

External Application Penetration Testing Technical Report

For

King Saud University

Document Info

Item	Description
Document Title	Mobile Penetration Test
Requestor	T.Fayazah Alshammary
Author/s	Shahad alshabri ,Ghada Alshathri,Haifa Almesfer,Albatool Alnujaym
Date Created	21 Nov 2020

Table of Content

1	EXECUTIVE SUMMARY	4
1.1	Introduction.....	4
1.2	Scope	4
1.3	Risk Rating	5
1.4	Threat Security Level.....	6
1.5	Summary Table.....	7
1.6	Summary Graph.....	7
1.7	Key Findings.....	8
2	CONCLUSION	8
3	DETAILED FINDINGS.....	9
3.1	Vulnerability Table.....	9
3.2	Limitations	9
3.3	Technical Description of Findings.....	10
	APPENDIX A: ABOUT THE TEAM	13
	APPENDIX B: OUR METHODOLOGY.....	14

1 Executive Summary

1.1 *Introduction*

Penetration testing is the practice of testing a computer system, network or web application to find security vulnerabilities that an attacker could exploit. Where the main objective of penetration testing is to identify security weaknesses to lead which needs to be protected. there are two types of penetration testing black box testing where the item being tested is not known to the tester and white box testing where the item being tested is known to the tester.

In this project we are going to run a penetration testing epically black box testing where we will apply all the knowledge that we have gain during the whole semester to run the test on android system through a simulator using some helpful tools which are Jadex , MoBsf as well as adb . with the help of these tools and our knowledge we are going to identify some vulnerabilities, and documents them by explain each vulnerability in details, risk ratting each one, and identify the threat security level, finally reference to the recourses that helped us.

Note: we did use Android studio emulator instead of Genymotion, because we are familiar with it considering we are already using it on another course.

1.2 *Scope*

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

Application Name	Platform	Version	Environment	Approach
PT	ANDROID	1.0	UBUNTO	BLACKBOX 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

1.4 Threat Security Level

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

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

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

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

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

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

1.5 Summary Table

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

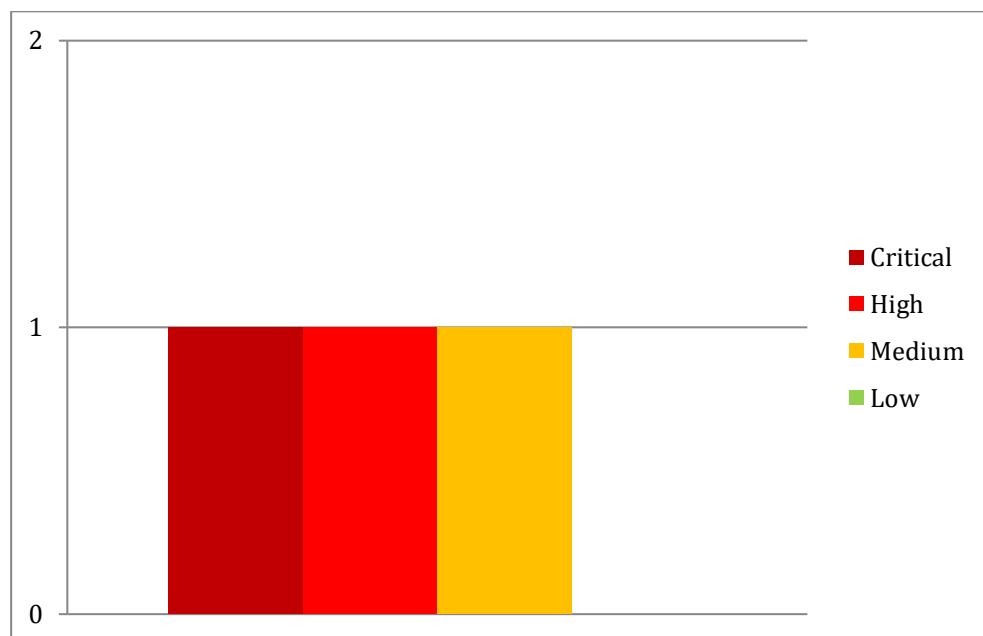
Mobile Application Penetration Testing

Critical	High	Medium	Low
1	1	1	-

1.6 Summary Graph

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

Figure 1 Application Penetration Testing



1.7 Key Findings

No.	Vulnerabilities Discovered	Platform	Severity Level
1	Mixed password with code	Android	Critical
2	<p>1.7.1.1 Limitation</p> <p>We did face a problem in this machine (Haifa's machine):</p> <ul style="list-style-type: none">The emulator stopped responding to any mouse click or even restarting it didn't work <p>We managed that by replacing it with another machine that has adb installed and did what we need to do with the emulator, since changing the emulator on the same machine and reinstalling it didn't work.</p> <p>So, the next 2 vulnerabilities have been exploited by the new machine (albatool's machine).</p> <p>Unprotected logs</p>	Android	High
3	Exported Activity	Android	Medium

2 Conclusion

We have found three vulnerabilities, the first one is the mixed password with code vulnerability where the password was mixed with the code as plain text, and it has a critical severity level, Popularity is high, the Simplicity level is medium, the technical impact is critical, and it has a high likelihood. We do recommend to separate the data with the code

Secondly, we have found the unprotected logs where we found the password in the logs, and it has a high severity level, Popularity is high, the Simplicity level is high, the technical impact is critical, and it has a medium likelihood. We do recommend to protect the logs and handle the errors appropriately by avoid revealing any sensitive information through them.

Finally, we have the third vulnerability which is the exported activity that gives attacker easy access to the system, and it has a medium severity level, Popularity is medium, the Simplicity level is low, the technical impact is high, and it has a medium likelihood. We do recommend to restrict the access by making exported = false in the code.

In the end, it was a very wonderful experience for us, despite the conditions of this semester and the other projects that we worked on, but this project was the best project
Thank you for giving us this great opportunity.

3 Detailed Findings

3.1 Vulnerability Table

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	Our observation and description about the vulnerability							
Impact	Describes the possible Technical Impact if the vulnerability is Exploited							
Proof of Concept								
Evidence to support the finding								
Recommendation	High Level recommendation to mitigate the vulnerability							
OWASP Reference	OWASP Reference							
Reference	Reference related to the vulnerability							

3.2 Limitations

We focused on installing specific tools such as android studio , Genymotion ,MobSF ,jdax , and adb and make sure that all of them work properly to help us run the penetration ,and we were limited by our scope , time and the required in this project which is a partial penetration testing and come up with three different vulnerabilities.

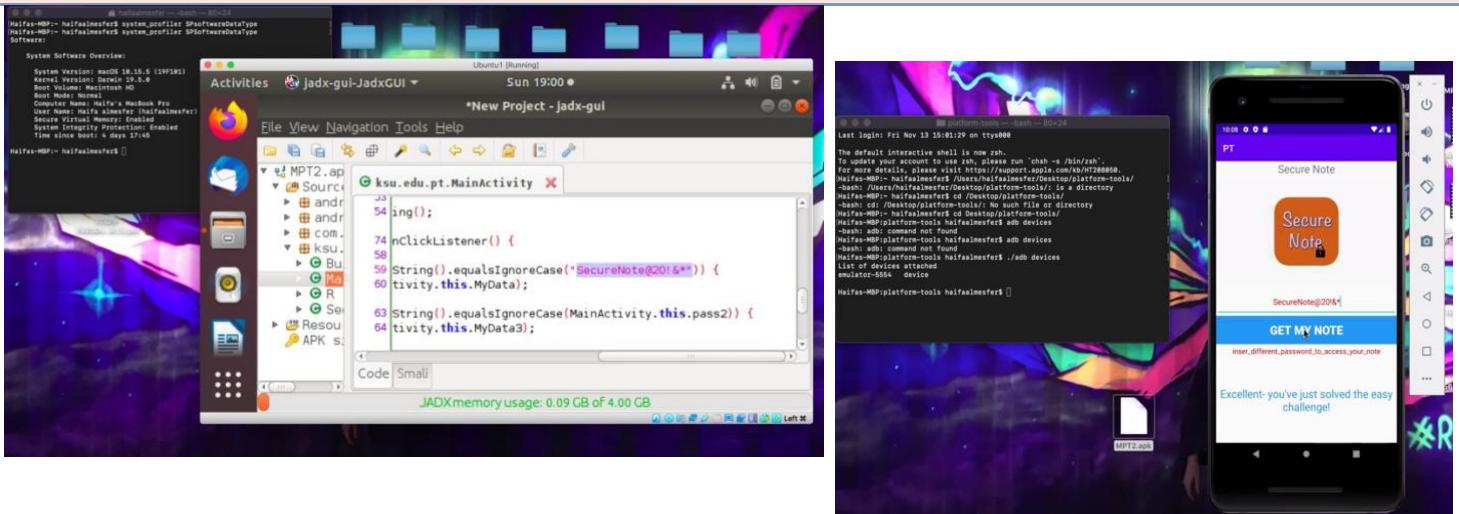
3.3 Technical Description of Findings

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

3.3.1 Mixed password with code

Severity Level	Critical				
Technical Impact	Critical	Likelihood		HIGH	
		Popularity	HIGH	Simplicity	MEDIUM
Observation	Our observation is that we found the password as a plain text in the source code by using JADX so that attackers can easily get the password without any effort.				
Impact	This will make the application vulnerable to anyone who can access the source code even if they have a little knowledge of technology.				

Proof of Concept



Recommendation	It is recommended not write the password in the source code (combining the data with the code) and we recommend storing the password by using hashing and salting techniques in protected database.
OWASP Reference	OWASP Reference
Reference	https://wiki.owasp.org/index.php/Use_of_hard-coded_password

3.3.1.1 Limitation

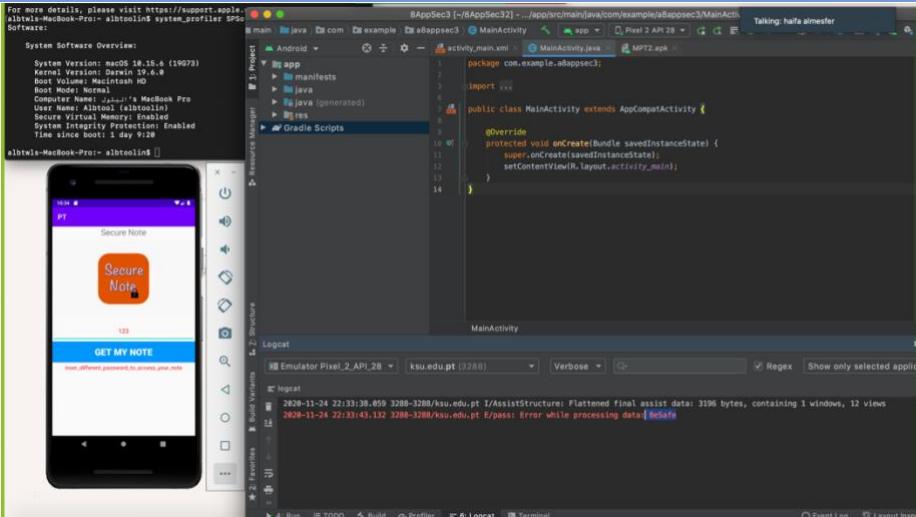
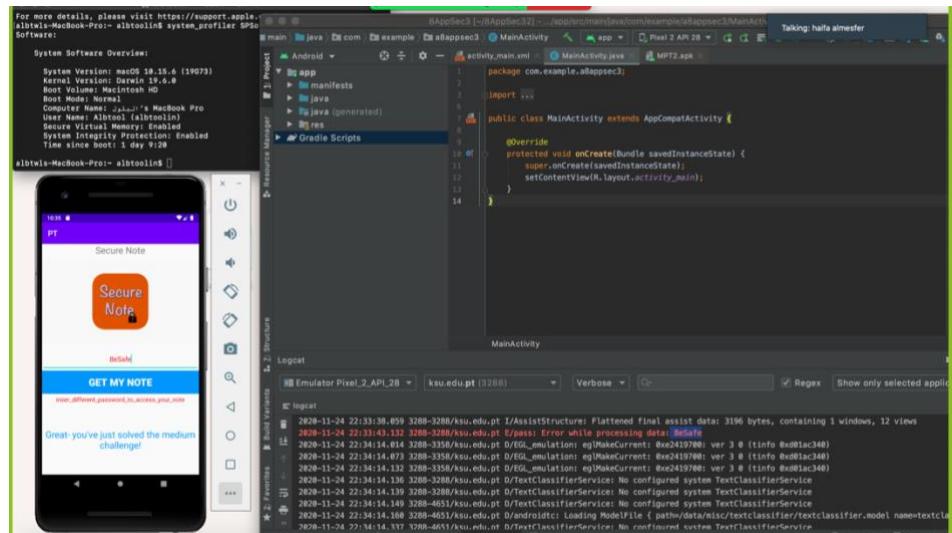
We did face a problem in this machine (Haifa's machine):

- The emulator stopped responding to any mouse click or even restarting it didn't work

