Project 2 : Device Security Analyser

I. Executive Summary:

Understanding security for smartphones is challenging. Although almost everyone uses these devices, their internal processes are known by only a handful of people. Hence it becomes very important to differentiate the secure phones from the insecure ones. The goal of this project is to design a simple yet effective way to design a system that identifies the security risks associated with the smartphone.
We have solved this problem using a hierarchical expert system. The current details of the smartphone hardware and software are recorded. They are fed into the expert system. The resultant of all this is a final score out of then that can explain the security status of the phone.

II. Requirements:

The requirement of this project was threefold. The first one was to understand the target system. Record the salient features that make or break the security of the system. The second part was formulating rules from these features. For example, A and B allow us to infer that C exists.
Inferences from such rules can be combined with other inferences to make up more rules. The last part was running these rules in an expert system shell and getting a final result of security score for the system. The major intention of using rules as opposed to hard-coded loops was ease of future updates

III. Specification:

Additional details about individual parts are specified here:

i. Read current settings in the system:

In this case, the system is a typical Android Phone. A typical Android phone has a lot of moving parts. We have to focus only on the parts relevant to security. Typically this includes Root Status of the phone, the permissions that apps access, list of the apps that are sideloaded, developer status of the phone etc. All this information needs to be recorded in real time.

ii. Make Rules from recorded details

The task here is to infer security information from the recorded data. All this recorded information is termed as the knowledge base. All of this data cannot be processed together in one iteration. Hence we must group information according to its relevance.
So a typical rule would be “If Airplane Mode on and WIFI off, then network security ensured”. This basically means it is close to impossible to breach network security if Airplane Mode is on

and WiFi is off. So here we inferring that Network Security is ensured because the conditions to achieve it are satisfied.
So directly recorded information can be used to infer new information. This inferred information can itself be used in new rules. As we use the results of previous rules, this expert system would be termed hierarchical. As new information is inferred, we update our knowledge base. As knowledge base is updated additional rules can run.

iii. Use Expert System to run Rules

The rules can be implemented using a series of IF-ELSE loops. The code would be fast, would work well and be the results would be verifiable. But now the rules are locked onto the system and any change would mean a change to the code. Hence we need to separate rules and implementation.
The tool used for such a task would be an Expert System Shell. Such a shell would accept the rules as a separate file. The shell would have access to the knowledge base as well. So the shell will run the rules as per the knowledge base and infer new information. This information would be updated in the knowledge base. In the end, the Expert System will give a final score to 'judge' the security of the Android phone.

IV. Description of the Domain problem:

We are using an expert shell which had been mentioned in the project description, which can be found here:
https://github.com/bennapp/forwardBackwardChaining
We preferred to use this over other present Expert System shells as this was written in Java, in comparison to the others written in C(CLIPS), Python(PyCLIPS), or even JESS(even though it is written in Java, there is no support for Android).
The expert shell is configured such that we accept rules in a file. The same file also contains the latest iteration of our knowledge base. Based on the availability of data some rules may be fired. As these initial rules are fired, additional data is generated by inference for the knowledge base. This additional data can enable more rules to run.
Since we are facts to infer new facts, and do not have a goal in mind, we chose to use forward chaining over backward chaining to reach the score.
We have tweaked the existing system to present a score at the end. The score is a percentage value which would symbolize how secure the current security of the phone is.
As our final tool is effectively an Android app, this shell is added as a library to the final app code. We pass the store the rules in a fixed static file and we make a new file on the fly that contains the rules and knowledge. This temporary file is used as an input for this expert shell.

