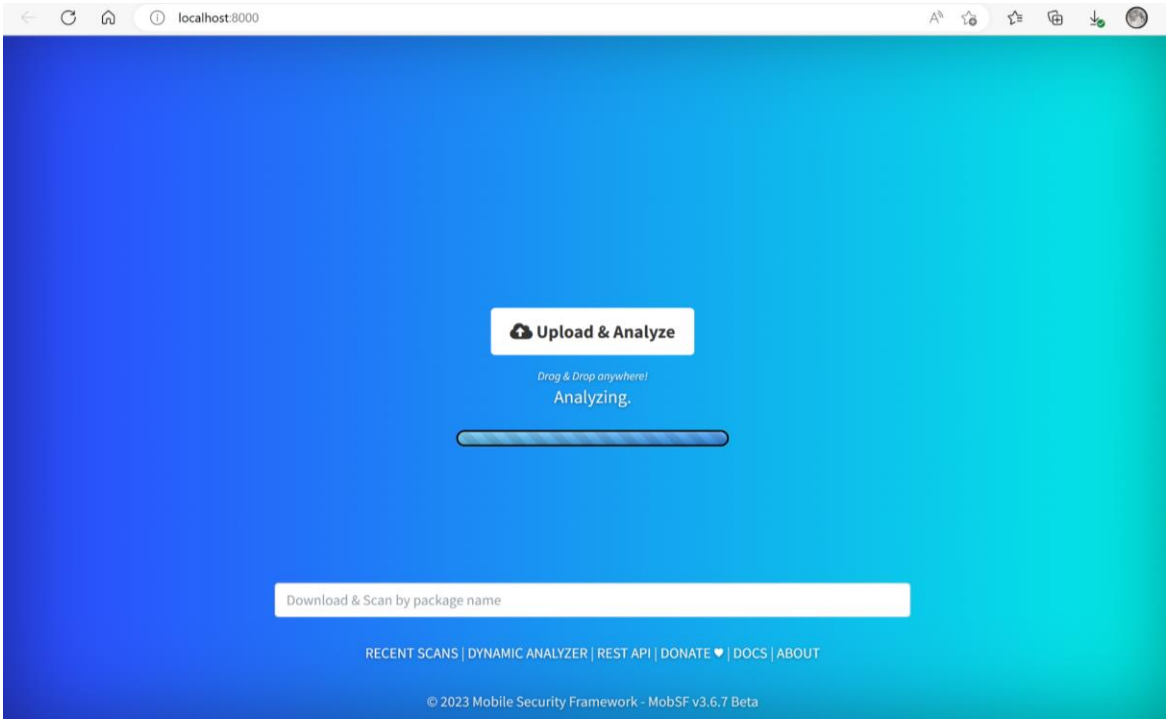
5.7 Then we opened the browser and searched “localhost:8000”. And this page appeared.