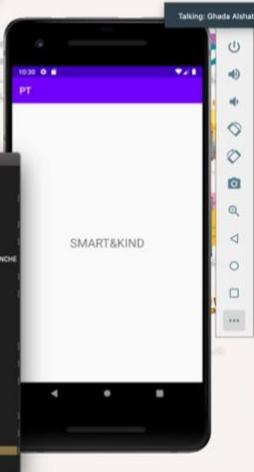
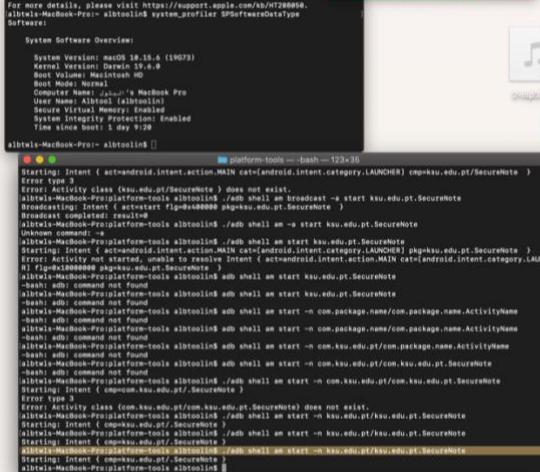
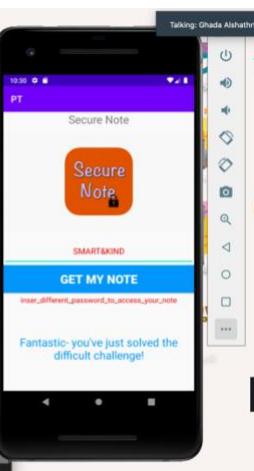
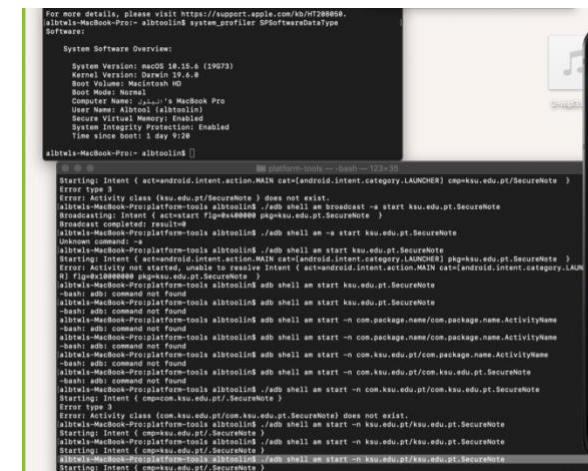
We managed that by replacing it with another machine that has adb installed and did what we need to do with the emulator, since changing the emulator on the same machine and reinstalling it didn't work.

So, the next 2 vulnerabilities have been exploited by the new machine (albatool's machine).

3.3.2 Unprotected logs

Severity Level	High				
Technical Impact	Critical	Likelihood		MEDIUM	
		Popularity	HIGH	Simplicity	HIGH
Observation	In the source code we observed that one of three passwords is related with logs, so we tried to show the logs while the app is running, when we enter a wrong password an error message appeared in the logs that contains the correct one.				
Impact	As long as the correct password not protected in the app logs, that makes it vulnerable to malicious actions where anyone can have access to the logs though basic knowledge.				
Proof of Concept					
 <pre>For more details, please visit https://support.apple.com/kb/SP905 albtwi-MacBook-Pro:~ albtwi\$ adb tools system_profiler SP905 Software: System Software Overview: System Version: macOS 10.15.4 (19G73) Kernel Version: Darwin 19.4.0 Boot Volume: Macintosh HD Boot Mode: Normal Computer Name: Albtwi's MacBook Pro User Name: Albtwi (albtwi@in) Secure Virtual Memory: Enabled System Integrity Protection: Enabled Time since boot: 1 day 9:20 albtwi-MacBook-Pro:~ albtwi\$ adb logcat I/AssistStructure(3288): Flattened final assist data: 3196 bytes, containing 1 windows, 12 views 2020-11-24 22:33:38.059 3288-3288/ksu.edu.pt I/AssistStructure: Flattened final assist data: 3196 bytes, containing 1 windows, 12 views 2020-11-24 22:33:43.132 3288-3288/ksu.edu.pt E/pass: Error while processing data ReSafe</pre>					
 <pre>For more details, please visit https://support.apple.com/kb/SP905 albtwi-MacBook-Pro:~ albtwi\$ adb tools system_profiler SP905 Software: System Software Overview: System Version: macOS 10.15.4 (19G73) Kernel Version: Darwin 19.4.0 Boot Volume: Macintosh HD Boot Mode: Normal Computer Name: Albtwi's MacBook Pro User Name: Albtwi (albtwi@in) Secure Virtual Memory: Enabled System Integrity Protection: Enabled Time since boot: 1 day 9:20 albtwi-MacBook-Pro:~ albtwi\$ adb logcat I/AssistStructure(3288): Flattened final assist data: 3196 bytes, containing 1 windows, 12 views 2020-11-24 22:33:38.059 3288-3288/ksu.edu.pt I/AssistStructure: Flattened final assist data: 3196 bytes, containing 1 windows, 12 views 2020-11-24 22:33:43.132 3288-3288/ksu.edu.pt E/pass: Error while processing data ReSafe 2020-11-24 22:34:14.014 3288-3358/ksu.edu.pt D/EGL_emulation: egMainGetCurrent: 0xe2419700; ver 3 0 (info 0x001a3c40) 2020-11-24 22:34:14.073 3288-3358/ksu.edu.pt D/EGL_emulation: egMainGetCurrent: 0xe2419700; ver 3 0 (info 0x001a3c40) 2020-11-24 22:34:14.132 3288-3358/ksu.edu.pt D/EGL_emulation: egMainGetCurrent: 0xe2419700; ver 3 0 (info 0x001a3c40) 2020-11-24 22:34:14.138 3288-3288/ksu.edu.pt D/TextClassifierService: No configured system TextClassifierService 2020-11-24 22:34:14.138 3288-3288/ksu.edu.pt D/TextClassifierService: No configured system TextClassifierService 2020-11-24 22:34:14.198 3288-4651/ksu.edu.pt D/AndroidRtc: Loading ModelFile </storage/emulated/0/rtctextclassifier/textclassifier.model> 2020-11-24 22:34:14.198 3288-4651/ksu.edu.pt D/TextClassifierService: No configured system TextClassifierService 2020-11-24 22:34:14.198 3288-4651/ksu.edu.pt D/TextClassifierService: No configured system TextClassifierService</pre>					
Recommendation	It is recommended to use non revealing error messages in logs, that could leads or gives a useful private data, and protect the log files that not anyone can have access on it.				
OWASP Reference	OWASP Reference				
Reference	https://wiki.owasp.org/index.php/Error Handling, Auditing and Logging#Best practices https://medium.com/@joecrobak/seven-best-practices-for-keeping-sensitive-data-out-of-logs-3d7bbd12904				

3.3.3 Exported Activity

Severity Level	Medium	Likelihood	MEDIUM							
Technical Impact	HIGH	Popularity	MEDIUM	Simplicity	LOW					
Observation	we observed that in mobSF there is an activity we can exploit and it's code "exported=true, so we look for it in the activity code ,we tried to access it by adb command, and we succeed									
Impact	This activity works like a back door, once it discovered it will be easy to the attacker to gain access to the app as an authorized user when he is not.									
Proof of Concept										
 										
 										
Recommendation	It is recommended to make sure of that every activity do what it required no more, and delete any activity that we don't need before launching the application , and restrict the access to any activity by its code "exported=false".									
OWASP Reference	OWASP Reference									
Reference	https://oldbam.github.io/android/security/android-vulnerabilities-insecurebank-activities https://securitygrind.com/exploiting-android-components-abusing-activities/									

Appendix A: About the Team

8AppSec3:		
Student Name	Serial Number	Role
Shahad Alshabri	16	Teamwork
Ghada Alshathri	9	Teamwork
Haifa Almesfer	26	Teamwork
Albatool Alnujaym	13	Teamwork

Appendix B: Our 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 as shown in Figure below and explained in following sections:

TASK 1 android code

We have not faced any problem with the android studio installation or with the android studio environment.

Install Android Studio

We didn't need to install the Android Studio because we already have it installed due to our needs for it on the software engineering course see Figure 1.

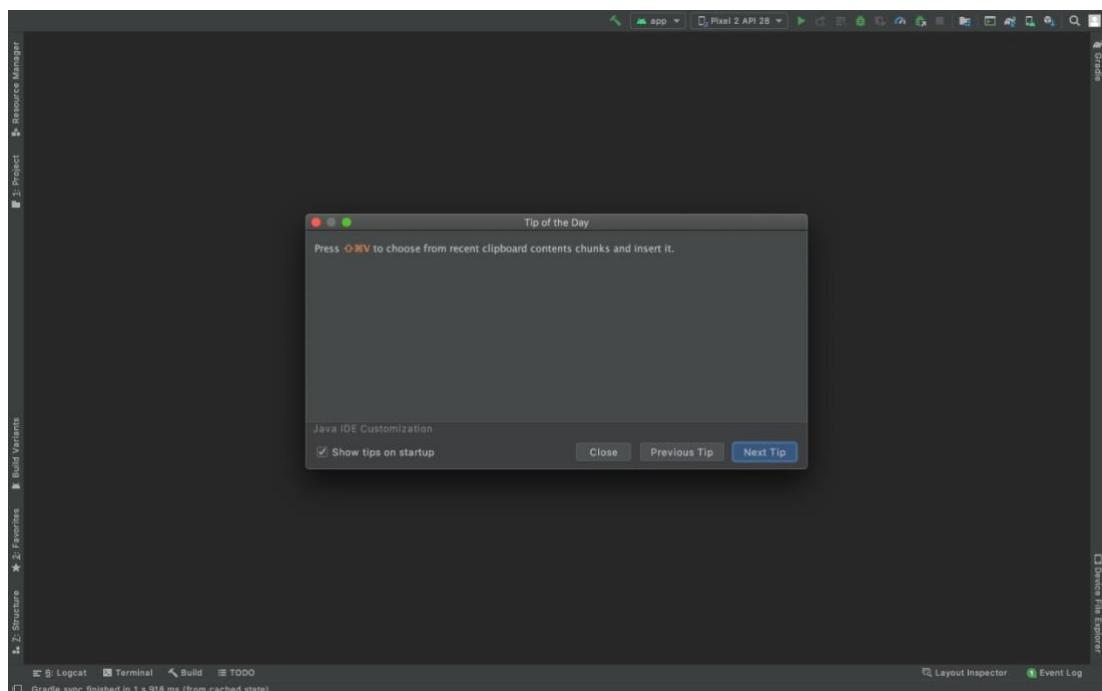


Figure 2 first page when we open android studio

Program and run a simple Android application

We have not faced any problem with the android studio environment. And we have practiced the login activity as you suggested.

Login Activity

The Figures below will show you the sequence of implementing the login activity by the android studio step by step.

