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

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 PENETRAITION 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.

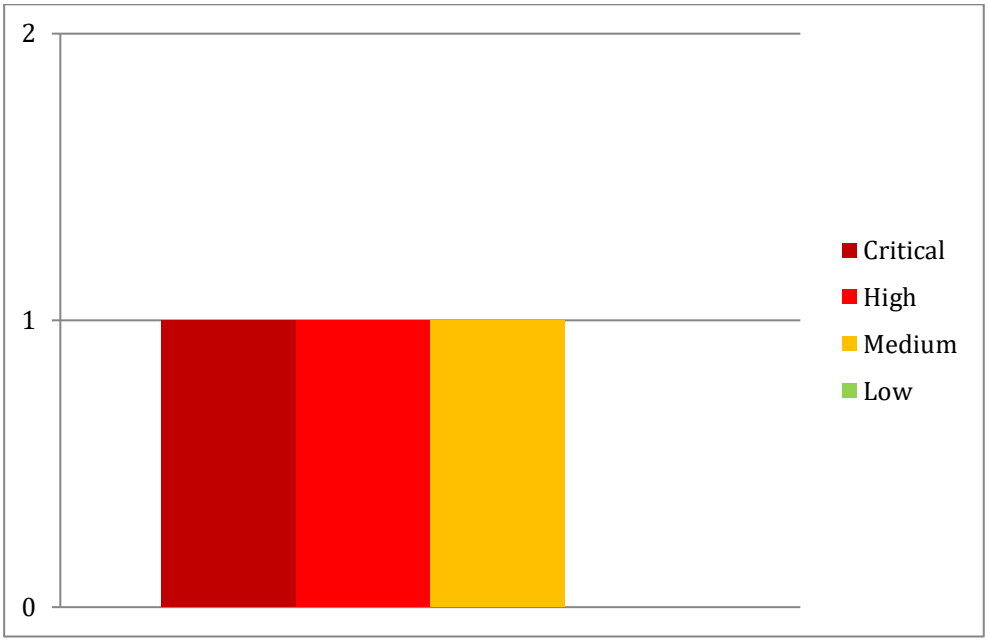
Mobile Application Penetration Testing

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	**1.7.1.1 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). Unprotected logs	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*

<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 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>

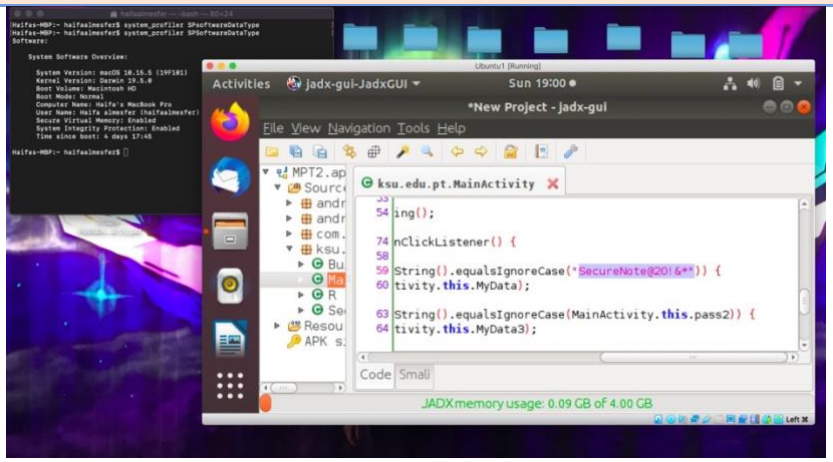
3.2 *Limitations*

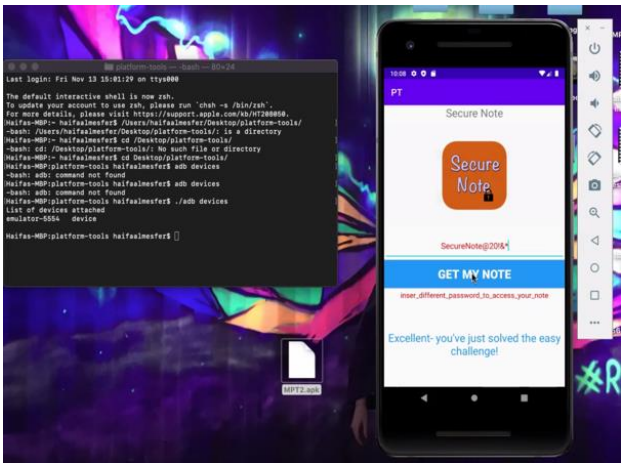
We focused on installing specific tools such as android studio , Genymotion ,MobSF ,jdax , and adb and make sure that all of them work properly to help us run the penetration ,and we were limited by our scope , time and the required in this project which is a partial penetration testing and come up with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

<table>
<tr><td>Severity Level</td><td colspan="4">Critical</td></tr>
<tr><td rowspan="2">Technical Impact</td><td rowspan="2">Critical</td><td colspan="2">Likelihood</td><td>HIGH</td></tr>
<tr><td>Popularity</td><td>HIGH</td><td>Simplicity</td><td>MEDIUM</td></tr>
<tr><td>Observation</td><td colspan="4">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.</td></tr>
<tr><td>Impact</td><td colspan="4">This well make the application vulnerable to anyone who can access the source code even if they have a little knowledge of technology.</td></tr>
<tr><td colspan="5">Proof of Concept</td></tr>
<tr><td colspan="5"> 


</td></tr>
<tr><td>Recommendation</td><td colspan="4">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.</td></tr>
<tr><td>OWASP Reference</td><td colspan="4">OWASP Reference</td></tr>
<tr><td>Reference</td><td colspan="4">https://wiki.owasp.org/index.php/Use_of_hard-coded_password</td></tr>
</table>

3.3.1.1 Limitation

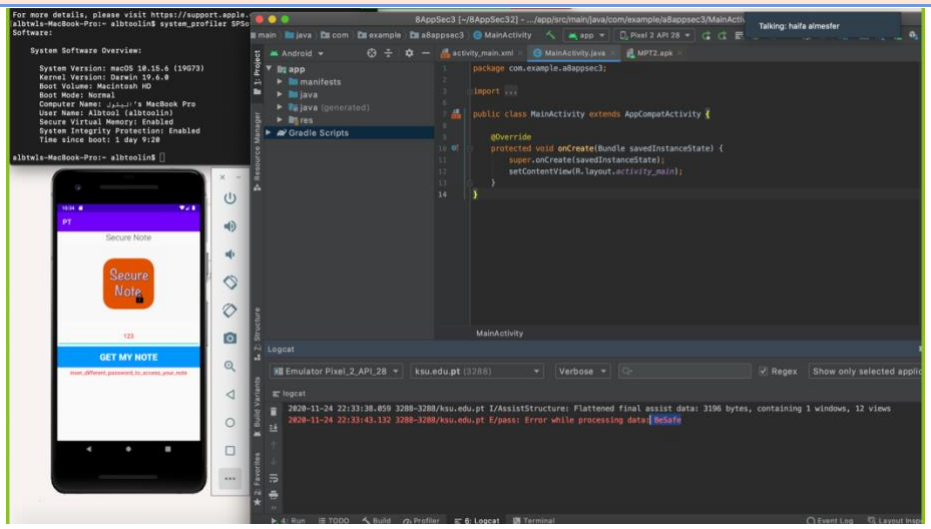
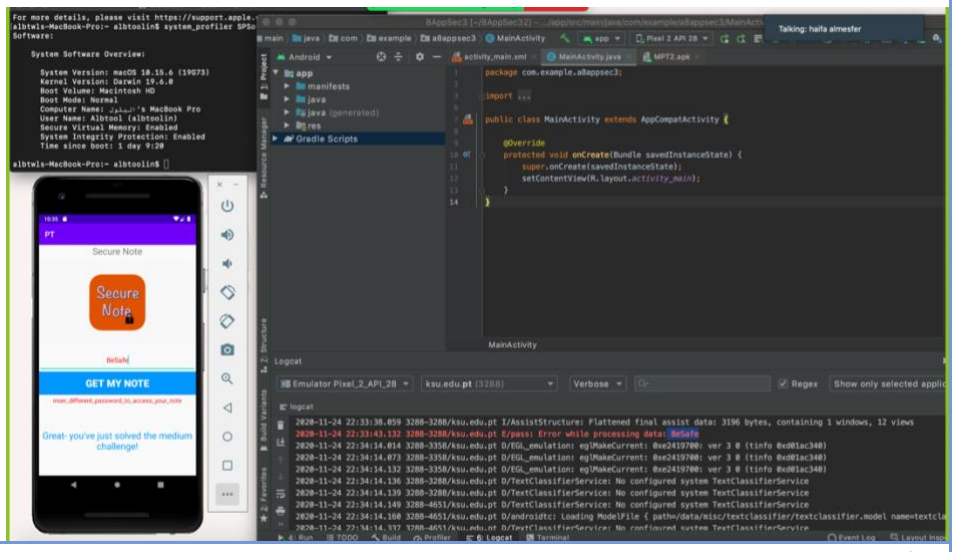
We did face a problem in this machine (Haifa's machine):

- 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.

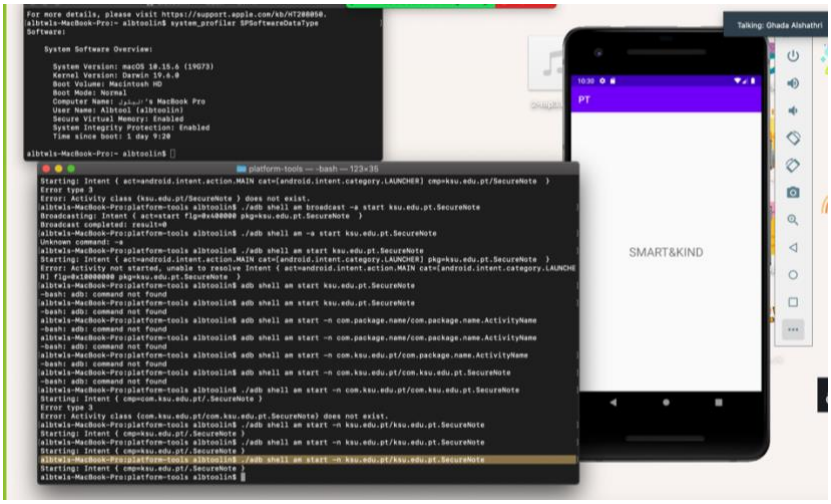
So, the next 2 vulnerabilities have been exploited by the new machine (albatool's machine).

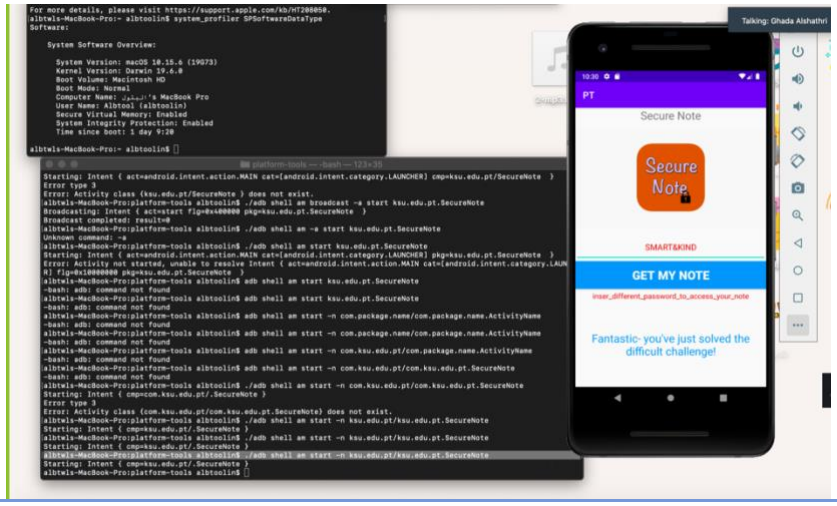
3.3.2 Unprotected logs

<table>
<tr><td>Severity Level</td><td colspan="4">High</td></tr>
<tr><td rowspan="2">Technical Impact</td><td rowspan="2">Critical</td><td colspan="2">Likelihood</td><td>MEDIUM</td></tr>
<tr><td>Popularity</td><td>HIGH</td><td>Simplicity</td><td>HIGH</td></tr>
<tr><td>Observation</td><td colspan="4">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.</td></tr>
<tr><td>Impact</td><td colspan="4">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.</td></tr>
<tr><td colspan="5">Proof of Concept</td></tr>
<tr><td colspan="5"> 
 </td></tr>
<tr><td>Recommendation</td><td colspan="4">It is recommended to use non reveling 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.</td></tr>
<tr><td>OWASP Reference</td><td colspan="4">OWASP Reference</td></tr>
<tr><td>Reference</td><td colspan="4">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</td></tr>
</table>

3.3.3 Exported Activity

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


</td></tr>
<tr><td>Recommendation</td><td colspan="4">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”.</td></tr>
<tr><td>OWASP Reference</td><td colspan="4">OWASP Reference</td></tr>
<tr><td>Reference</td><td colspan="4">https://oldbam.github.io/android/security/android-vulnerabilities-insecurebank-activities
https://securitygrind.com/exploiting-android-components-abusing-activities/</td></tr>
</table>

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.

