Project – 1

Name: Ishan Aakash Patel

Student ID: 146151238

Course: CYT-230

Exploring mobile app security tools

Part 1: Investigate Mobile Security Tools

1. Drozer

Overview: Drozer is a comprehensive security assessment framework for Android. Developed by MWR InfoSecurity, it allows security professionals to evaluate the security of Android applications and devices. Drozer provides a wide range of tools for vulnerability assessment and exploitation, making it a powerful resource for Android app security.

Key Features:

- **Security Assessment:** Drozer can identify and exploit vulnerabilities in Android applications, helping to ensure they are secure.
- **Modular Architecture:** The framework is highly extensible, allowing users to add custom modules for specific tasks or testing scenarios.
- **Integration:** It integrates well with other security tools and frameworks, providing a cohesive testing environment.
- **Interactive Shell:** Drozer offers an interactive shell for executing commands and scripts directly on the device or emulator.
- **Automated Testing:** The framework supports automated testing processes, which can save time and improve efficiency.

Use Cases:

- **Security Professionals:** To conduct thorough security assessments of Android applications.
- **Developers:** To test their applications for vulnerabilities before release.
- **Organizations:** To ensure their Android applications adhere to security standards and are safe for users.

Pros:

- Free and open-source, making it accessible to anyone.
- Comprehensive toolset specifically designed for Android security.
- Highly extensible, allowing for customization and addition of new functionalities.
- Strong community support with regular updates and enhancements.

Cons:

- Limited to Android applications and devices.
- Requires technical knowledge and expertise to use effectively.
- The setup process can be complex for beginners.

Preferred For: Drozer is ideal for security professionals and developers who focus on Android application security. Its comprehensive and modular approach makes it a versatile tool for identifying and addressing security issues in Android apps.

Example Usage:

1. **Installation and Setup:**
    - Download Drozer from its official repository.
    - Install Drozer on your machine and set up an Android emulator or connect a physical device.
    - Launch the Drozer server and connect the client to the target device.
2. **Basic Commands:**
    - Use commands like run app.package.list to list all installed packages on the device.
    - Execute run app.package.info -a <package_name> to get detailed information about a specific application.
3. **Exploitation:**
    - Identify potential vulnerabilities using Drozer's built-in modules.
    - Exploit discovered vulnerabilities to test the application's resilience.

Conclusion: Drozer is a powerful and flexible tool for Android application security assessment. Its extensive feature set and modular design make it a top choice for security professionals seeking to ensure the safety and security of Android applications.

2. Frida

Overview: Frida is a dynamic instrumentation toolkit designed for developers, reverse-engineers, and security researchers. It allows users to inject custom scripts into running applications to trace, modify, and explore their behavior. Frida supports multiple platforms, including Android, iOS, Windows, macOS, and Linux, making it a versatile tool for security testing and application analysis.

Key Features:

- **Dynamic Instrumentation:** Frida enables users to inject JavaScript code into running applications, allowing for real-time analysis and modification.
- **Cross-Platform Support:** Frida supports a wide range of platforms, making it useful for testing applications across different operating systems.
- **Extensible:** Users can write custom scripts to perform specific tasks, automate testing, or explore application behavior.
- **Interactive Shell:** Frida provides an interactive shell for executing commands and scripts, making it easier to experiment and test.

- **Open-Source:** Frida is free and open-source, with a strong community contributing to its development and improvement.

Use Cases:

- **Developers:** To debug and test their applications in real-time.
- **Security Researchers:** To analyze and modify the behavior of applications for security testing and vulnerability analysis.
- **Reverse Engineers:** To understand how applications work and identify potential security flaws.

Pros:

- Free and open-source, with a large and active community.
- Supports multiple platforms, making it versatile for various testing scenarios.
- Highly customizable and extensible through custom scripts.
- Provides real-time analysis and modification capabilities.

Cons:

- Requires technical expertise and familiarity with scripting to use effectively.
- Can be complex to set up and configure, especially for beginners.
- Potential performance overhead when instrumenting applications.

Preferred For: Frida is best suited for developers, security researchers, and reverse engineers who need a powerful and flexible tool for analyzing and modifying applications. Its dynamic instrumentation capabilities make it an excellent choice for real-time application analysis and security testing.

Example Usage:

1. **Installation and Setup:**
    - Install Frida using Python's package manager (pip install frida).
    - Set up Frida on the target device (e.g., an Android or iOS device).
2. **Basic Commands:**
    - Use the Frida CLI to attach to a running process (e.g., frida -U -n <app_name>).
    - Inject custom JavaScript code to trace or modify application behavior.
3. **Scripting:**
    - Write custom scripts to automate testing or explore specific application functionalities.
    - Use Frida's API to interact with the target application and perform dynamic analysis.

Conclusion: Frida is a versatile and powerful toolkit for dynamic application analysis and security testing. Its cross-platform support and extensibility through custom scripts make it a valuable resource for developers, security researchers, and reverse engineers seeking to understand and secure applications.

Step 2: Research a Free or Open-Source Mobile Security Tool

MobSF (Mobile Security Framework)

Overview: MobSF (Mobile Security Framework) is an automated, open-source mobile application security testing framework that supports Android, iOS, and Windows platforms. It provides comprehensive security analysis, including static and dynamic analysis, malware analysis, and web API testing. MobSF is designed to be user-friendly and integrates various security tools into a single platform.

Key Features:

1. **Static Analysis:** MobSF performs static analysis by decompiling APK, IPA, and APPX files. It scans for vulnerabilities, insecure configurations, and sensitive information leaks in the source code.
2. **Dynamic Analysis:** MobSF can analyze the runtime behavior of mobile applications by monitoring the interactions between the app and the operating system. It uses an Android emulator or real devices for this purpose.
3. **Malware Analysis:** The framework includes capabilities to detect malware by analyzing the app's behavior and code patterns.
4. **Web API Testing:** MobSF tests the security of web APIs used by mobile applications to ensure they are not vulnerable to common web attacks.
5. **Report Generation:** The tool generates detailed reports highlighting security issues and providing remediation suggestions.

Pros:

- **Comprehensive Analysis:** MobSF provides both static and dynamic analysis, offering a thorough assessment of mobile applications.
- **Cross-Platform Support:** It supports multiple platforms, making it versatile for testing various types of mobile apps.
- **User-Friendly Interface:** The tool has an intuitive web interface that simplifies the analysis process.
- **Automated Testing:** MobSF automates many aspects of security testing, saving time and effort for security professionals.
- **Active Community and Support:** The tool is maintained by an active community, ensuring regular updates and improvements.

Cons:

- **Resource Intensive:** Dynamic analysis, in particular, can be resource-intensive and may require a powerful machine or server.
- **Initial Setup:** Setting up MobSF, especially for dynamic analysis, can be complex and may require some configuration.
- **Limited Real Device Support:** While MobSF supports dynamic analysis, it is primarily designed for emulators. Testing on real devices may require additional setup.

Comparison with Tools from Step 1:

1. **Drozer vs. MobSF:**
    - **Drozer** is a specialized tool for Android security assessment, focusing on identifying and exploiting vulnerabilities within Android apps. It provides an interactive shell and modular architecture for extensive security testing.
    - **MobSF**, on the other hand, offers a more comprehensive and automated approach, supporting multiple platforms and including features like malware analysis and web API testing. For a quick, high-level assessment, MobSF is more user-friendly and less technical than Drozer.
2. **Frida vs. MobSF:**
    - **Frida** is a dynamic instrumentation toolkit that allows real-time code injection and modification across various platforms. It is highly flexible and extensible but requires significant expertise and scripting knowledge.
    - **MobSF** is more accessible to users with limited technical expertise, offering a wide range of automated testing features out-of-the-box. For users needing quick insights without deep technical intervention, MobSF is more suitable.

Recommendation:

- For comprehensive mobile app security assessment, I prefer **MobSF** due to its user-friendly interface, automated testing capabilities, and support for multiple platforms. It provides a holistic view of mobile app security, including static, dynamic, and malware analysis, making it suitable for both Android and iOS applications.
- However, for more in-depth and hands-on security testing on Android apps, **Drozer** remains a top choice due to its powerful exploitation and vulnerability assessment tools.
- For dynamic and real-time analysis, **Frida** is unmatched in flexibility, although it requires a higher level of technical expertise.

Step-3 : Frida Installation

Step 1: Get the zip file for Frida server from github.

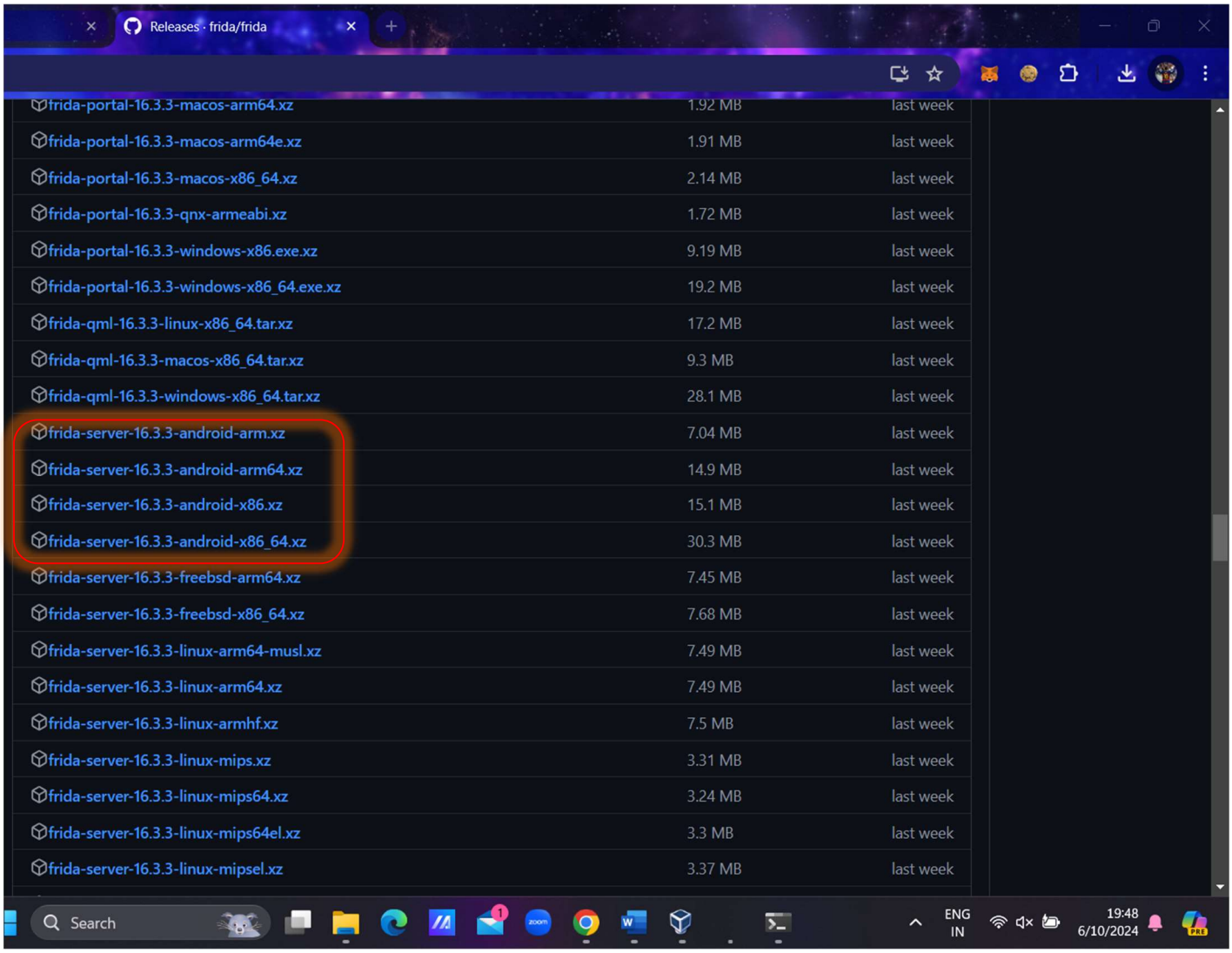
Step 2: Move the unzip file to your android emulator using adb. (Make sure adb is preinstalled)

Step 3: Start the adb shell and make sure you have the root privileges.