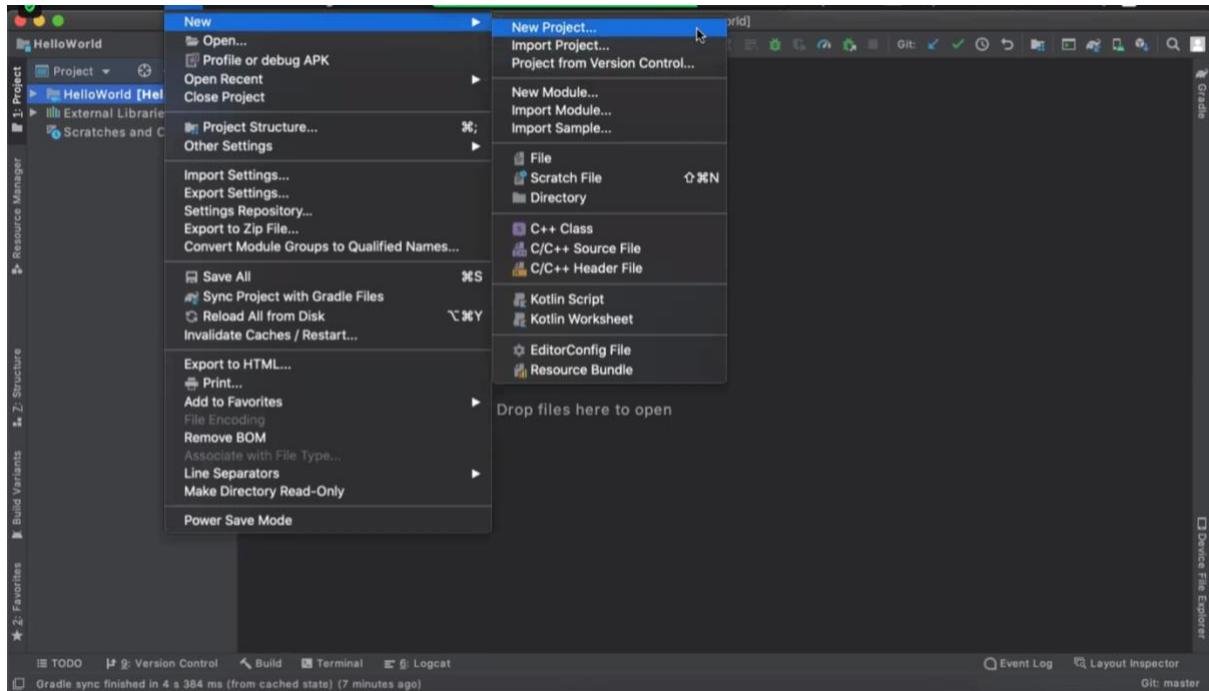


Figure 3 [1] crate a new project

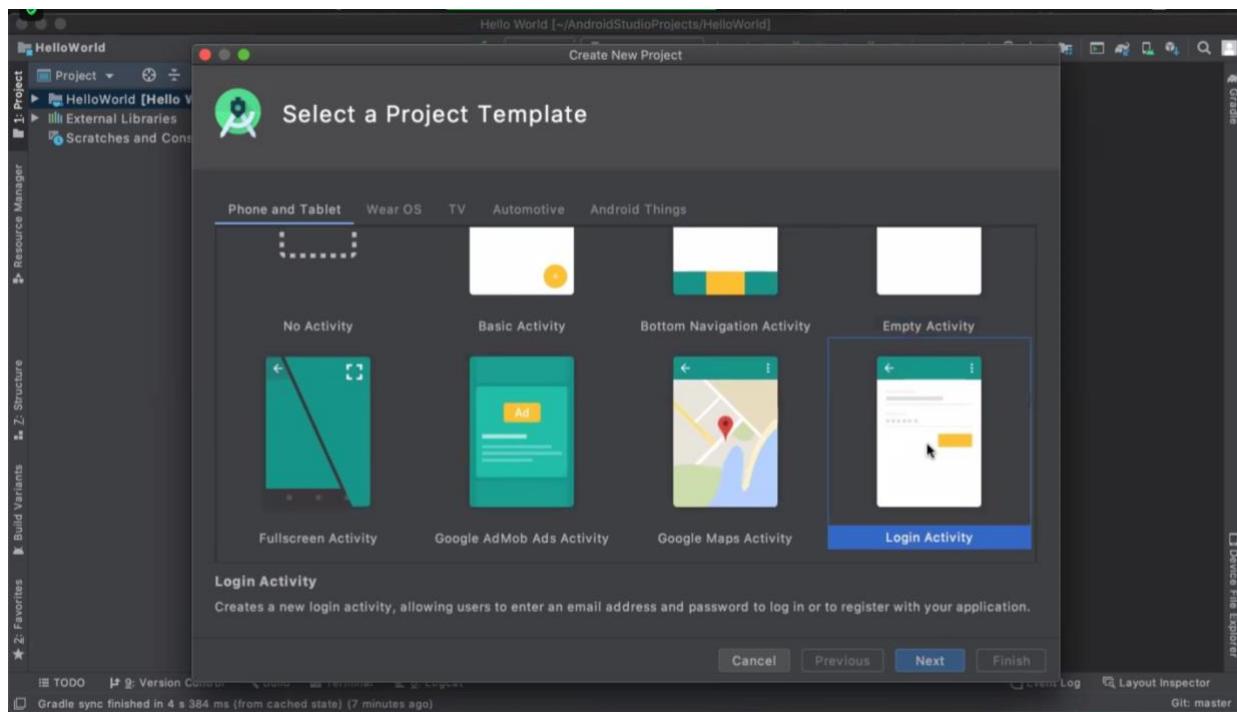


Figure 4 [2] select the project templet

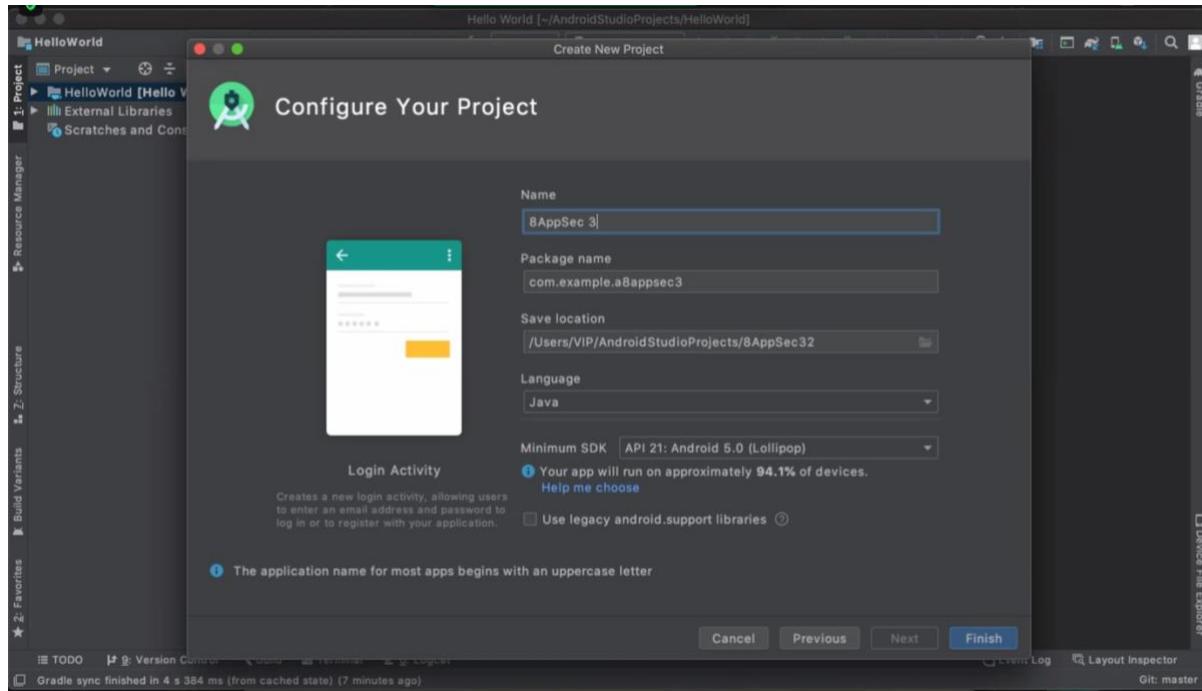


Figure 5 [3]

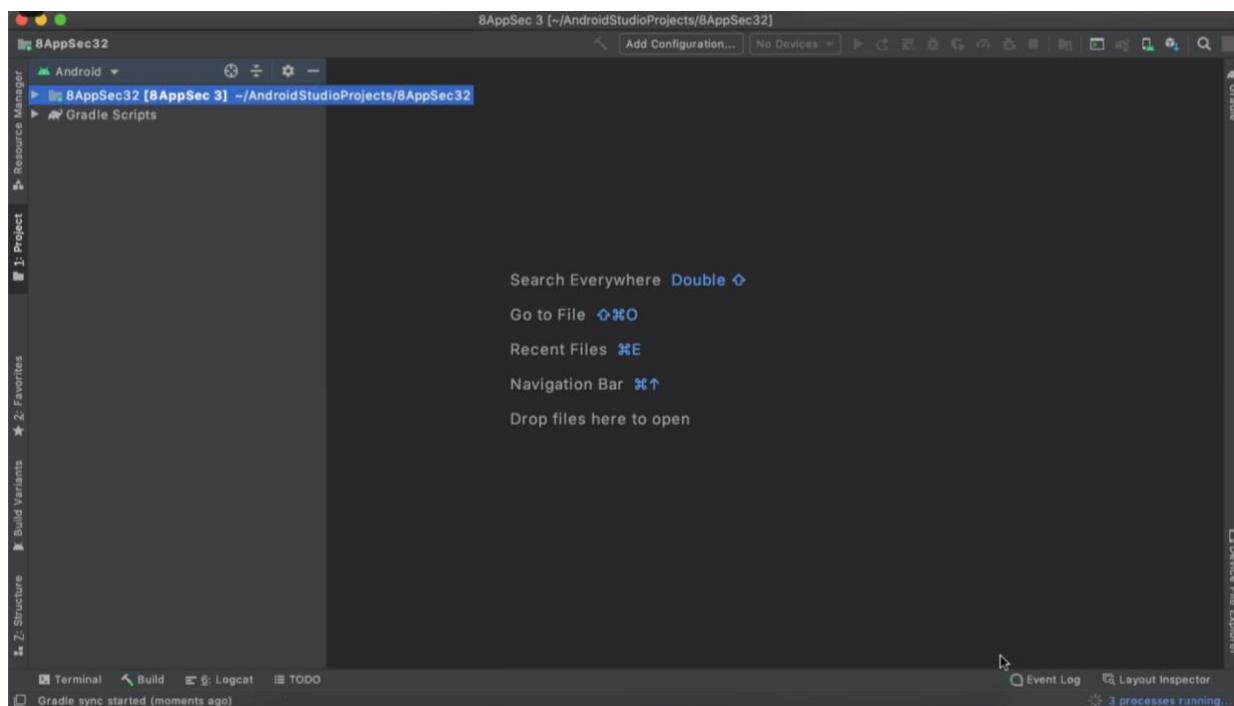


Figure 6 [4]

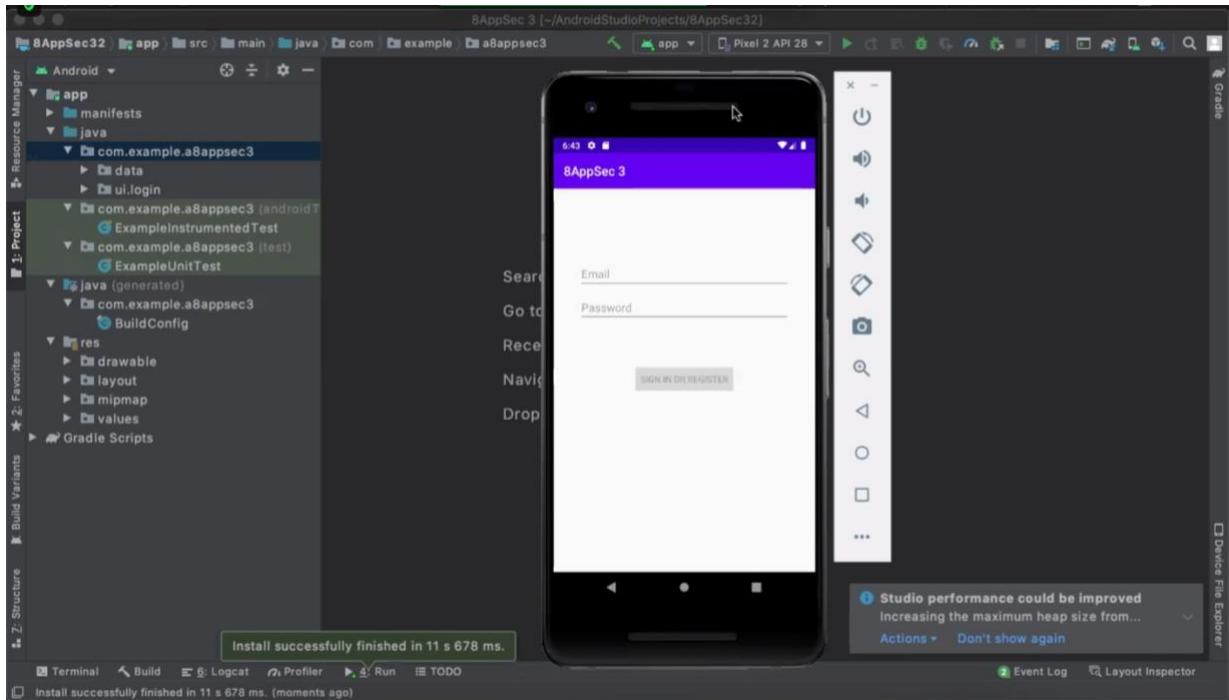


Figure 7 [5] running the application

```

private LoginViewModel loginViewModel;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);
    loginViewModel = ViewModelProviders.of(this).get(LoginViewModel.class);

    final EditText usernameEditText = findViewById(R.id.username);
    final EditText passwordEditText = findViewById(R.id.password);
    final Button loginButton = findViewById(R.id.login);
    final ProgressBar loadingProgressBar = findViewById(R.id.loading);

    loginViewModel.getLoginFormState().observe(owner: this, (loginFormState) -> {
        if (loginFormState == null) {
            return;
        }
        loginButton.setEnabled(loginFormState.isDataValid());
        if (loginFormState.getUsernameError() != null) {
            usernameEditText.setError(getString(loginFormState.getUsernameError()));
        }
        if (loginFormState.getPasswordError() != null) {
            passwordEditText.setError(getString(loginFormState.getPasswordError()));
        }
    });

    loginViewModel.getLoginResult().observe(owner: this, (loginResult) -> {
        if (loginResult == null) {
            return;
        }
        loadingProgressBar.setVisibility(View.GONE);
    });
}

LoginActivity : onCreate()

```

Figure 8 [6] the source code

TASK 2 mpt environment setup

We have faced some problems in this task epically with JADX and mobSF which we will talk about later on the next section of the report.

Genymotion Software

we decided to use a simulator instead of the real android phone and we already have it on the host machine as you recommended. The Figures below will show you the sequence of activating genymotion android emulator step by step.