V. Feasibility Study

There are other ways to achieve the solution we have achieved. The current version of the project is an Expert System which used Ben App as an Expert Shell. The values that are fed into the system are binary all the time. Their design decisions will be debated here:

i. Expert System vs Regular System

The end result of evaluating the security of an Android phone can ideally be done without an Expert System. But the results of such a test cannot be trusted. After a representation of the knowledge base is built, then the system will move forward for getting a score. But this score cannot be considered valid as incoherent elements as merged together to create a single score. So for example, the root status of a phone is coupled with WIFI toggle status and Permissions Data. Assuming that such disparate elements can work well together would be a mistake.
Compared to this the approach presented by Expert System is much more rigorous and thorough. In such a situation, WIFI status will be paired with Airplane Mode Status to give us an overall Network Security Picture. App permissions and status of sideloaded apps can give us information on App Security. Then later Network Security and App Security can be combined. So this approach pairs and groups aspects of the phone at their relevant levels.

ii. JESS vs Clips vs Expert Shell used by us

We had decided to make an Android app. Unfortunately, JESS is not compatible with Android. So we had to rule out JESS to meet our project specifications.
Clips is compatible with Android, but the system is designed in C. This prevents it from directly interacting with JAVA Android code. There is an option to use Clips with Android Native Development Kit(NDK). But that would require gaining additional programming skills. Given the timeframe of this project, it was best to stick with the currently implemented expert system shell.This shell is written in Java and is compatible with Android, which made it a very good reason for it to be used.
Speaking in terms of speed, Clips given its C base would probably be the fastest. But as our Android app isn't so CPU intensive, there was no real need to go in that direction.

iii. Binary Score vs Gradient Score

Gradient Score is something that makes more organic sense. For example, Android Phone with PassPhrase Security is more secure than a phone with Pattern Lock. So technically both types have lock security but are not necessarily the same. Hence we wrote most of our initial code to return decimal values.
But this expert system shell is not tuned for non-Boolean inputs. It is technically possible to have multiple values for a variable but is not ideal. Hence we changed most of our system to assign boolean values for variables.
iv. Machine Learning vs Expert System

It is possible to use Machine Learning to predict the potential security risks for a system. Data can be used to build a model that learns from previous security failures and tries to overcome those issues in real time. Such a system would work best in a real world scenario. This whole process can run independent of the end user who never has to worry about security.
Acquiring the right data will be challenging. And checking the system for security risks at all time will also be expensive. And furthermore security risks keep changing. Then the cost to retrain the model will be really high.
Comparatively Expert Systems are less nuanced and more stable. Updates can also be made rather easily.

VI. Implementation

For the purposes of this project, the rules database and the knowledge base are housed together in a file for input to the Expert Shell. It a prerequisite for the expert shell that was designed out of comfort for practical use cases. The the rule file along with knowledge base in specified below:

```
WiFi_Off ^ Airplane => NWSEC
Lock => DEVSEC
OS_Diff ^ Root_Off ^ Encryption => SWSEC
Unknown_sources_absent ^ Permission => APPSEC
NWSEC => 1
DEVSEC => 1
SWSEC => 1
APPSEC => 1
SWSEC ^ APPSEC => 2
DEVSEC ^ APPSEC => 2
DEVSEC ^ SWSEC => 2
NWSEC ^ APPSEC => 2
NWSEC ^ SWSEC => 2
NWSEC ^ DEVSEC => 2
NWSEC ^ DEVSEC ^ SWSEC => 3
NWSEC ^ DEVSEC ^ APPSEC => 3
NWSEC ^ SWSEC ^ APPSEC => 3
DEVSEC ^ SWSEC ^ APPSEC => 3
NWSEC ^ DEVSEC ^ SWSEC ^ APPSEC => 4
```

WiFi_Off
Airplane
Lock
OS_Diff
Root_Off
Encryption
Unknown_sources_absent
Permission

The app doesn't populate the values for `NWSEC, APPSEC, DEVSEC, SWSEC`. As the rules are fired, those values may be become TRUE. By default the Expert System assumes them to be FALSE.

