Android Security Management and Vulnerability Detection Application

CSCI.630.01-02

Vignesh Nair
vn3246@rit.edu

Meghana Sathish
mxs7620@rit.edu

Saurabh Gandhele
smg6512@rit.edu

The intention of our application is to make an expert system which will analyze the security features of a user's Android system and accordingly give it ratings. This will help a user to understand the risk of his android device in terms of security and then he can take effective measures to augment his security. Our Project is implemented with the help of Android studio which is the official integrated development environment for the Android system. In order to proceed with our project, we go through the following steps:

1. Design rules within the application.
2. Implement logic for forward or backward chaining.
3. Design the UI of the app.
4. Compile, build and deploy the application.

Programming languages like Java, C++ and Kotlin can be used to develop Android apps. We have used Java in our application. Android package, APK is an archive file which has the suffix .apk that contains all the necessary contents of an Android app and it is the file that is used by the Android devices to install the app. To compile the code with all the data and resource files into an APK, Android SDK is used.

Android apps are protected in a security sandbox which has the following features:

- Each app is a different user in the android operating system which is a multi-user Linux system.
- Every app in the system is assigned a unique Linux user ID by default. This ID is only known to the system and not the app.
- System sets the permissions for the app is that the access is restricted.
- An app in the system runs in isolation to other apps as each process in the system has its own virtual machine.

- When any of the components of the app has to be started, Android system starts the process and it ends that process when the job is done.

Principle of least privilege is implemented in the Android system which ensures that by default, every app has access to only those components which is needed by the app to get the work done. This is done to make sure a very secure environment is created.

Important components in an Android app are the following:

Activities: To interact with the user, this is the starting point. It represents a user interface with a single screen.

Services: To perform operations that run for long time or to work for remote process, this component runs in the background.

Broadcast receivers: It enables the app to respond to broadcast announcements and the system to deliver the events even to the apps outside of the regular user flow.

Content providers: As the name suggests, these components manage the app data which can be stored in the file system, in a SQLite database or any other storage location that the app can access.

Android Studio is the standard official IDE that we have used in our project for Android development.
It offers many features like Gradle-based build system, emulator, code templates and GitHub integration. Each project has Android app modules, Library modules and Google App Engine modules.

Code the app: This is the first step involved in making the app. Android studio provides code templates which can be used. Modules can be added for new device. Java class must be created and user interface with themes and icons can be created.

Build and run the app: Android Studio has an Android Emulator that allows us to deploy the new projects and once the app is installed, the Instant Run option allows us to make changes to the code without building a new APK.

Configure the build: Gradle which is an advanced build toolkit is used to automate and manage the build process. Each build configuration can use it’s own set of code and resources.

Debug, test and Deploy app: Debugger is used to debug. JUnit test can be set for testing or other testing frameworks like Mockito can be used. The app is now suitable to publish to be made available to the users.

Requirements:

The intention of our application is to make an expert system which will analyze the security features of a user's Android system and accordingly give it ratings. This will help a user to understand the risk of his android device in terms of security and then he can take effective measures to augment his security. In order to deploy this app we need a system running android 6.0 or higher, a USB cable to transfer the APK file. This project helps us not only understand the security features of android, but also on how expert systems, forward chaining using propositional logic works.

Purpose: The main goal of this project is to design and develop an application that will search through the downloaded apps by a user and report on the vulnerable ones. Through this app, users can find out other applications which can potentially turn out to be malicious in future. The key idea is to find other information such as manifest, signature, version code, etc. and compare it with the ones available on Google Play Store.

Scope: Expected Outcome: A running android application which evaluates the security factors of the android device and displays the vulnerabilities of the same.