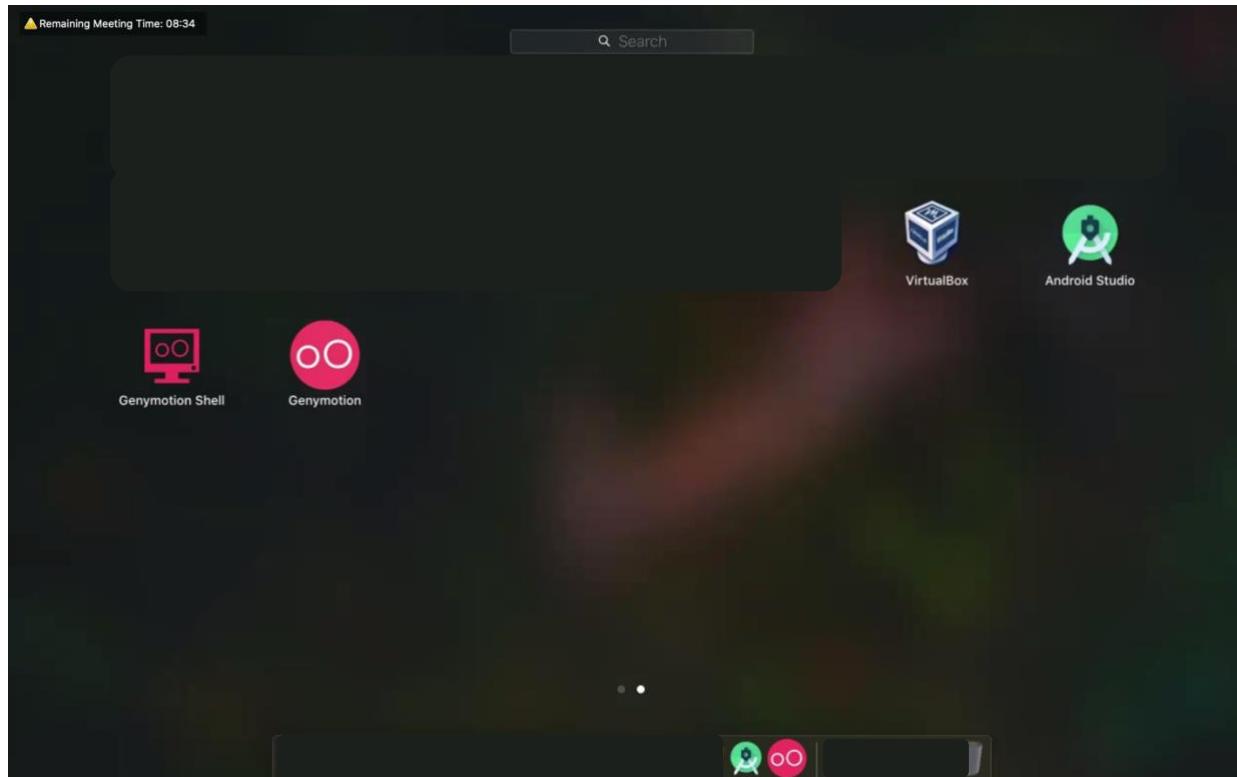


Figure 9 [1]opening the Genymotion to activate it

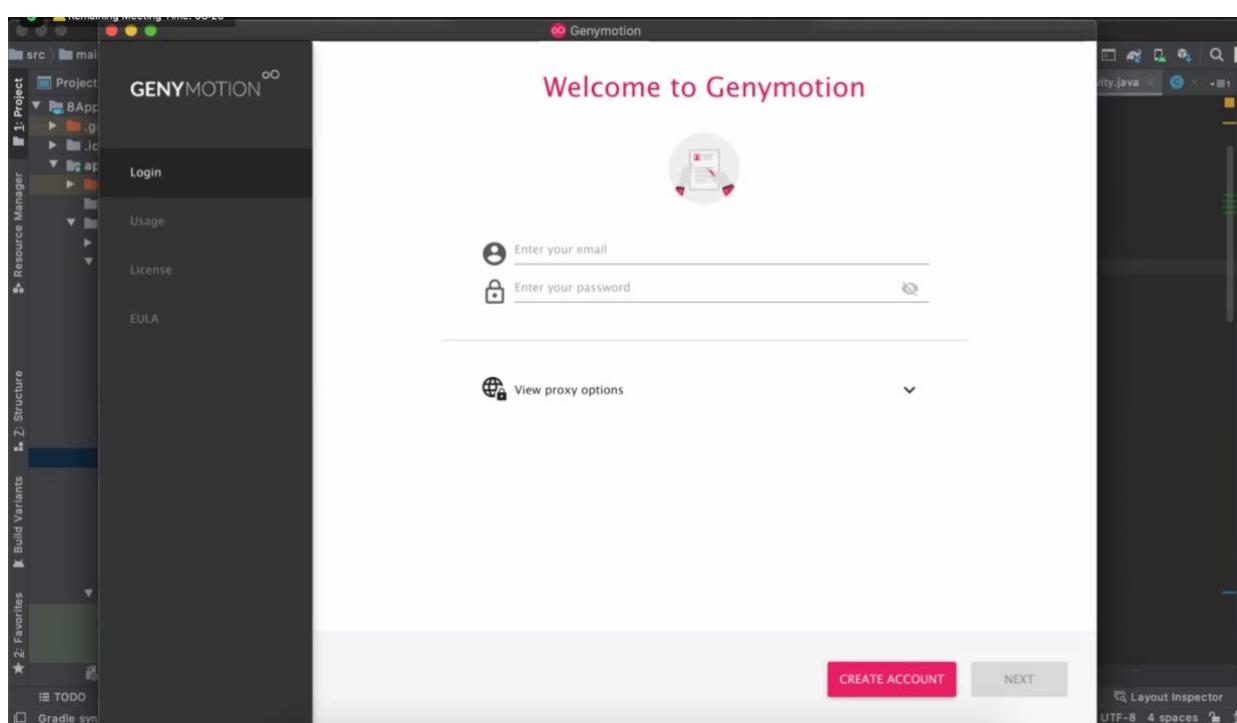


Figure 10 [2]

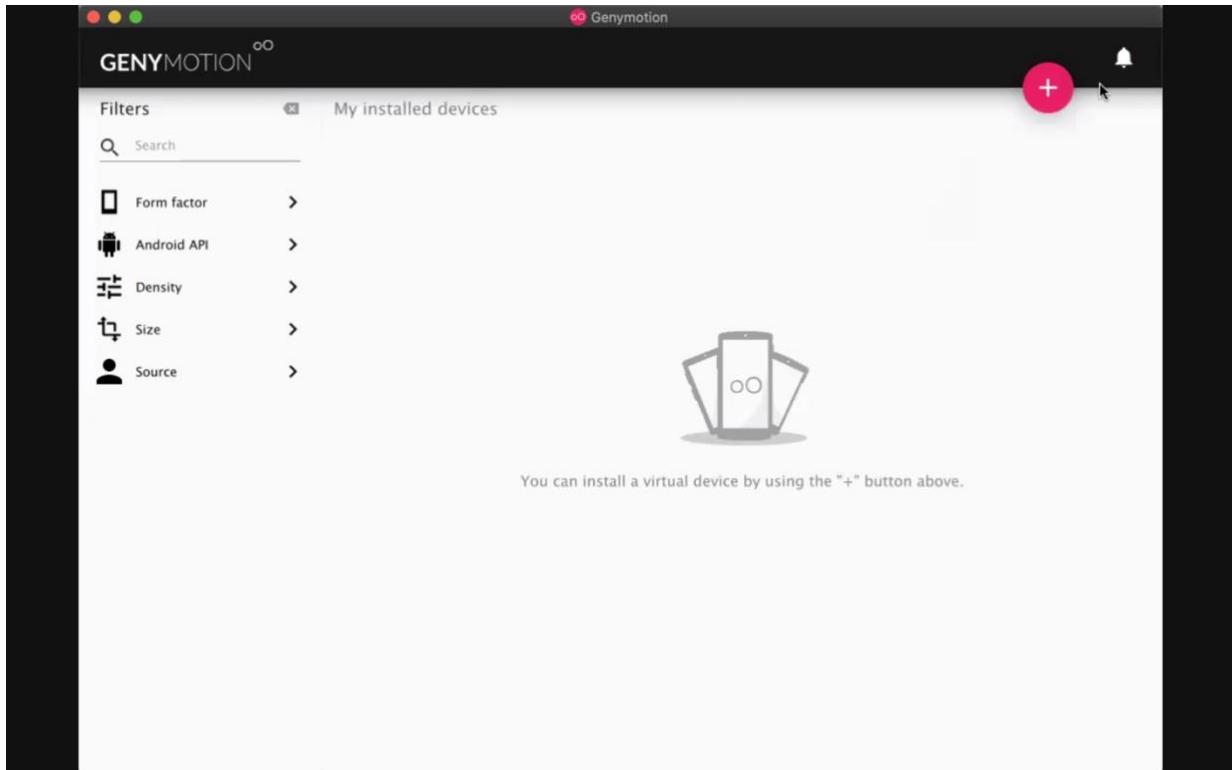


Figure 11 [3]

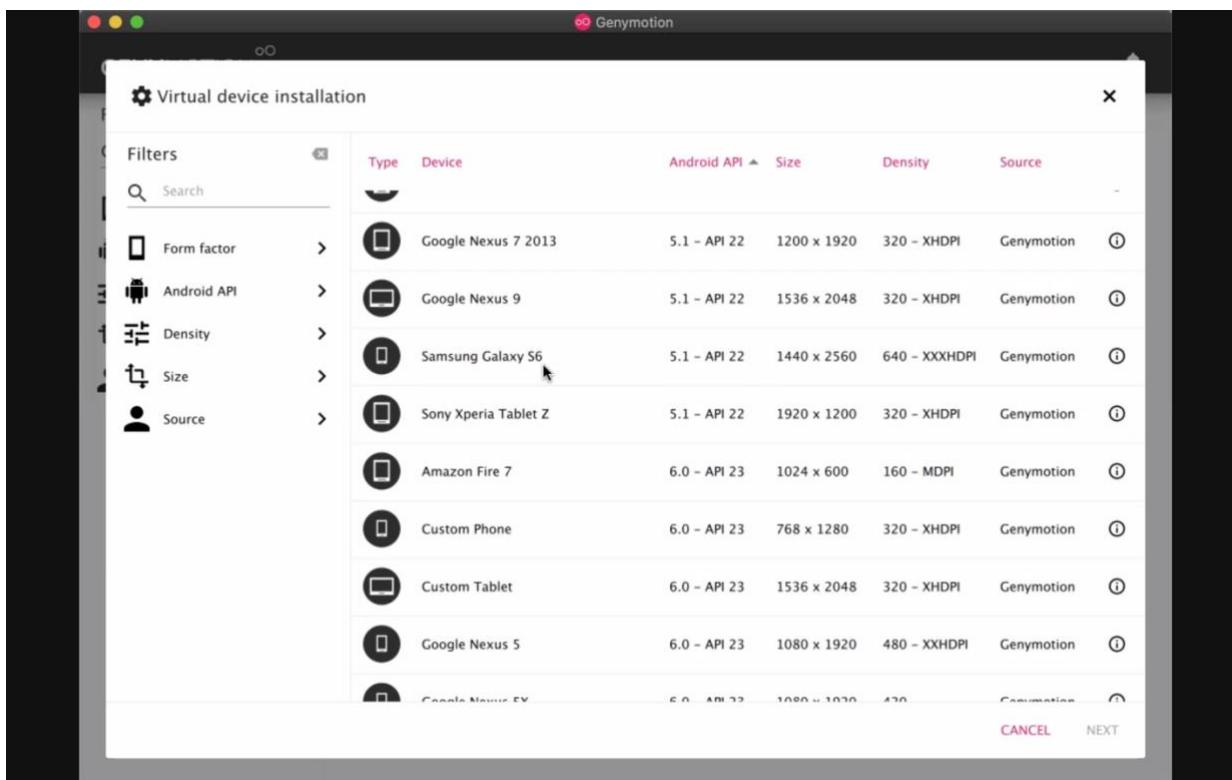


Figure 12 [4]

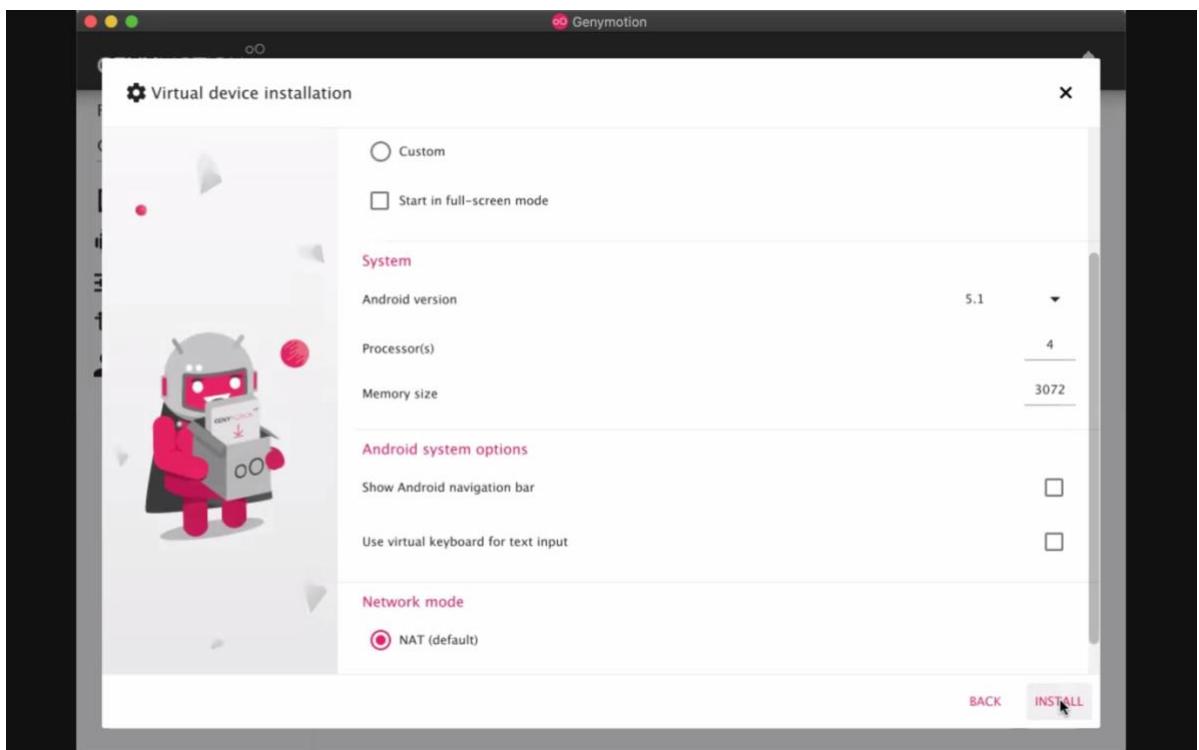


Figure 13 [5]