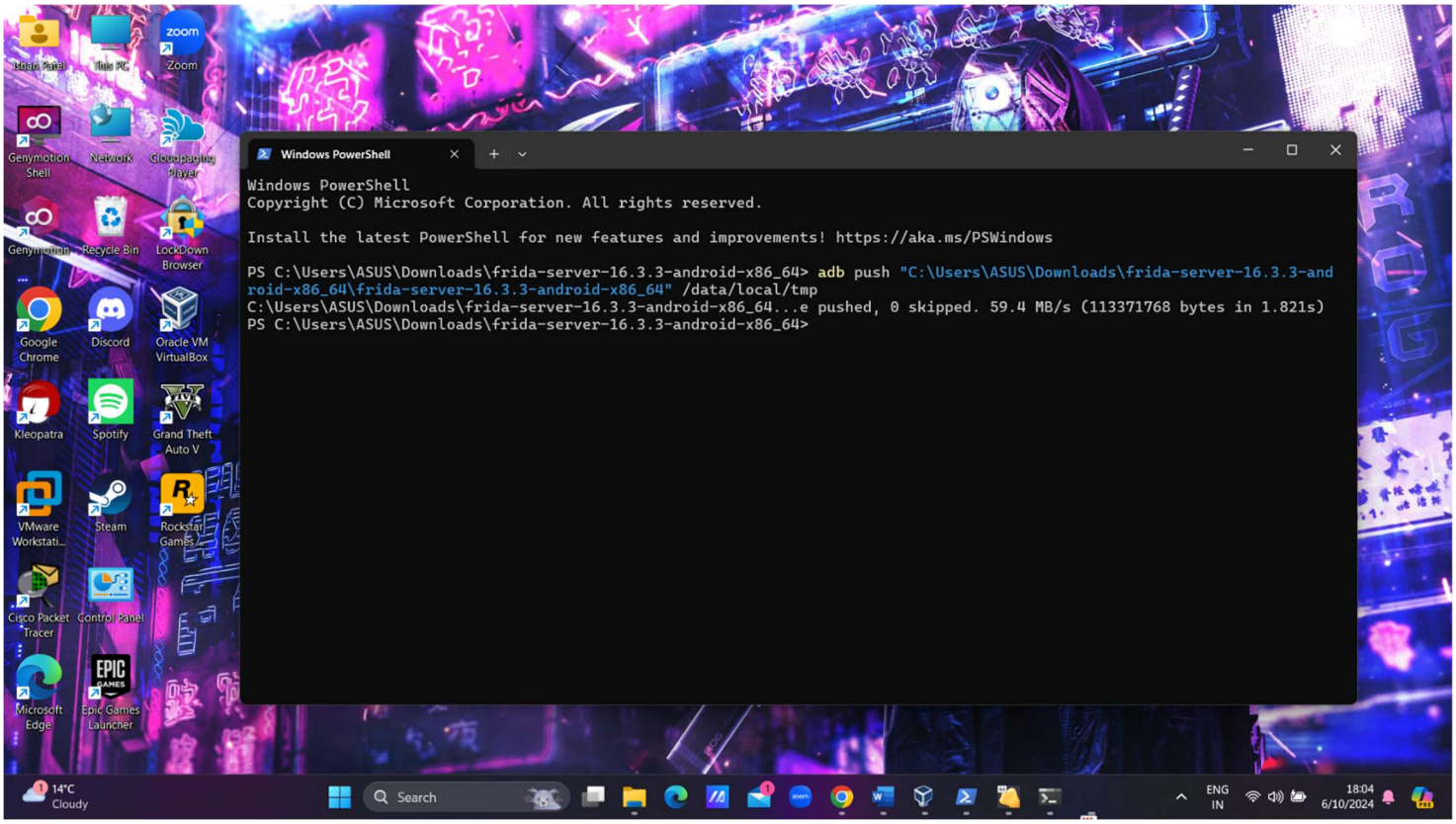
Step 4: locate the file and change the necessary permissions and then start it.

ADB is a powerful tool that provides developers and users with a wide range of capabilities for interacting with Android devices and emulators from the command line. Its versatility makes it indispensable for tasks such as app development, testing, debugging, and device management.

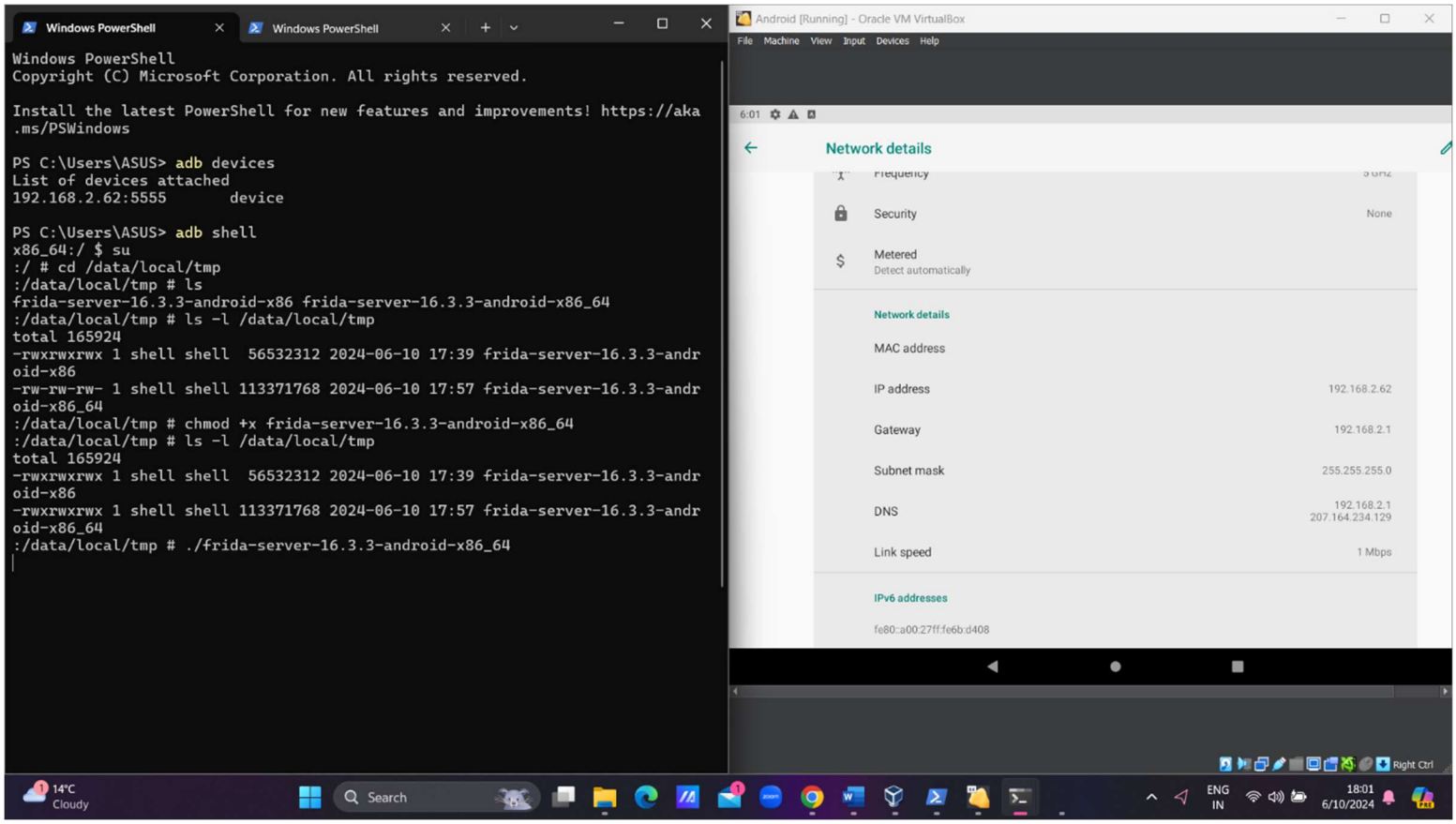
Step – 1 : Get the zip file for Frida server from github.



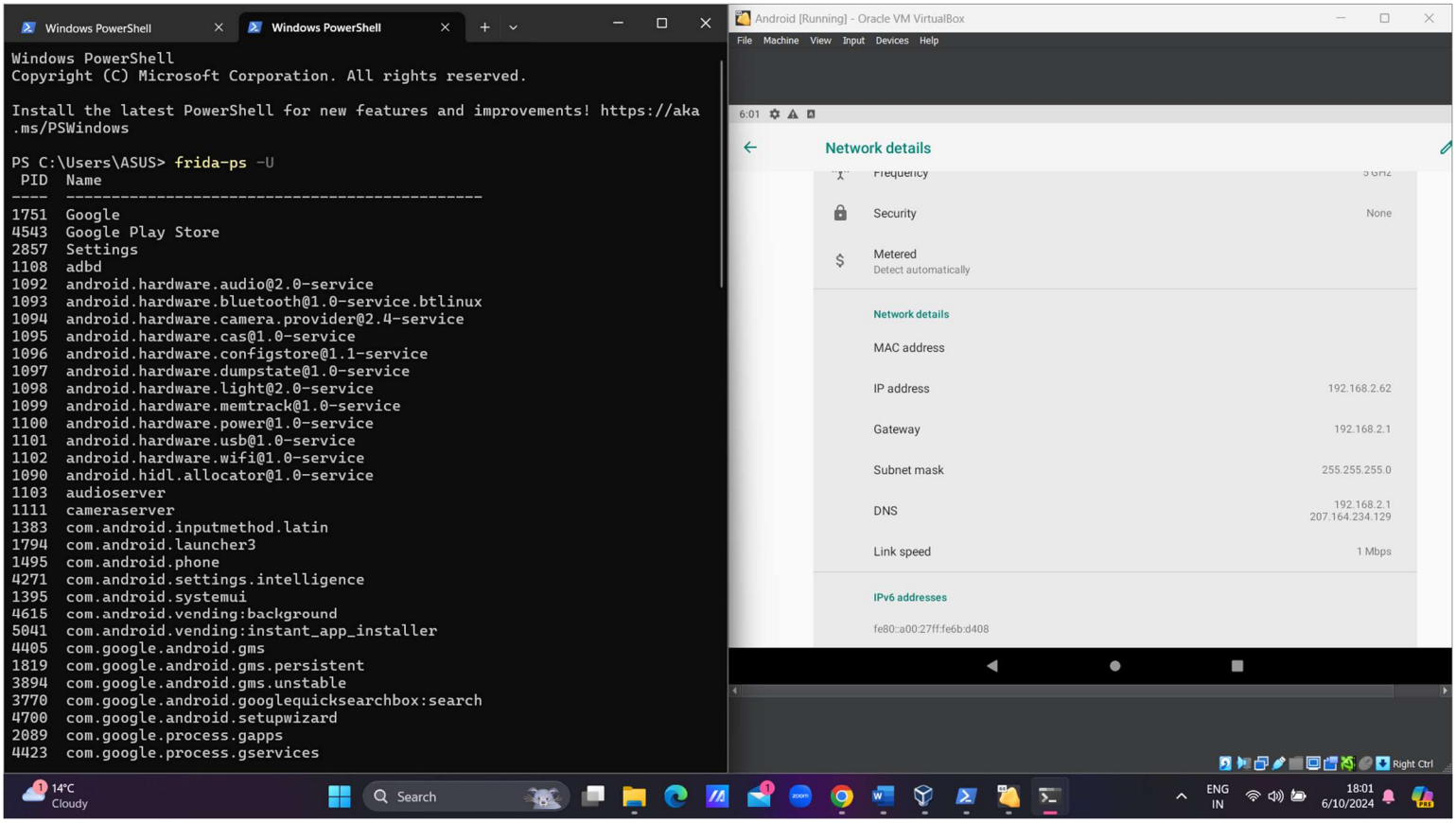
Step 2 : Move the unzip file to your android emulator using adb. (Make sure adb is preinstalled)



Step 3: Start the adb shell and make sure you have the root privileges.