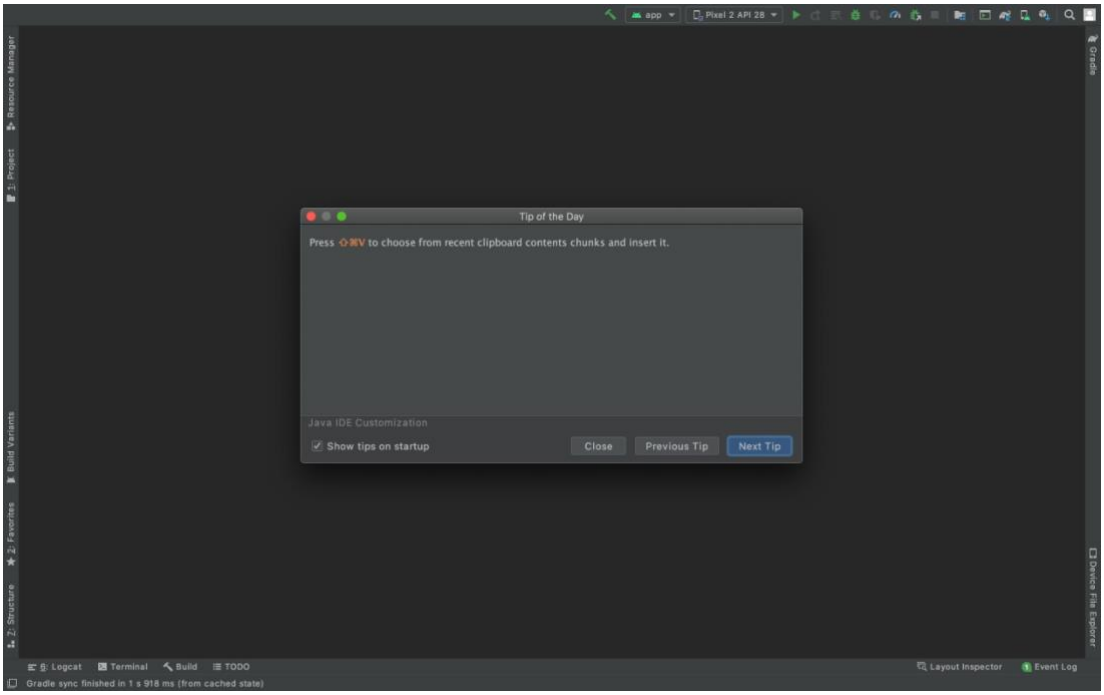


Figure 2 first page when we open android studio

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.

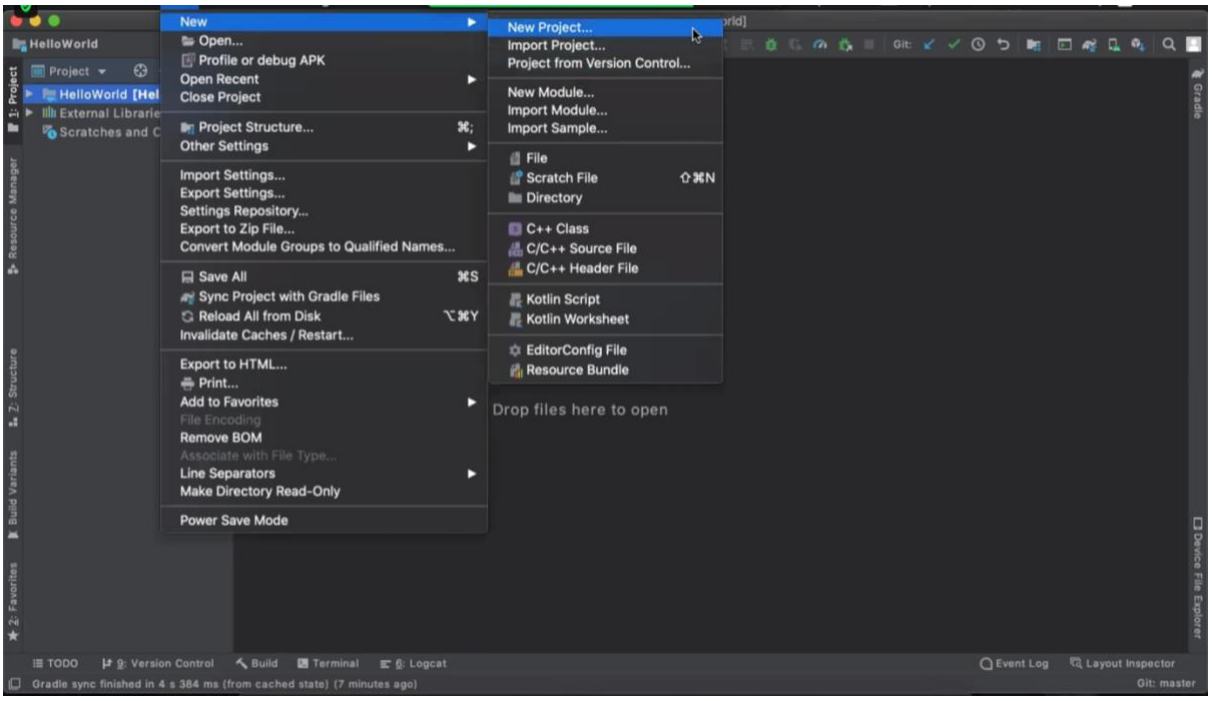


Figure 3 [1] crate a new project

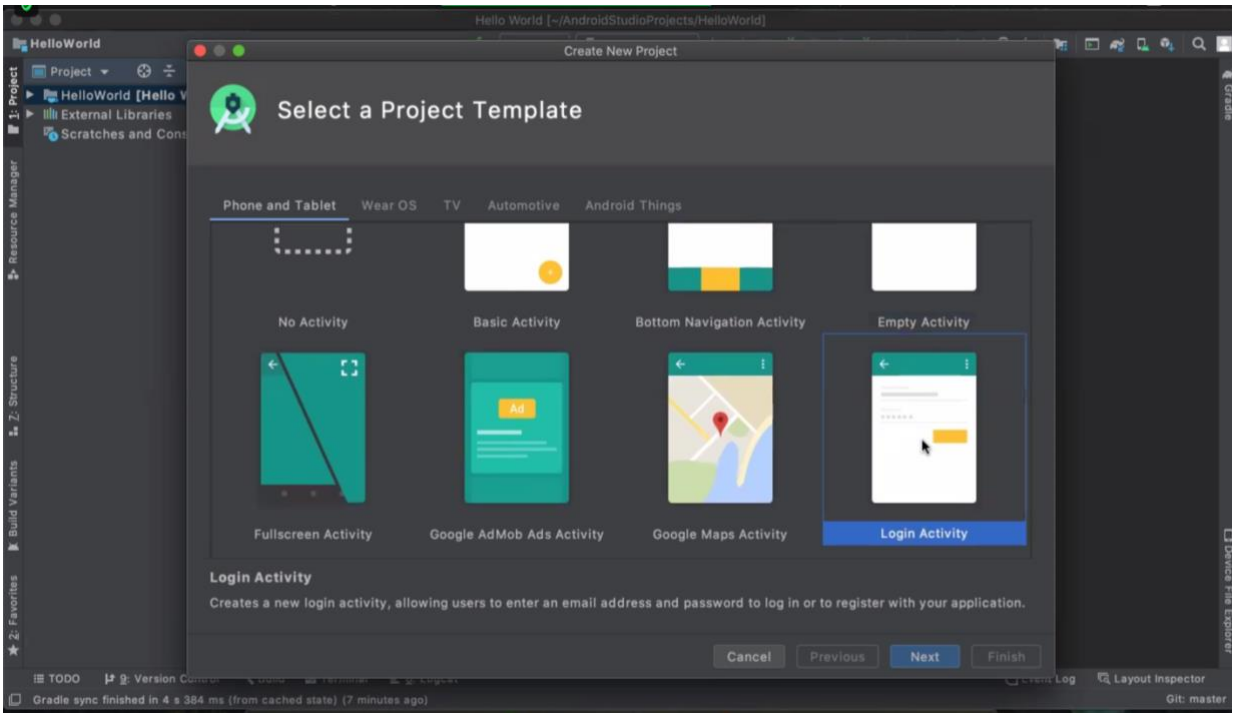


Figure 4 [2] select the project templet

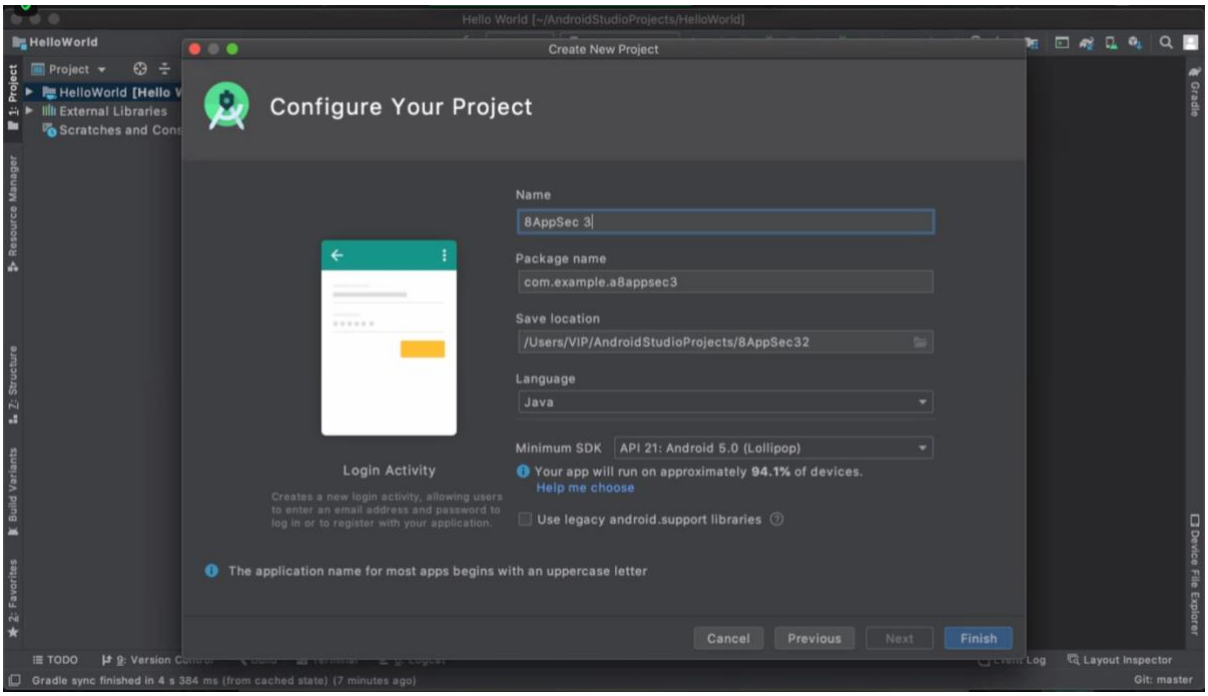


Figure 5 [3]

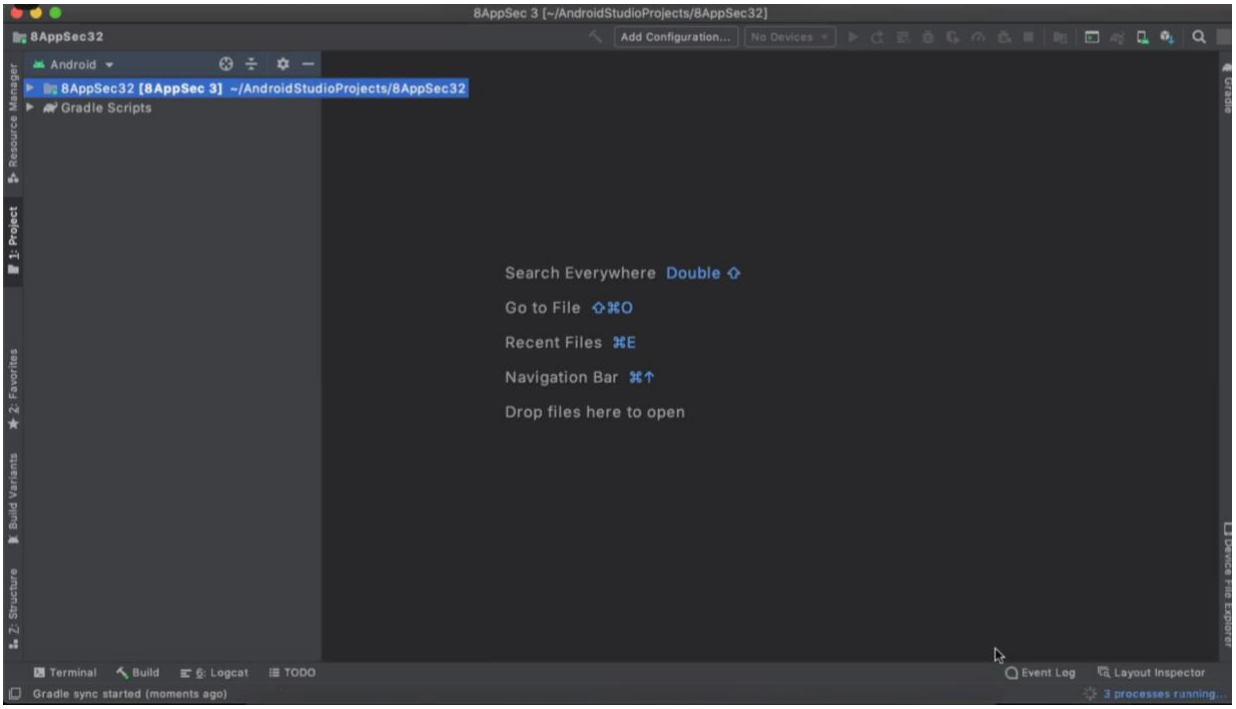


Figure 6 [4]