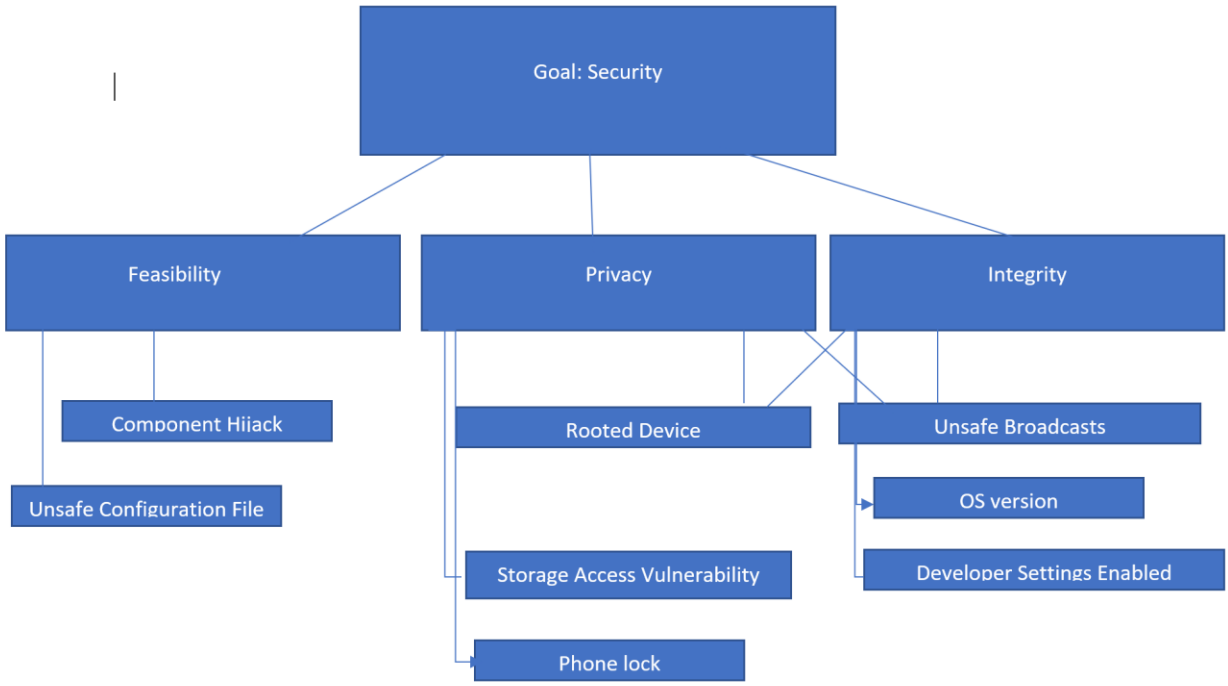
Required Framework: • Android Studio. • Android Emulator/Android Device. Knowledge Areas Needed for this project: • Android programming. • Decent knowledge of security issues on Android. Forward chaining expert system which uses propositional logic

Android Security hierarchy:

For the implementation of our project, we have formulated a hierarchy to build a well-planned expert system. Following is the hierarchy we plan to use for our expert system:

Generally, information security concerns three aspects: privacy, integrity and feasibility Privacy means that only authorized users can access to the information while other users cannot. Integrity refers to that information integrity cannot be destroyed.

Feasibility means that the system which provides information service should work normally.



- **Feasibility:**

Android security feasibility is about how accurate the data is and how well it is being applied. An app with an unverified intent has a greater chance of having its feasibility compromised. Following are the factors of feasibility:

- Component Hijack
- Unsafe Configuration File

Component Hijack:

If Activity component of an app is hijacked and if we intercept a user's password while it is being inputted in real time, we can find ways to enter the android system and change how the application information works. For example, If the app for a bank is compromised, a money transfer is issued, and the users money can get stolen and redirected while the user can be shown as if the original transaction was successful

Unsafe Configuration File:

If the configuration file of an app is not stored properly say for some reason the developer decides to store it on external storage. If the file is accessed by a malicious user, they can completely control how the contents of the app are managed.

Privacy:

It is important to preserve the privacy of the user as far as security realm is concerned. Privacy is one of the cornerstones in case of information security. All of what we do, is for personal use especially in mobile phones. This personal information, if prone to divulging can cause various problems like identity theft, credit card information being stolen and in extreme criminal cases, even blackmail. As far as the overall Android security system is concerned, there are four main factors to privacy:

- Storage access vulnerability
- Rooted Device
- Locked Device
- Unsafe Broadcast

- **Storage access Vulnerability:**

The different data storage options available on Android: [4]

1. Internal file storage
2. External file storage
3. Shared preferences
4. Databases

Android has implemented this protection of data storage which works sufficiently in most cases. But there are the following loopholes which can be used for breaching security:

1. Usage of MODE WORLD READABLE/WRITABLE which makes the files accessible to malicious apps.
2. Writing files to external storage. Which makes the files easily accessible? If these files have sensitive data, they are likely to be compromised.