Step 4: locate the file and change the necessary permissions and then start it.

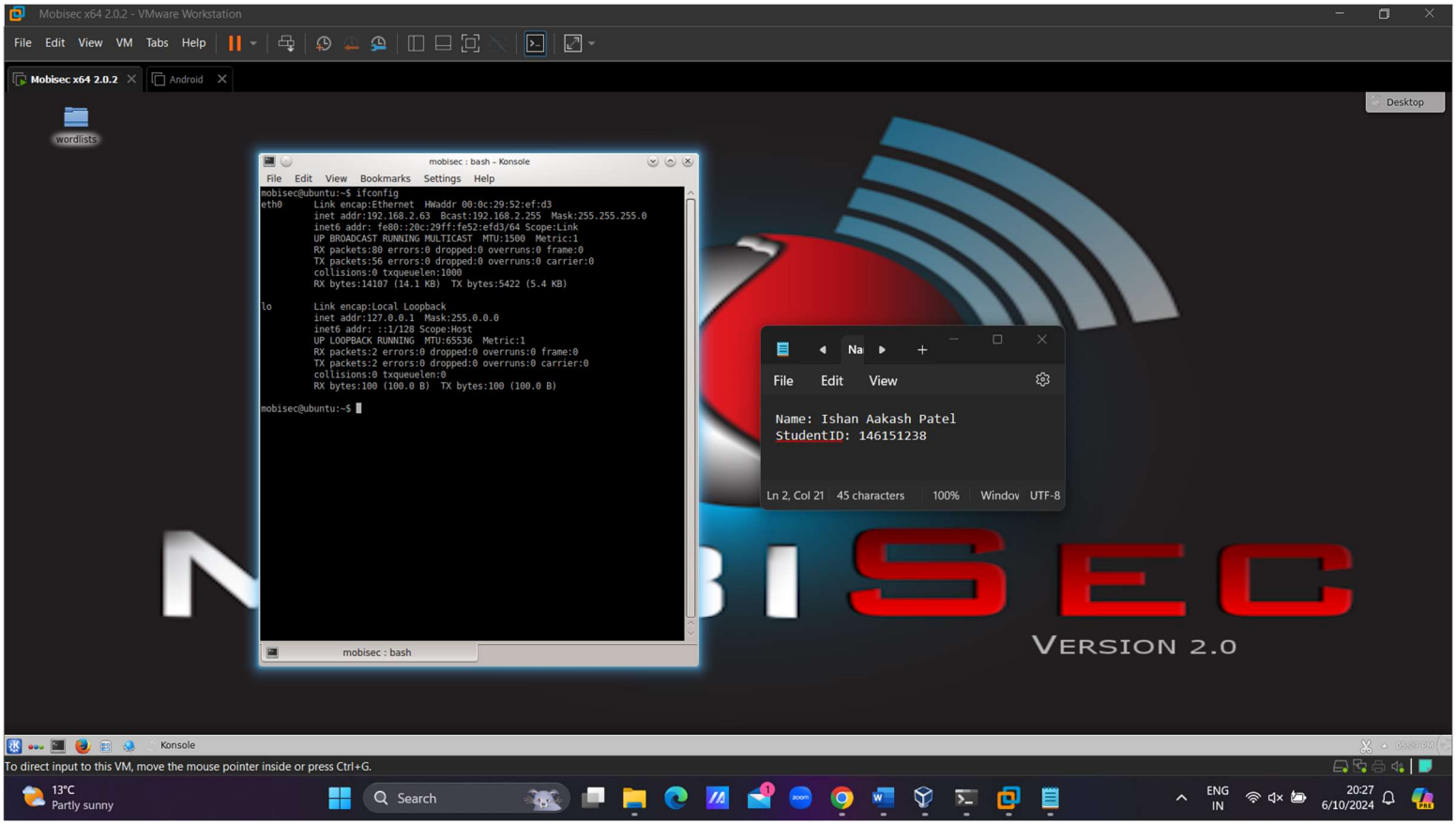


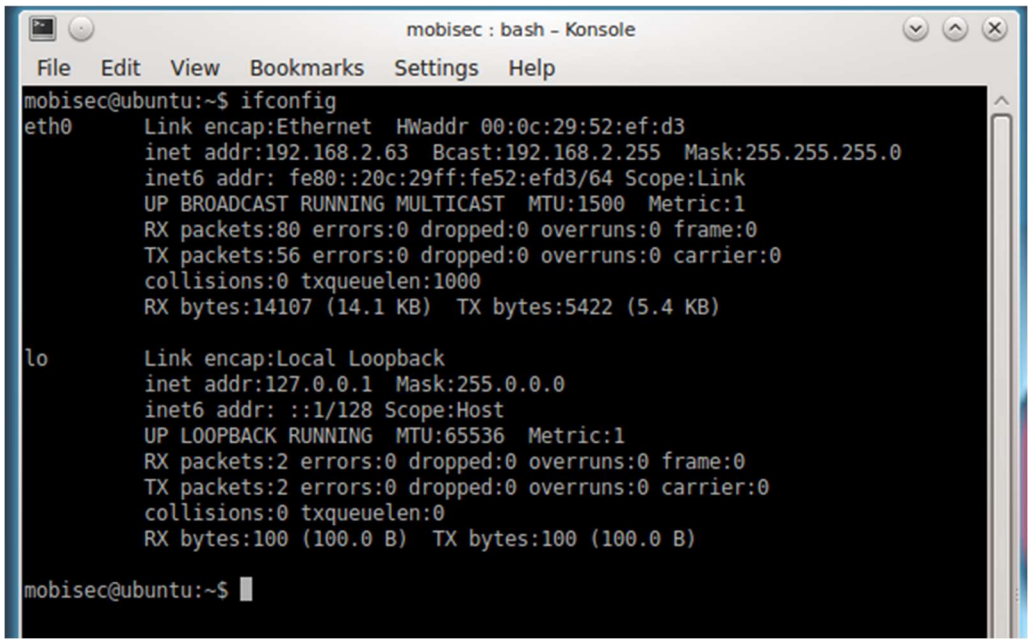
Here you can see all the process which are running in the android emulator and in the similar way you can make your own scripts to run in ant individual process or app to capture the traffic, capture call recordings, etc.