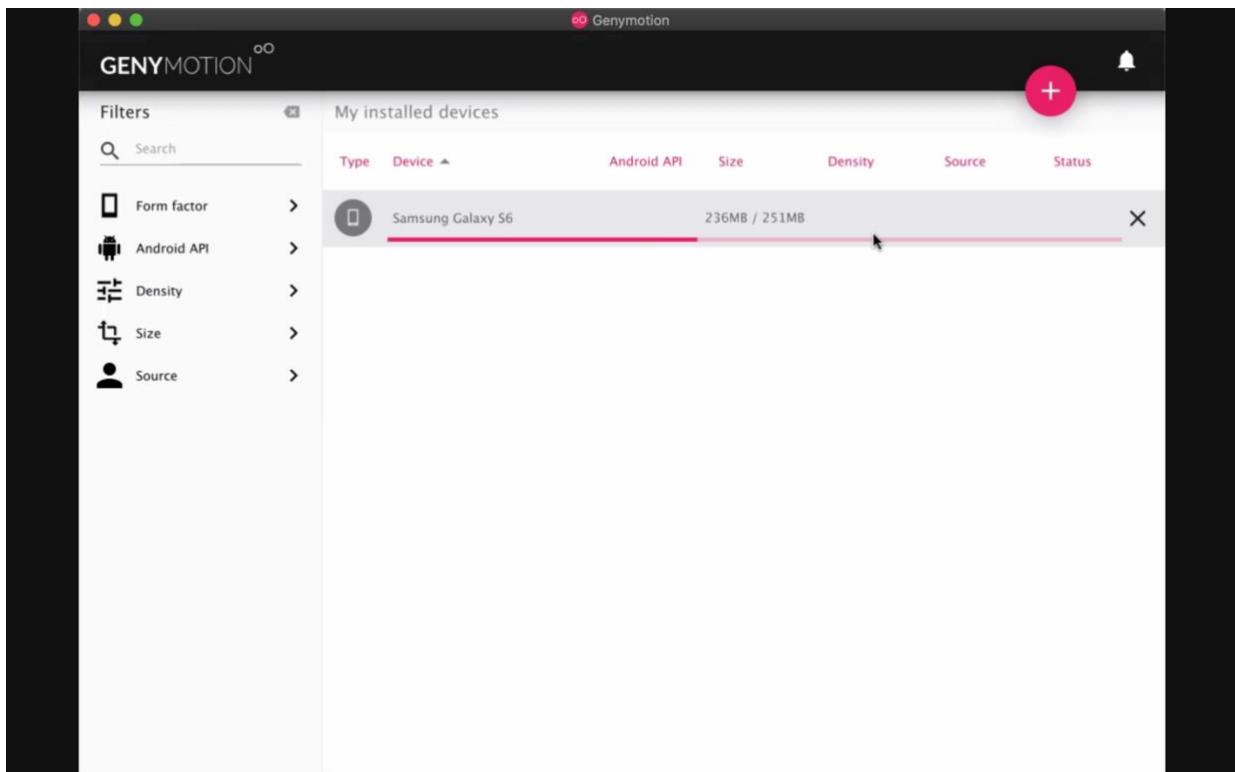


Figure 14 [6]

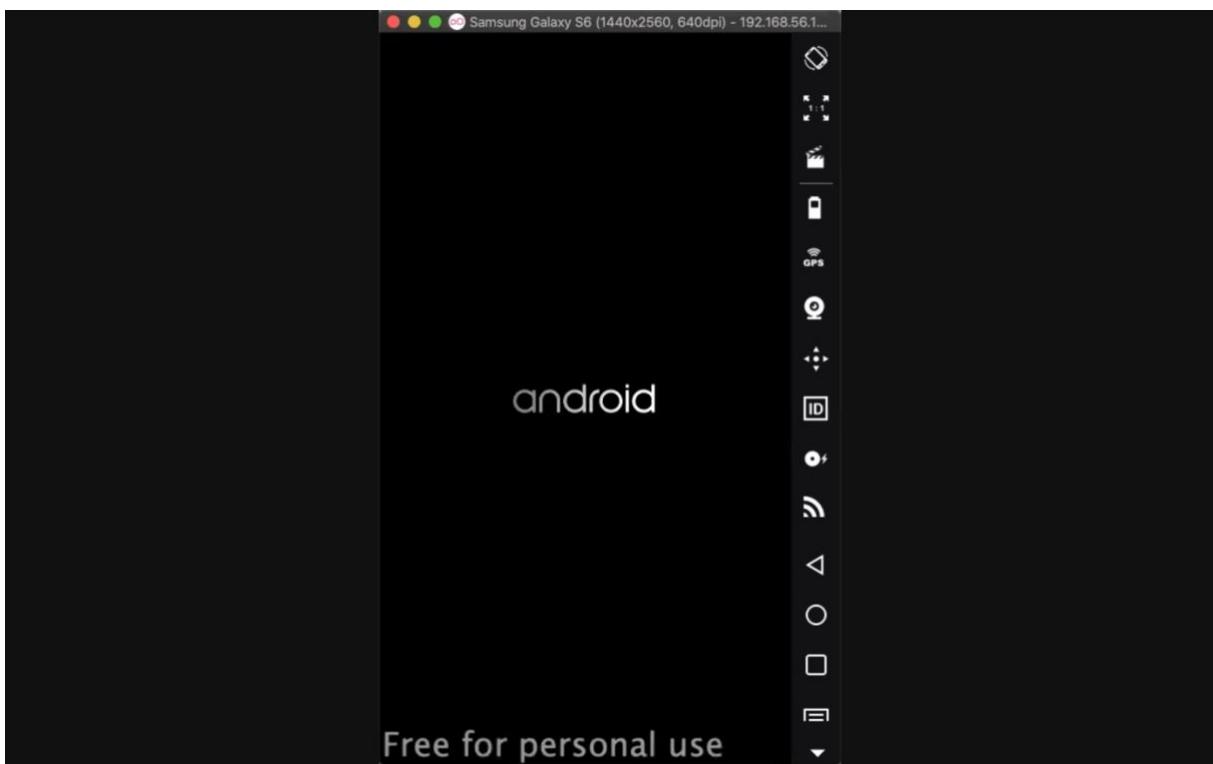


Figure 15 finally the Genymotion is activated

Installing ADB

Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device. The adb command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.

7. On your phone's screen, you should see a prompt to allow or deny USB Debugging access. Naturally, you will want to grant USB Debugging access when prompted (and tap the always allow check box if you never want to see that prompt again).
8. Finally, re-enter the command from step #6. If everything was successful, you should now see your device's serial number in the command prompt. Yay! You can now run any ADB command on your device! Now go forth and start modding your phone by following our extensive list of tutorials!

How to Install ADB on macOS

1. [Download the ADB ZIP file for macOS](#)
2. Extract the ZIP to an easily-accessible location (like the Desktop for example).
3. Open Terminal.
4. To browse to the folder you extracted ADB into, enter the following command: `cd /path/to/extracted/folder/`
5. For example, on my Mac it was: `cd /Users/Doug/Desktop/platform-tools/`
6. Connect your device to your Mac with a compatible USB cable. Change the USB connection mode to "file transfer (MTP)" mode. This is not always required for every device, but it's best to just leave it in this mode so you don't run into any issues.
7. Once the Terminal is in the same folder your ADB tools are in, you can execute the following command to launch the ADB daemon: `adb devices`
8. On your device, you'll see an "Allow USB debugging" prompt. Allow the connection.
9. Finally, re-enter the command from step #7. If everything was successful, you should now see your device's serial number in macOS's Terminal window. Congratulations! You can now run any ADB command on your device! Now go forth and start modding your phone by following our extensive list of tutorials!

How to Install ADB on Linux

1. [Download the ADB ZIP file for Linux](#)
2. Extract the ZIP to an easily-accessible location (like the Desktop for example).
3. Open a Terminal window.
4. Enter the following command: `cd /path/to/extracted/folder/`
5. This will change the directory to where you extracted the ADB files.
6. So for example: `cd /Users/Doug/Desktop/platform-tools/`
7. Connect your device to your Linux machine with your USB cable. Change the connection mode to "file transfer (MTP)" mode. This is not always necessary for every device, but it's recommended so you don't run into any issues.

Open "<https://dl.google.com/android/repository/platform-tools-latest-darwin.zip>" in a new tab

POCO X3 gets its first custom ROM:
Nitrogen OS based on Android 10

October 7, 2020

[More Forum Links](#)

SUGGESTED APPS

Official XDA Forum App
The XDA App is the fastest way to access the forums on mobile.

Navigation Gestures
Customizable gesture control for any Android device.

XDA Labs
Labs is an independent app store that gives developers full control over their work.

Figure 16

7. On your phone's screen, you should see a prompt to allow or deny USB Debugging access. Naturally, you will want to grant USB Debugging access when prompted (and tap the always allow check box if you never want to see that prompt again).
8. Finally, re-enter the command from step #6. If everything was successful, you should now see your device's serial number in the command prompt. Yay! You can now run any ADB command on your device! Now go forth and start modding your phone by following our extensive list of tutorials!

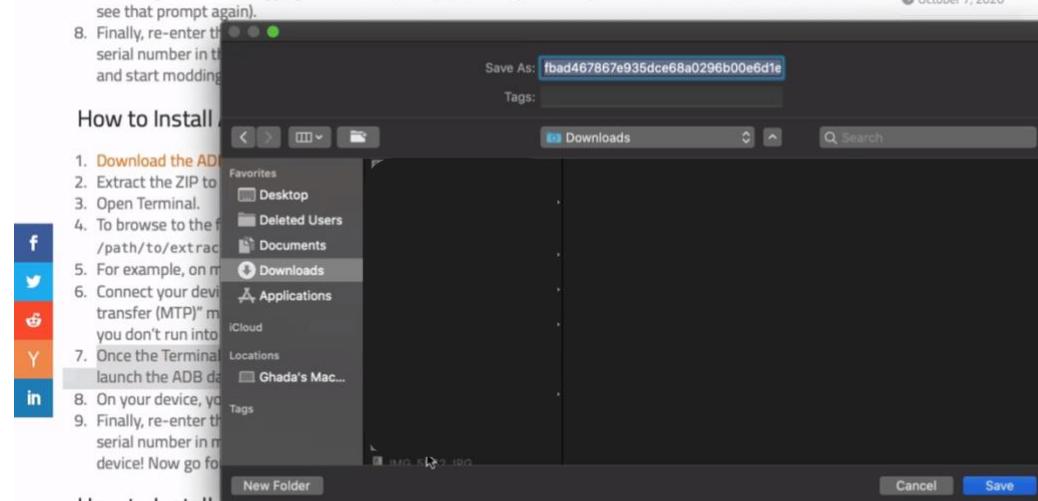
How to Install ADB on Linux

1. [Download the ADB ZIP file for Linux](#)
2. Extract the ZIP to an easily-accessible location (like the Desktop for example).
3. Open Terminal.
4. To browse to the folder you extracted ADB into, enter the following command: `cd /path/to/extracted/folder/`
5. For example, on my Mac it was: `cd /Users/Doug/Desktop/platform-tools/`
6. Connect your device to your Linux machine with your USB cable. Change the connection mode to "file transfer (MTP)" mode. This is not always necessary for every device, but it's recommended so you don't run into any issues.
7. Once the Terminal is in the same folder your ADB tools are in, you can execute the following command to launch the ADB daemon: `adb devices`
8. On your device, you'll see an "Allow USB debugging" prompt. Allow the connection.
9. Finally, re-enter the command from step #7. If everything was successful, you should now see your device's serial number in the command prompt. Yay! You can now run any ADB command on your device! Now go forth and start modding your phone by following our extensive list of tutorials!

How to Install ADB on Linux

1. [Download the ADB ZIP file for Linux](#)
2. Extract the ZIP to an easily-accessible location (like the Desktop for example).
3. Open a Terminal window.
4. Enter the following command: `cd /path/to/extracted/folder/`
5. This will change the directory to where you extracted the ADB files.
6. So for example: `cd /Users/Doug/Desktop/platform-tools/`
7. Connect your device to your Linux machine with your USB cable. Change the connection mode to "file transfer (MTP)" mode. This is not always necessary for every device, but it's recommended so you don't run into any issues.

Figure 17



POCO X3 gets its first custom ROM:

Nitrogen OS based on Android 10

October 7, 2020

Official XDA Forum App
The XDA App is the fastest way to access the forums on mobile.

Navigation Gestures
Customizable gesture control for any Android device.

XDA Labs
Labs is an independent app store that gives developers full control over their work.