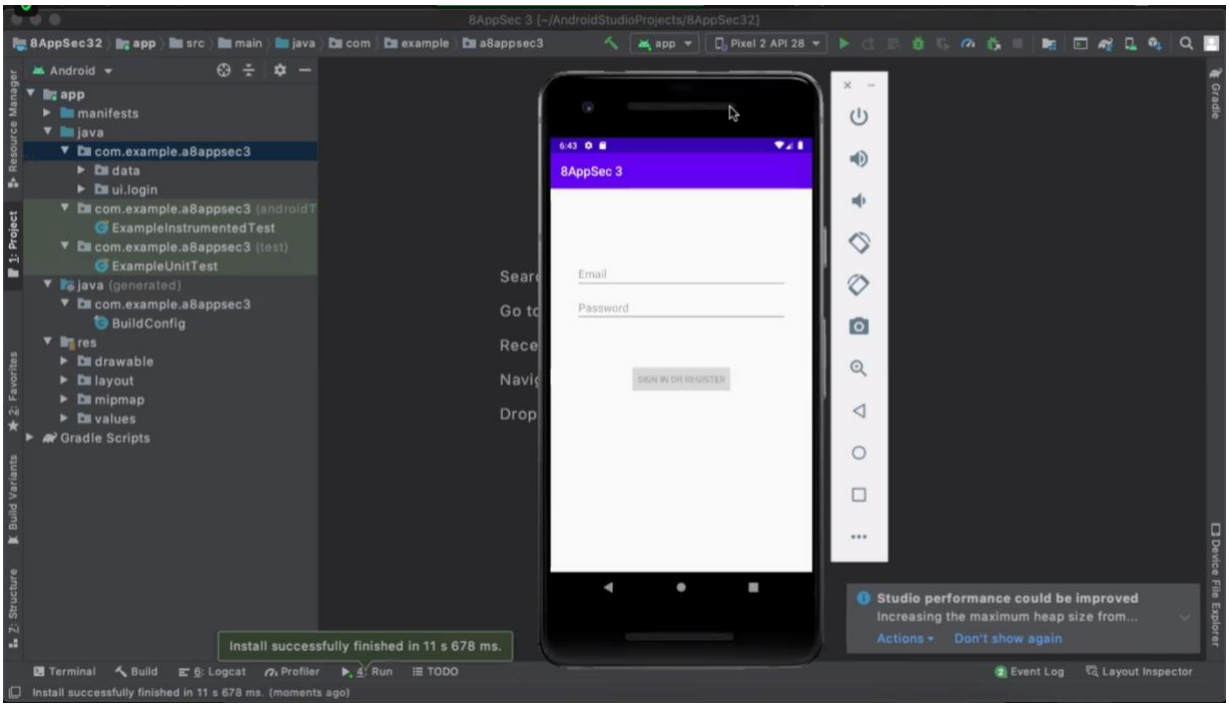


Figure 7 [5] running the application

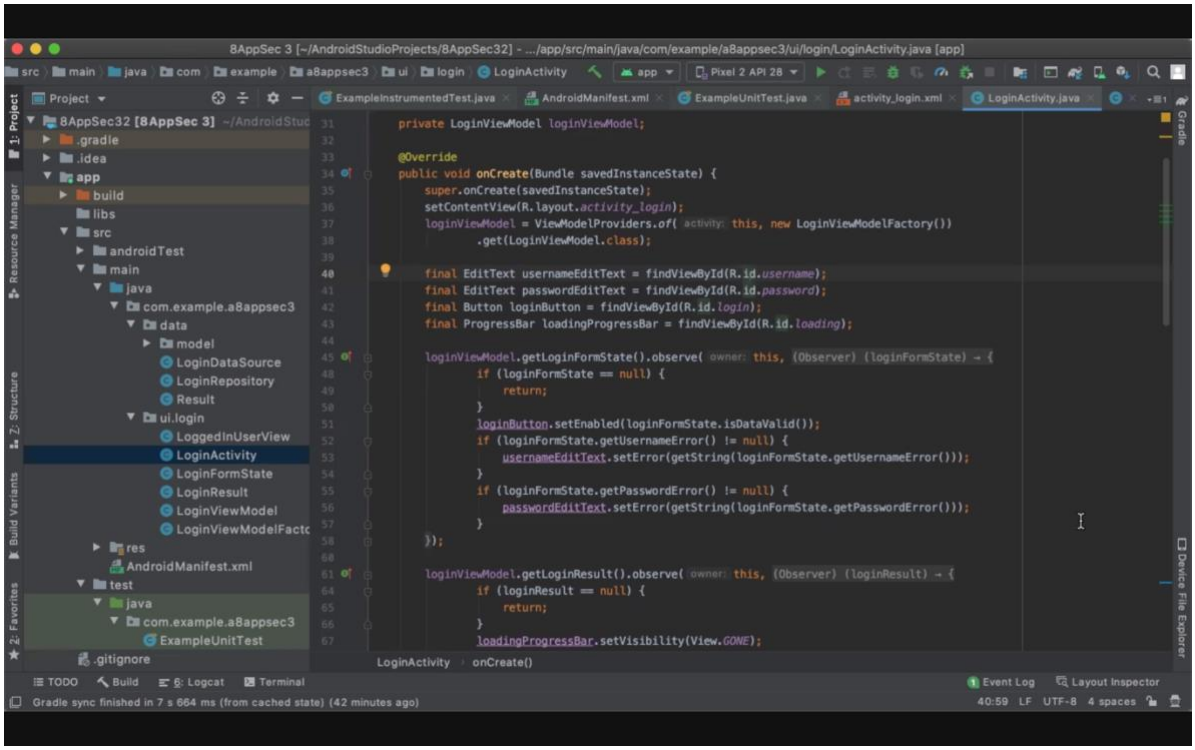


Figure 8 [6] the source code

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.

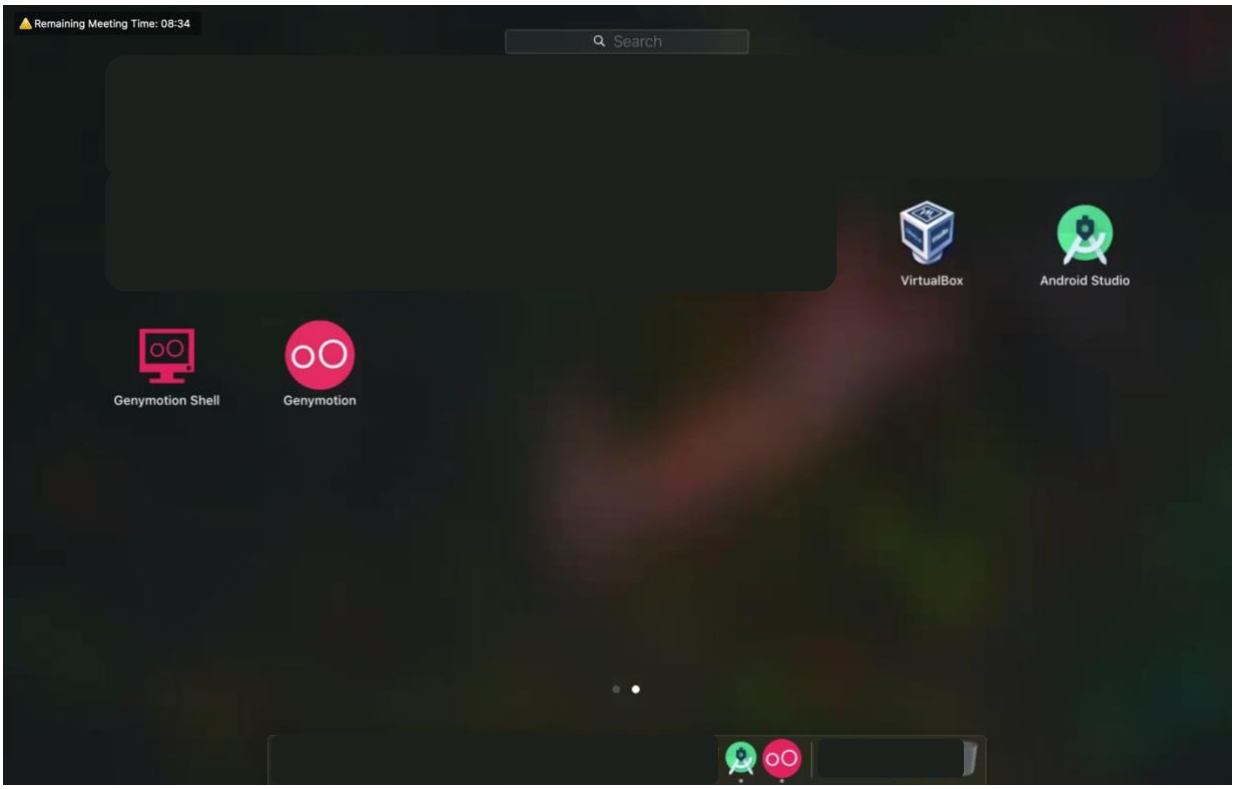


Figure 9 [1]opening the Genymotion to activate it

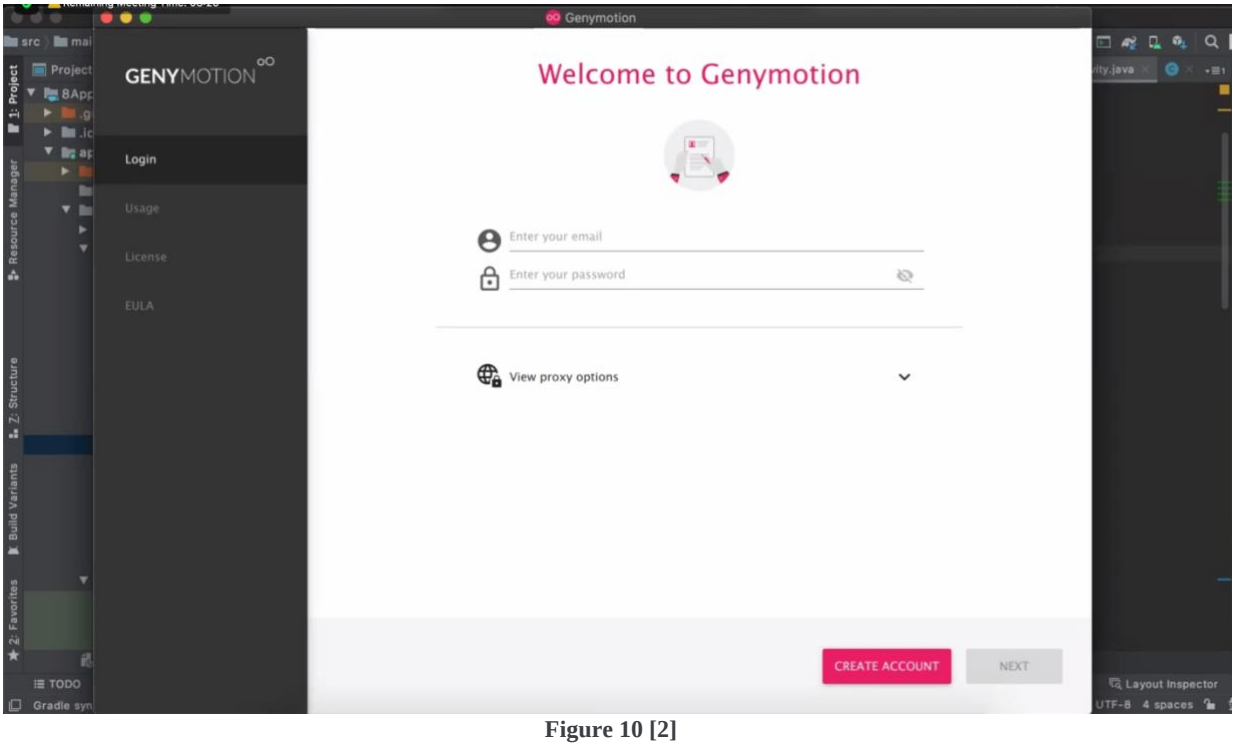


Figure 10 [2]

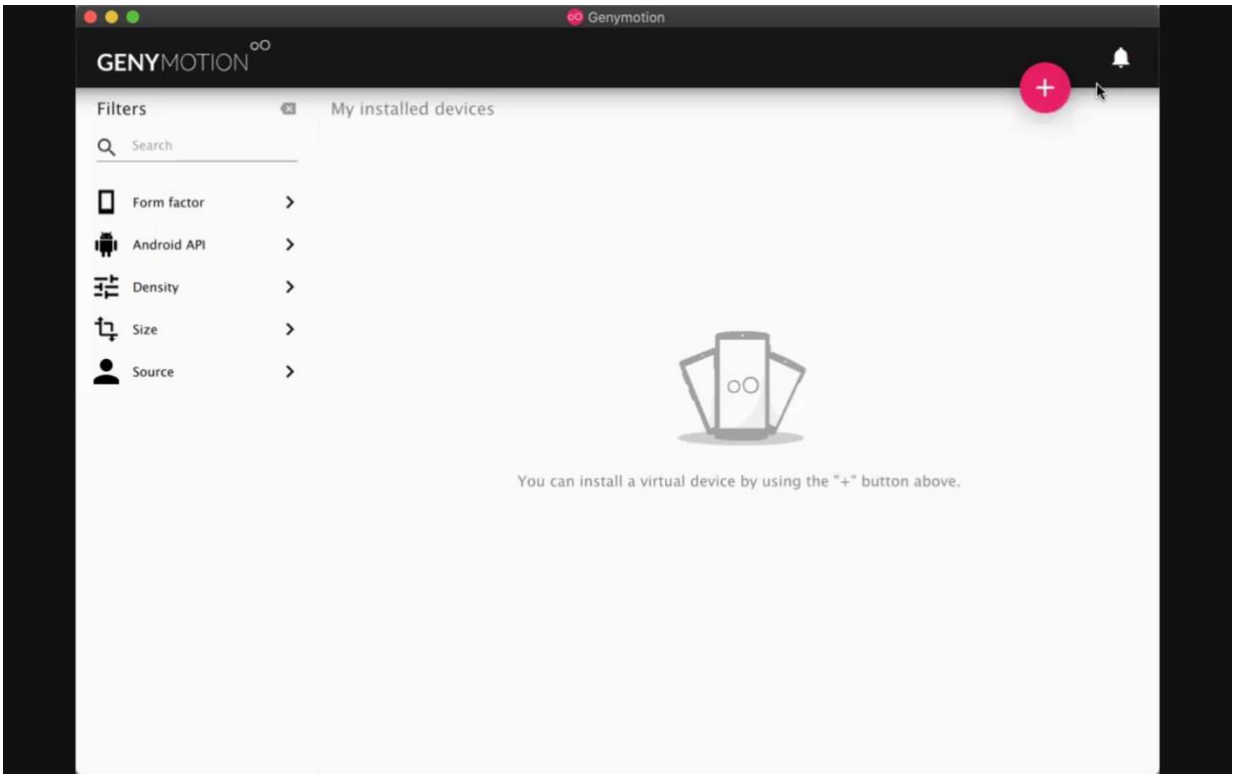


Figure 11 [3]