Part – 2

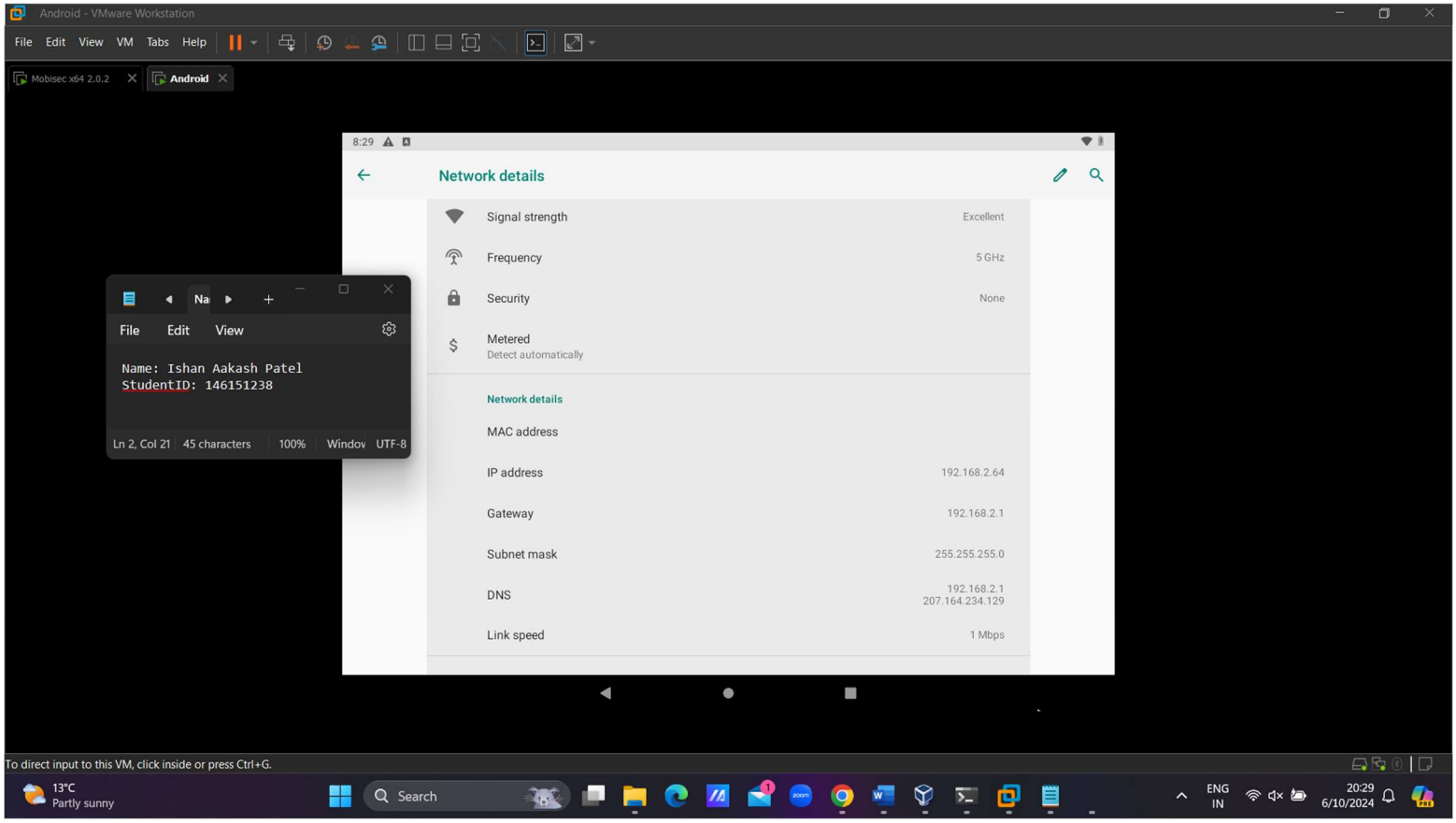
Lab Experiment Using MobiSec

Step 1: Installation of Mobisec Virtual Machin

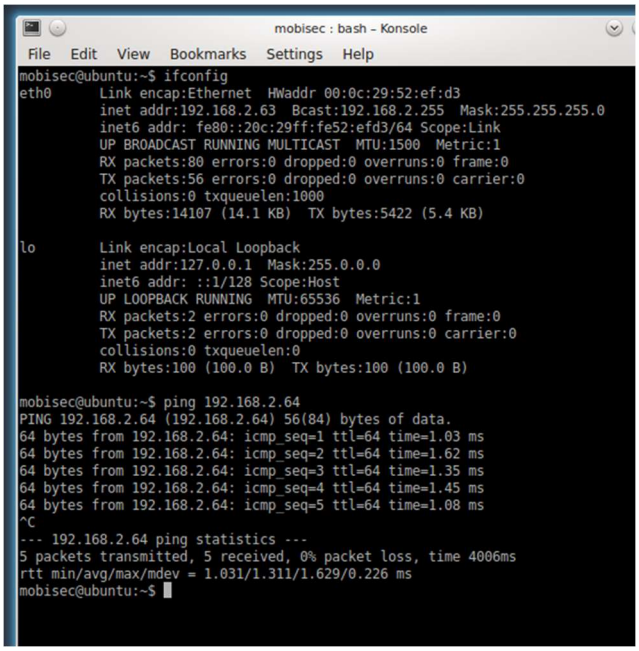




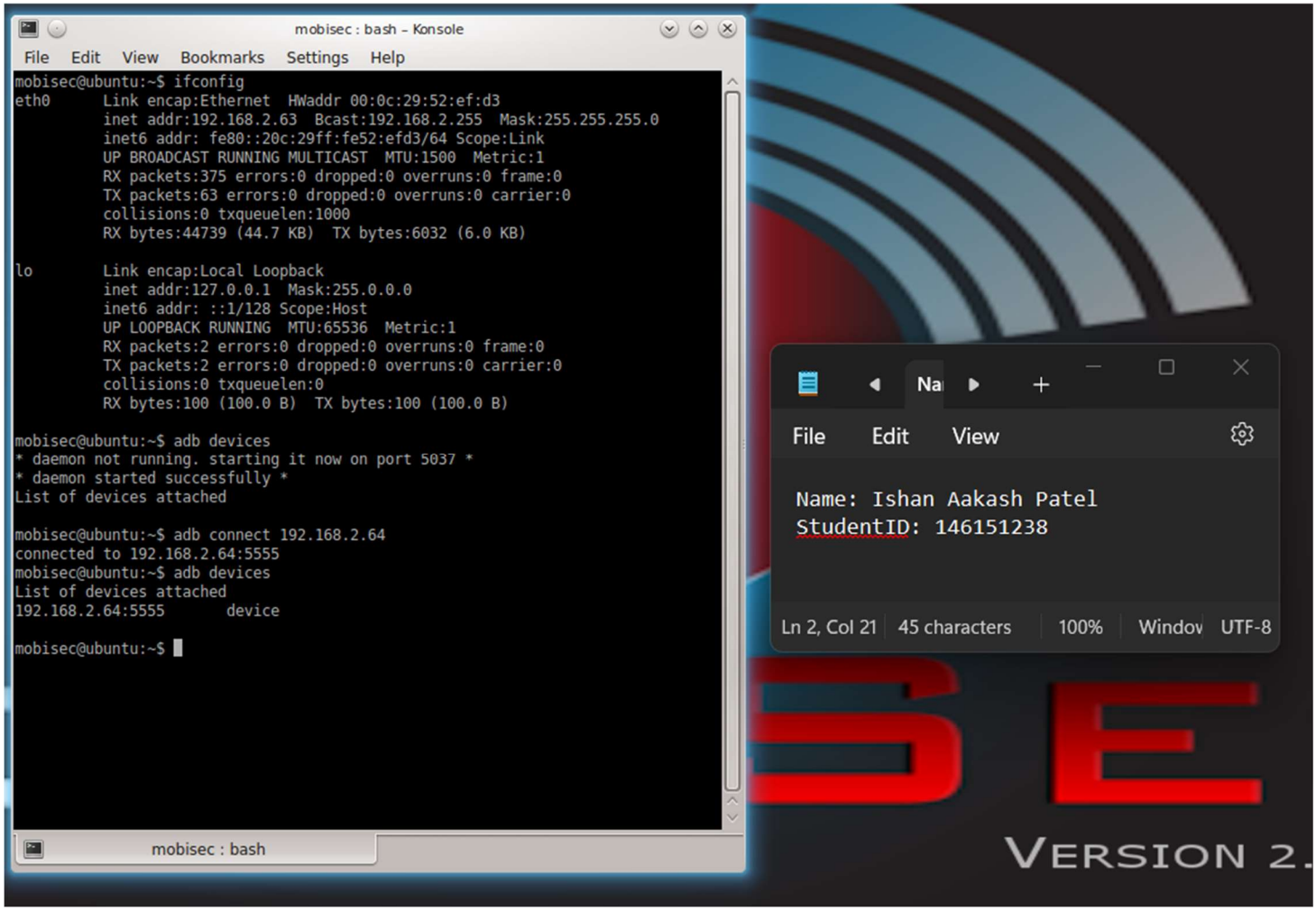
Android x86 Emulator



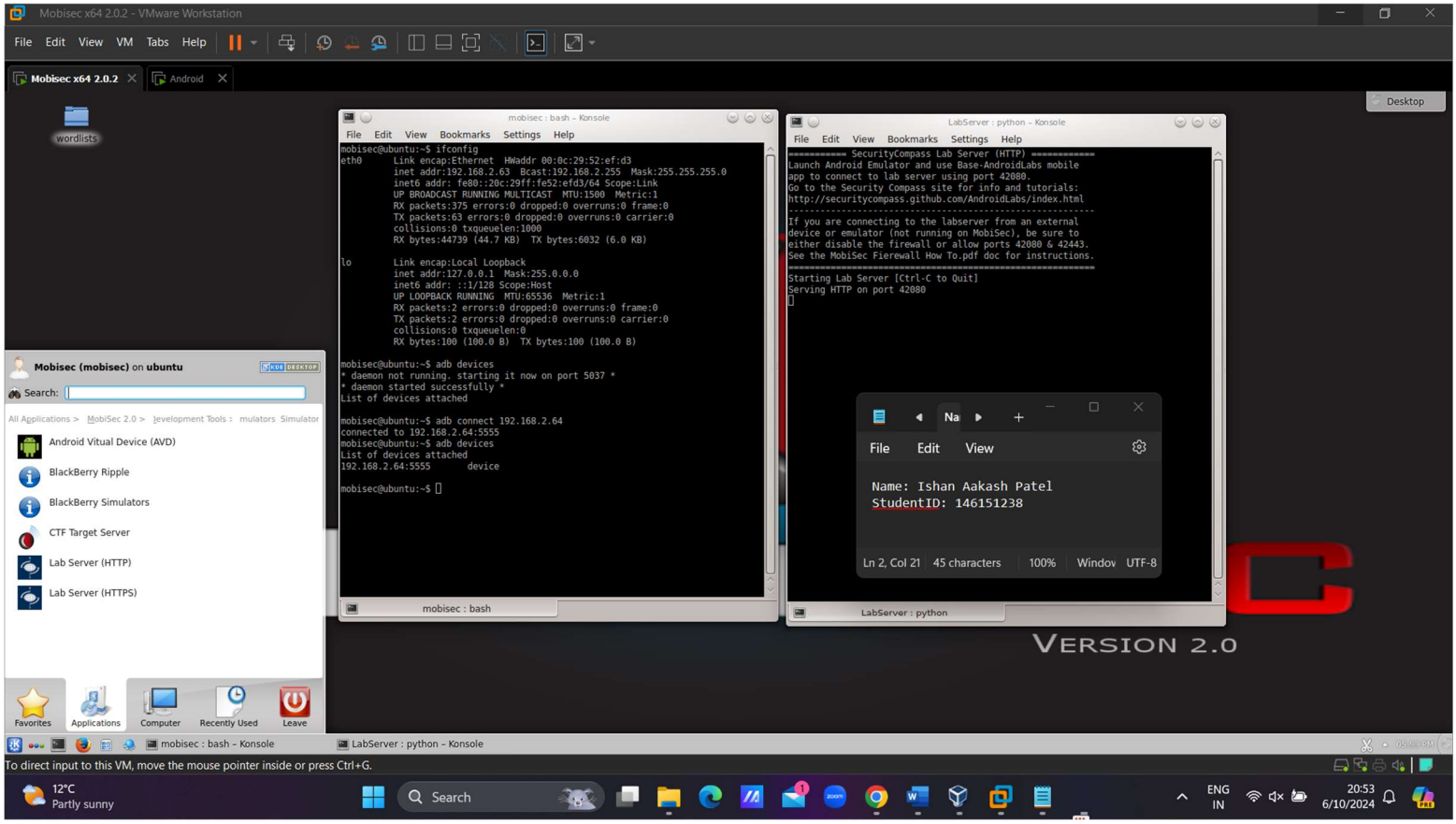
Make sure both of them are in the same network. In the below screenshot you can see that I can ping from mobisce to android x86.



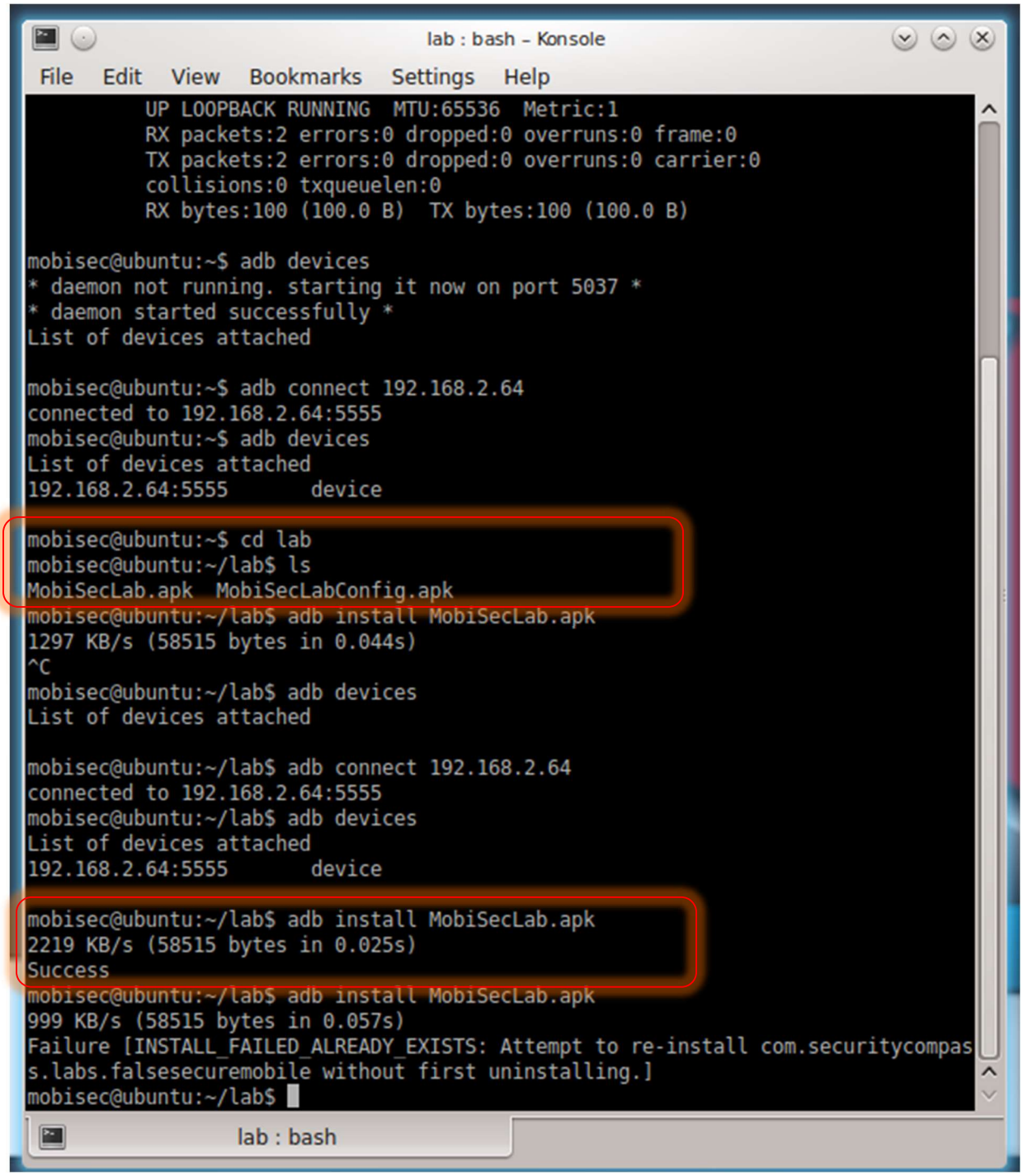
Step 2: Now using adb devices connect to the android x86 emulator.