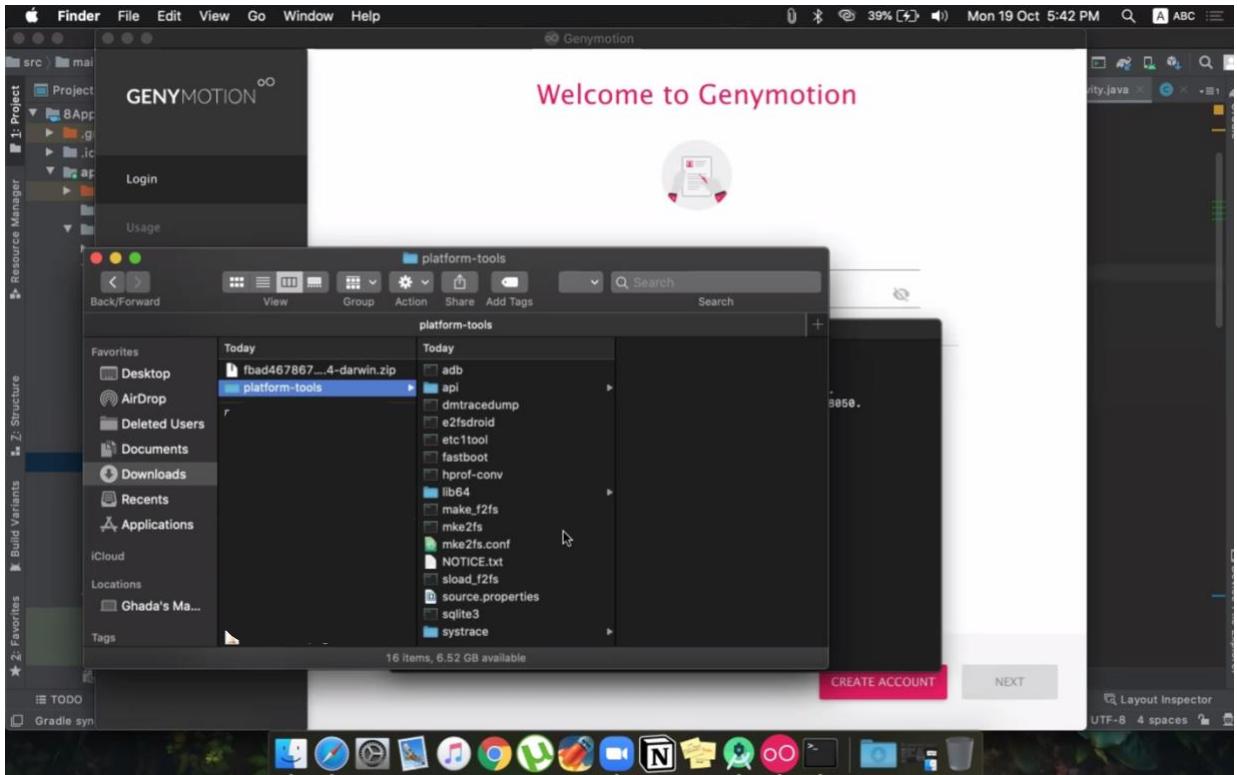


Figure 18

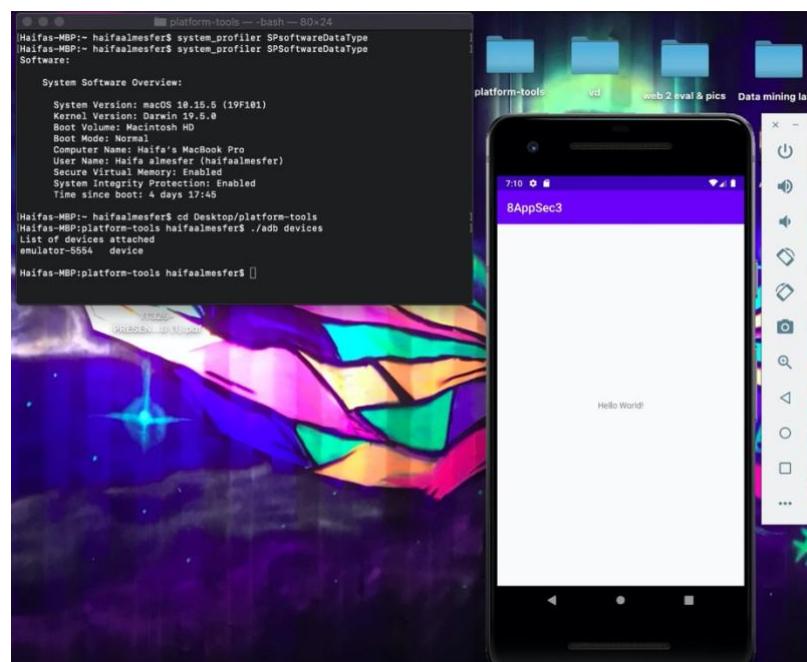


Figure 18 adb run

Installing Jadx

We have chosen to install Jadx that stand for (Dex to Java decompiler) which is a CLI/GUI tool to produce Java source code from Android Dex and APK files, instead of Apktool because it suit us well for its advantages such as , it is ability to Produces human-friendly source code and for Allowing you to display the app structure after decompiling. . The Figures below will show you the sequence of installing the Jadx.

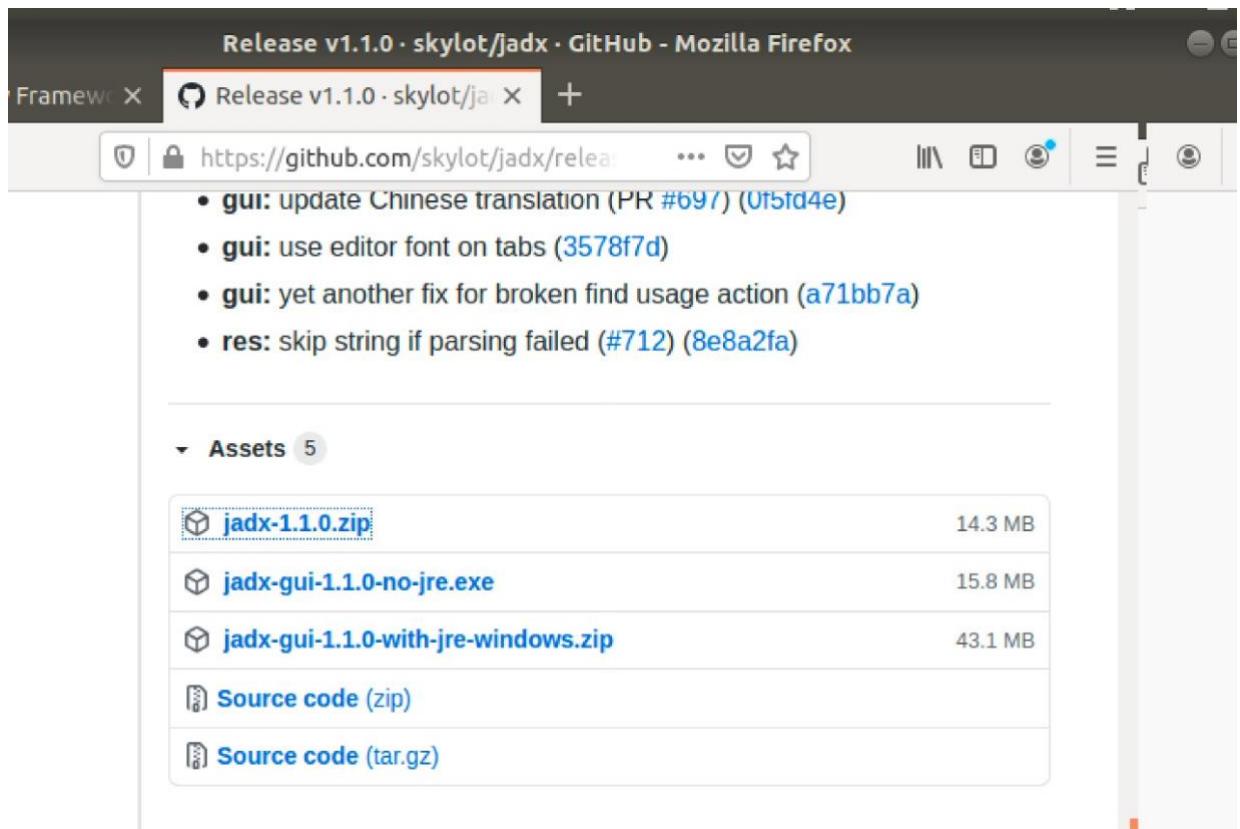


Figure 19 the source we used

```

root@haifa-VirtualBox: /usr/local/bin
File Edit View Search Terminal Help
haifa@haifa-VirtualBox:~$ sudo su
[sudo] password for haifa:
root@haifa-VirtualBox:/home/haifa# cd /usr/local/bin
root@haifa-VirtualBox:/usr/local/bin# ls
root@haifa-VirtualBox:/usr/local/bin# cd /usr/local/bin/
root@haifa-VirtualBox:/usr/local/bin# ls
root@haifa-VirtualBox:/usr/local/bin# clear
root@haifa-VirtualBox:/usr/local/bin# wget https://github.com/skylot/jadx/releases/download/v1.1.0/jadx-1.1.0.zip
--2020-11-01 21:33:51-- https://github.com/skylot/jadx/releases/download/v1.1.0/jadx-1.1.0.zip
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-production-release-asset-2e65be.s3.amazonaws.com/8859474/e3d11600-1920-1ea-8b76-f06f1e814678?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F201101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20201101T183353Z&X-Amz-Expires=300&X-Amz-Signature=7e6801c08a176036617388416066a3e7843d069df8a31a8f34fb9a5633ea5c47&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=8859474&response-content-disposition=attachment%3B%20filename%3Djadx-1.0.zip&response-content-type=application%2Foctet-stream [following]
--2020-11-01 21:33:52-- https://github-production-release-asset-2e65be.s3.amazonaws.com/8859474/e3d11600-1920-1ea-8b76-f06f1e814678?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20201101%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20201101T183353Z&X-Amz-Expires=00&X-Amz-Signature=7e6801c08a176036617388416066a3e7843d069df8a31a8f34fb9a5633ea5c47&X-Amz-SignedHeaders=host&actor_id=0&key_id=0&repo_id=8859474&response-content-disposition=attachment%3B%20filename%3Djadx-1.0.zip&response-content-type=application%2Foctet-stream
Resolving github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)... 54.231.40.11
Connecting to github-production-release-asset-2e65be.s3.amazonaws.com (github-production-release-asset-2e65be.s3.amazonaws.com)|54.231.40.11|:443... connected.

```

Figure 20

```

root@haifa-VirtualBox: /usr/local/bin
File Edit View Search Terminal Help
root@haifa-VirtualBox:/usr/local/bin# unzip jadx-1.1.0.zip -d jadx-1.10
Archive: jadx-1.1.0.zip
  inflating: jadx-1.10/README.md
  inflating: jadx-1.10/LICENSE
  creating: jadx-1.10/bin/
  inflating: jadx-1.10/bin/jadx
  inflating: jadx-1.10/bin/jadx-gui.bat
  inflating: jadx-1.10/bin/jadx.bat
  inflating: jadx-1.10/bin/jadx-gui
  inflating: jadx-1.10/NOTICE
  creating: jadx-1.10/lib/
  inflating: jadx-1.10/lib/commons-lang3-3.9.jar
  inflating: jadx-1.10/lib/jcommander-1.78.jar
  inflating: jadx-1.10/lib/jsr305-3.0.2.jar
  inflating: jadx-1.10/lib/rxjava2-swing-0.3.7.jar
  inflating: jadx-1.10/lib/apksig-3.5.2.jar
  inflating: jadx-1.10/lib/util-2.3.4.jar
  inflating: jadx-1.10/lib/error_prone_annotations-2.3.2.jar
  inflating: jadx-1.10/lib/image-viewer-1.2.3.jar
  inflating: jadx-1.10/lib/dx-1.16.jar
  inflating: jadx-1.10/lib/asm-7.2.jar
  inflating: jadx-1.10/lib/slf4j-api-1.7.29.jar
  inflating: jadx-1.10/lib/dexlib2-2.3.4.jar
  inflating: jadx-1.10/lib/logback-core-1.2.3.jar
  inflating: jadx-1.10/lib/jadx-core-1.1.0.jar
  inflating: jadx-1.10/lib/quava-28.1-jre.jar
  inflating: jadx-1.10/lib/jfontchooser-1.0.5.jar
  inflating: jadx-1.10/lib/antlr-runtime-3.5.2.jar
  inflating: jadx-1.10/lib/listenablefuture-9999.0-empty-to-avoid-conflict-with-quava.jar

```

Figure 21

```
jadx-1.1.0.zip          100%[=====] 14.32M 264KB/s in 89s
2020-11-01 21:35:22 (165 KB/s) - 'jadx-1.1.0.zip' saved [15014403/15014403]
root@haifa-VirtualBox:/usr/local/bin# ls
jadx-1.1.0.zip
root@haifa-VirtualBox:/usr/local/bin# █
```