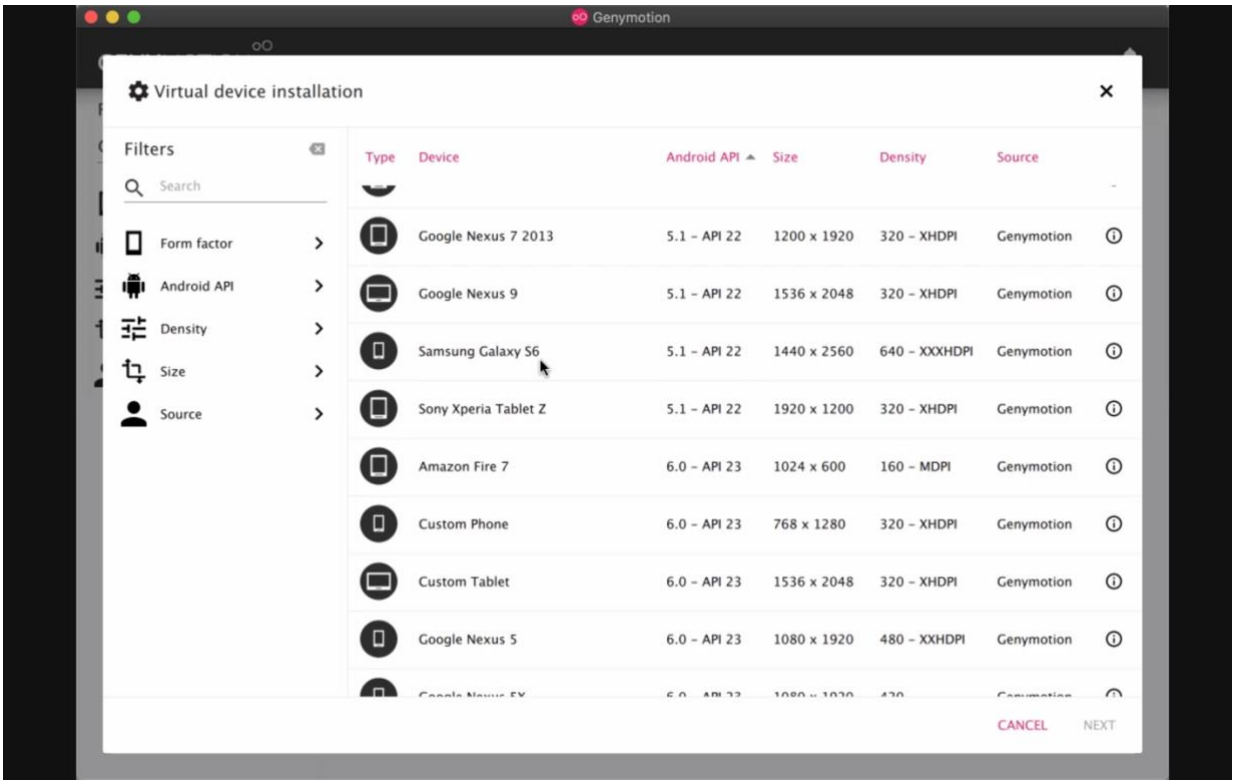


Figure 12 [4]

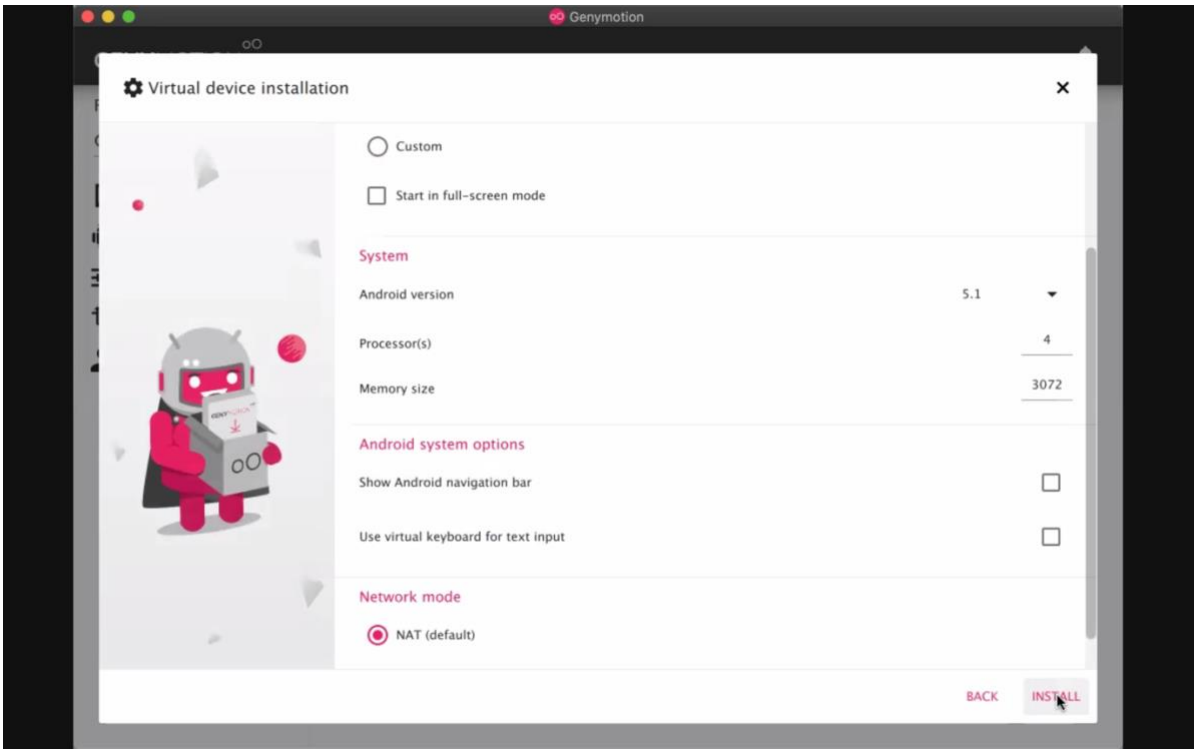


Figure 13 [5]

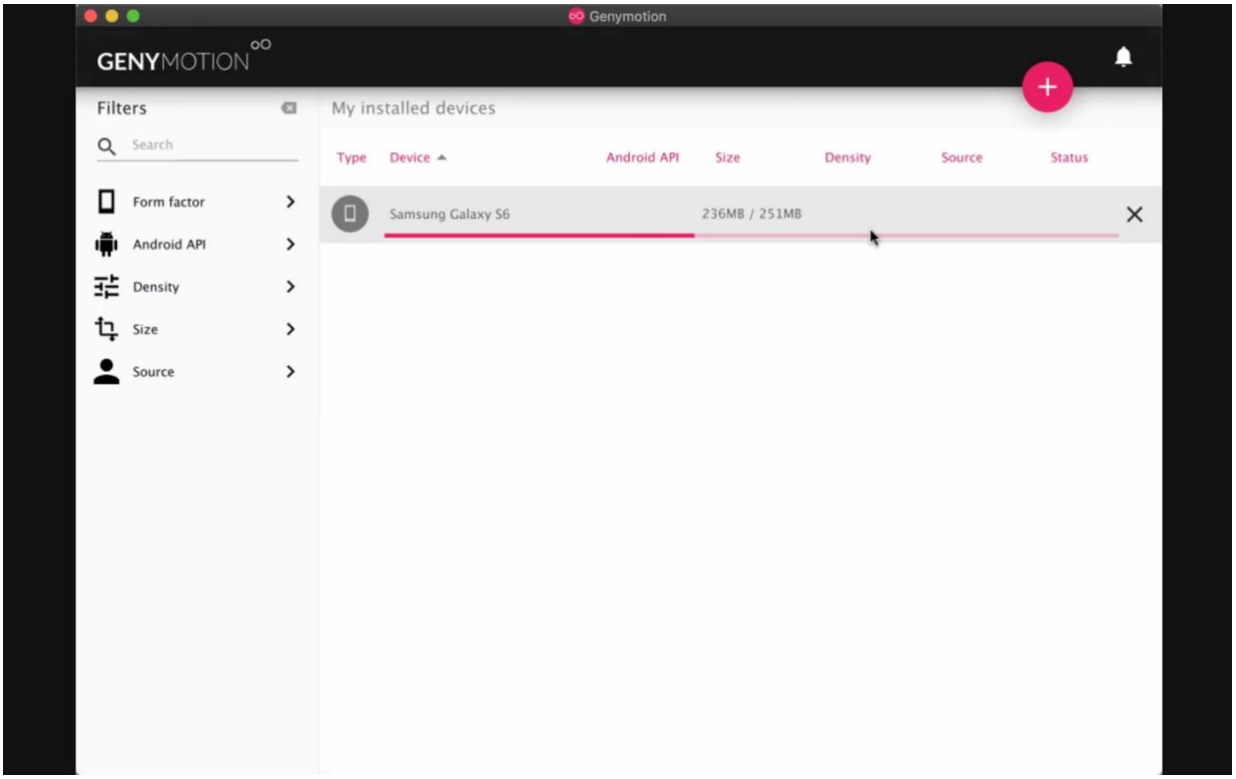


Figure 14 [6]

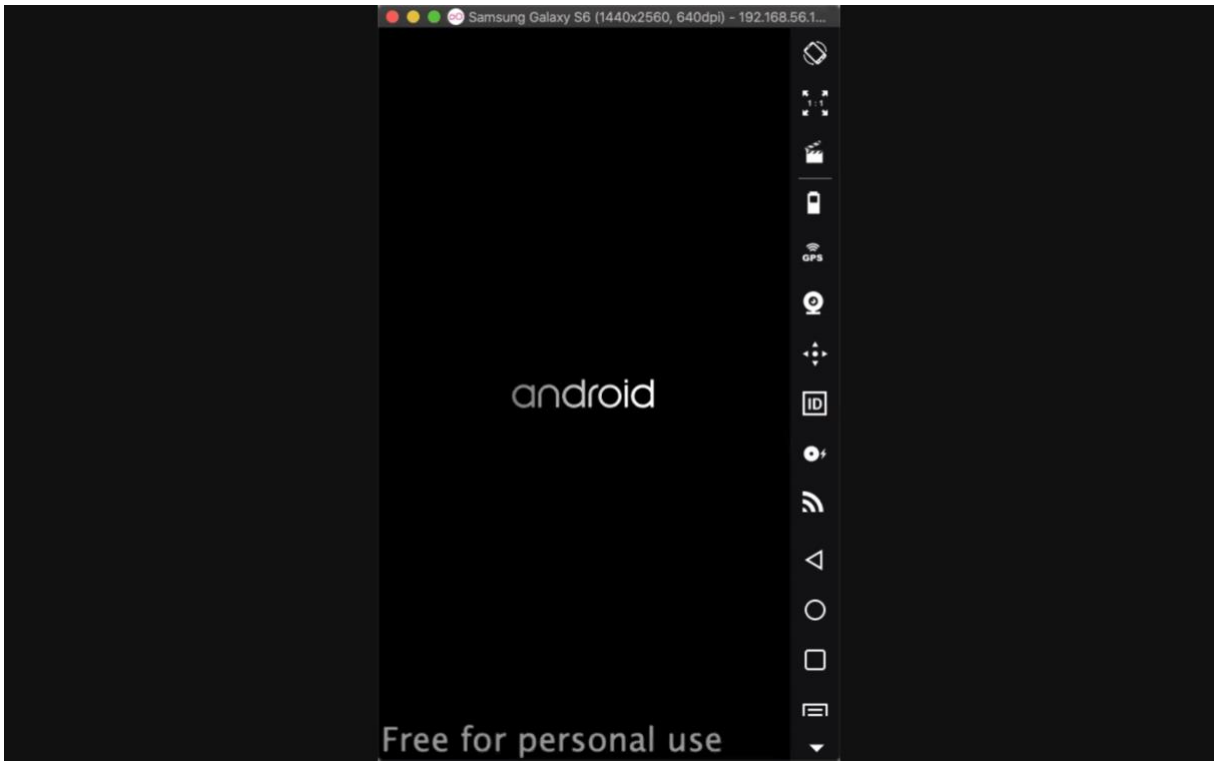


Figure 15 finally the Genymotion is activated

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.

