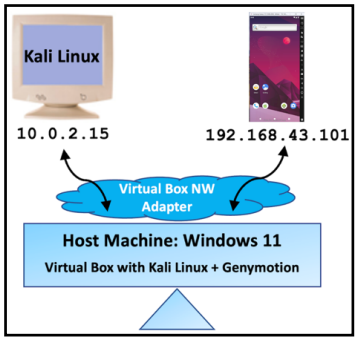
Faculty of Computing and Information Technology
University of the Punjab
(Term Spring 25)

CC-403 Information Security Student ID:_____

Class Activity 04

Marks: 70



This activity is about **Pen-Testing, Reverse Engineering, Malware Development and Malware Analysis of Android Apps** (related HO # 2.12 to 2.15). You should attempt the following tasks on your own laptops in the lab virtual environment using an appropriate `AVD` inside `Genymotion` and connecting to it via `adb` running on Kali. For reversing and static analysis of `apk` files you can use tools like `apktool`, `jadx`, `dex2jar`, `jd-gui`, `keytool`, `apksigner`, `MobSF`, `strings` and so on. For dynamic analysis you can use utilities like `Wireshark`, `BurpSuite`, `Frida`, `Drozer` and so on. TAs of all the sections will announce the RV, day, and time of viva after coordinating with the CRs of their respective sections. Happy Learning to all.

Task 01: Android App Development: [15]

Build a basic Login app having a single screen (good colourful and professional looking UI) that prompts the user to enter username and password and displays an appropriate Success/Failure message. You may hardcode the credentials inside the strings.xml file. You are required to test your app inside Android Studio Emulator, inside a Virtual Device on Genymotion as well as on your personal physical Android device. Once done try to publish your app on Google Play Store. **(No hand-written report for this task is required)**

Task 02: Android DIVA App Pentesting: [15]

Run the Android app `DIVA` on `AVD` using `Genymotion` and connect your Kali machine to `AVD` using `adb`. Decompile and load the source files of all the thirteen activities of DIVA using `jd-gui`. TAs can ask you to run any activity and ask questions to check your understanding of related concepts. **(No hand-written report for this task is required)**

Task 03: Android Malware Development: [15]

The objective of this task is to turn a benign app (e.g., a calculator) into a trojanized app with a Meterpreter backdoor. Download a legitimate Android app of your choice from `https://apkpure.com` or `https://f-droid.org`. Now generate a Metasploit payload using `msfvenom`, decompile both `apks`, and then merge payload into legitimate app. You also have to update the `AndroidManifest.xml` file. Now rebuild, resign the apk. Finally test the trojanized app on your AVD using Genymotion and see if you get a `meterpreter` session with elevated privileges. Use `ngrok` to get full credit **(No hand-written report for this task is required)**

Task 04: Reverse Engineering & Static Malware Analysis of Android `apk`: [25]

Download `apk` on your Kali machine from `github` using the following command:

```
$ wget https://github.com/ashishb/android-malware/blob/master/rouge_skype/skype.apk
```

Once you have downloaded, you need to perform the following mentioned steps using appropriate tools. **Need to submit a hand-written report for all the following sub-tasks, the commands you have used, and your analysis of the output.** You must submit this report to the TAs on the day of your viva-voce exam, and be ready to answer their questions and run the necessary commands on your laptops to get evaluated.

- **Initial `apk` Inspection:** Unzip `skype.apk` (which is a `JAR` file) in a directory with your Rollno (all lower case). Check the contents of that directory and write down your observations on the files and sub-directories that are there at the first level and their contents (whether binary or ASCII). Now use `apktool` to decompile all the assets and resources. Give a bird's eye view to all the directory contents specially understand contents of `AndroidManifest.xml` file, contents of the `smali/` directory, and the `res/` sub-directories. Write down all the commands that you used to perform these tasks and submit a report on the output and your observations. **[5]**
- **DEX to Java Decompilation:** Convert the `classes.dex` file to `.jar` file and then view all the `.class` files using `jd-gui`. Here you will see a whole bunch of obfuscated names of files and directories. When you see the code in these files, you may feel that this is not actually an Android app as the code is doing lot of other unnecessary stuff (Mention that in your report). View the contents of all the sub-directories and `.java` files, understand what the code is doing at an abstract level, and finally write down a detailed report on your observations. **[5]**

- **Analyzing Embedded Assets:** Now view the contents of the `assets/` directory, where again you will see lot of obfuscated file names. Check the type of these files using the `file` tool and you will see that some of these are java archive files, some are 32 bit dynamically linked `elf` files, some are binary data files, and so on. Start by decompiling the jar file(s) using `apktool` and when you view its contents you will feel that there exists another complete app within the original app having its own `AndroidManifest.xml` file, containing whole lot of user permissions, activities, services, broadcast receiver and other components. Do check out the contents of the `res/` directory of this inner app, and along with other items do check out the contents of `res/values/strings.xml` file. Carry out a detailed analysis of this sub-task and write down a detailed report on your observations. **[5]**
- **Nested App Code Analysis:** Now it is time to view the source code of this inner app. For this you need to unzip the jar file of this inner app, then use `dex2jar` to convert the `classes.dex` file to `classes-dex2jar.jar` file and then view all the `.class` files using `jd-gui`. Carry out a detailed analysis of this sub-task and write down a detailed report on your observations. **[5]**
- **Online Analysis:** Use the online `MobSF` and `VirusTotal` service, load the `skype.apk` and generate its static analysis report. From the two reports you need to write down your own report covering the important aspects with a brief description of each. **[5]**