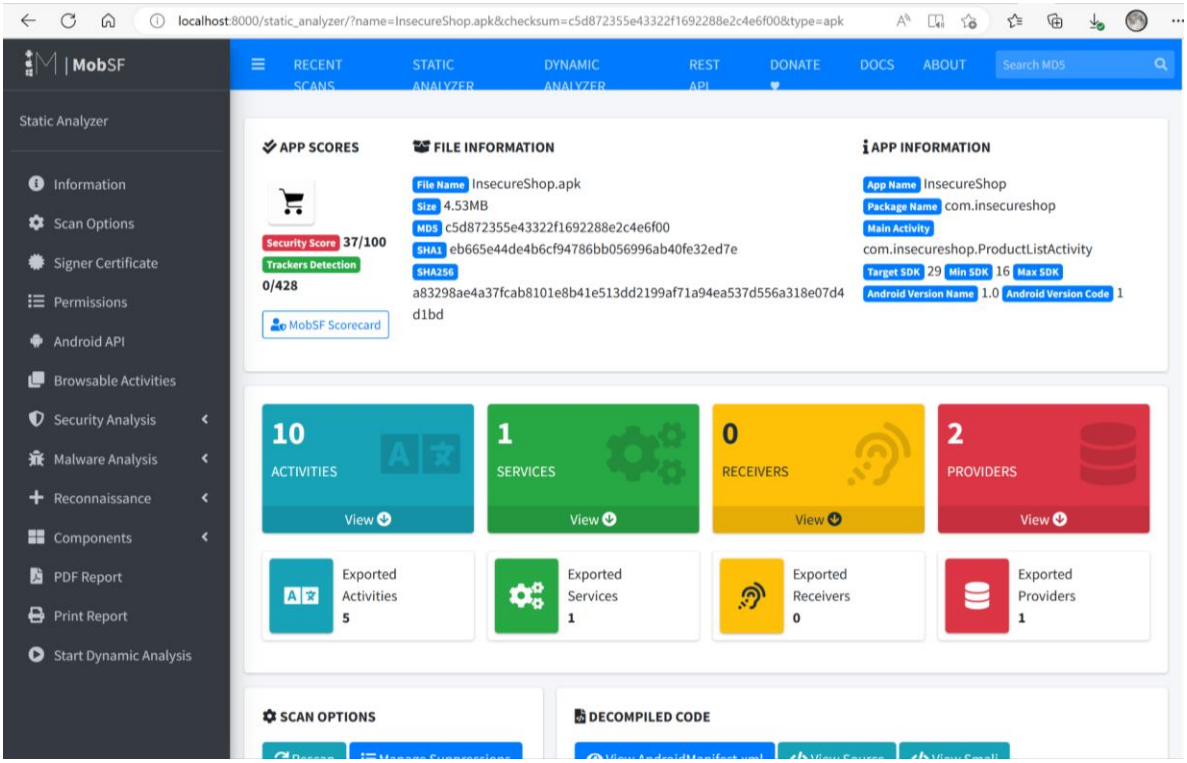
5.8 We upload the insecureshop.apk file here.