3. Data may be leaked from activities and intents while they are saved as extras. These extras can be read from anywhere unless an additional permission is added. A rooted phone is highly vulnerable in this case.

- **Phone Screen Lock:**

Every Android System has an option to keep a locked phone screen. This can be of the following forms: None, Swipe, Pattern, PIN, Password, Fingerprint.

- **Unsafe broadcast:**

It enables the app to respond to broadcast announcements and the system to deliver the events even to the apps outside of the regular user flow. Broadcasts are vulnerable to passive eavesdropping and active denial of service attack. A malicious broadcast receiver intercepts an Intent or prevents it from reaching other broadcast receivers.12% of apps are vulnerable to this attack [4]. There are two ways in which an Android system has unsafe broadcast vulnerability:

1. Broadcast Theft-passive Eavesdropping

2. Broadcast Theft – DOS Attack

- **Rooted Device:**

When a device is rooted, the user is called as a superuser. This gives them many additional permissions, which basically gives full potential access to the phone. Rooting is counterproductive in many ways. While it gives you the ability to do various things like install custom ROM, experiment with the phones' hardware capabilities, etc, it also weakens basic android security principles.

Integrity:

Integrity is about if information cannot be destroyed. It is important that the application on an android system has ways to preserve information and make sure it lasts. As far as the overall Android security system is concerned, there are four main factors to Integrity:

- Rooted Device
- Developer settings enabled

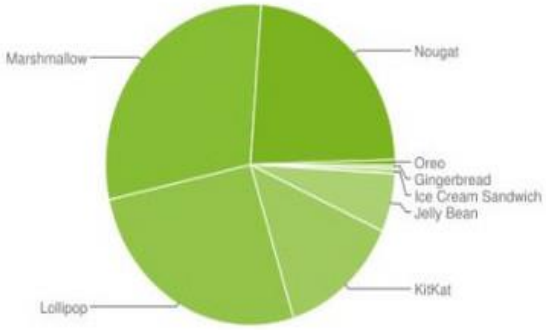
- Unsafe Broadcast
- OS version

As seen, some of these overlap with privacy. A lot of the factors do tend to overlap with each other because an android system is intricate, interconnected and complicated. So an event in a certain area will affect adjacent areas too.

- **Developer options enabled:** Developer settings are present in any app which gives access to the developer to make any changes, improve the performance of the app or even test it. It allows debugging over USB, show CPU usage and capture bug reports. By default, it is hidden. If the developer options are enabled and are accessible to others, then it can lead to vulnerability and the security can be breached.

- **OS version:** Following is the distribution of OS version of the Android system. This information is provided in the official page of android development. A system always needs to be improved and the intention of updating a system to its latest is that most of the vulnerabilities is serviced and maintained. [6]

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.0%
4.2.x		17	3.0%
4.3		18	0.9%
4.4	KitKat	19	13.4%
5.0	Lollipop	21	6.1%
5.1		22	20.2%
6.0	Marshmallow	23	29.7%
7.0	Nougat	24	19.3%
7.1		25	4.0%
8.0	Oreo	26	0.5%



Data collected during a 7-day period ending on December 11, 2017.
Any versions with less than 0.1% distribution are not shown.

Feasibility study:

Comparison to other expert systems:

Jess, CLIPS and Prolog are other expert system that support building hierarchy and rules. Since, we chose to implement an Android application, there is very little support provided by the above frameworks. Also, integrating these libraries can be painstaking in the android studio and so we avoided it. Hence, we decided to implement our own expert shell.