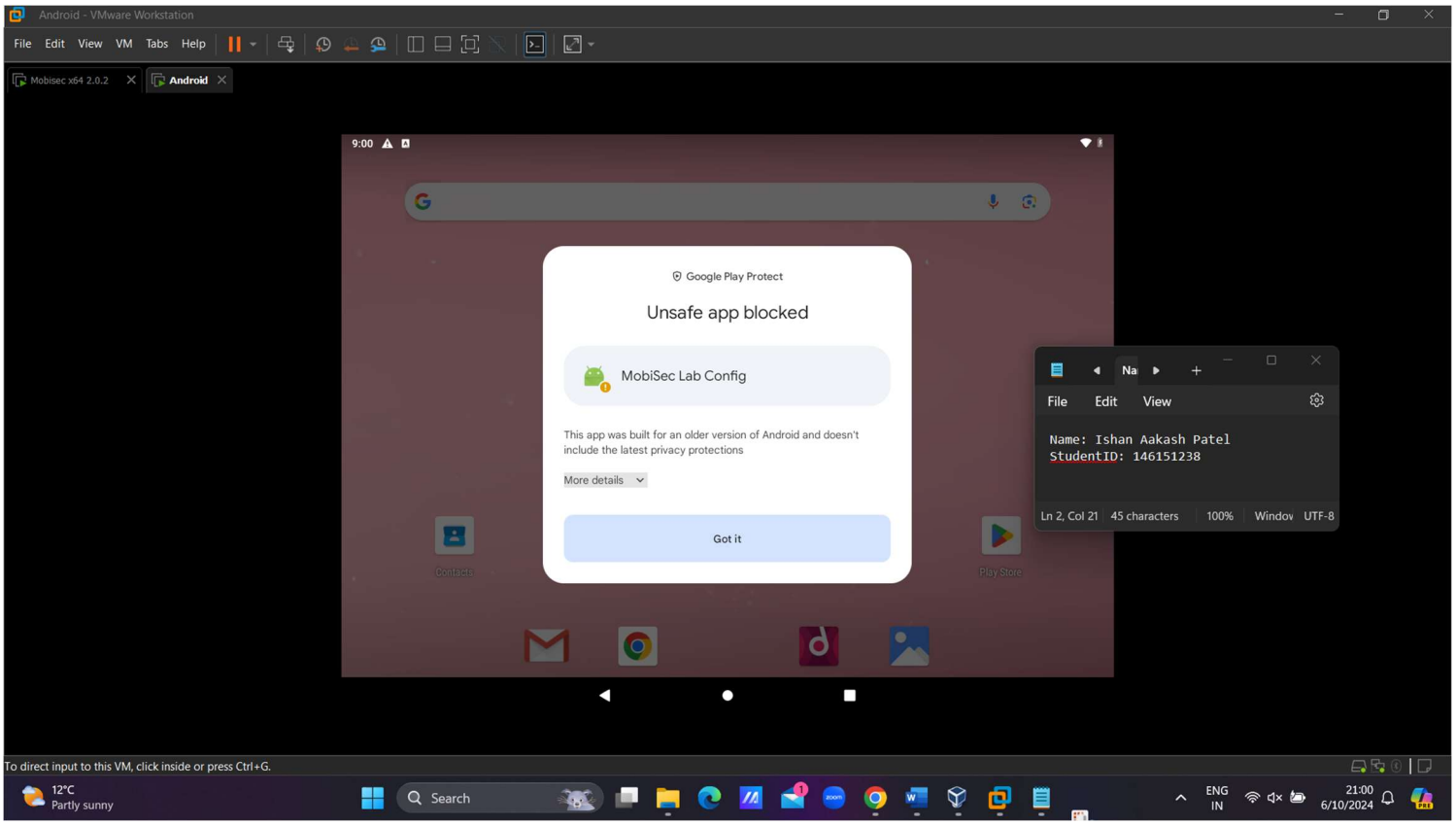
Step 3: Now start the http services in the mobisec vm.



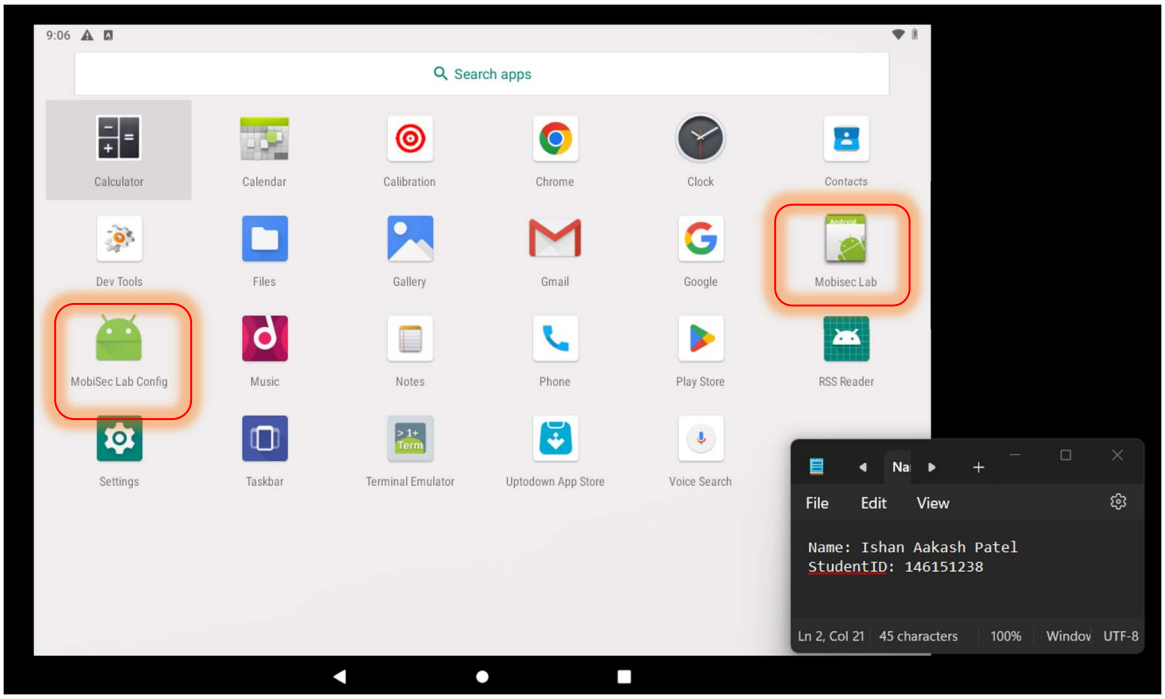
Step 4: Now using adb move the lab apk files to the android x86 emulator.



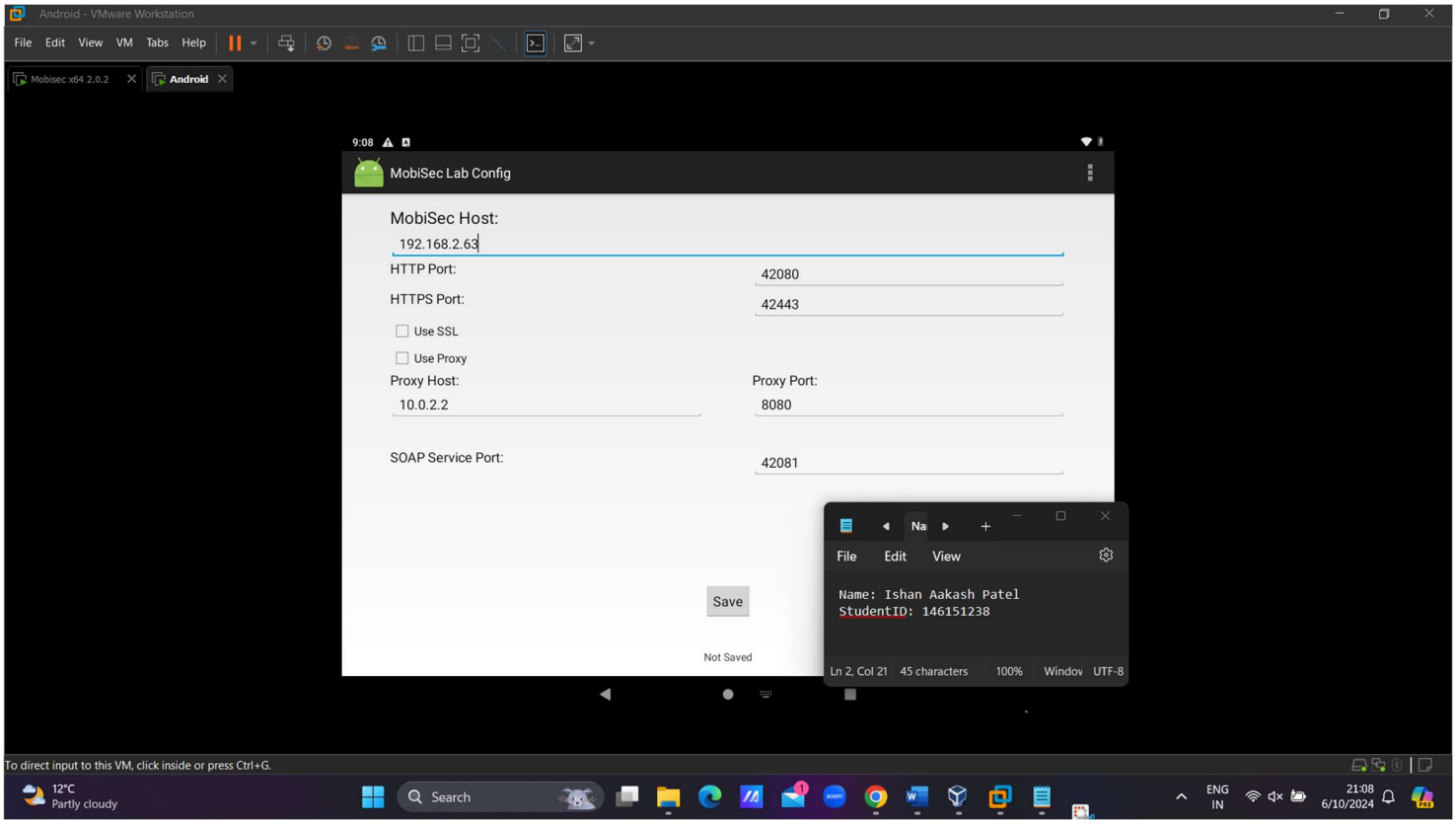
A pop-up will come to the android x86 emulator which you will have to install anyway.