Description of the expert system applications:

As is very evident from the text until now, our app heavily relies on Expert System to solve the problem. Apart from our app there are a lot more instances where Expert system is great at solving problem. Some other applications of expert systems are:

a. Schedule Manager:

Conventionally there a lot of constraints for schedule of school timetable or airplane timetable. Such constraints can be converted into rules. Such a expert system will give the result by considering all rules and can help save a lot of time.

b. Stock Investment Guides:

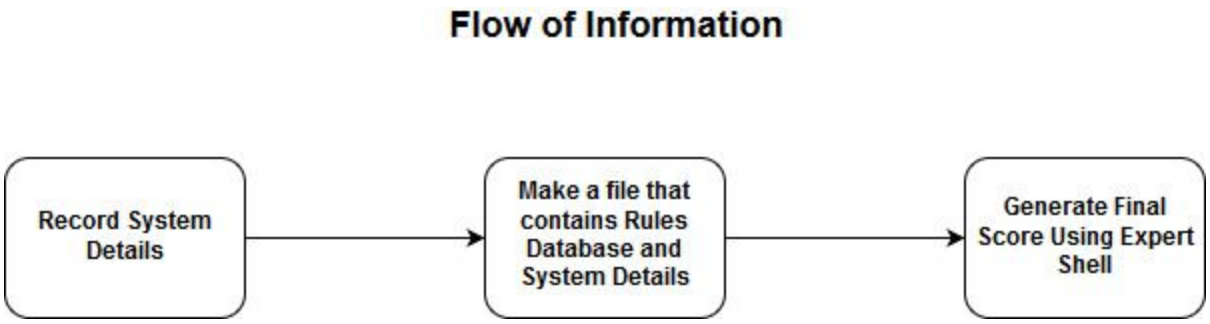
There are certain rules that always govern the performance of a particular stock. This means that we can build a system that predicts the possible rise or fall of stock. This will help the average stock investor made better decisions.

c. Alternate App Recommender:

Here we are building up on the app that we have made. We already know which apps are high risk due to their permissions. We can recommend alternate apps. If we have a database of apps, their utilities and their permissions, we can recommend safer apps.

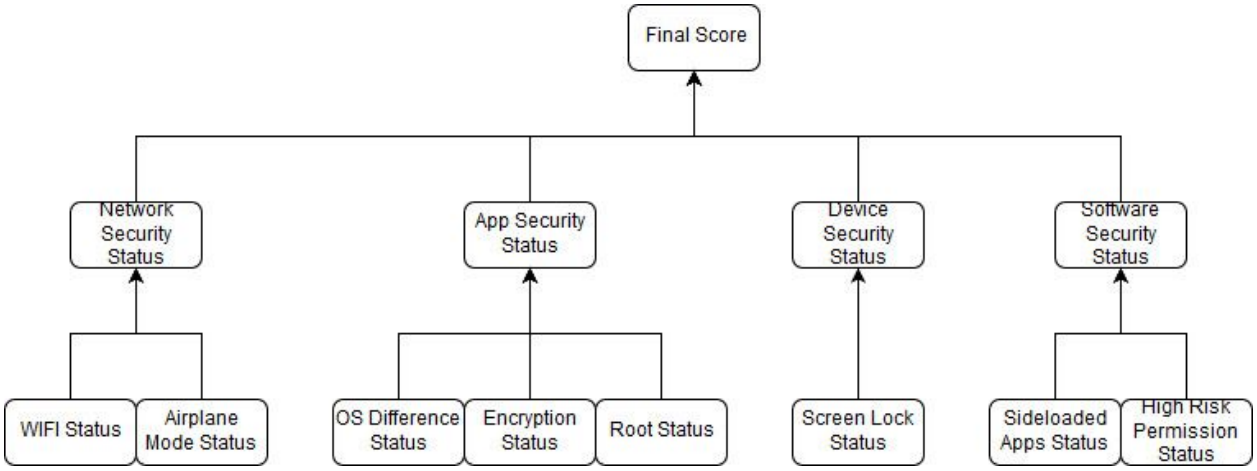
i. Structure:

The system is structured to achieve the flow of information specified below:



ii. Working of Rules:

The system is an expert based forward chaining. The build up the final security score is detailed below:



iii. Content:

System Detail	**Types**	**Meaning**
Wifi Status	On	Wifi is Off
	Off	Wifi is On
Airplane Status	On	Airplane Mode is On
	Off	Airplane Mode is Off
OS Difference	Pass	If API of Android is greater than 25
	Fail	If API of Android is lesser than or equal to 25
Encryption Status	Pass	Device is Encrypted
	Fail	Device is not Encrypted
Root	Pass	Device is not Rooted
	Fail	Device is Rooted
Screen Lock Status	Pass	Screen Lock is Enabled
	Fail	Screen Lock is not Enabled
Side Loaded Apps	Pass	Apps installed through Play Store
	Fail	Apps Installed as apk

Apps with High Risk Permissions (Permissions graded)	High	Lot of Apps with High Risk Permissions
	Low	Few Apps with High Risk Permissions

iii. Software Requirements:
The app can run only an Android system which is higher than Android(6.0) or Marshmellow.

iv. Hardware Requirements:
All modern smartphones can qualify for the test proposed by this app. No special hardware is required for this app.

v. User Interface:
The user interface is designed to pretty straightforward. User can calculate Score for their phone. After calculating the score, the user can choose to view the list of Apps that were manually installed and their respective permissions. The user can also view the current Hardware and Software System Details. Finally the user can also view tips to increase the score.

vi. Limitations:
The biggest limitation of this test would be the lack of nuance. Due to time and expert shell constraints we were not able to incorporate the nuance required to best guide the user. The app is built on rules that are designed to function as all or nothing. Unfortunately the real life scenarios are more complicated that.

Furthermore the app is designed at a very high level. If there are genuine system vulnerabilities the operating system or apps, those cannot be identified by the app. It can only detect previously identified risks.

vii. Classes:

a. Expert Shell:
   This is made up of multiple classes like And, Sentence, Variable, Operator, Entail. The rule database and knowledge base are passed to Entail class
b. InitVariables:
   The knowledge base is built here. The formatting of these values is also done here.
c. FileManipulations:
   This class makes the rule database and knowledge base file required for Expert System
d. PermissionLister:
   All user installed by user need to displayed along with their permissions. That processing happens in this class
e. Device Information:

The class picks up details of the system and formats that data for printing on a single page

VII. Testing:

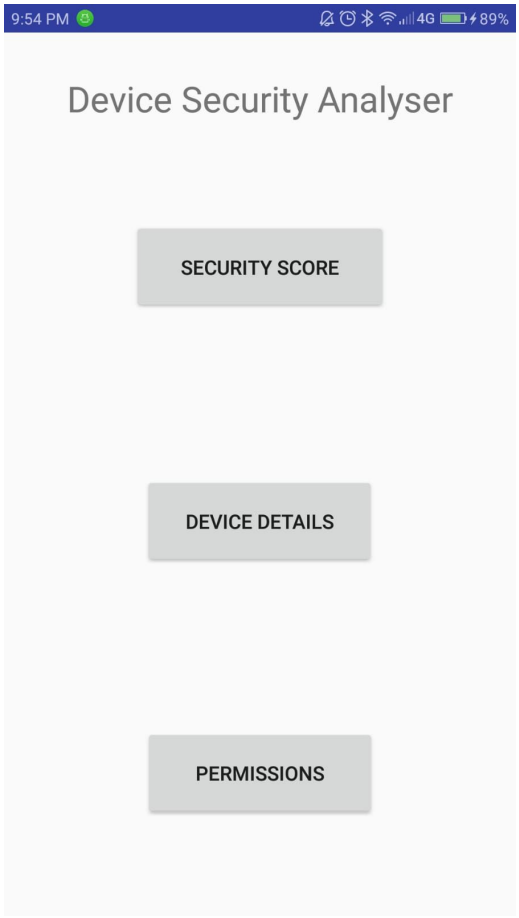
The resultant of the code is the app. So if app is functioning correctly, we can infer that the code is working correctly. So we check multiple aspects of the app using simple test cases. The test cases are details below:

Page	Test Case	Result
Home Page	Check if all Buttons are present and aligned	Pass
Home Page	Check if all Buttons are Clickable	Pass
Result Page	Check if Score is Visible	Pass
Result Page	Check if all 8 tests are displayed there	Pass
Result Page	Check if all 8 tests have relevant subtext	Pass
App-Permissions Page	Check if all App Boxes are visible and clickable	Pass
Device Details Page	Check if Device details like Name, Android version are loaded	Pass
Device Details Page	Check if Airplane mode and WiFi Status shown is correct	Pass
Device Details Page	Check if Phone Root Information is correct	Pass
Device Details Page	Check if Dev Setting Enabled Information is correct	Pass
Device Details Page	Check if change in settings is reflected	Pass

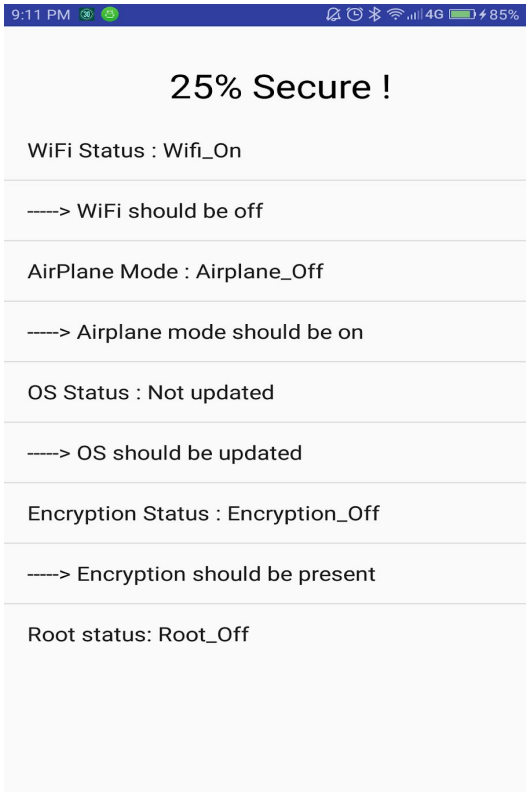
These tests were conducted on 3 Android devices. The App passed the tests on all 3 devices.

VIII. GUI

Home Page:
This is the first page that user loads the app. From here the user can Run the Security test, Get Device Information or get Installed Apps and their Permissions.

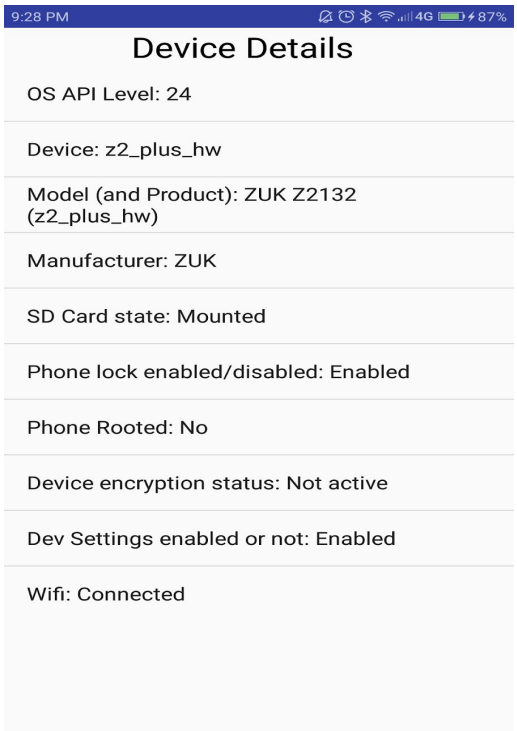


Test Run Page:
Here the security score of the current configuration is visible. This page also shows the sub-categories of security parameters and their details. The goal of this page is the provide the score and break down the score. This will allow the user to learn more about the phone and improve the security.