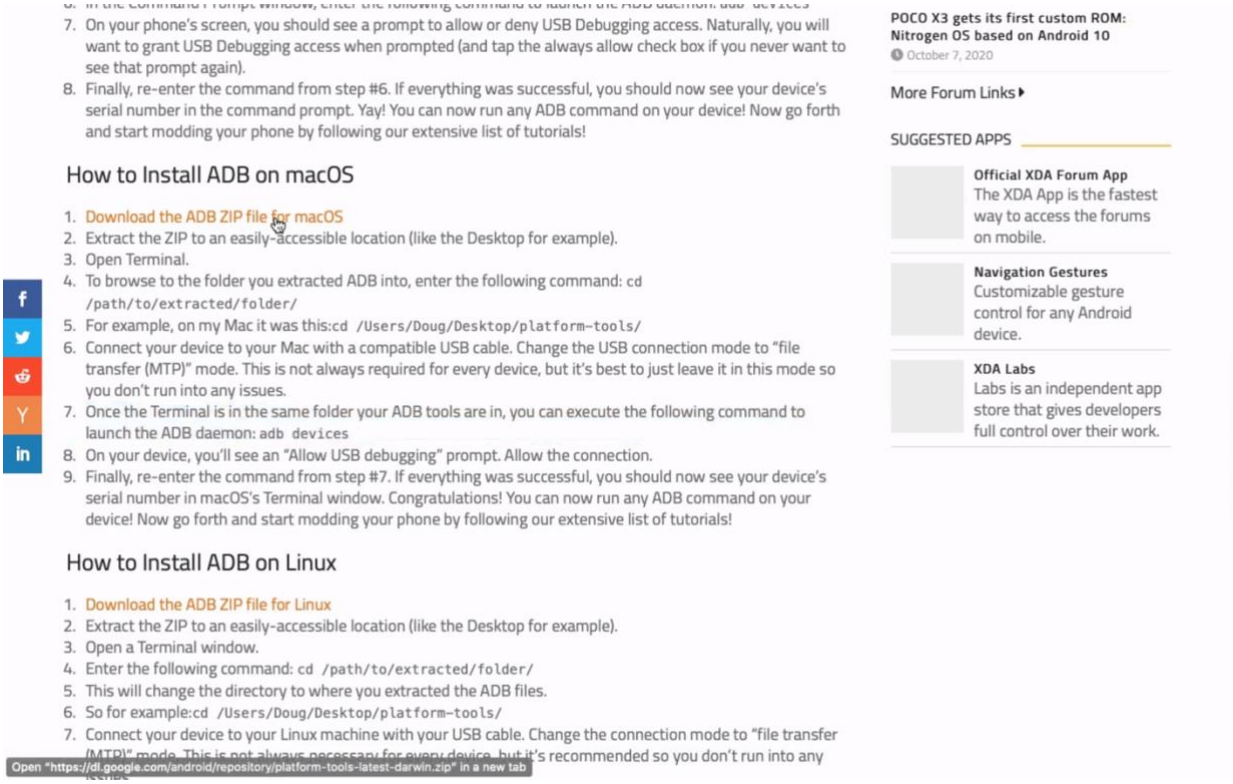


Figure 16

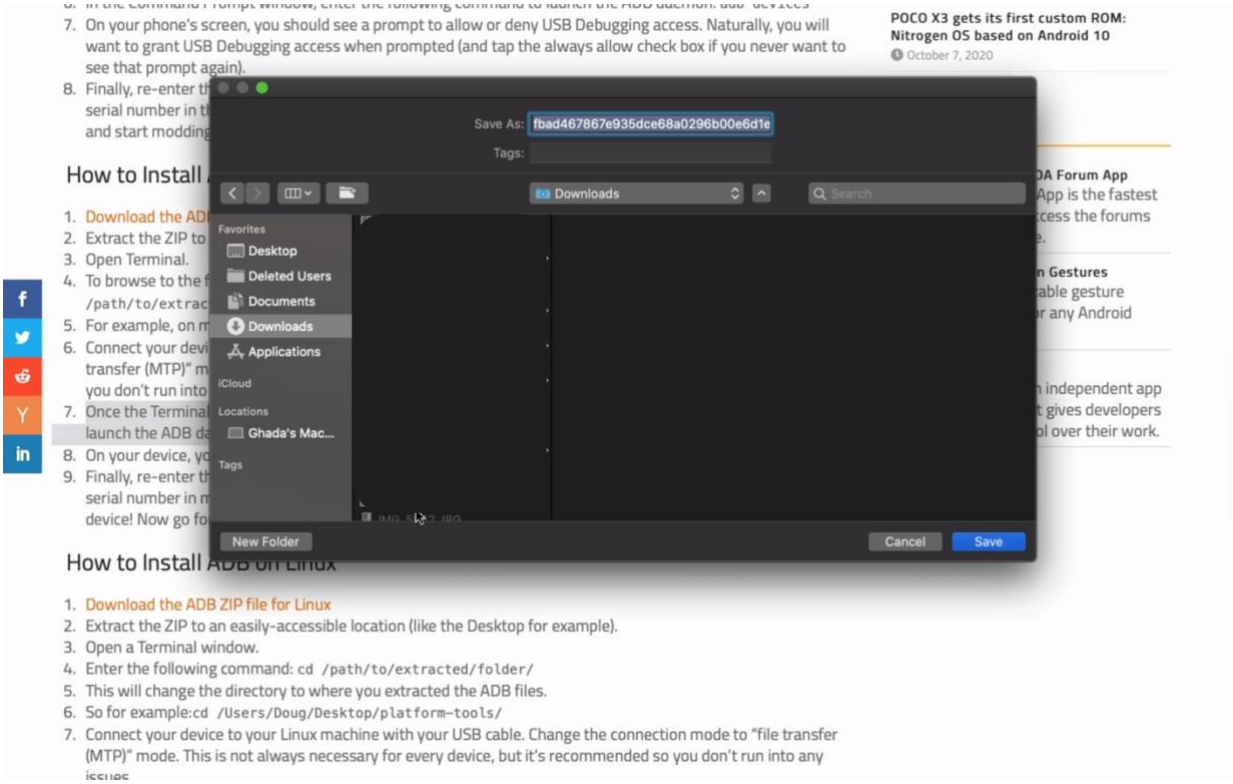


Figure 17

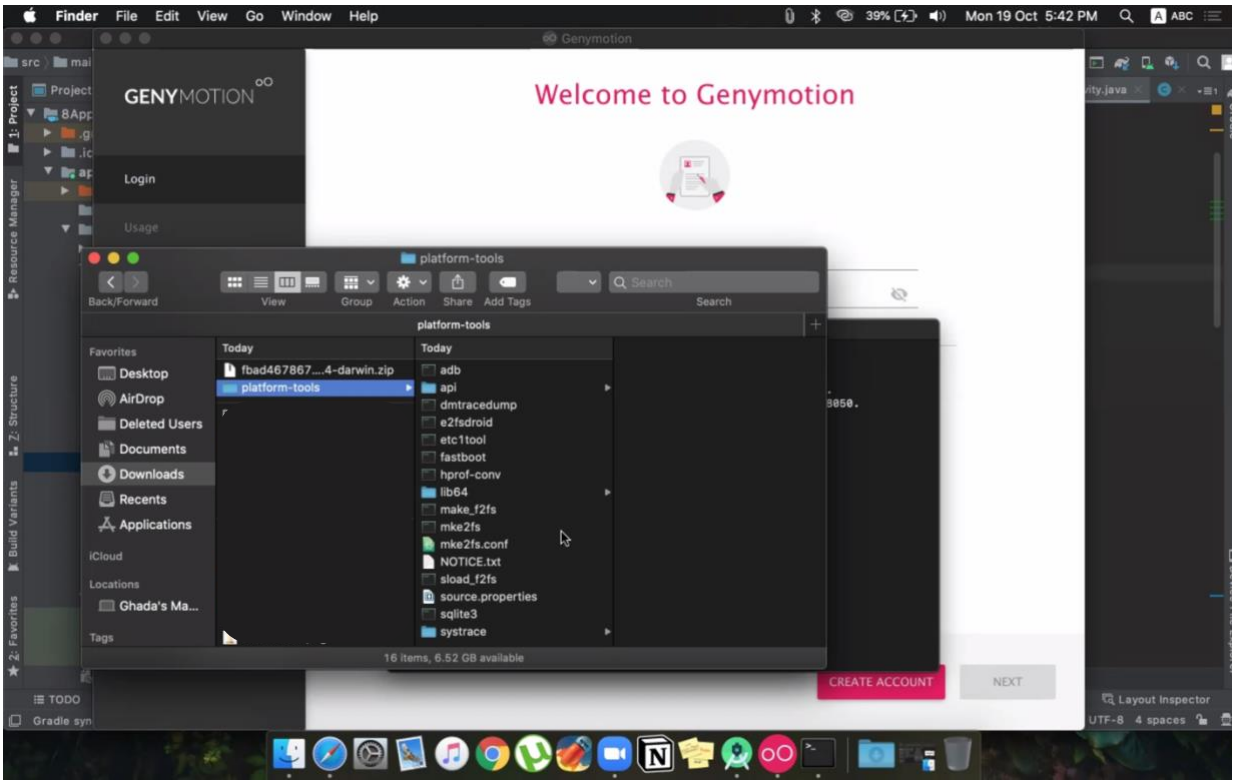


Figure 18

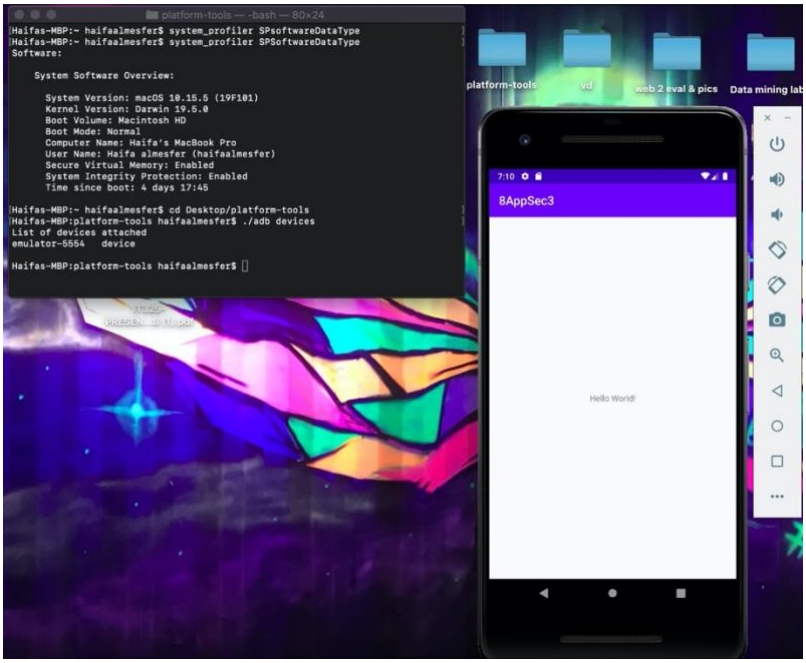


Figure 18 adb run

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 it 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.