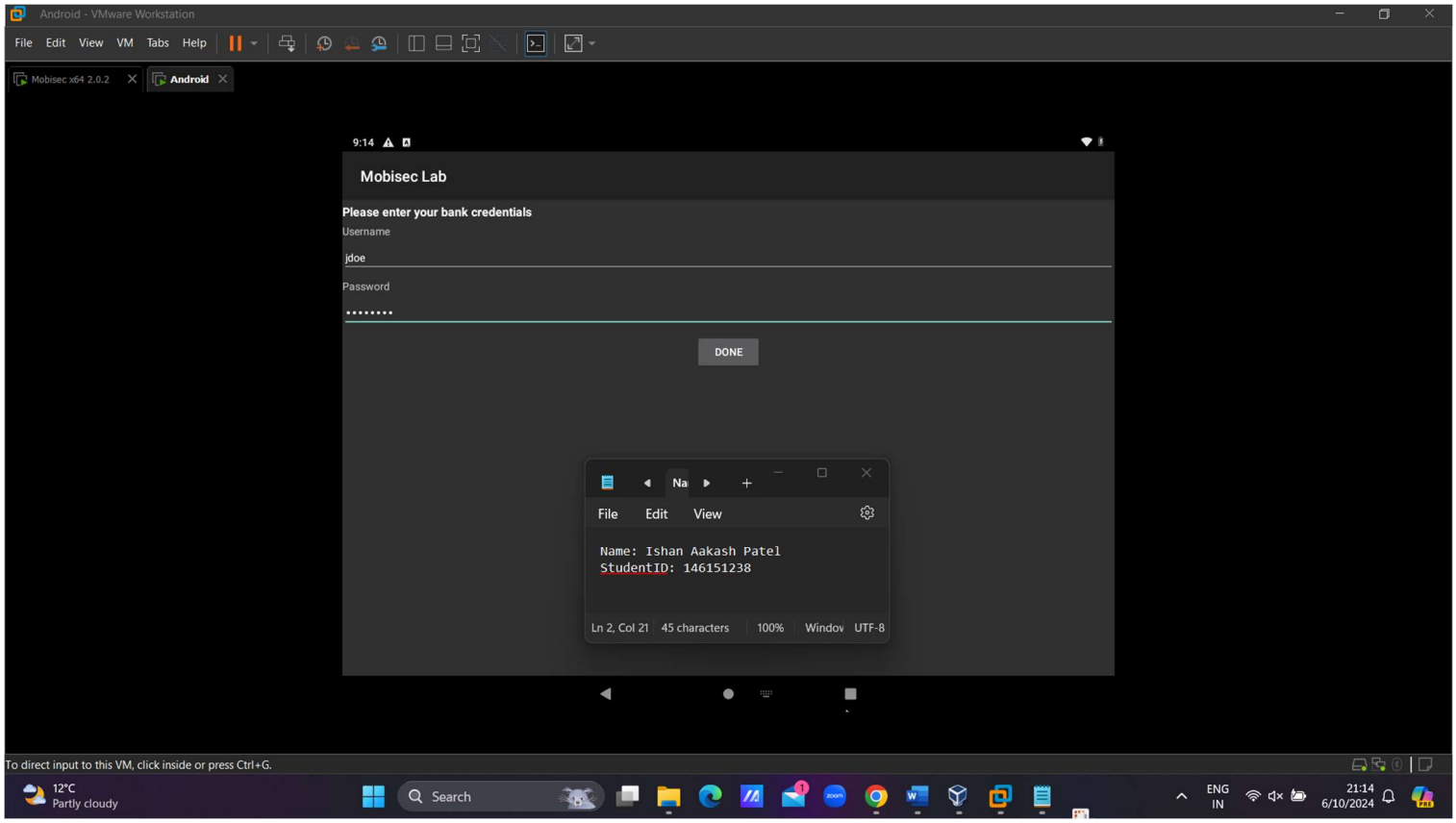


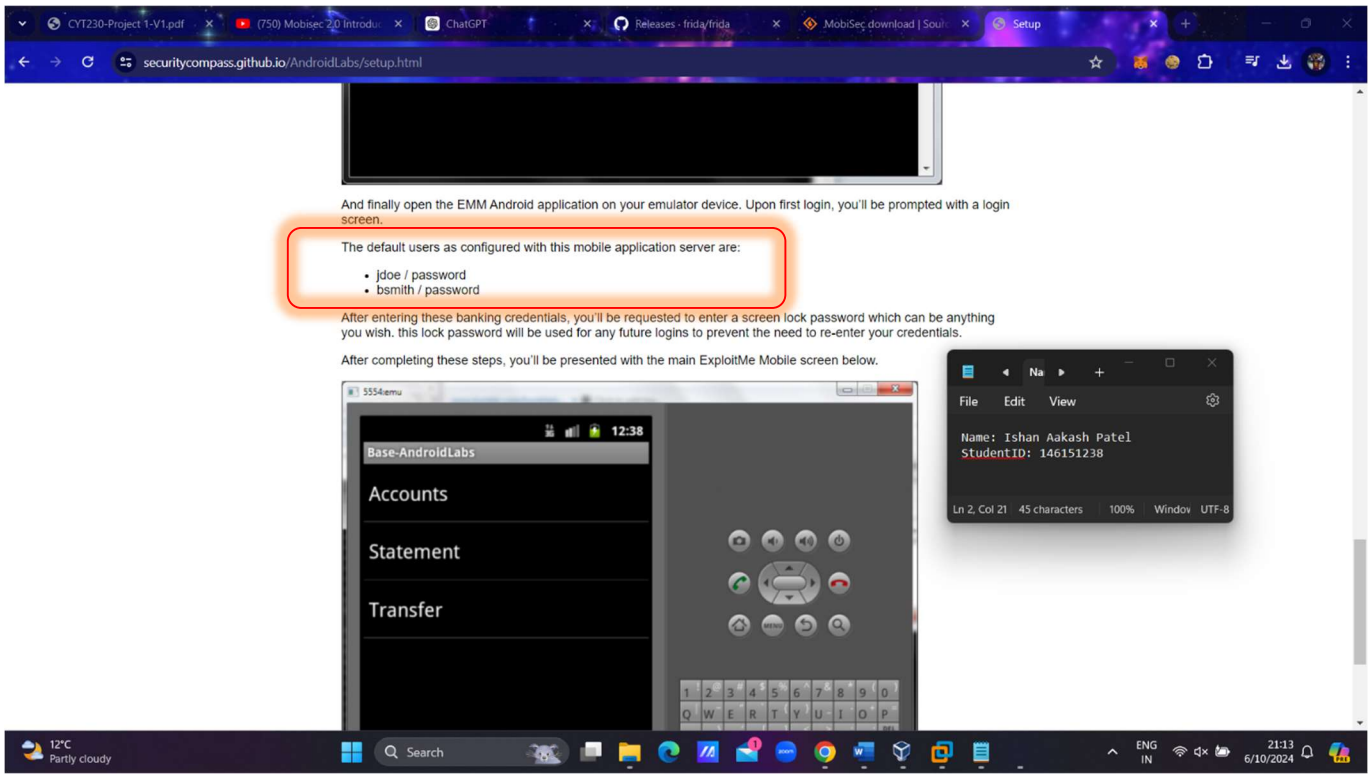
As you can see in the below screenshot the lab is installed.



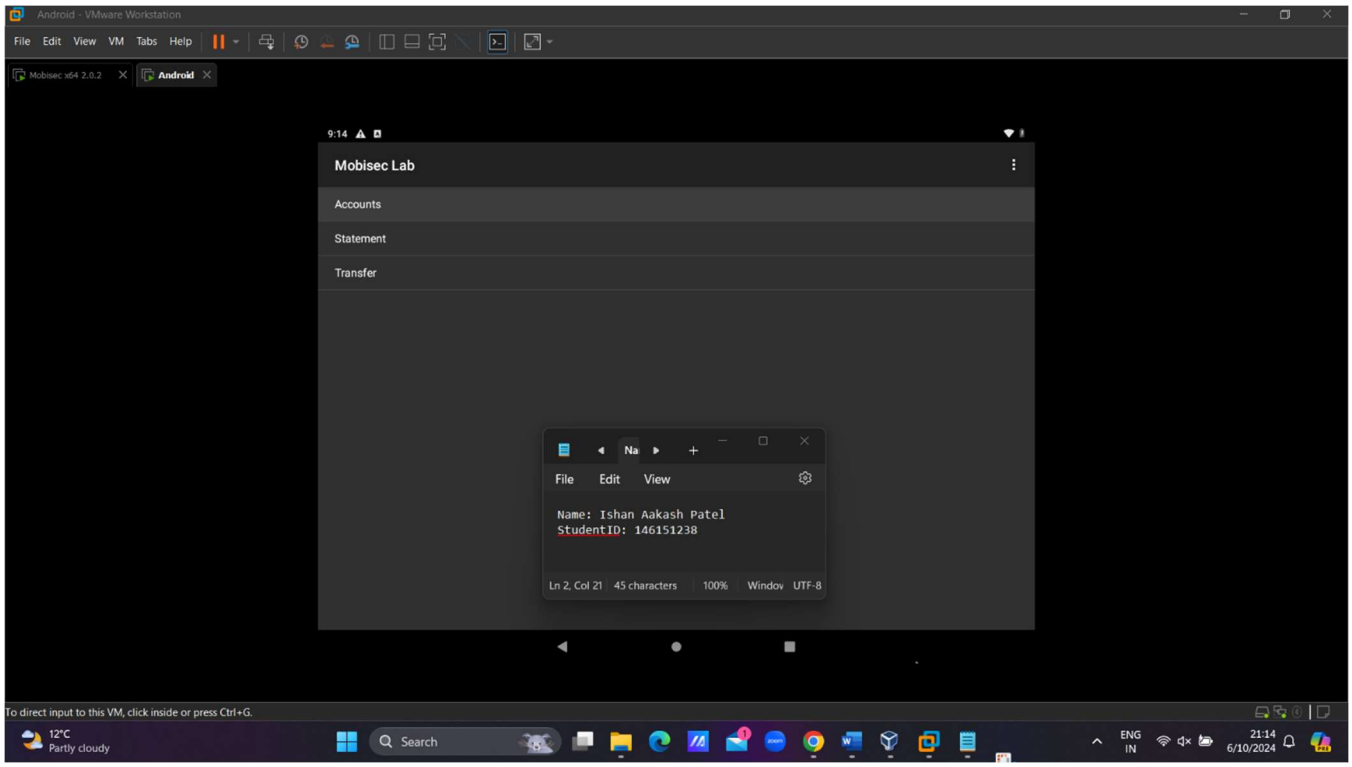
Step 5: Now open the config file and set the mobisec IP and then open the lab and setup the account.