Our expert system: Forward Chaining Propositional based:

For our project implementation we will be using forward chaining. In order to understand forward chaining, we need to focus on propositional logic. Forward chaining methods are data driven. Forward chaining is also called as production systems. To create a working production system, we need the following:

- To create some form of the rule set.
- The current system in implemented in a running and working storage unit.
- A conclusion or an inference system that understands the rules and applies it.

The rules are of the form:

left hand side (LHS) => right hand side (RHS).

Implementation:

For the user interface, there are namely three files that represent every page/screen on the android layout. These files are namely:

1. Activity_main.xml
2. First_layout.xml
3. second_layout.xml

The activity_main file displays the opening page when the application is launched. It displays welcome msg and provides a button for the user to begin the device analysis. Once this button is clicked, the first_layout file gets called. This first_layout file displays all the System parameters with their Score depending on they

are enabled/disabled in the device. There is a green tick image displayed on the side if the system parameter has a score assigned. There is a red cross image displayed on the side if the system parameter has no score. On clicking the next button below, the second_layout file displays the total score and the security level/risk of the device. This security level is calculated from the rules supplied in the KB.txt file and the hierarchy designed for the system. There is an exit button provided below to quit the application.

We have used the forward chaining method as a reference to our application. Hence, there are eight Java files. The Main Activity java file contains the code for displaying and setting the text view values, providing on click listeners to the buttons and other function calls. The Entail java files contains code for the expert system shell that calls the forward chaining method and reads the rules provided in the KB.txt file. It parses the rules and depending upon the operator specified, other initialization classes get invoked such as And, Or, Operator and Variable. Upon calculation of the score and traversing the hierarchy, the security level gets classified and gets returned to the main activity file and displays it.

Hierarchy of the Expert System: (KB.txt file)

DM => 0
DM ^ SL => LOW
US ^ LOW => 1
DM ^ SL ^ LOW => 2
VR ^ LOW => 3
US ^ VR => MED
US ^ VR ^ MED => 4
LOW ^ MED => 5
AV ^ MED => 6
AV ^ LOW ^ MED => HIGH
AV ^ LOW ^ MED ^ HIGH => 7
DM
SL
US
VR
AV

Where,
DM = Developer Mode
SL = Screen Lock

US = Unknown Sources
VR = OS Version
AV = AntiVirus

Assigning the Score:

The following parameters are considered while assigning a score:

Developer Mode:
If the developer mode is enabled, the score assigned is 0.
If the developer mode is disabled, the score assigned is 1.

Unknown Sources:
If the apps to install from unknown sources is enabled, the score assigned is 0.
If the apps to install from unknown sources is disabled, the score assigned is 1.

Screen Lock:
If there is any kind off screen lock enabled, the score assigned is 1.
If there is no screen lock, the score assigned is 0.

OS Version:
If the OS version no found in the device is 8, the score assigned is 2.
If the OS version no found in the device is 7, the score assigned is 1.
If the OS version no found in the device is <= 6, the score assigned is 0.

Antivirus Installed:
If the device has an Anti-Virus app installed, the score assigned is 2.
If the device does not have an Anti-Virus app installed, the score assigned is 0.

Depending on the score, the Expert System shell on receiving it tries to classify the score according to the Security level. The Rules which are been fed to the expert system will try to match the score and follow the hierarchy. The hierarchy will then be evaluated to return the level of security. For example, if the score found out in the device is 7, the last rule from the file will be traversed which is : **AV ^ LOW ^ MED ^ HIGH => 7**. Then the rule for HIGH will be traversed which is : **AV ^ LOW ^ MED => HIGH.** Then the rule for MED will be traversed which is : **US ^ VR => MED.** Then the rule for LOW will be traversed which is : **DM ^ SL => LOW.** Thus, the score of 7 classifies to a HIGH level of security in the device.

Conclusion & Results:

The expert system which we have implemented gives a good evaluation about the security level of the android device. Users can thus test the security on their phone and get an estimation about the risk and potential threat to their device. Based on the five system parameters that we selected, some users did change their settings in the device to boost their score and hence make it high secure. Also, a score on the level of 0 - 7 is a good measuring attribute which provides a hierarchy in classifying it as low, medium and high depending on the rules provided.