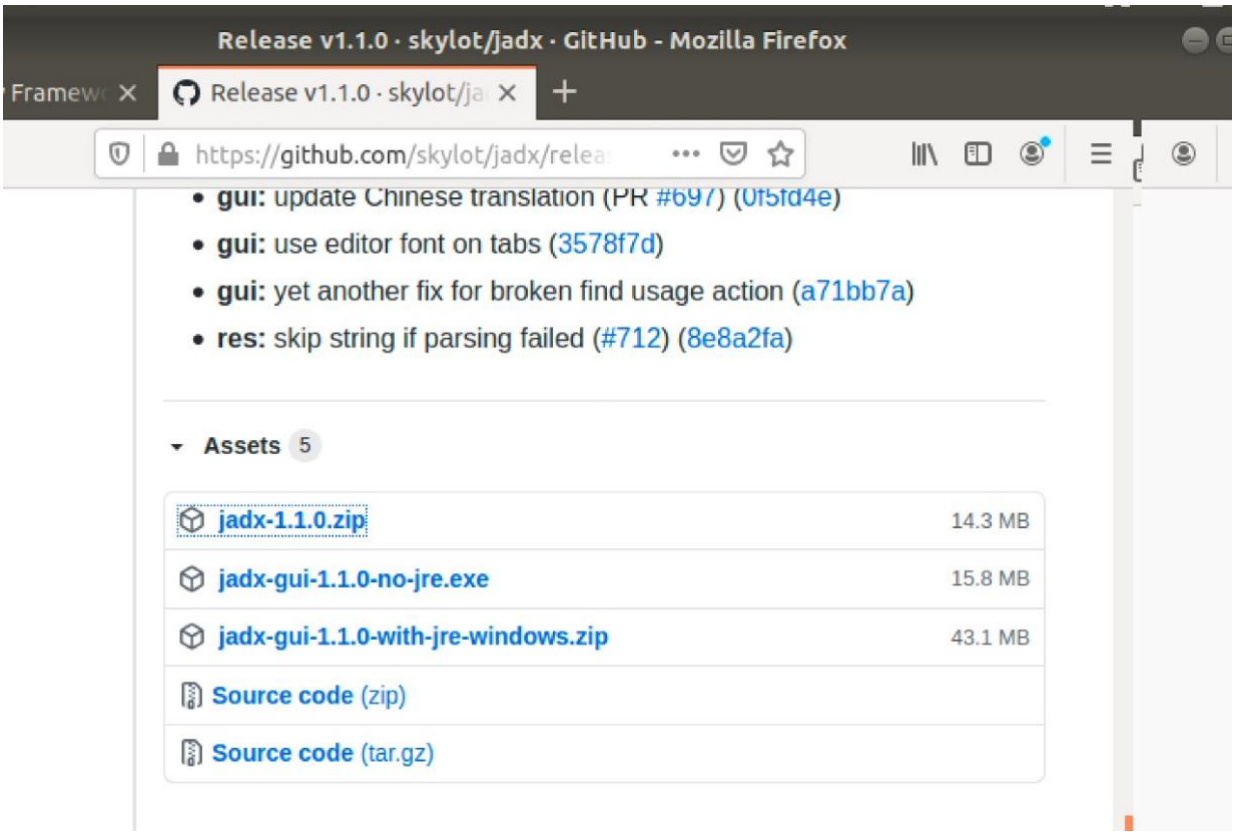


Figure 19 the source we used

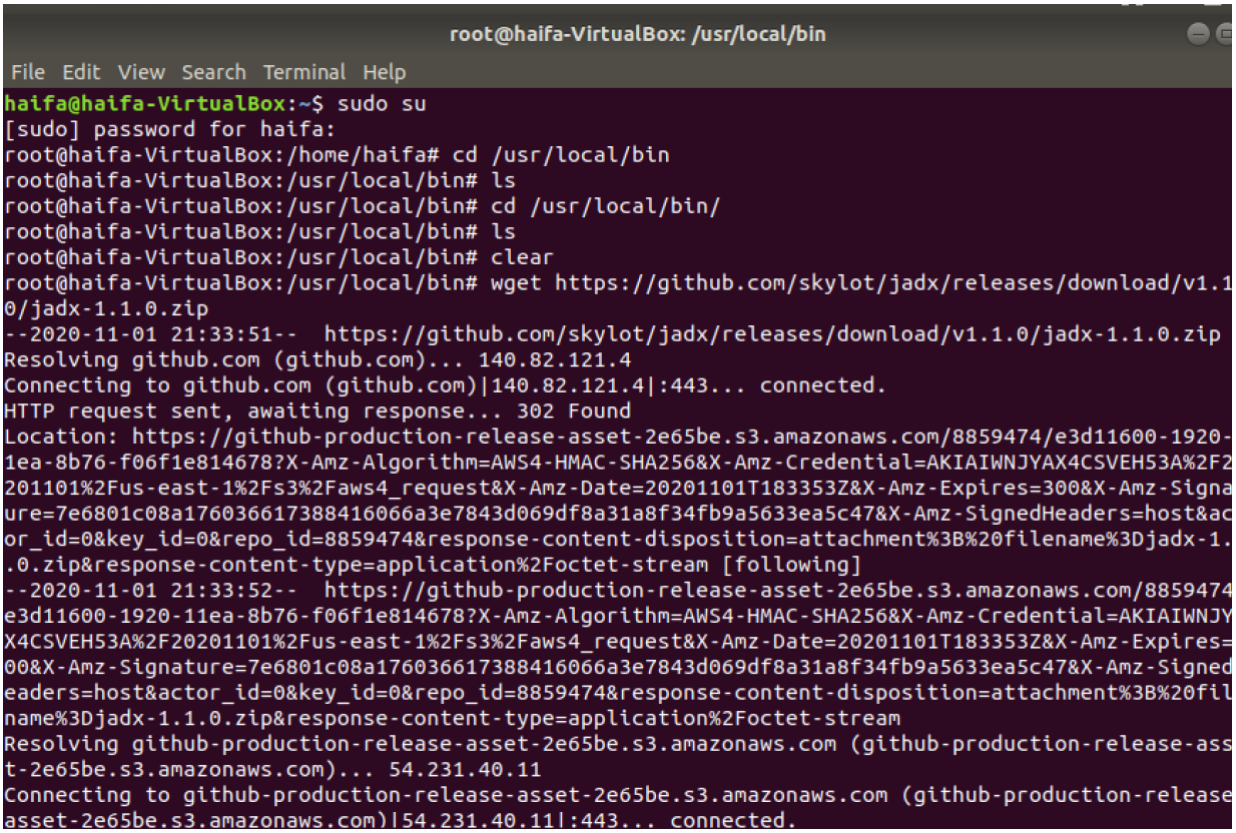


Figure 20

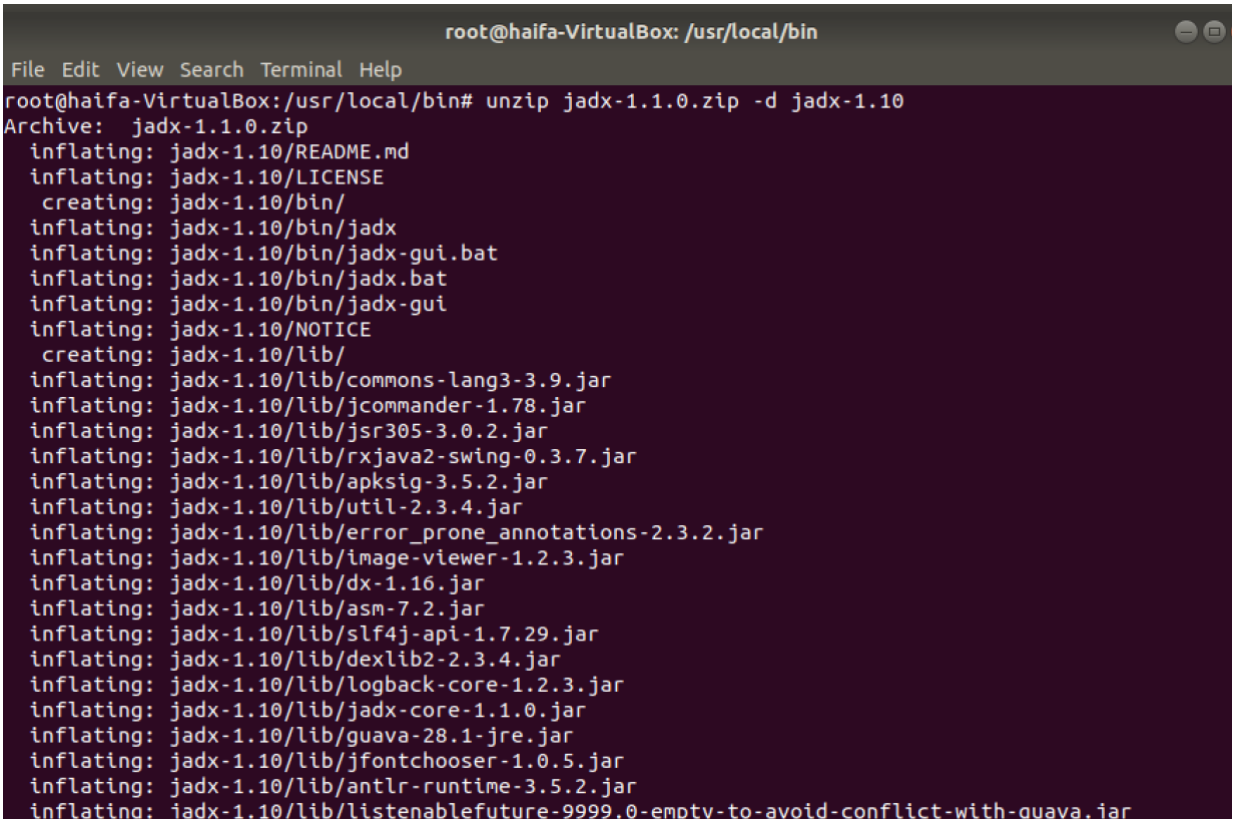


Figure 21

```
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

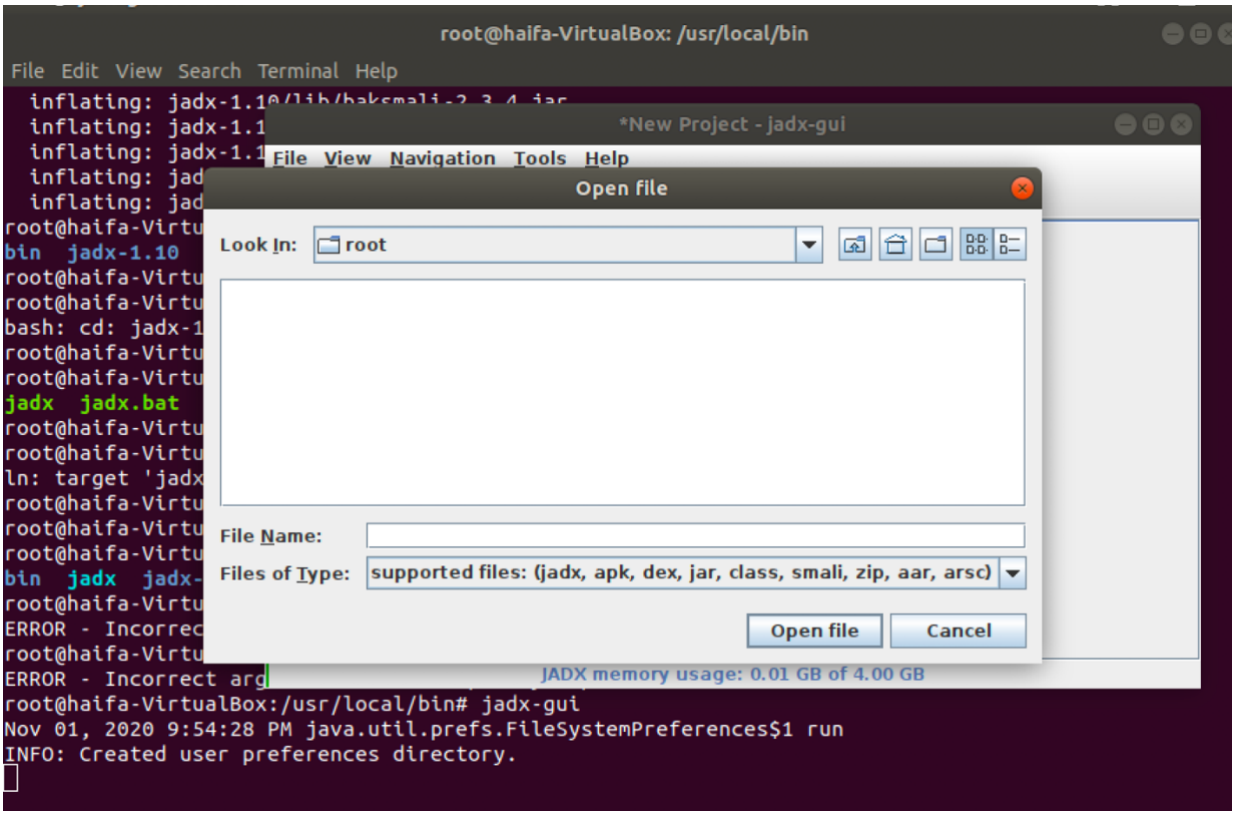


Figure 25

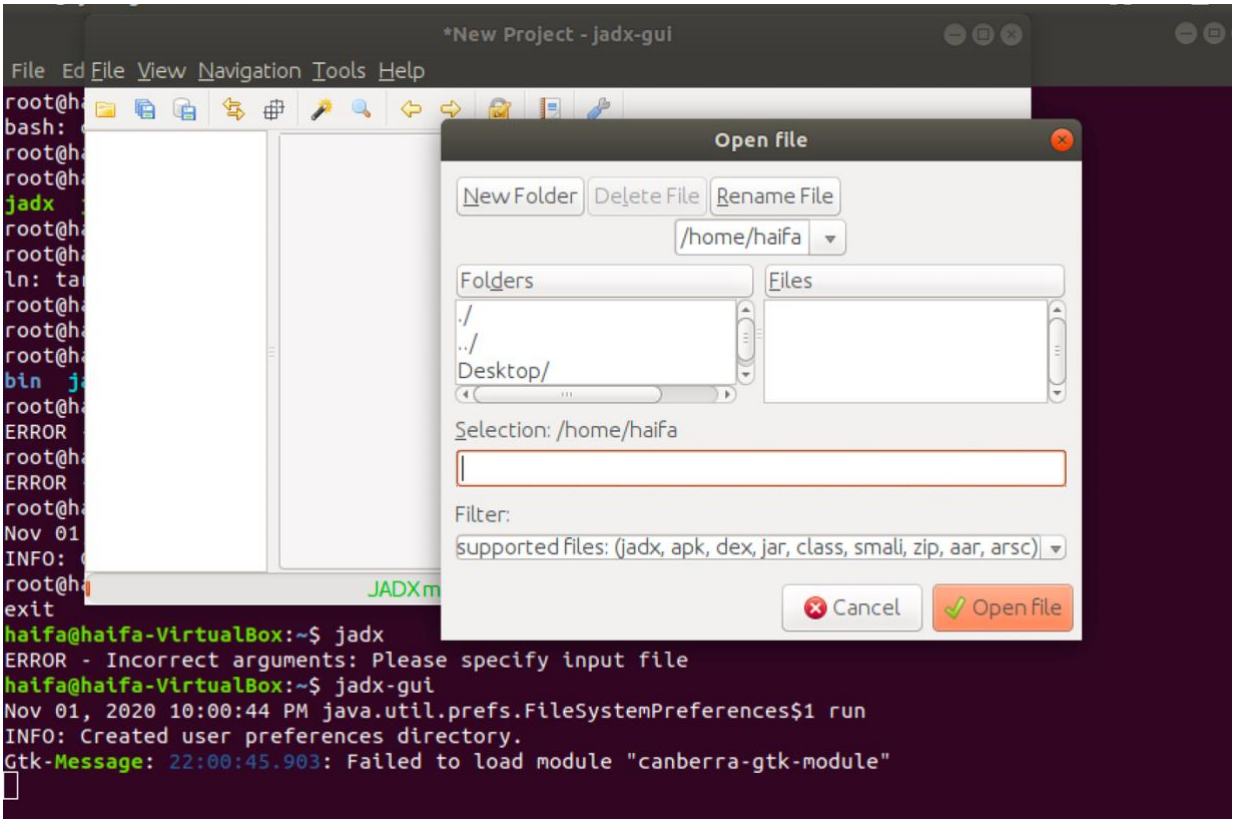