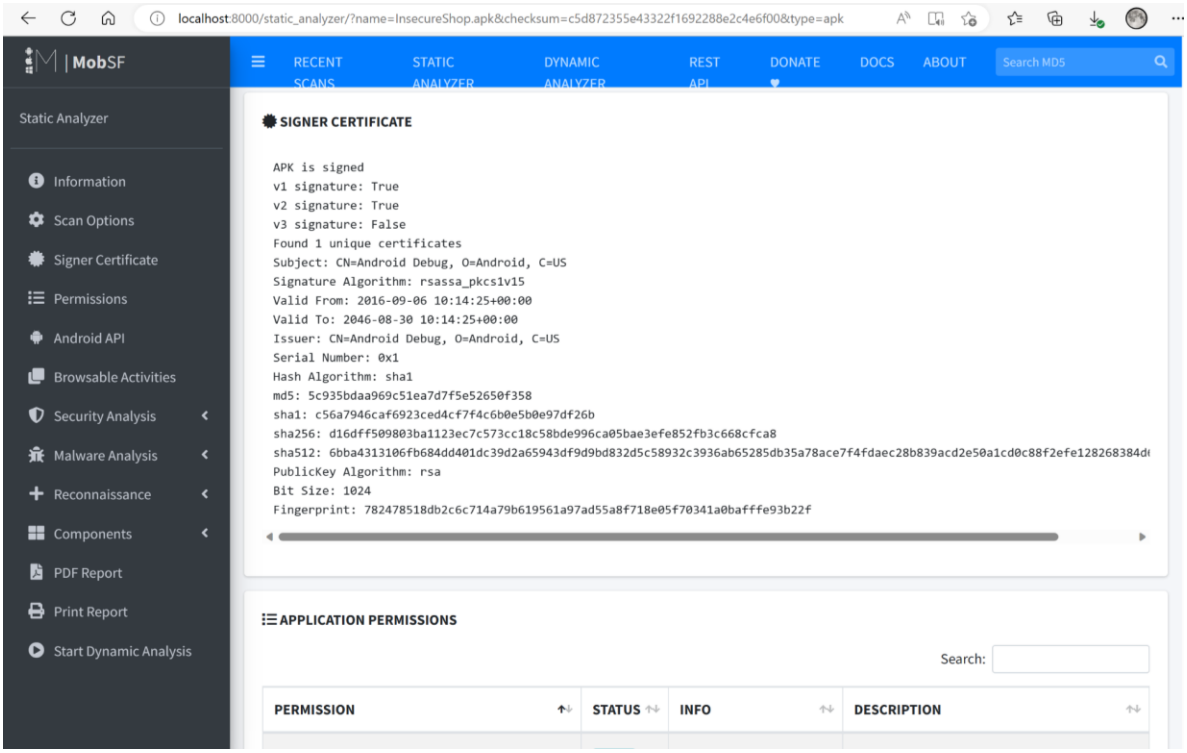


5.9 MobSF starts analyzing the apk file.



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





4 Detailed Findings

4.1 *Limitations*

During the mobile penetration testing the 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, wkhtmktox, 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.

<table>
<tr><td>Severity Level</td><td colspan="4">Risk Severity Level of the Vulnerability</td></tr>
<tr><td rowspan="2">Technical Impact</td><td rowspan="2">Impact level</td><td colspan="2">Likelihood</td><td colspan="2">Popularity and Simplicity</td></tr>
<tr><td>Popularity</td><td>Popularity of the Exploit</td><td>Simplicity</td><td>Simplicity to Exploit</td></tr>
<tr><td>Observation</td><td colspan="4">Our observation and description about the vulnerability</td></tr>
<tr><td>Impact</td><td colspan="4">Describes the possible Technical Impact if the vulnerability is Exploited</td></tr>
<tr><td>Platform</td><td colspan="4">Describe the platform</td></tr>
<tr><td colspan="5">Proof of Concept</td></tr>
<tr><td colspan="5">Evidence to support the finding</td></tr>
<tr><td>Recommendation</td><td colspan="4">High Level recommendation to mitigate the vulnerability</td></tr>
<tr><td>OWASP Reference</td><td colspan="4">OWASP Reference</td></tr>
<tr><td>Reference</td><td colspan="4">Reference related to the vulnerability</td></tr>
</table>

4.1.1 Insecure Communication

Severity Level	**High**				
Technical Impact	High	**Likelihood**		**Medium**	
		Popularity	Medium	**Simplicity**	Medium
Observation	**While observing CustomWebViewClient.java, we found out that method onReceivedSslError 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 handler.proceed() 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	**CWE:** CWE-295: Improper Certificate Validation **OWASP Top 10:** M3: Insecure Communication **OWASP MASVS:** MSTG-NETWORK-3	com/insecureshop/util/CustomWebViewClient.java

```
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 = {"\u0000$\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();
        }
    }
}
```