Below are displayed some sample result screenshots of the android application.

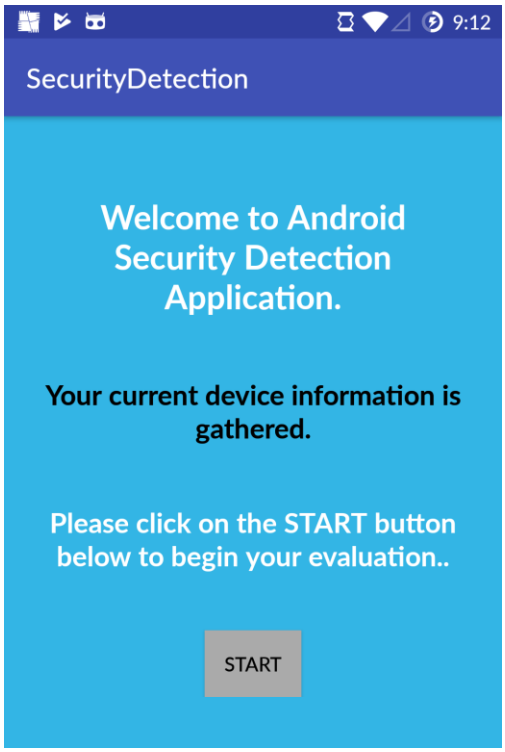


Fig. 1

Figure 1 is displaying the initial screen of the application when it is launched. It displays the welcome message and asks user to press Start button to begin evaluation. Once the Start button is pressed, user is directed to the next screen.

