Documentation

1. Executive Summary

We address the problem of security evaluation of android devices using a Hierarchical Expert System. The system is developed using handcrafted rules. It makes use of facts admitted in the knowledge base and rules that are run using rule engine. In order to evaluate security of an android device, we calculate a security score based on 12 metrics in total. These metrics are some of the characteristics of a mobile device like its settings, locks applied, API level etc. These metrics follow a chain rule in the calculation process i.e. a combination of some metrics is given as an input to calculate to some other value which eventually is used to calculate the final security score. By implying this method, we address both hardware and software security issues in a device. This is because, some of the metrics are hardware specific while some others are derived from software settings of a device.

We use JESS rule engine for implementing this expert system. We encode all our rules and knowledge base using JESS. Whereas, the GUI of this system is developed using Java Swing class. The GUI makes it easy for a layman to input metrics of his/her android device. Based on the metric values entered in the GUI, it pops out a security score in the range of 0 to 10.

2. Requirements

Hardware requirements:

- A laptop or computer
- 4 GB RAM
- A hard disk
- Internet connection not required

Software requirements:

- Java minimum version 7
- JESS
- Terminal

3. Specification

The main idea behind this project is to learn about the rule based expert system and implement it on a real world problem. For the given problem, It is developed in the form of a desktop app that any normal person could use to evaluate the security of his android device. Once the app is installed, it asks for some metrics of an android smartphone. A user can manually enter answers to these simple questions via a GUI provided. The GUI contains radio buttons for entering values of metrics. These values entered by user are taken as input to the backend. In the backend, there exist a JESS code that contains a rule base and fires different rules based on the values given. A normalised security score is calculated in the backend and flashed on GUI.

Most of the input metrics take boolean values of either Enabled or Disabled. The desired value for some of them is enabled while it is disabled for others. Here a desired value signifies the value that would increase a device's security score. Additionally, there are few metrics which take numeric value as input. For entering values to these metrics, we have provided dropdown list in the GUI.

This system can be used as a stand alone tool to evaluate security score of handheld devices and additionally can be used to generate data for a more advanced security solution. In the later stage of this project, we are supposed to develop a machine learning based security evaluation tool. A machine learning based model requires lots of data to train and improve its accuracy. We can use this rule based expert system to generate data for building a machine learning based model.

An important feature for any app is to have forward compatibility. Which means, a system must be easily upgradeable in the future. Our expert system follows this principle and can be updated based on the newer android versions. This would help us to keep the system more robust and cohesive with the existing features on an android ecosystem.

4. Description of Domain problem

We have used JESS rule engine to execute this hierarchical expert system. It is a rule engine for the Java platform. To use it, we must specify logic in the form of rules using one of two formats: the Jess rule language or XML. We also provide some of our own data for the rules to operate on. When this rule engine is run, it carries out the rules specified. Rules can create new data, or they can do anything that the Java programming language can do. Although Jess can run as a standalone program, usually the Jess library is embedded in a Java code and manipulated using its own Java API or the basic facilities offered by the javax.rules API. A Jess language code can be developed in any text editor.

For development of the GUI for this expert system, we have used Java Swing Class. Swing is a part of JFC (Java Foundation Classes). It contains a large set of components which allow high level of customization and provide rich functionalities, and is used to create window based applications. Java swing components are lightweight, platform independent, provide powerful components like tables, scroll panels, buttons, list, color chooser, etc.

5. Feasibility study

The project implementation is pretty simple. The flow of the program is such that, the Graphical User Interface is developed in Java Swing GUI widget toolkit and methods necessary to execute the Jess Rule Engine are also developed in Java. All the rules, facts, slots and templates are developed in clp file using Jess Rule Engine which is invoked by the same Java program. The other possible alternative for this solution could be to develop Graphical User Interface using Jess Rule Engine along with rules, facts, slots and templates with the same Engine.