Figure 22

```
inflating: jadx-1.10/lib/annotations-18.0.0.jar
root@haifa-VirtualBox:/usr/local/bin# ls
bin jadx-1.10 jadx-1.1.0.zip lib LICENSE NOTICE README.md
root@haifa-VirtualBox:/usr/local/bin# rm jadx-1.1.0.zip
root@haifa-VirtualBox:/usr/local/bin# cd jadx-1.1.0/bin/
bash: cd: jadx-1.1.0/bin/: No such file or directory
root@haifa-VirtualBox:/usr/local/bin# cd jadx-1.10/bin/
root@haifa-VirtualBox:/usr/local/bin/jadx-1.10/bin# ls
jadx jadx.bat jadx-gui jadx-gui.bat
root@haifa-VirtualBox:/usr/local/bin/jadx-1.10/bin# cd ../..
root@haifa-VirtualBox:/usr/local/bin# ln -s jadx-1.10/bin/ jadx jadx
ln: target 'jadx' is not a directory
root@haifa-VirtualBox:/usr/local/bin# ln -s jadx-1.10/bin/jadx jadx
root@haifa-VirtualBox:/usr/local/bin# ln -s jadx-1.10/bin/jadx-gui jadx-gui
root@haifa-VirtualBox:/usr/local/bin# ls
bin jadx jadx-1.10 jadx-gui lib LICENSE NOTICE README.md
```

Figure 23

```
root@haifa-VirtualBox:/usr/local/bin# jadx-gui
Nov 01, 2020 9:54:28 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
root@haifa-VirtualBox:/usr/local/bin# exit
exit
haifa@haifa-VirtualBox:~$ jadx
ERROR - Incorrect arguments: Please specify input file
haifa@haifa-VirtualBox:~$ jadx-gui
Nov 01, 2020 10:00:44 PM java.util.prefs.FileSystemPreferences$1 run
INFO: Created user preferences directory.
Gtk-Message: 22:00:45.903: Failed to load module "canberra-gtk-module"
haifa@haifa-VirtualBox:~$ █
```

Figure 24

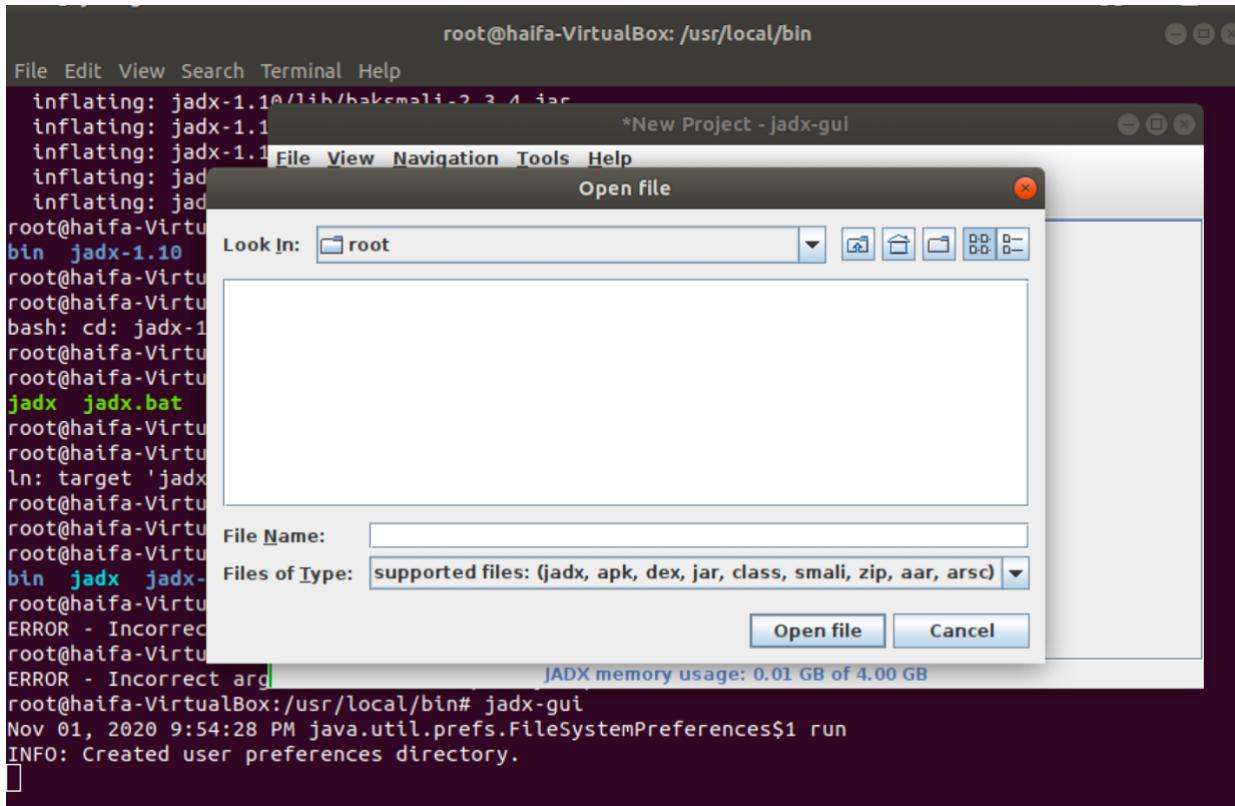


Figure 25

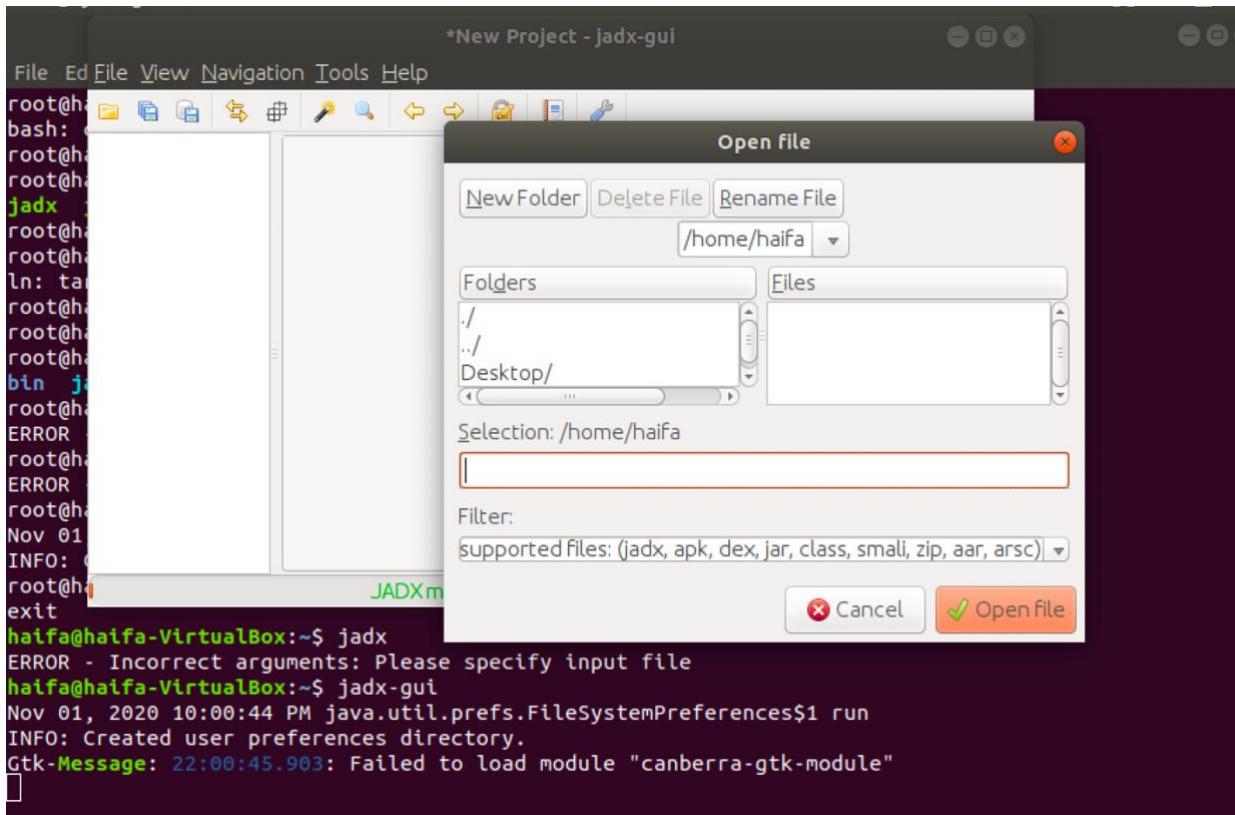
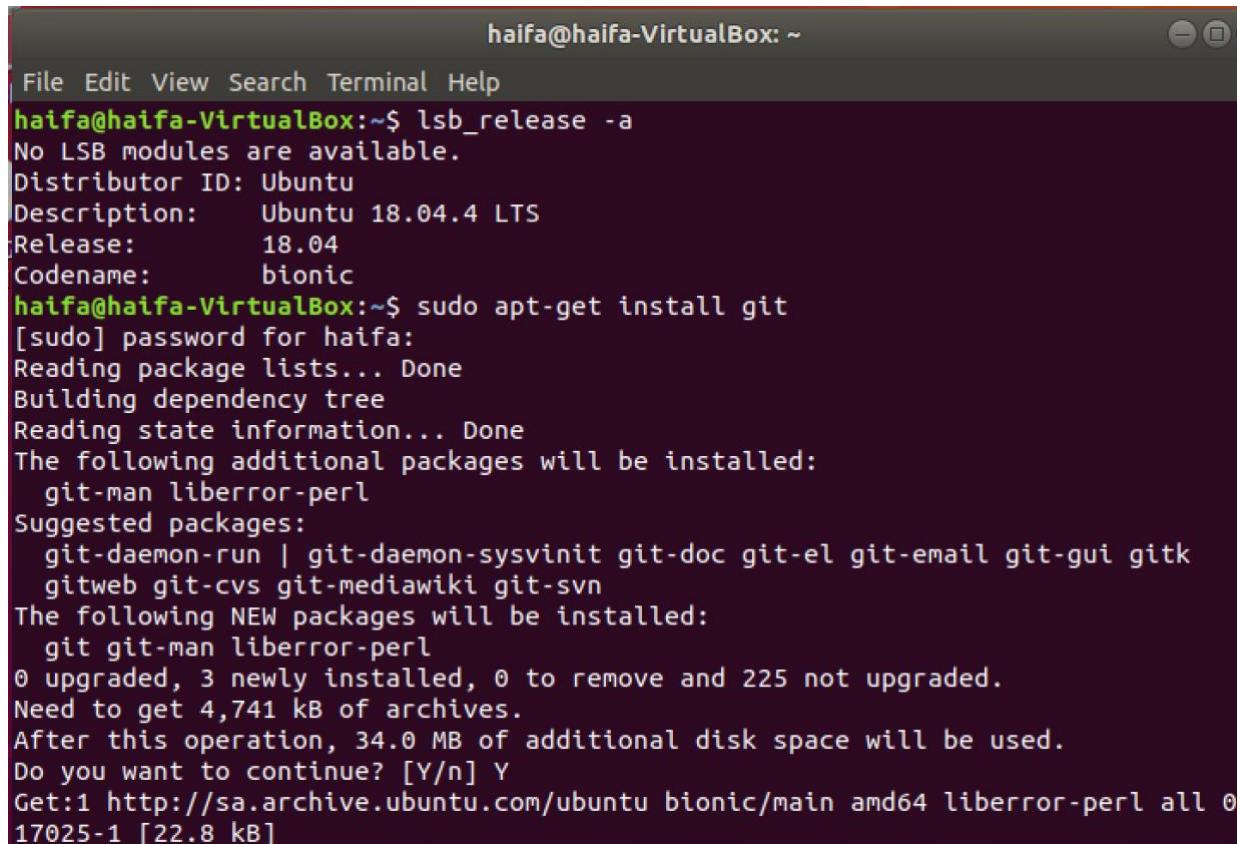


Figure 26

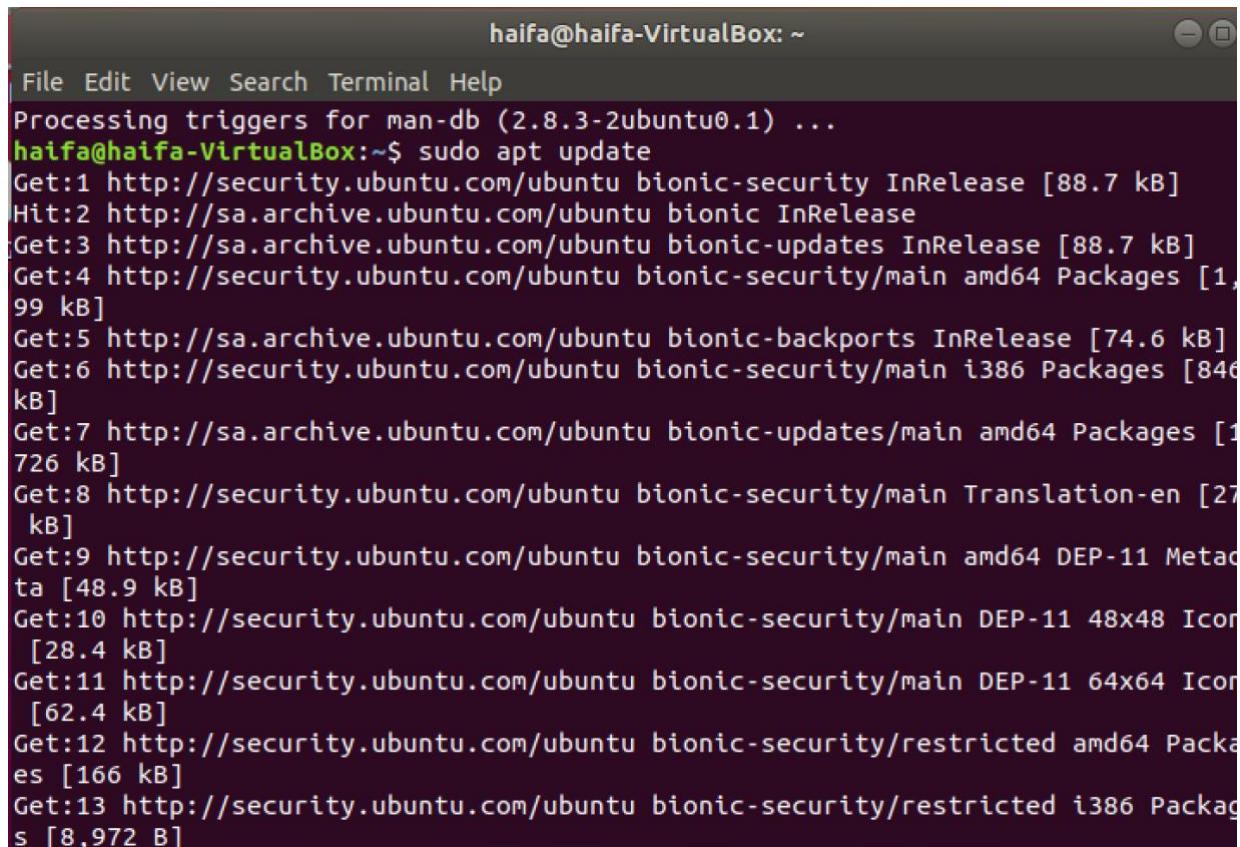
Installing mobSF

We have chosen to install mobSF that stand for (Mobile Security Framework) which is allows to run a pen-testing, malware analysis and security assessment framework capable of performing static and dynamic analysis.



```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
haifa@haifa-VirtualBox:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.4 LTS
Release:        18.04
Codename:       bionic
haifa@haifa-VirtualBox:~$ sudo apt-get install git
[sudo] password for haifa:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email git-gui gitk
  gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 225 not upgraded.
Need to get 4,741 kB of archives.
After this operation, 34.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://sa.archive.ubuntu.com/ubuntu bionic amd64 liberror-perl all 0
17025-1 [22.8 kB]
```