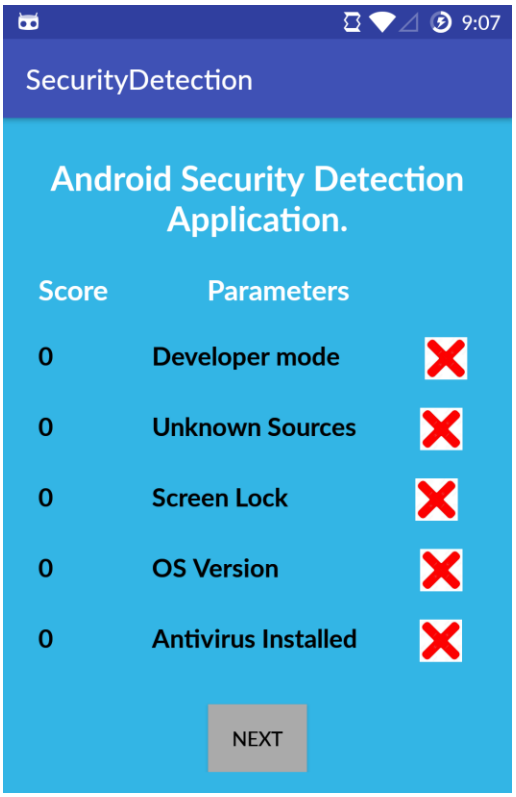


Fig. 2

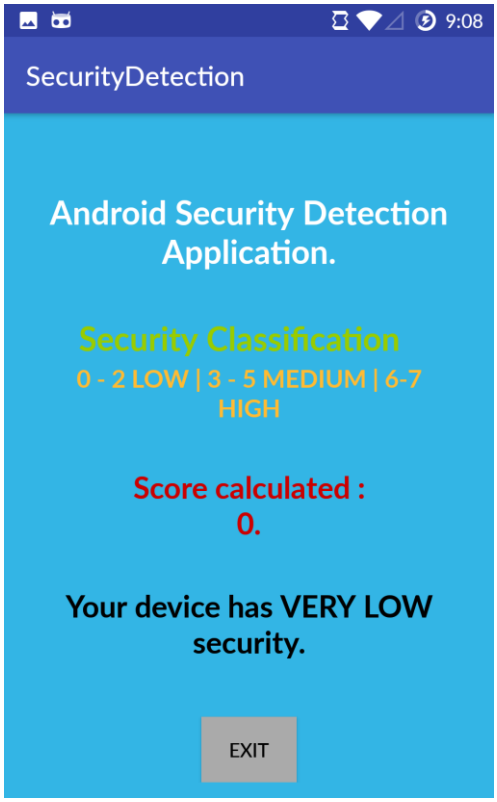


Fig. 3

This is a sample evaluation obtained that is displayed on the second screen. Figure 2 is displaying all the system parameters with their score. The red cross image on every parameter indicates a score 0 assigned to it. The next button is for the classification of the score obtained. On pressing the Next button, user will be directed to the third page which is in the Figure 3. The overall score was passed to the expert system and the rules classified the score as low from the text file supplied. The message is displayed, and the analysis is complete. User can press the exit button to exit the application and may want to relaunch the application after changing the parameters.

Fig. 4

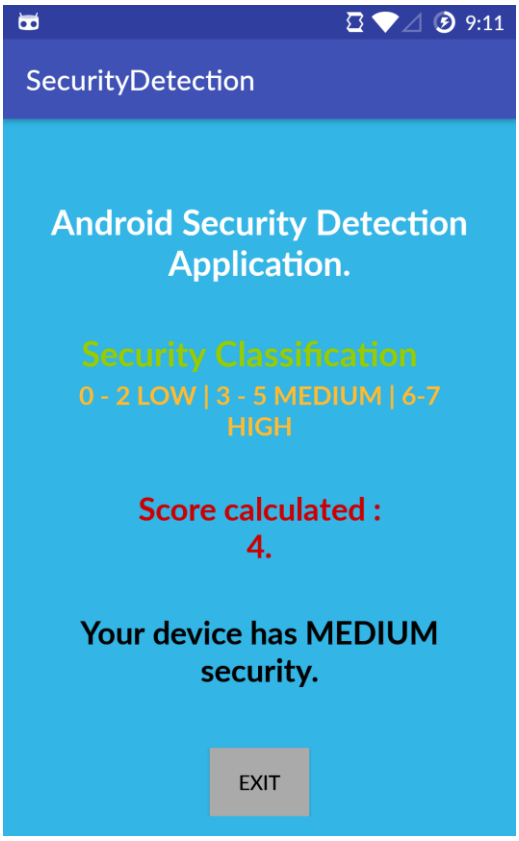


Fig. 5

Figure 4 and 5 are likewise the previous results mentioned above. In this case, the user has changed the settings. Unknown sources is disabled, Screen Lock is activated and Anti-Virus application is installed. Thus, there is a green check mark for every score > 0. Hence, on evaluating the above parameters, a score of 4 was calculated and passed to the expert system shell. The shell on reading the score and parsing the rules specified classified the level of security as medium. Hence, the message is displayed for the device having medium security.

Fig. 6

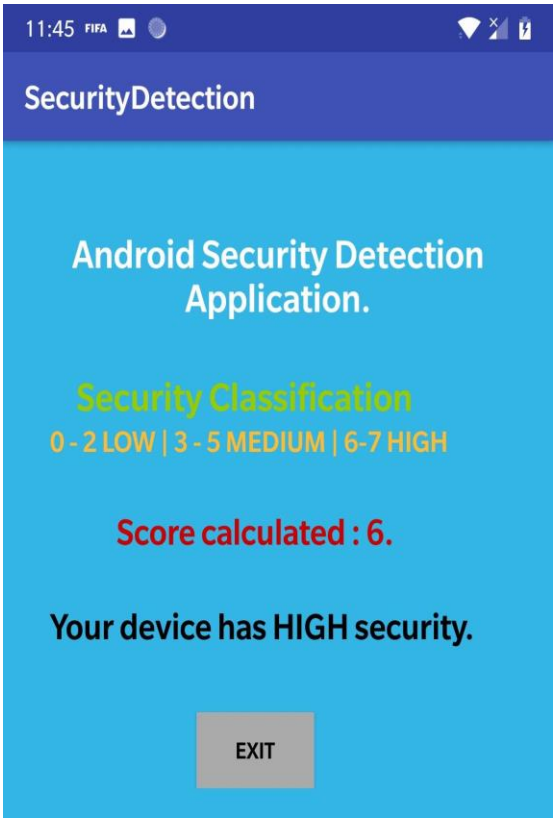


Fig. 7

Figure 6 and 7 are likewise the figures displayed above. In this scenario, the device has an Anti-Virus installed, most updated version of OS, activated screen lock, unknown sources disabled. Only the developer mode is enabled and hence a score of 0 for it with the red cross symbol. The score after summing is evaluated by the expert system and the rules are applied which eventually classifies the device as highly secure.