One of the other possible solutions would be to develop a full functioning expert system using Clips. Since Clips is quite rich in resources and it has wide documentation available on the internet, it would not be hard to develop Graphical User Interface and execute the program.

Another option would be to develop an expert system using Pyclips but unfortunately, Pyclips is not flexible and it’s quite difficult to implement.

Comparison

Jess	Clips	Pyclips
High number of features and libraries are available to use	Intermediate number of features and libraries are available to use	Low number of features and libraries are available to use
Jess rule engine has very high flexibility	Clips has high flexibility	Pyclips has low flexibility
Resource consumption of Jess is very high	Resource consumption of Clips is intermediate	Pyclips does not consume resources much

6. Implementation

a) The android device based metrics used in this expert system are as follows,

No.	*Metric*	*Possible value*
1	Installation from Unknown Sources	Enabled / Disbaled
3	Device Filesystem Encryption	Enabled / Disabled
4	Harmful applications	No. of such applications
5	Screen lock	Enabled / Disabled
6	Developer options	Enabled / Disabled
7	SIM PIN Lock	Enabled / Disabled
8	Bluetooth Discoverability	Visible / Invisible
9	ADB over USB	Enabled / Disabled
10	Android API level	1-10, based on API level
11	Google Play Protect	Enabled / Disabled
12	Device Tampering Test(Rooted Device, Bootloader Unlocked, SELinux status)	Passed / Failed

b) Expert System applications:

This expert system can be used by companies to evaluate the security of devices on their rolls. For example, a courier company which provides mobile devices to their delivery men needs a system to evaluate the security of those devices. They can use this system for the purpose of security evaluation.

c) Structure and contents:

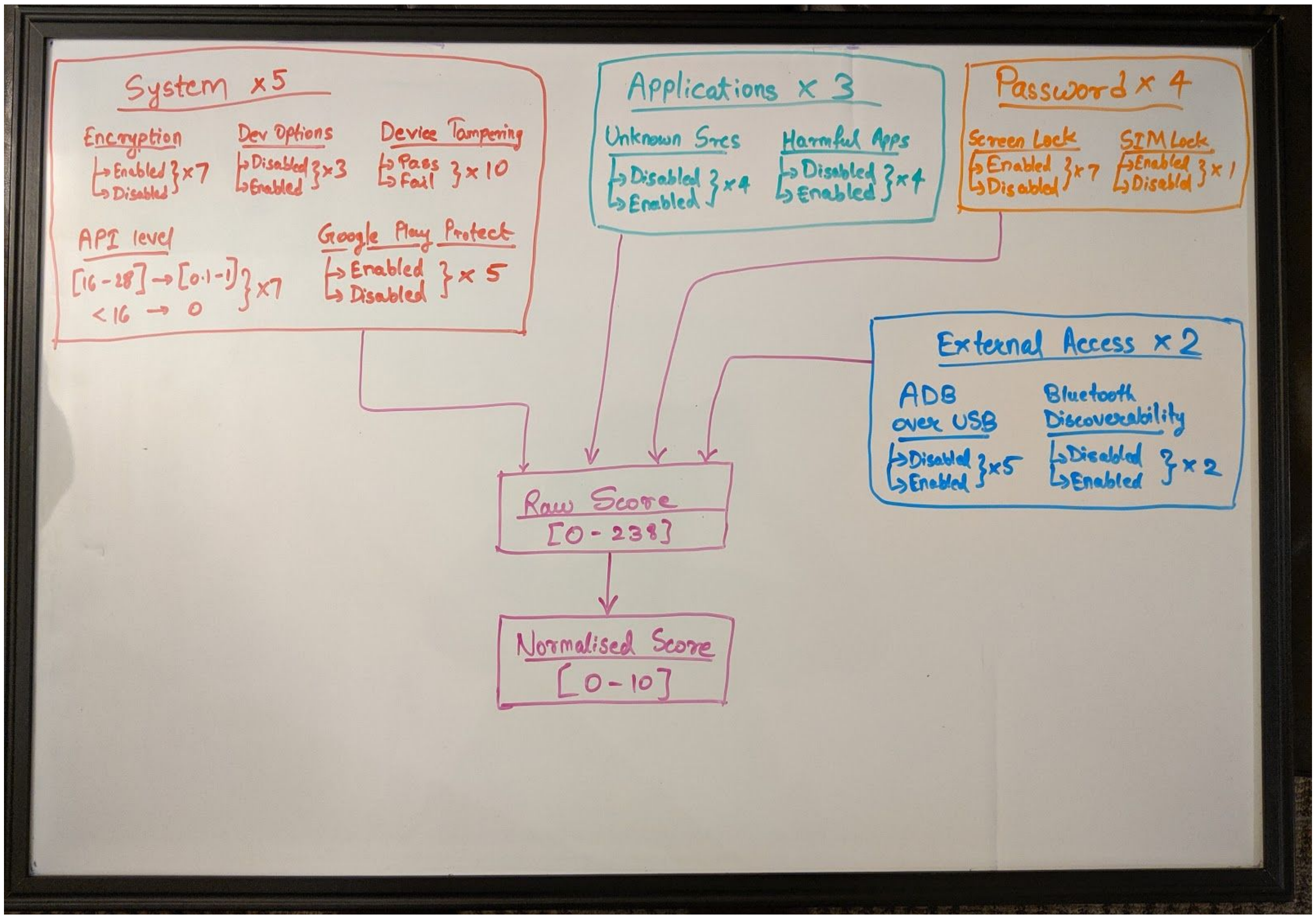


Fig. Project structure for Hierarchical Expert System

As shown in the figure above, all the metrics are divided into various clusters based on their categories. Each metric is multiplied with their respective weight and a cluster score is calculated. The cluster score is further multiplied by their weights and a summation is calculated. The weight values are assigned based on their importance and degree of vulnerability. The summation of all the cluster scores gives us a raw score. This score could be anywhere in the range of 0 to 238. This score is then normalized in the range 0 to 10. This final score evaluated is the security score of a given android device.

d) Limitations:

- Although this expert system is a good solution that considers both hardware and software perspectives of security evaluation, still has some lacunas in it. The biggest feature of this system i.e. a rule based system is itself the biggest limitation of it. Being a rule based system gives it a rigidity due to which it doesn't consider many other features of an android device.