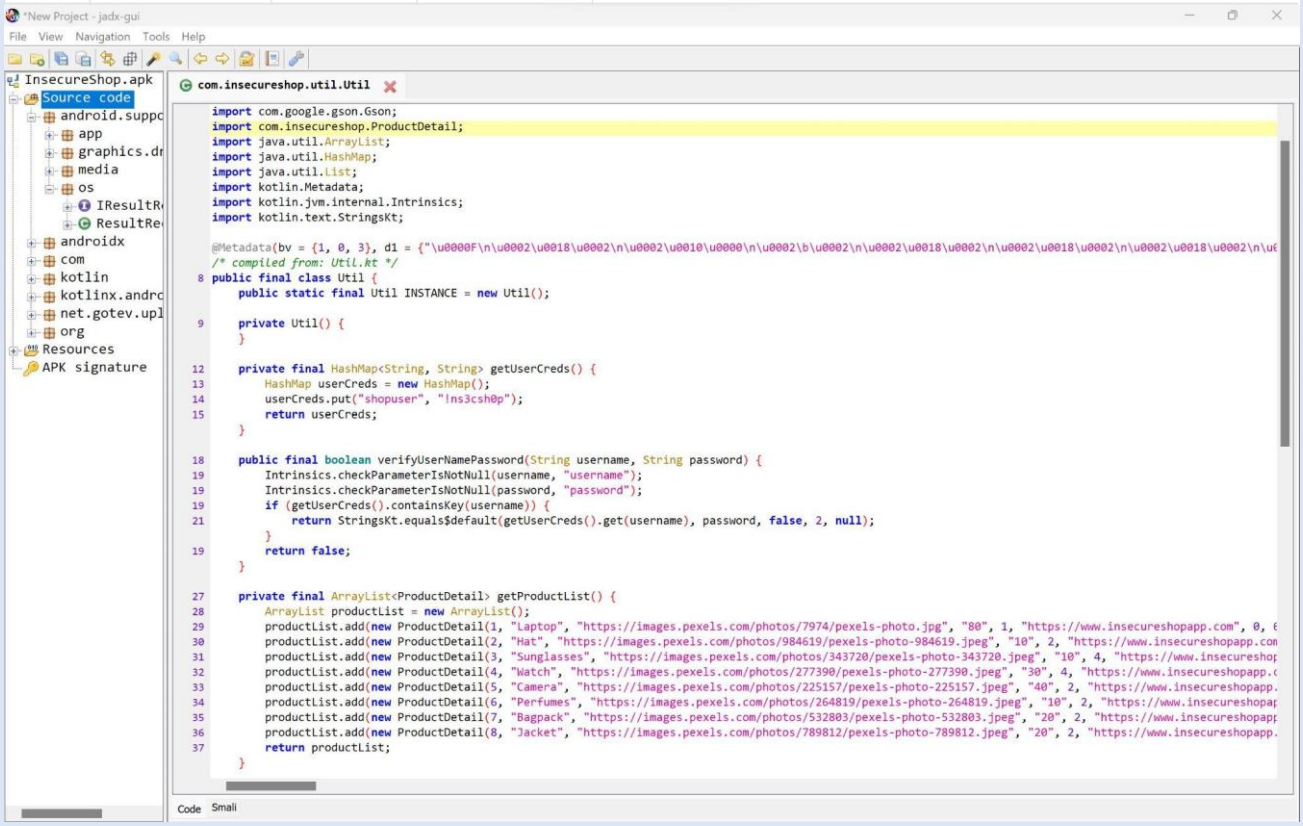
Recommendation	**- Use strong encryption algorithms to protect data confidentiality and integrity. - Use transport layer security (TLS) or secure sockets layer (SSL) protocols to encrypt data in transit and protect it from interception or tampering.**
OWASP reference	**M3: Insecure Communication | OWASP Foundation**
Reference	**owasp-mstg/0x05g-Testing-Network-Communication.md at master · MobSF/owasp-mstg (github.com)**

4.1.2 Hardcoding sensitive data in the source code.

Severity Level	**Medium**				
Technical Impact	High	**Likelihood**		**High**	
		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	**CWE:** CWE-312: Cleartext Storage of Sensitive Information **OWASP Top 10:** M9: Reverse Engineering **OWASP MASVS:** MSTG-STORAGE-14	com/bumptech/glide/load/Option.java com/bumptech/glide/load/engine/DataCacheKey.java com/bumptech/glide/load/engine/EngineResource.java com/bumptech/glide/load/engine/ResourceCacheKey.java com/bumptech/glide/manager/RequestManagerRetriever.java



Recommendation	**- 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.**

OWASP reference	**M9: Reverse Engineering | OWASP Foundation**
Reference	**https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE** **owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)** **https://cwe.mitre.org/data/definitions/798.html**

4.1.3 Insecure Logging of sensitive data.

Severity Level	**High**				
Technical Impact	High	**Likelihood**		**High**	
		Popularity	Medium	**Simplicity**	Medium
Observation	**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.**				
Impact	**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.**				
Platform	**Android**				
Proof of Concept					