Figure 26

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.

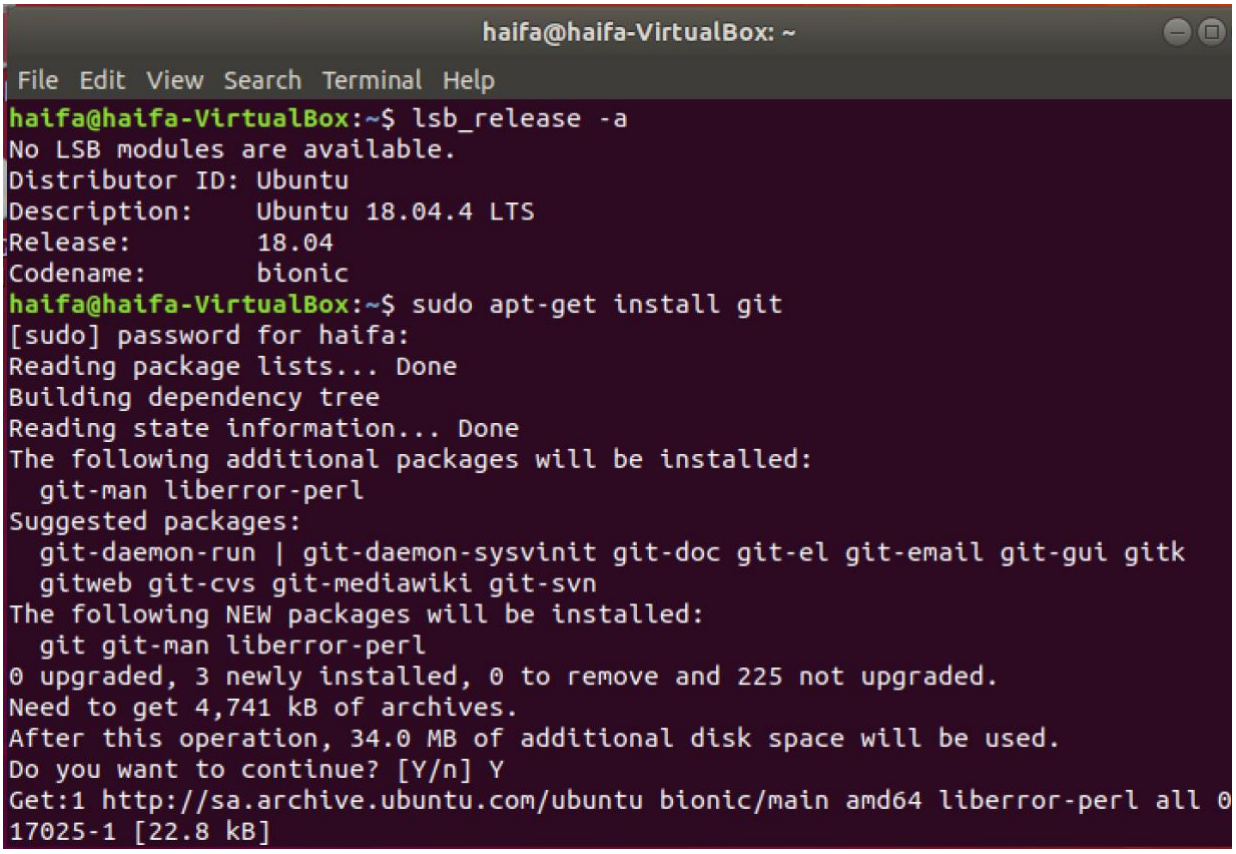


Figure 27

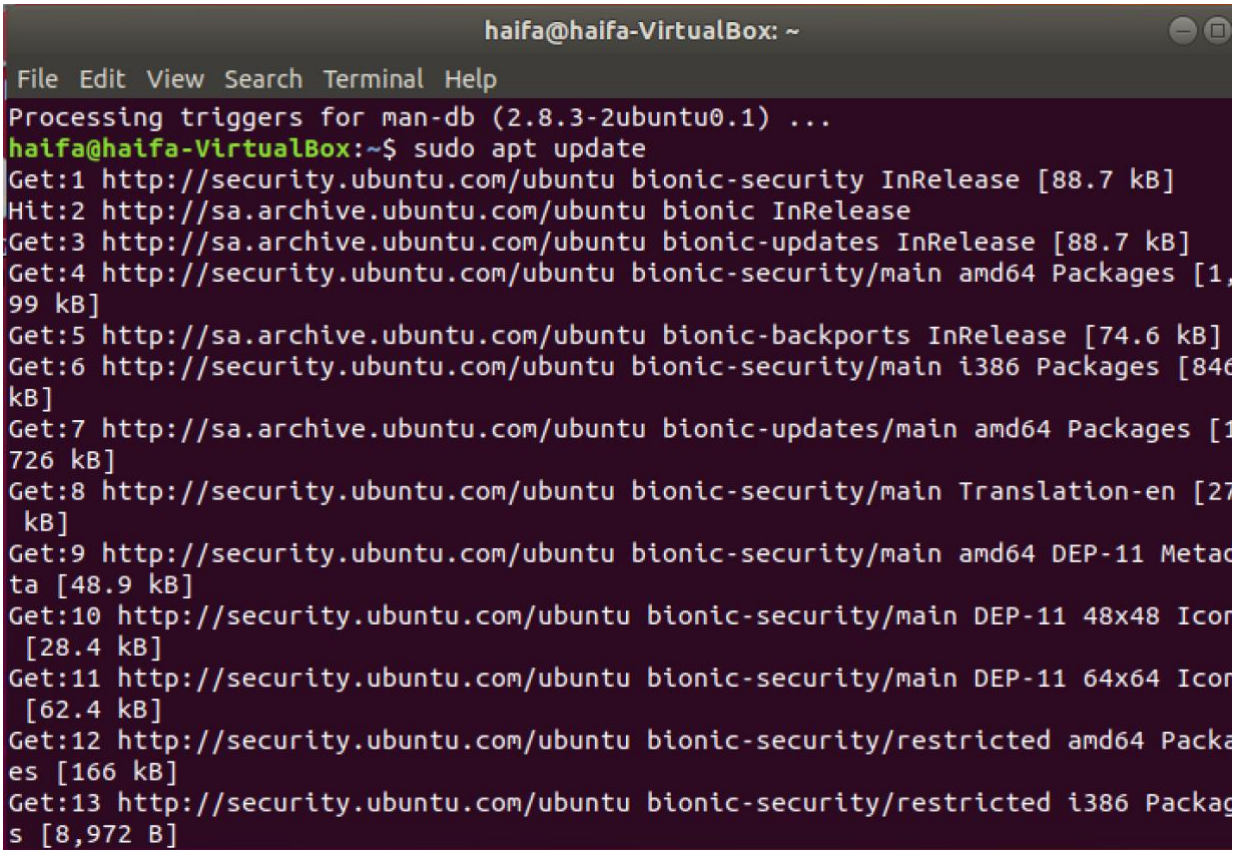


Figure 28

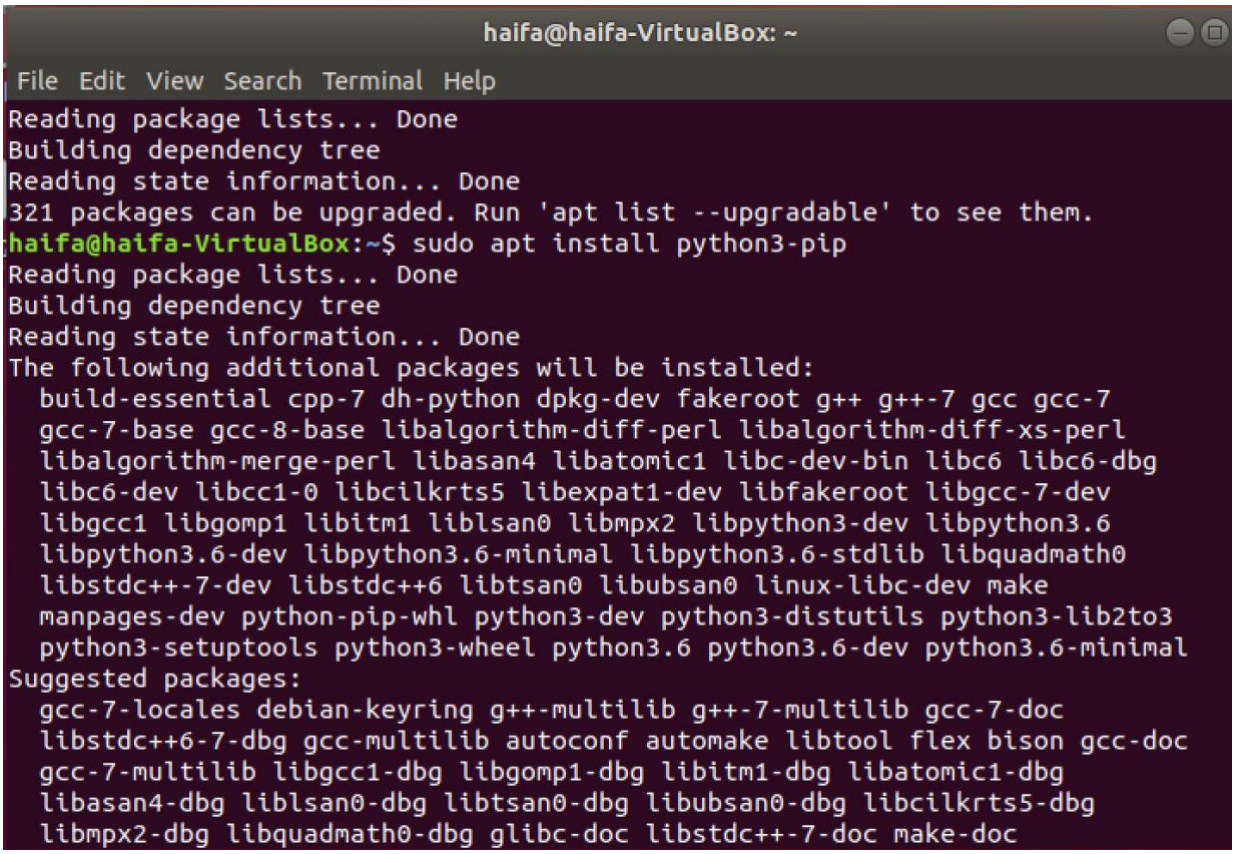


Figure 29

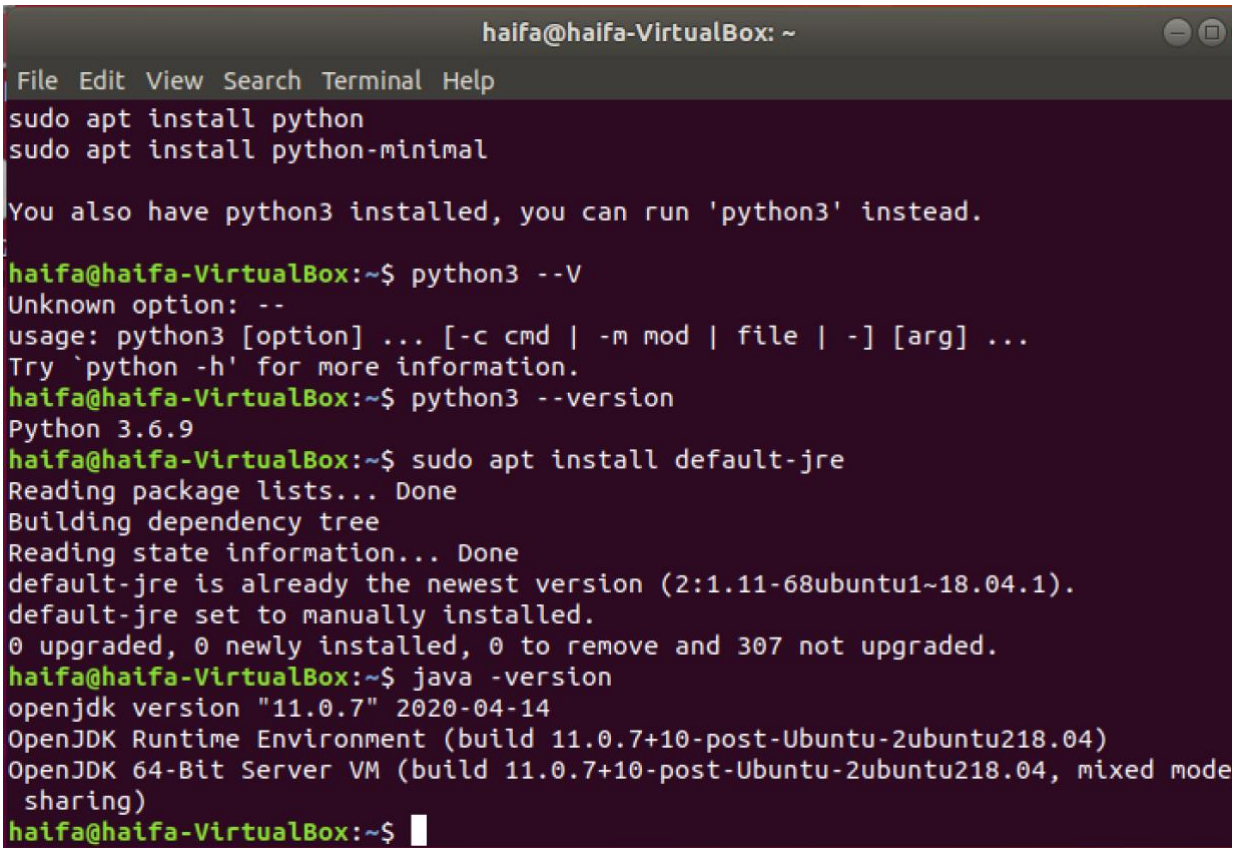


Figure 30

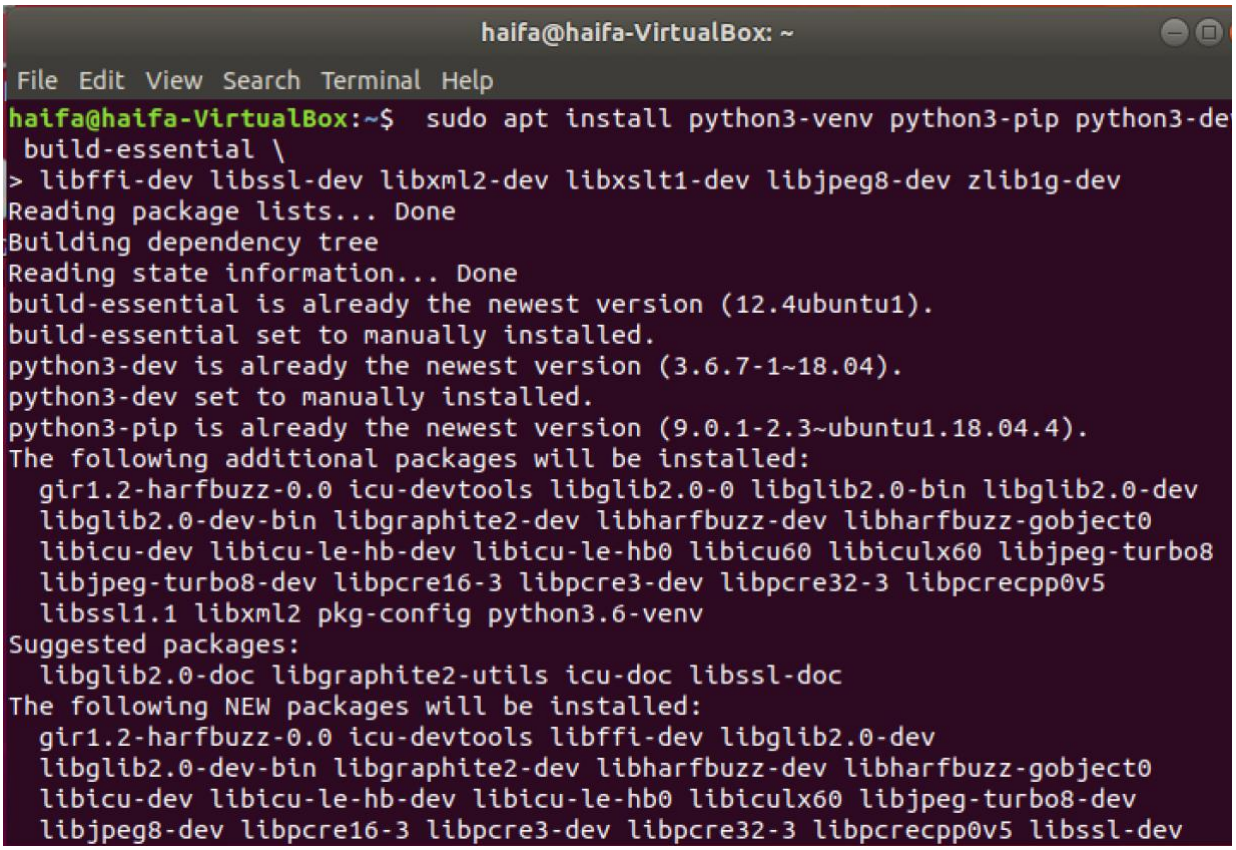


Figure 31