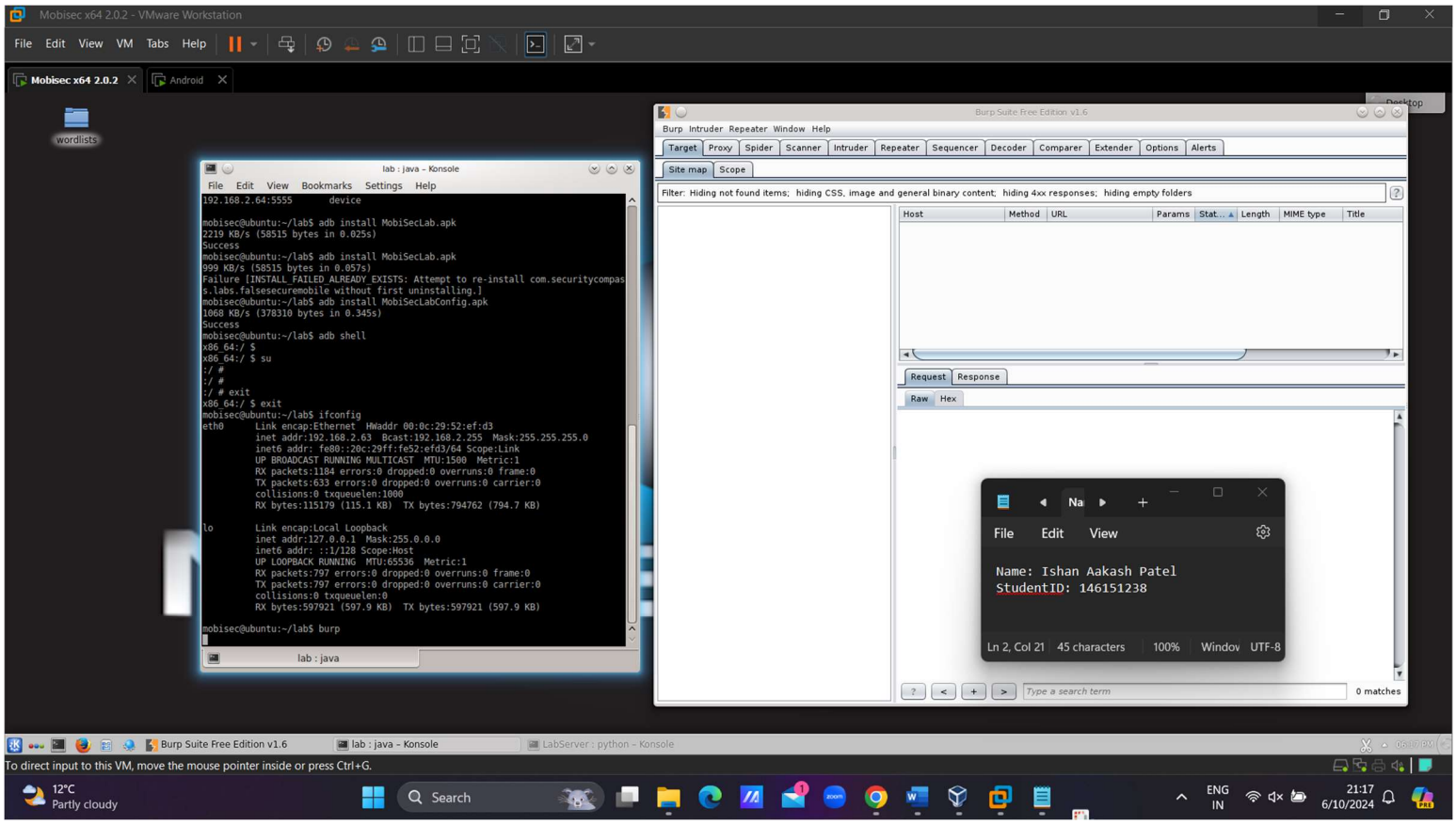

Android - VMware Workstation
File Edit View VM Tabs Help
Mobisec x64 2.0.2
Android
9:14
Mobisec Lab
Please enter your bank credentials
Username
jdoe
Password
DONE
File Edit View
Name: Ishan Aakash Patel
StudentID: 146151238
Ln 2, Col 21 45 characters 100% Windov UTF-8
To direct input to this VM, click inside or press Ctrl+G.
12°C
Partly cloudy
Search
ENG
IN
21:14
6/10/2024


securitycompass.github.io/AndroidLabs/setup.html
And finally open the EMM Android application on your emulator device. Upon first login, you'll be prompted with a login screen.
The default users as configured with this mobile application server are:
jdoe / password
bsmith / password
After entering these banking credentials, you'll be requested to enter a screen lock password which can be anything you wish. this lock password will be used for any future logins to prevent the need to re-enter your credentials.
After completing these steps, you'll be presented with the main ExploitMe Mobile screen below.
12:38
Base-AndroidLabs
Accounts
Statement
Transfer
Name: Ishan Aakash Patel
StudentID: 146151238
Ln 2, Col 21 45 characters 100% Windov UTF-8
12°C
Partly cloudy
Search
ENG
IN
21:13
6/10/2024

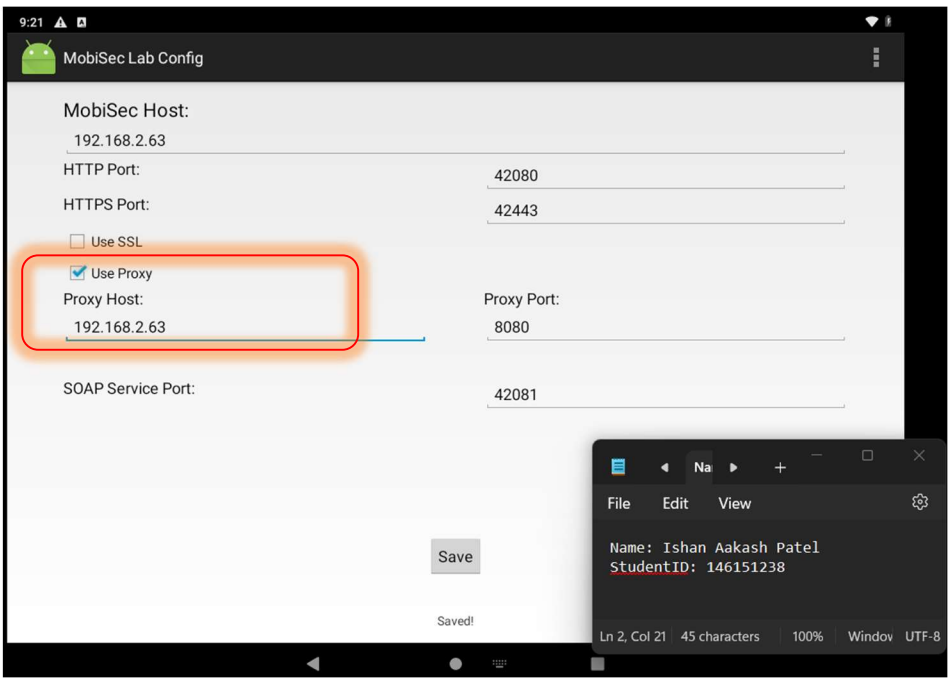
Account Setup Done.



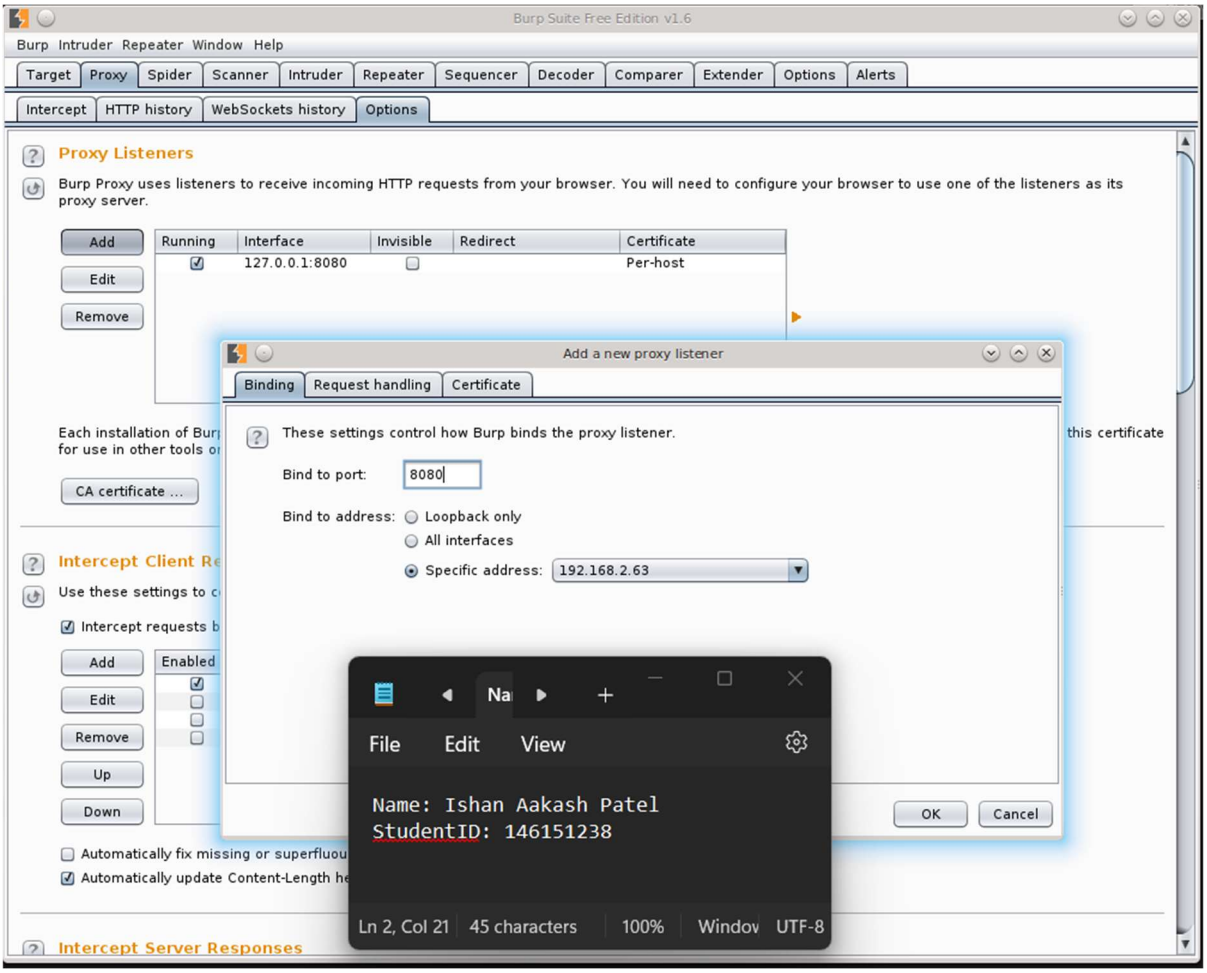
Step 6: Now we have to capture the traffic of android x86 from MobiSec Burp.