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

```
public final void onLogin(View view) {
    Intrinsics.checkParameterIsNotNull(view, "view");
    ActivityLoginBinding activityLoginBinding = this.mBinding;
    if (activityLoginBinding == null) {
        Intrinsics.throwUninitializedPropertyAccessException("mBinding");
    }
    TextInputEditText textInputEditText = activityLoginBinding.edtUserName;
    Intrinsics.checkExpressionValueIsNotNull(textInputEditText, "mBinding.edtUserName");
    String username = String.valueOf(textInputEditText.getText());
    ActivityLoginBinding activityLoginBinding2 = this.mBinding;
    if (activityLoginBinding2 == null) {
        Intrinsics.throwUninitializedPropertyAccessException("mBinding");
    }
    TextInputEditText textInputEditText2 = activityLoginBinding2.edtPassword;
    Intrinsics.checkExpressionValueIsNotNull(textInputEditText2, "mBinding.edtPassword");
    String password = String.valueOf(textInputEditText2.getText());
    Log.d("userName", username);
    Log.d("password", password);
    boolean auth = Util.INSTANCE.verifyUserNamePassword(username, password);
```

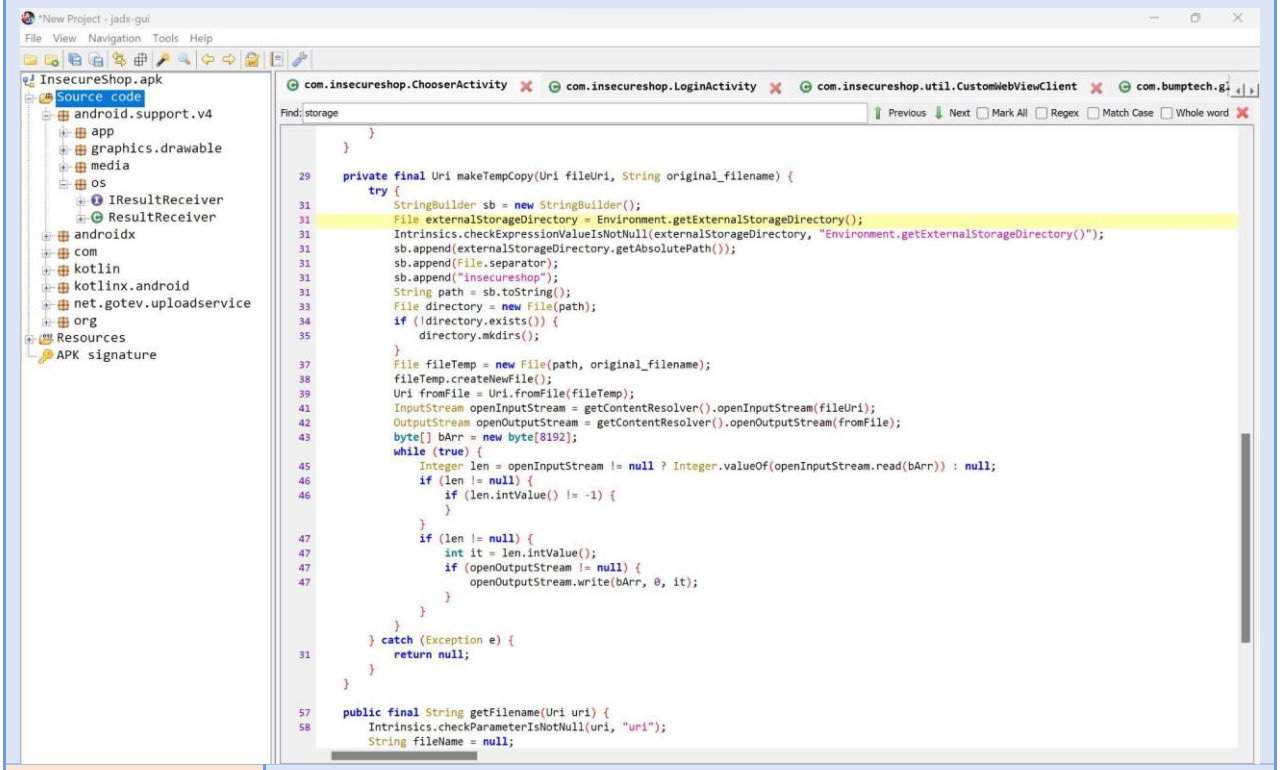
Recommendation	**- Use secure logging frameworks, and tools that can perform the encryption.** **- Conduct regular security assessments and penetration testing to identify and address any vulnerabilities in the application's logging practices.**
OWASP reference	**OWASP Top Ten 2017 | A10:2017-Insufficient Logging & Monitoring | OWASP Foundation**
Reference	**owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)**