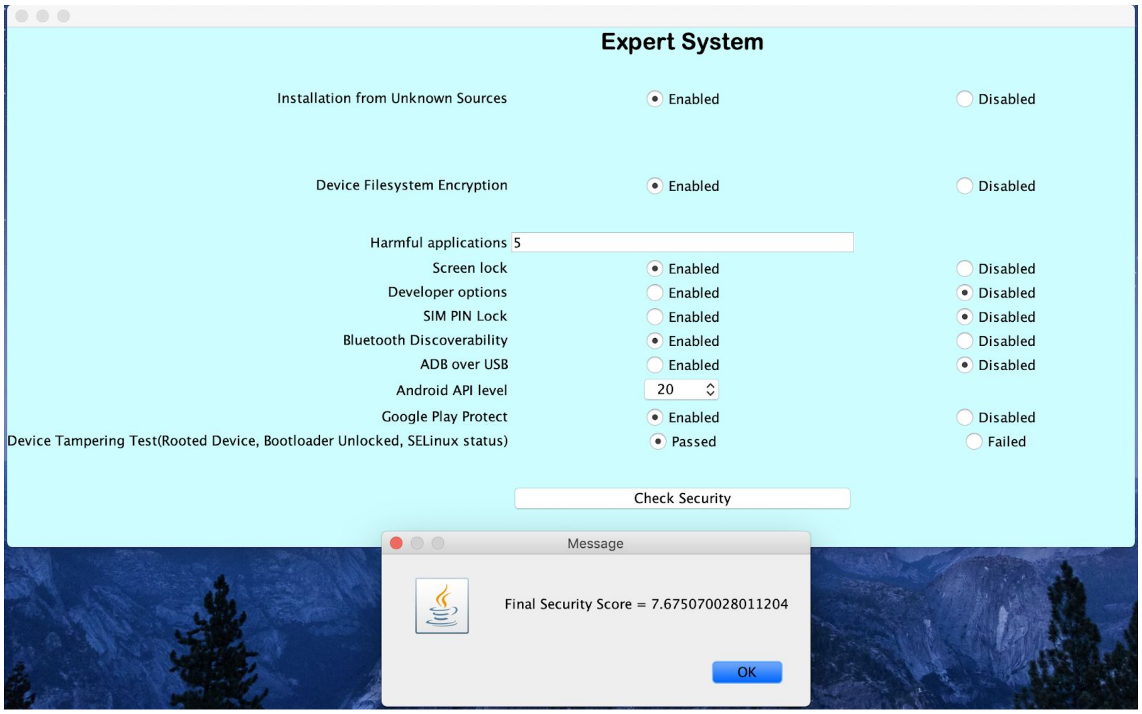
- It is not a complete security evaluation system. This is because it does not consider various web attacks on the device.
- An android app may fire unlimited GET, POST requests through its REST APIs. These requests may inform app developer about various device details.
- Although this app has a very nice workflow, it has some security limitations as well. The front end and backend connection of this expert system does not guarantee any protection against security breach. That is, a hacker may change the rule base or the way a security score is evaluated by making assembly level changes to this app’s codebase.

e) Software requirements
- Any java version