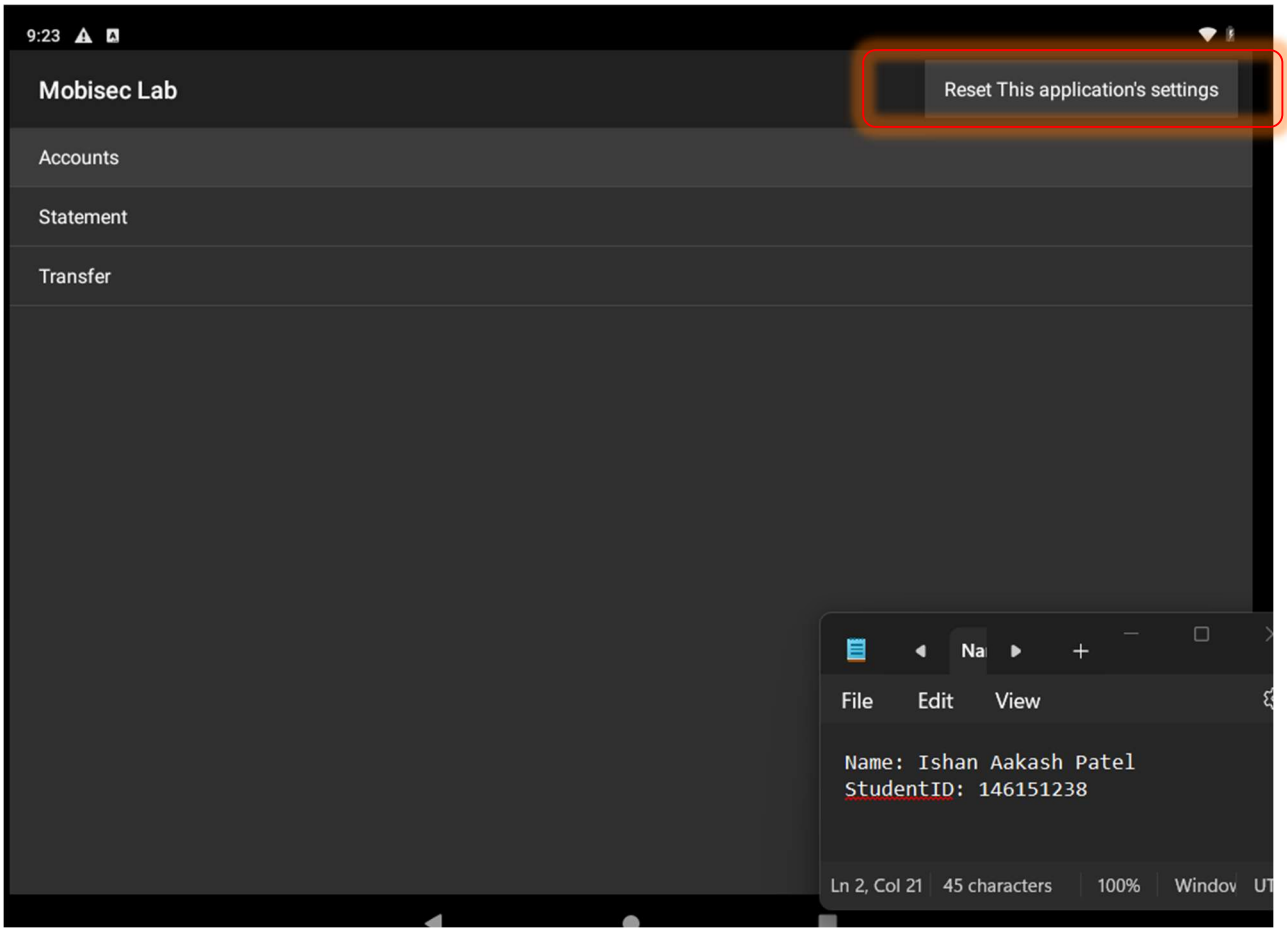
Add the proxy to lab config.



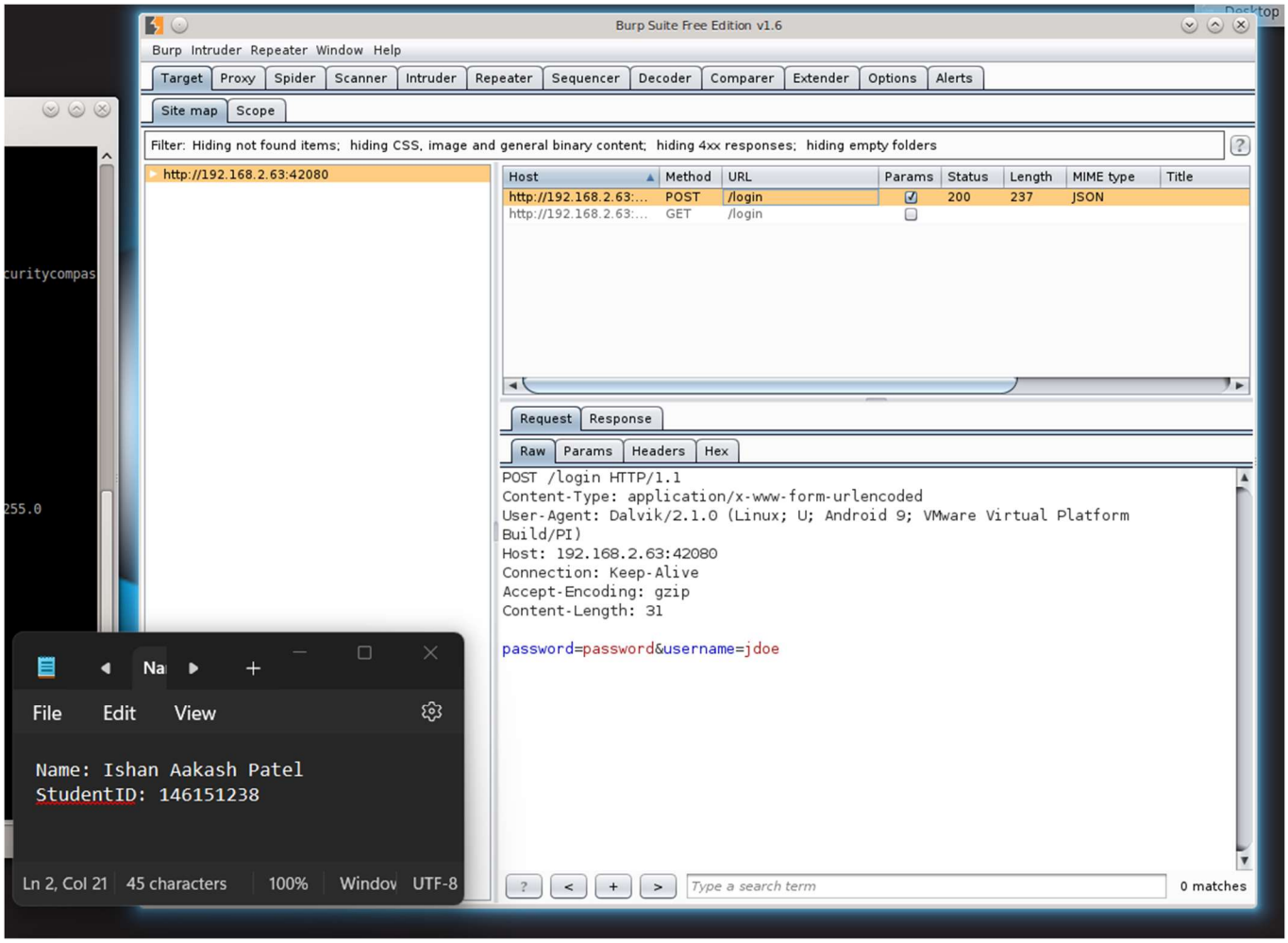
Set the burpsuite.

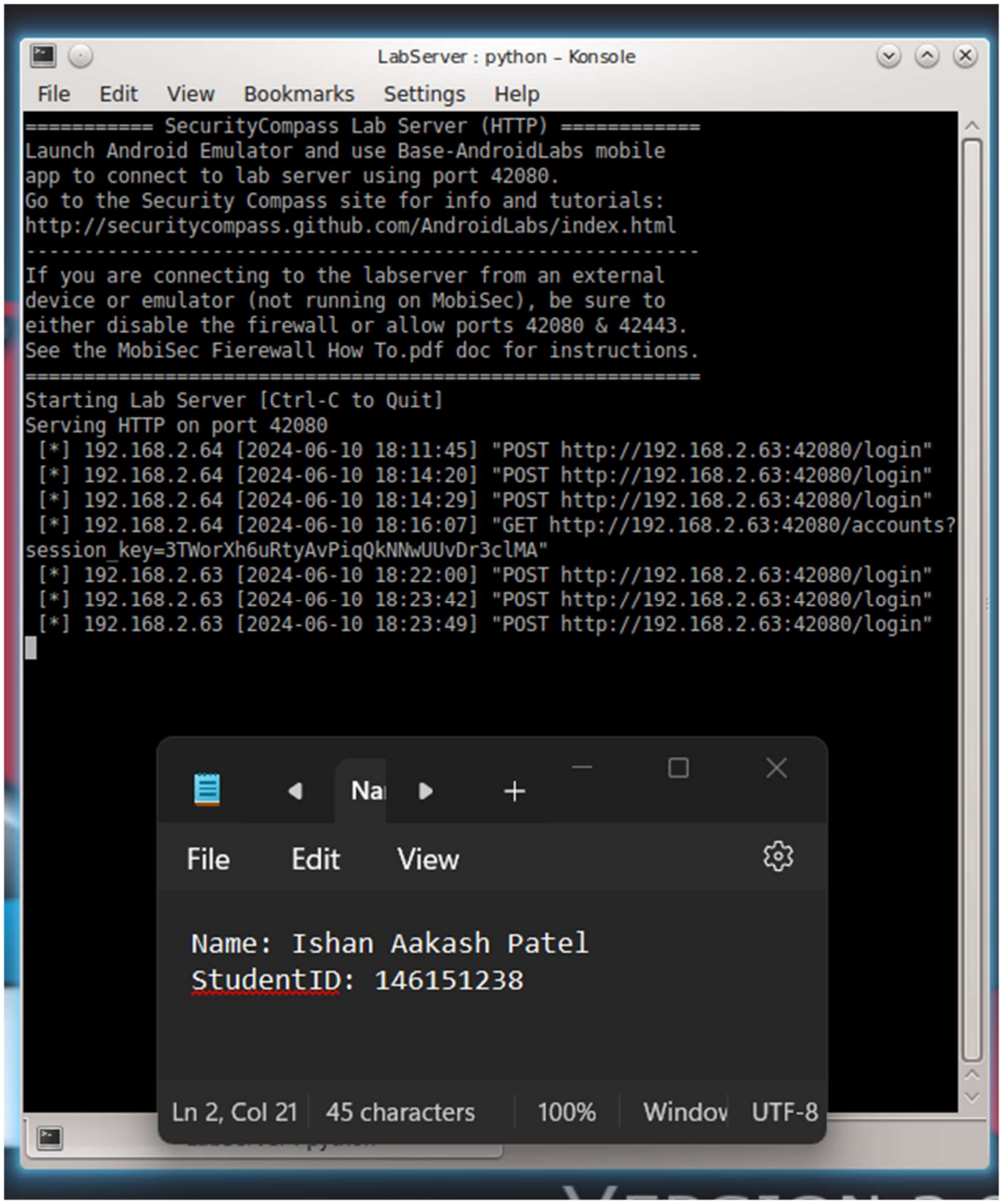


Step 7: Now testing time – Reset the account (lab.apk) and then set up the account to detect some activity.



After setting up the account again. Check below




LabServer : python – Konsole
File Edit View Bookmarks Settings Help
========== SecurityCompass Lab Server (HTTP) ============
Launch Android Emulator and use Base-AndroidLabs mobile
app to connect to lab server using port 42080.
Go to the Security Compass site for info and tutorials:
http://securitycompass.github.com/AndroidLabs/index.html
If you are connecting to the labserver from an external
device or emulator (not running on MobiSec), be sure to
either disable the firewall or allow ports 42080 & 42443.
See the MobiSec Fierewall How To.pdf doc for instructions.
Starting Lab Server [Ctrl-C to Quit]
Serving HTTP on port 42080
[*] 192.168.2.64 [2024-06-10 18:11:45] "POST http://192.168.2.63:42080/login"
[*] 192.168.2.64 [2024-06-10 18:14:20] "POST http://192.168.2.63:42080/login"
[*] 192.168.2.64 [2024-06-10 18:14:29] "POST http://192.168.2.63:42080/login"
[*] 192.168.2.64 [2024-06-10 18:16:07] "GET http://192.168.2.63:42080/accounts?
session_key=3TWorXh6uRtyAvPiqQkNNwUUvDr3clMA"
[*] 192.168.2.63 [2024-06-10 18:22:00] "POST http://192.168.2.63:42080/login"
[*] 192.168.2.63 [2024-06-10 18:23:42] "POST http://192.168.2.63:42080/login"
[*] 192.168.2.63 [2024-06-10 18:23:49] "POST http://192.168.2.63:42080/login"
File Edit View
Name: Ishan Aakash Patel
StudentID: 146151238
Ln 2, Col 21 45 characters 100% UTF-8

Thank you…