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**				

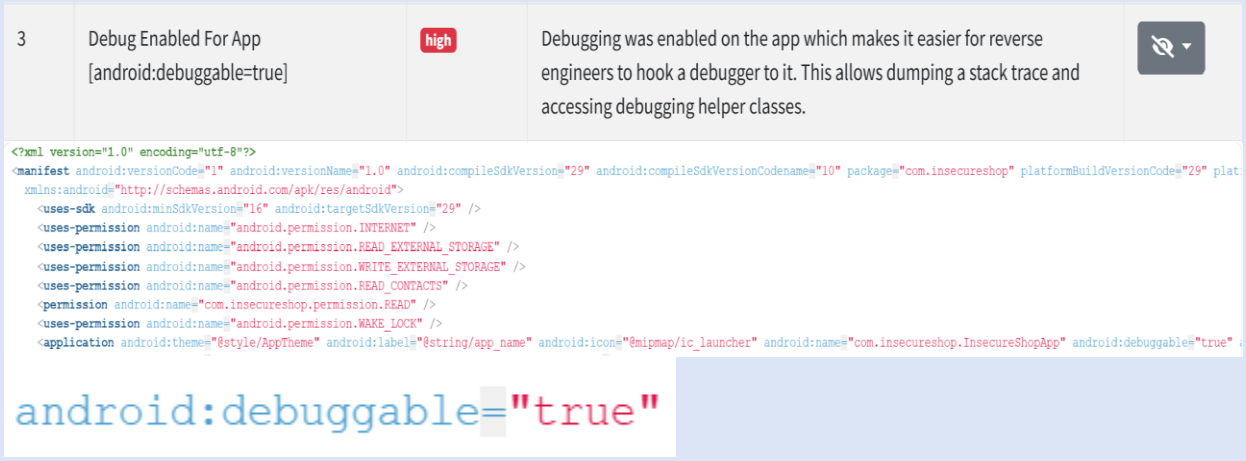
	other malicious activities.
Platform	**Android**
Proof of Concept	

4	App can read/write to External Storage. Any App can read data written to External Storage.	warning	**CWE:** CWE-276: Incorrect Default Permissions **OWASP Top 10:** M2: Insecure Data Storage **OWASP MASVS:** MSTG-STORAGE-2	com/insecureshop/ChooserActivity.java



Recommendation	**- 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.**
OWASP Reference	**M2: Insecure Data Storage | OWASP Foundation**
Reference	**owasp-mstg/0x05d-Testing-Data-Storage.md at master · MobSF/owasp-mstg (github.com)** **Unpacking Android Security: Part 2 — Insecure Data Storage | by Ed Holloway-George | ProAndroidDev**

4.1.5 Application is enabled for debugging.

<table>
<tr><td>Severity Level</td><td colspan="4">High</td></tr>
<tr><td rowspan="2">Technical Impact</td><td rowspan="2">High</td><td colspan="2">Likelihood</td><td>MEDIUM</td></tr>
<tr><td>Popularity</td><td>MEDIUM</td><td>Simplicity</td><td>LOW</td></tr>
<tr><td>Observation</td><td colspan="4">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.</td></tr>
<tr><td>Impact</td><td colspan="4">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.</td></tr>
<tr><td>Platform</td><td colspan="4">Android</td></tr>
<tr><td colspan="5">Proof of Concept</td></tr>
<tr><td colspan="5">


</td></tr>
<tr><td>Recommendation</td><td colspan="4">Disable debugging mode to prevent access and reverse engineering.</td></tr>
<tr><td>OWASP Reference</td><td colspan="4">M9: Reverse Engineering | OWASP Foundation</td></tr>
<tr><td>Reference</td><td colspan="4">android:debuggable | Android Developers
https://cwe.mitre.org/data/definitions/489.html
https://docs.fluidattacks.com/criteria/vulnerabilities/058/</td></tr>
</table>

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	jadex 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)