Figure 27



```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
haifa@haifa-VirtualBox:~$ sudo apt update
Get:1 http://security.ubuntu.com/ubuntu bionic-security InRelease [88.7 kB]
Hit:2 http://sa.archive.ubuntu.com/ubuntu bionic InRelease
Get:3 http://sa.archive.ubuntu.com/ubuntu bionic-updates InRelease [88.7 kB]
Get:4 http://security.ubuntu.com/ubuntu bionic-security/main amd64 Packages [1,
99 kB]
Get:5 http://sa.archive.ubuntu.com/ubuntu bionic-backports InRelease [74.6 kB]
Get:6 http://security.ubuntu.com/ubuntu bionic-security/main i386 Packages [846
kB]
Get:7 http://sa.archive.ubuntu.com/ubuntu bionic-updates/main amd64 Packages [1
726 kB]
Get:8 http://security.ubuntu.com/ubuntu bionic-security/main Translation-en [27
 kB]
Get:9 http://security.ubuntu.com/ubuntu bionic-security/main amd64 DEP-11 Metad
ata [48.9 kB]
Get:10 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 48x48 Icon
[28.4 kB]
Get:11 http://security.ubuntu.com/ubuntu bionic-security/main DEP-11 64x64 Icon
[62.4 kB]
Get:12 http://security.ubuntu.com/ubuntu bionic-security/restricted amd64 Packag
es [166 kB]
Get:13 http://security.ubuntu.com/ubuntu bionic-security/restricted i386 Packag
es [8,972 B]
```

Figure 28

```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
Reading package lists... Done
Building dependency tree
Reading state information... Done
321 packages can be upgraded. Run 'apt list --upgradable' to see them.
haifa@haifa-VirtualBox:~$ sudo apt install python3-pip
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  build-essential cpp-7 dh-python dpkg-dev fakeroot g++ g++-7 gcc gcc-7
  gcc-7-base gcc-8-base libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan4 libatomic1 libc-dev-bin libc6 libc6-dbg
  libc6-dev libcc1-0 libcilkrt5 libexpat1-dev libfakeroot libgcc-7-dev
  libgcc1 libgomp1 libitm1 liblsan0 libmpx2 libpython3-dev libpython3.6
  libpython3.6-dev libpython3.6-minimal libpython3.6-stdlib libquadmath0
  libstdc++-7-dev libstdc++6 libtsan0 libubsan0 linux-libc-dev make
  manpages-dev python-pip-whl python3-dev python3-distutils python3-lib2to3
  python3-setuptools python3-wheel python3.6 python3.6-dev python3.6-minimal
Suggested packages:
  gcc-7-locales debian-keyring g++-multilib g++-7-multilib gcc-7-doc
  libstdc++6-7-dbg gcc-multilib autoconf automake libtool flex bison gcc-doc
  gcc-7-multilib libgcc1-dbg libgomp1-dbg libitm1-dbg libatomic1-dbg
  libasan4-dbg liblsan0-dbg libtsan0-dbg libubsan0-dbg libcilkrt5-dbg
  libmpx2-dbg libquadmath0-dbg glibc-doc libstdc++-7-doc make-doc
```

Figure 29

```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
sudo apt install python
sudo apt install python-minimal

You also have python3 installed, you can run 'python3' instead.

haifa@haifa-VirtualBox:~$ python3 --V
Unknown option: --
usage: python3 [option] ... [-c cmd | -m mod | file | -] [arg] ...
Try `python -h` for more information.
haifa@haifa-VirtualBox:~$ python3 --version
Python 3.6.9
haifa@haifa-VirtualBox:~$ sudo apt install default-jre
Reading package lists... Done
Building dependency tree
Reading state information... Done
default-jre is already the newest version (2:1.11-68ubuntu1~18.04.1).
default-jre set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 307 not upgraded.
haifa@haifa-VirtualBox:~$ java -version
openjdk version "11.0.7" 2020-04-14
OpenJDK Runtime Environment (build 11.0.7+10-post-Ubuntu-2ubuntu218.04)
OpenJDK 64-Bit Server VM (build 11.0.7+10-post-Ubuntu-2ubuntu218.04, mixed mode
sharing)
haifa@haifa-VirtualBox:~$
```

Figure 30

```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
haifa@haifa-VirtualBox:~$ sudo apt install python3-venv python3-pip python3-dev
  build-essential \
> libffi-dev libssl-dev libxml2-dev libxslt1-dev libjpeg8-dev zlib1g-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.4ubuntu1).
build-essential set to manually installed.
python3-dev is already the newest version (3.6.7-1~18.04).
python3-dev set to manually installed.
python3-pip is already the newest version (9.0.1-2.3~ubuntu1.18.04.4).
The following additional packages will be installed:
  gir1.2-harfbuzz-0.0 icu-devtools libglib2.0-0 libglib2.0-bin libglib2.0-dev
  libglib2.0-dev-bin libgraphite2-dev libharfbuzz-dev libharfbuzz-gobject0
  libicu-dev libicu-le-hb-dev libicu-le-hb0 libicu60 libiculx60 libjpeg-turbo8
  libjpeg-turbo8-dev libpcre16-3 libpcre3-dev libpcre32-3 libpcrecpp0v5
  libssl1.1 libxml2 pkg-config python3.6-venv
Suggested packages:
  libglib2.0-doc libgraphite2-utils icu-doc libssl-doc
The following NEW packages will be installed:
  gir1.2-harfbuzz-0.0 icu-devtools libffi-dev libglib2.0-dev
  libglib2.0-dev-bin libgraphite2-dev libharfbuzz-dev libharfbuzz-gobject0
  libicu-dev libicu-le-hb-dev libicu-le-hb0 libiculx60 libjpeg-turbo8-dev
  libjpeg8-dev libpcre16-3 libpcre3-dev libpcre32-3 libpcrecpp0v5 libssl-dev
```

Figure 31

```
haifa@haifa-VirtualBox: ~
File Edit View Search Terminal Help
Setting up python3-venv (3.6.7-1~18.04) ...
Setting up libpcre3-dev:amd64 (2:8.39-9) ...
Setting up libglib2.0-dev-bin (2.56.4-0ubuntu0.18.04.6) ...
Setting up libssl-dev:amd64 (1.1.1-1ubuntu2.1~18.04.6) ...
Setting up libglib2.0-dev:amd64 (2.56.4-0ubuntu0.18.04.6) ...
Setting up libicu-le-hb-dev:amd64 (1.0.3+git161113-4) ...
Setting up libicu-dev (60.2-3ubuntu3.1) ...
Setting up libxml2-dev:amd64 (2.9.4+dfsg1-6.1ubuntu1.3) ...
Setting up libharfbuzz-dev:amd64 (1.7.2-1ubuntu1) ...
Setting up libxslt1-dev:amd64 (1.1.29-5ubuntu0.2) ...
Processing triggers for install-info (6.5.0.dfsg.1-2) ...
Processing triggers for libc-bin (2.27-3ubuntu1) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
haifa@haifa-VirtualBox:~$ git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git
Cloning into 'Mobile-Security-Framework-MobSF'...
remote: Enumerating objects: 124, done.
remote: Counting objects: 100% (124/124), done.
remote: Compressing objects: 100% (82/82), done.
remote: Total 16617 (delta 56), reused 75 (delta 42), pack-reused 16493
Receiving objects: 100% (16617/16617), 940.36 MiB | 1.54 MiB/s, done.
Resolving deltas: 100% (7875/7875), done.
Checking out files: 100% (378/378), done.
haifa@haifa-VirtualBox:~$
```

Figure 32

```
haifa@haifa-VirtualBox: ~/Mobile-Security-Framework-MobSF
File Edit View Search Terminal Help
haifa@haifa-VirtualBox:~$ cd Mobile-Security-Framework-MobSF
haifa@haifa-VirtualBox:~/Mobile-Security-Framework-MobSF$ ./setup.sh
[INSTALL] Found Python3
pip 9.0.1 from /usr/lib/python3/dist-packages (python 3.6)
[INSTALL] Found pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/cb/28/91f26bd088ce8e22169
32100d4260614fc3da435025ff389ef1d396a433/pip-20.2.4-py2.py3-none-any.whl (1.5MB
  100% |██████████| 1.5MB 741kB/s
Installing collected packages: pip
Successfully installed pip-20.2.4
[INSTALL] Using python virtualenv
[INSTALL] Activating virtualenv
Collecting pip
  Downloading https://files.pythonhosted.org/packages/cb/28/91f26bd088ce8e22169
32100d4260614fc3da435025ff389ef1d396a433/pip-20.2.4-py2.py3-none-any.whl (1.5MB
  100% |██████████| 1.5MB 382kB/s
Collecting wheel
  Downloading https://files.pythonhosted.org/packages/a7/00/3df031b3ecd5444d572
41321537080b40c1c25e1caa3d86cdd12e5e919c/wheel-0.35.1-py2.py3-none-any.whl
Installing collected packages: pip, wheel
  Found existing installation: pip 9.0.1
    Uninstalling pip-9.0.1:
      Successfully uninstalled pip-9.0.1
```

Figure 33

Figure 34

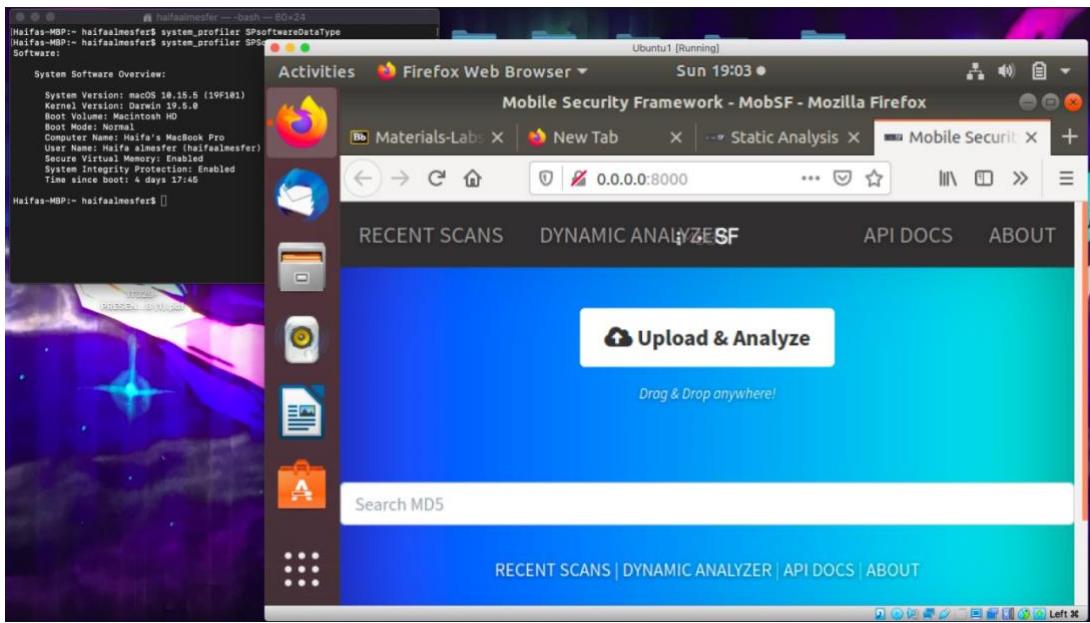


Figure 35 MobSF run

References :

<https://www.geeksforgeeks.org/differences-between-black-box-testing-vs-white-box-testing/>

<https://searchsecurity.techtarget.com/definition/penetration-testing>