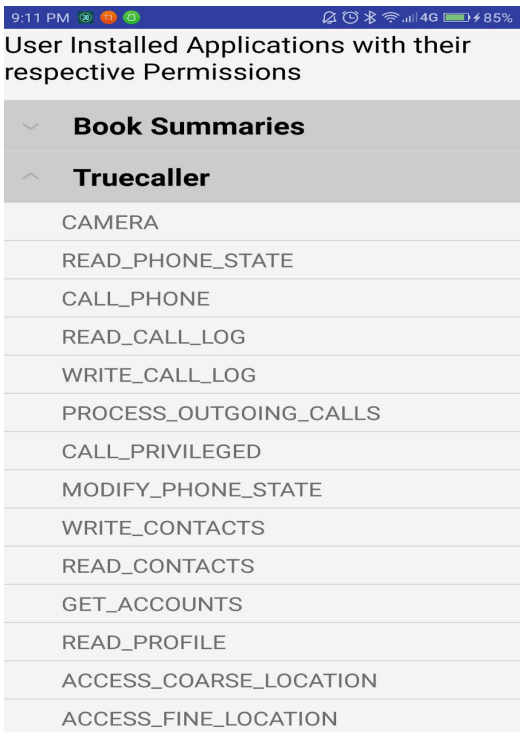


Device Details Page:

This page specifies the current configuration of the phone. This page is typically used to review the details of the phone before the test can be run again.



App Permissions Page:
This page details all the user installed apps and their relevant permissions. This will allow the user to understand which apps require what permissions. This will allow the user to gain context on which apps are to be uninstalled so that the score can be improved.



IX. Development
The team consists of two people namely Sahaj Gandhi and Pranav Rane. The tasks achieved by every member are detailed below:

Sahaj Gandhi:

- System Conceptualization: Recognizing the system requirements and designing the relevant classes for each task
- GUI Development: Development of pages, buttons
- Tweaks to Expert System: There were significant challenges in using the expert systems shell being used as is. So some changes were made to the original source code to better meet our ends.
- Class Development - Device Information
- Rule Designing
- Phase 1 - Documentation
- Testing

Pranav Rane:

- Class Development - Device Information, FileManipulations, Permissions Lister
- Rule Designing
- Phase 1 - Documentation
- Phase 2 - Documentation
- Testing

X. Code Setup

There are two ways to test the project:
i. Install Application as Apk
The instructions for setup are:

- Download the app
- Navigate to Downloads and Locate the .apk file
- Open the file and Install the application
- If Unknown Sources aren't Activated, then Activate them in the following way:
    - Go to Settings>Security
    - Check the option "Unknown Sources"
    - Tap Okay on Prompt Message
    - Select Trust
- Ensure all Permissions are selected
- Use the application, refer Section VIII for details

ii. Run code Through Android Studio's Phone Emulator

- Install Android Studio from https://developer.android.com/studio/install. Follow the instructions as per the operating system
- Open 'Android Studio'
- Unzip the Project Zip in a location of your preference
- Open Android Studio
- Load the Project Folder into Android Studio by File>Open and navigating to the folder where the project folder is located
- Project Loading may take some time
- Once the project is loaded, Run the project by Clicking the Green Play Button
- Select a Deployment Target, something with API greater than 25
- If no Virtual Devices are available, then Install one. The instructions are:
    - Choose 'Create New Virtual Device'
    - Choose the Default Phone and Select the highest level of API available
    - Keep Selecting Next Until and Select Finish in the end
    - This should install the Virtual Environment for testing
- This should run the application. The application will open automatically on the Virtual Phone
- Use the application, refer Section VIII for details