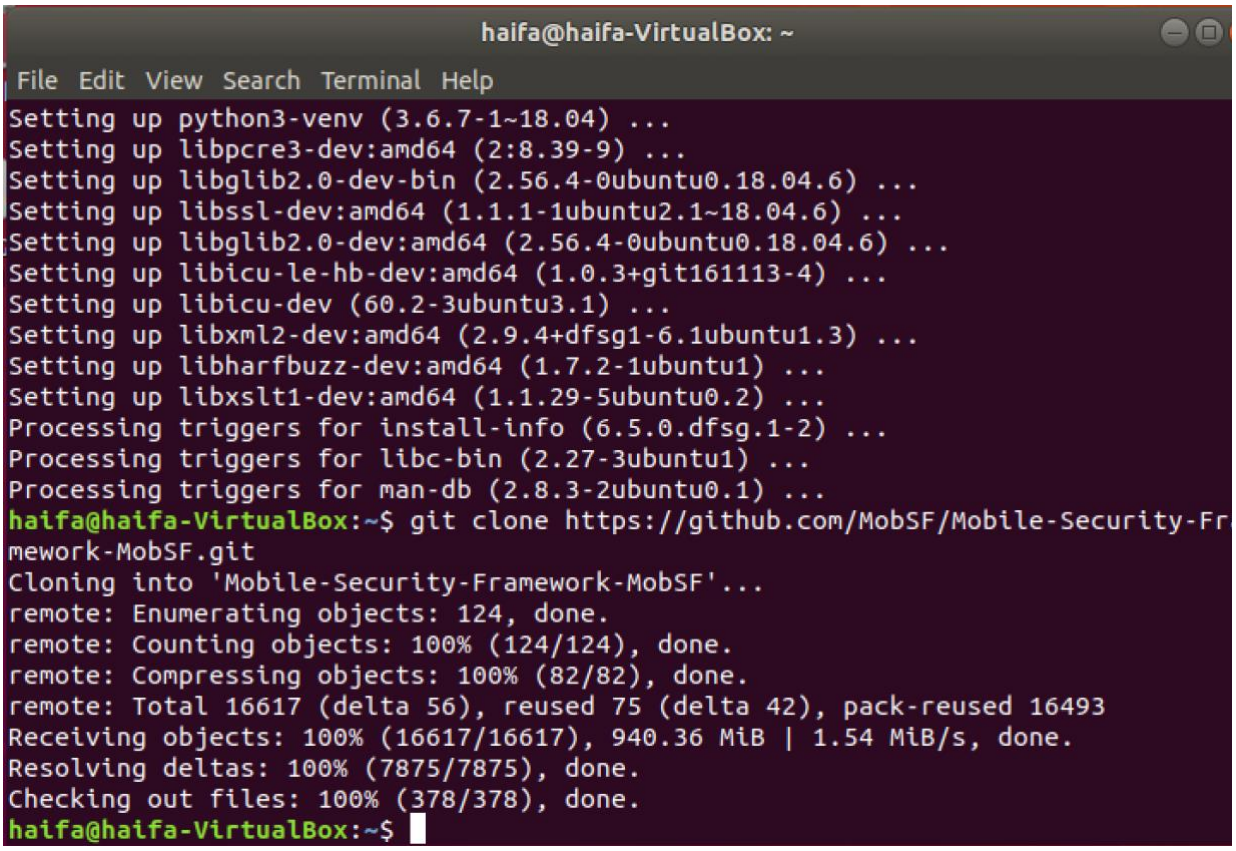


Figure 32

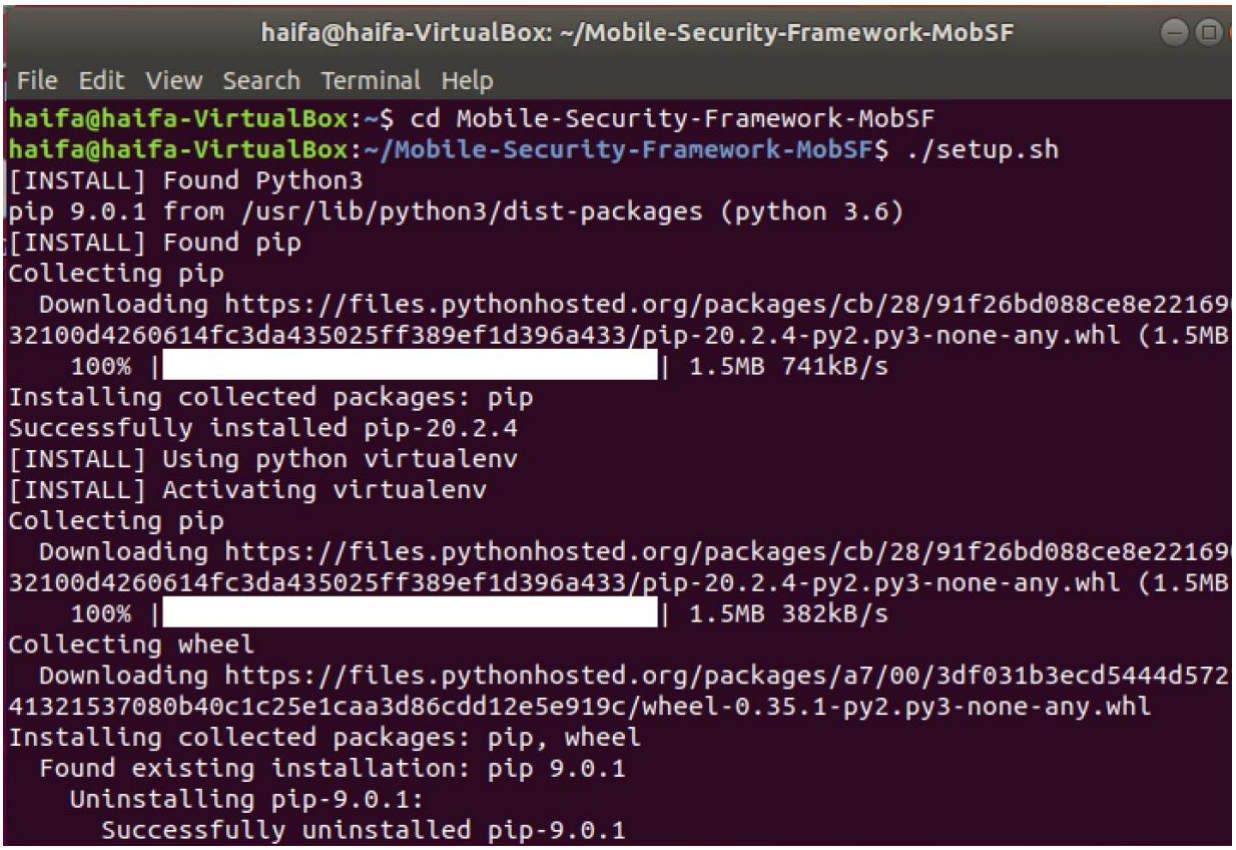


Figure 33

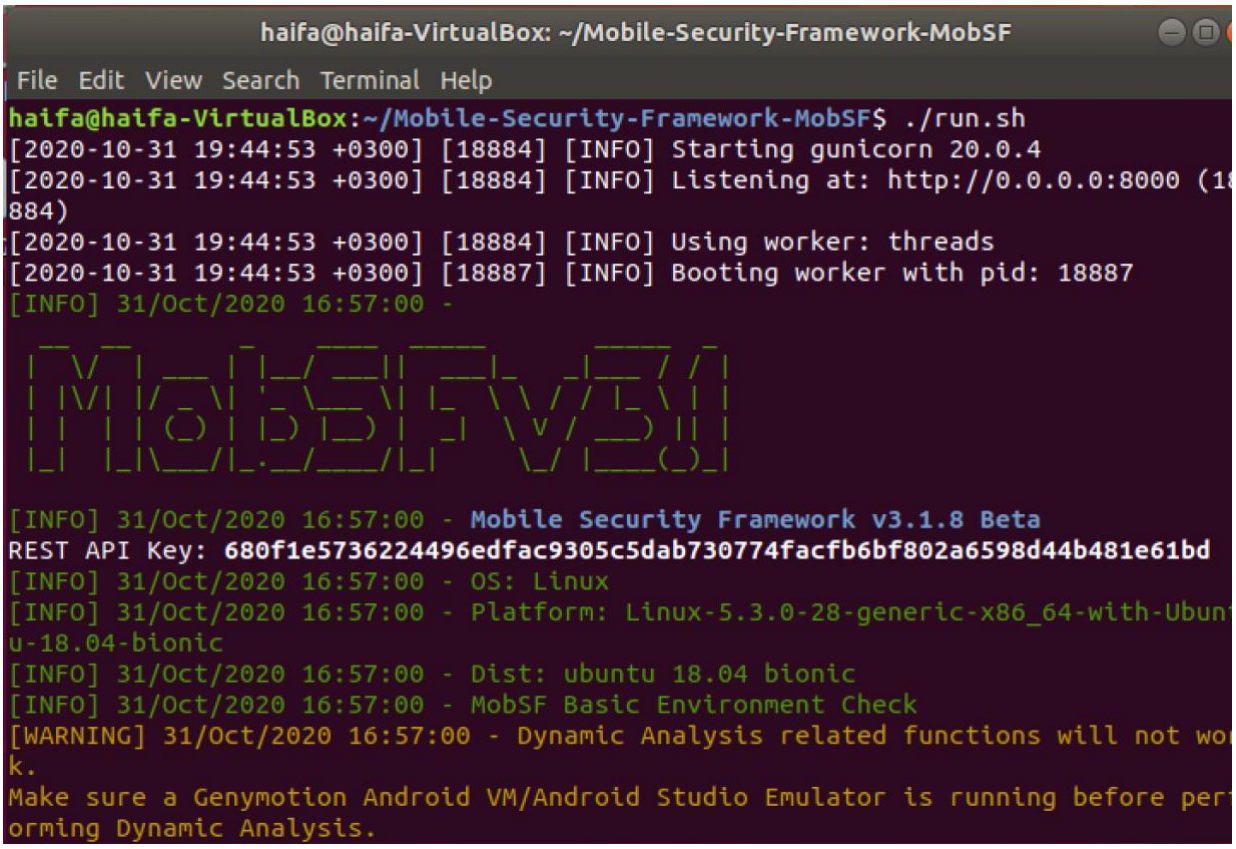


Figure 34

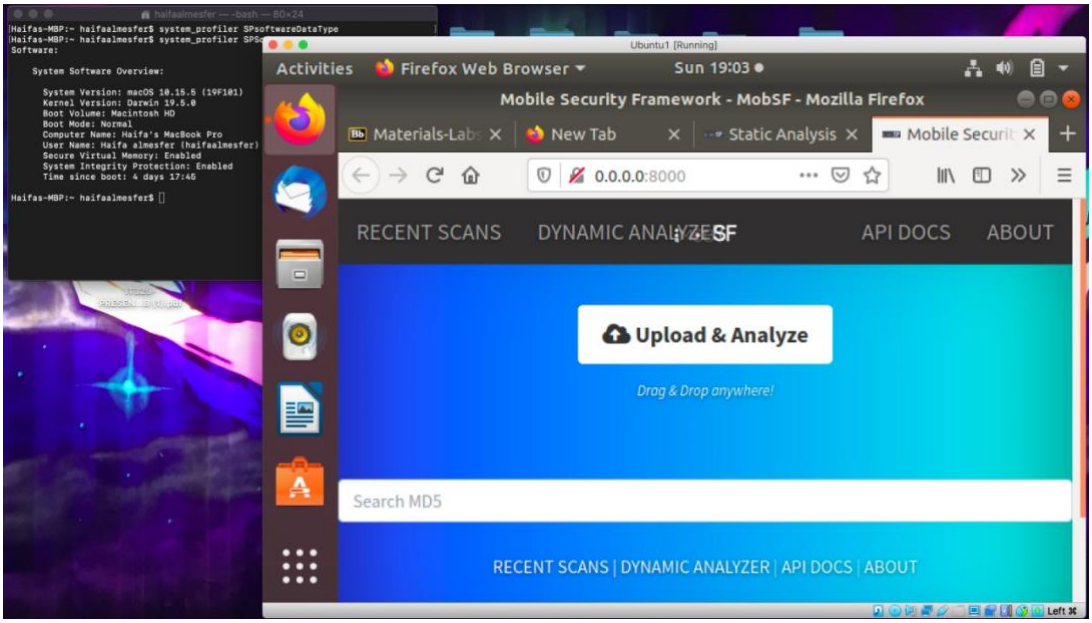


Figure 35 MobSF run

References :

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

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