- JESS rule engine

f) Hardware requirements
- A laptop or a computer with any configuration

7. Testing Description

Following are some of the test cases that we executed:



Expert System

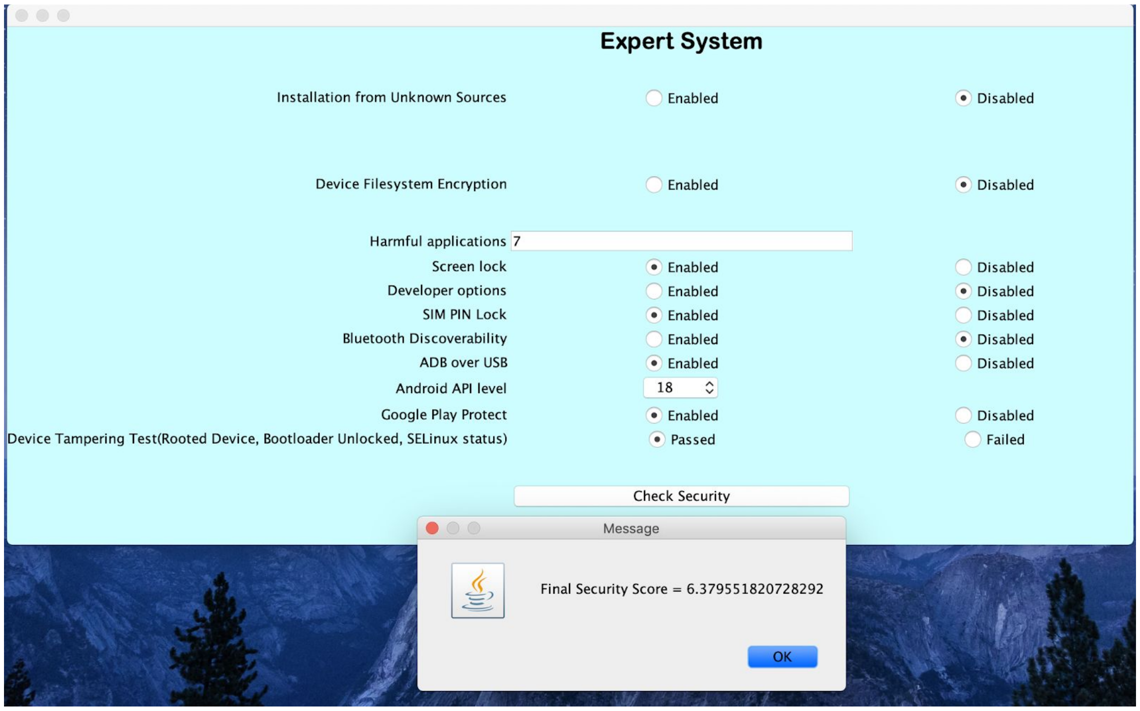
Installation from Unknown Sources	[✓ Enabled]	Disabled
Device Filesystem Encryption	[✓ Enabled]	Disabled
Harmful applications	5	
Screen lock	[✓ Enabled]	Disabled
Developer options	Enabled	[✓ Disabled]
SIM PIN Lock	Enabled	[✓ Disabled]
Bluetooth Discoverability	[✓ Enabled]	Disabled
ADB over USB	Enabled	[✓ Disabled]
Android API level	20	
Google Play Protect	[✓ Enabled]	Disabled
Device Tampering Test(Rooted Device, Bootloader Unlocked, SELinux status)	[✓ Passed]	Failed