User Guide:

To run the Android application, there are two approaches:

A) Using the APK file.

1) Unzip the submitted zip file.
2) Locate the app-debug.apk file.
3) Install this file on your android device.
4) After successful installation of the application, open/run it to evaluate the security of the device.

B) Import the Project into Android Studio.

1) Unzip the submitted zip file.
2) Locate the folder/zip file VDA.
3) Map this entire project onto your Android Studio using the import project option.
4) After importing, clean and rebuild the project to make sure there are no errors in the gradle build and dependencies.
5) Run the application either using a virtual device, or by USB debugging your android phone.

Testing

Once the android application was developed, unit testing was carried out to make sure that the SYSTEM parameters generated, and RULES provided function correctly. Parameters such as Developer Mode, Screen Lock, Unknown Sources were tested by manually enabling/disabling the properties in the Android phone. OS version was tested on different devices because there is no way to manually edit this property on a device. Phones that supported Android Version 8, 7 and 6 were tested to ensure the OS Version attribute. The Antivirus feature was tested by installing an antivirus application from the google play store (Norton, AVG, Avast, etc).

Apart from unit testing, System and Integration testing was also performed to make sure the application does not show erratic behavior and does not crash. For this reason, the application was tested on one of more devices of popular mobile brands such as Samsung, Motorola, LG and OnePlus. For every test carried out, the device correctly calculated the score and classified the security as per the rules provided. Also, the application was thoroughly tested on the Android Studio Emulators and Virtual devices. Typically, NEXUS 5X API.

Development Process:

The following was handled by the team members in our project:

Saurabh Gandhele:

- Scope Definition
- Phase 1 literature review of 3 papers.
- Layout of the security system.
- Debugging Entail.java
- Coding the base UI, Compilation and testing
- Knowledge base rules decision making.

- Pipelining the project.
- Implementation of low-med-high linguistic variables for the expert system

Vignesh Nair:

- Scope Definition
- Phase 1 literature review of 3 papers.
- Defining system of Android Hierarchy for the expert system.
- Debugging Entail.java
- Knowledge base rules decision making.
- Phase 2 literature review of 3 papers.

Meghana Satish:

- Phase 1 literature review of 2 papers.
- Compile and testing.
- Android Security research.
- UI design formulation.
- Phase 2 literature review of 1 paper.

References:

[1] Andrew Hoffman, Darren Pollard and Leon Reznik, “Hierarchical Security Evaluation Framework and its Implementation on Android Smartphones”, pp 1-6.

[2] L. Reznik, "Integral instrumentation data quality evaluation: The way to enhance safety, security, and environment impact," *2012 IEEE International Instrumentation and Measurement Technology Conference Proceedings*, Graz, 2012, pp. 2138-2143.
doi: 10.1109/I2MTC.2012.6229325.

[3] I. Khokhlov and L. Reznik, "Data security evaluation for mobile android devices," *2017 20th Conference of Open Innovations Association (FRUCT)*, St. Petersburg, 2017, pp. 154-160.
doi: 10.23919/FRUCT.2017.8071306.

[4] Faysal Hossain Shezan , Syeda Farzia Afroze , Anindya Iqbal, Department of Computer Science and Engineering Bangladesh University of Engineering and Technology ”Vulnerability Detection in Recent Android Apps: An Empirical Study”, 2017 International Conference on Networking, Systems and Security (NSysS)

[5] Wan Yang, Wang Guolin, Feng Xiangyang, “An Evaluation Model for Information Security of Android Application Based on Analytic Hierarchy Process”

[6] ”Distribution Dashboard” https://developer.android.com/about/dashboards/.

[7] Expert System Shell for Java files.
“https://github.com/bennapp/forwardBackwardChaining”.