Check Security

Message

Final Security Score = 7.675070028011204

OK



Expert System

Installation from Unknown Sources: Enabled [✓ Disabled]

Device Filesystem Encryption: Enabled [✓ Disabled]

Harmful applications: 7

Screen lock: [✓ Enabled] Disabled

Developer options: Enabled [✓ Disabled]

SIM PIN Lock: [✓ Enabled] Disabled

Bluetooth Discoverability: Enabled [✓ Disabled]

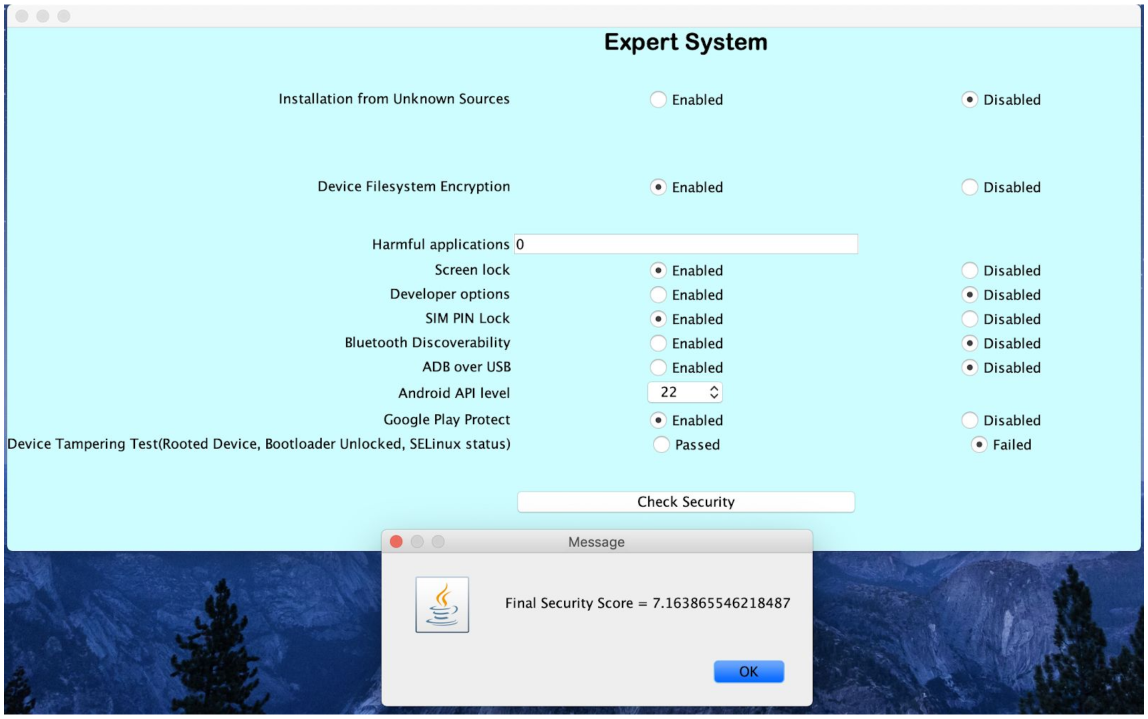
ADB over USB: [✓ Enabled] Disabled

Android API level: 18

Google Play Protect: [✓ Enabled] Disabled

Device Tampering Test(Rooted Device, Bootloader Unlocked, SELinux status): [✓ Passed] Failed

Final Security Score = 6.379551820728292



Expert System

Installation from Unknown Sources: Enabled [✓ Disabled]

Device Filesystem Encryption: [✓ Enabled] Disabled

Harmful applications: 0

Screen lock: [✓ Enabled] Disabled

Developer options: Enabled [✓ Disabled]

SIM PIN Lock: [✓ Enabled] Disabled

Bluetooth Discoverability: Enabled [✓ Disabled]

ADB over USB: Enabled [✓ Disabled]

Android API level: 22

Google Play Protect: [✓ Enabled] Disabled

Device Tampering Test(Rooted Device, Bootloader Unlocked, SELinux status): Passed [✓ Failed]

Final Security Score = 7.163865546218487

8. User’s guide and GUI

The Graphical User Interface is built using Swing Java Toolkit. It is made up of Java inbuilt classes such as JLabel and JComboBox palettes. There are total 18 RadioBoxe palettes present in the parent panel. These are used to represent boolean values. There is also a textbox palette to represent number of harmful apps and ComboBox palette to choose API level.

After selecting all the inputs, button needs to be clicked to evaluate the security system. After that, pop up is shown with the security evaluation score,

9. Development process documentation

This project was collectively developed by Sarang Narkhede, Ayush Soni and Hitesh Vaidya. An extensive literature survey was done by all the project group members for development of this system. Through this literature survey we found out various aspects related to security of android devices. Following are those points,

- We found out generally used common ways of breaching the security of android smartphones.
- Ways in which some of the existing rule based security evaluation systems have been developed.
- Various metrics that could be used for security evaluation
- Metric values that are desirable for increasing the security evaluation score of any smartphone
- The total number of android APIs released so far
- Version wise number of users of Android APIs and the number of security threats for each version

A comparative study of various options for implementation tools was performed by Sarang Narkhede in the development process. Through this study we found out that JESS was perhaps the best rule engine to develop this expert system. All the group members studied how JESS rule engine works and the syntax for it.

All the group members collectively came up with various metrics that could be used for security evaluation. These metrics were then shortlisted to the final eleven metrics that are used in the system. We decided what values these metrics would take and which out of them more desirable.

One of the given conditions for the development of this system was to build a chain rule i.e. output of one calculation is used as an input for another one. We made sure that this condition is met. From the figure above, we can understand that we calculate various cluster scores from the weighted sum of all the cluster specific metric values. A further weighted summation these cluster scores is done to calculate the raw score. This raw score is then normalized in the final step. This structure containing various clusters and chained calculation was developed by Ayush Soni and Hitesh Vaidya.

A layman could use this expert system only if it had a good GUI for the interaction purpose. The GUI for this system was developed by Sarang Narkhde. He studied about Java Swing class and the way in which we could connect Java Swing class with JESS rule engine. Along with Ayush Soni he connected the GUI to JESS rule engine.

All the device metrics were encoded in JESS to form a rule base. The metric values were then assigned weight values and various rules were assigned to the system. This rule engine was developed collectively by Ayush Soni and Hitesh Vaidya.

The documentation of this project was done collectively by all the group members.