Foundation of Intelligent Systems

Project Part-3

Use of machine learning techniques to implement Expert System

By
Abdul Hakim Shanavas
(ahs7665@rit.edu)

Table of Contents

Part-A

Part-B

Executive Summary

In this project we focused on approximating our android app's expert system by training an artificial neural network and evaluating the model's performance on different choice of neurons in a hidden layer. Evaluation of the model was based on the time consumption, resource consumption and accuracy of the model. We created our dataset based on the combinations of our inputs of our android app metrics. We defined the ground truth based on the outcome of our expert system to the dataset. By trying out different combinations of our input metrics we generated our dataset which has 279 records with 7 attributes. The predicted label is the security evaluation of the device which is categorized into 5 categories (Danger - 1, Critical – 2, Low – 3, Medium – 4, Perfect - 5). Some of our metrics have only binary outputs such as Screen Lock (Tells if the device is locked or not) and some have a range of values. Based on the score and the range of each metric and performing combinations on those inputs we created our dataset.

The dataset was then split into training set and testing set of roughly 67% and 33% respectively. We used python language and packages such as Sklearn.Multi-Layer_Perceptron to build our neural network model. As suggested in the project description file, we started to train our neural network with initial number of neurons in the hidden layer equal to half the size of inputs. Then we gradually increased the number of neurons almost doubling each time and tested our model with the test data by generating the accuracy, f1 score, precision, recall, weighted average and macro average. We even calculated the confusion matrix. We also started with one hidden layer and increased the number of hidden layers and profiled our python script using time and memory profiling. Based on the learning rate, we plotted loss curve while fine tuning our model with different initial neurons and number of hidden layers.

From the results, we found out that increasing the number of neurons made the model better at predicting unknown test data but on the other hand increased the time and memory consumed by the program. One more interesting thing which we found was, after a certain limit increasing the number of neurons to the same hidden layer does not make any impact on the accuracy rather it only increases time and memory. However, instead of setting the total number of initial neurons to just one layer, if we splitted them into two layers or more, the accuracy of the model is significantly better and stable, also the time and memory is almost same because the total number of neurons remain the same.

Requirements

In this project we are required to build a machine learning model to approximate our expert system output. The reason behind this is expert system is computationally expensive, which can be a major drawback for a constrained mobile device. Machine learning model on the other hand predicts the output by approximation. In this project we went with creating a neural network model from the dataset generated by entering combinations of inputs to our expert system. Another phase of this project is to make design decisions of our neural network model such as the number of neurons in the hidden layer by simulating different choices and selecting the best combination in terms of time and memory consumption and also based on the accuracy of the model.

Therefore, this project is divided into three tasks namely 1. Dataset generation, 2. Creating neural network model by splitting the dataset into training and testing dataset and testing the model on unknown data, 3. Making design decisions of creating the neural network by trying different choices of number of neurons in the hidden layer. All these tasks are evaluated based on the time and resource consumption.

Implementation

This project has three phases or tasks.

1. Dataset generation:
   We have totally seven input attributes in our android application. Some of the attributes are binary and some have a range of values. Based on these we were able to generate 279 records. Our android application evaluates security based on the metrics and categorizes the level of security into five categories. Therefore, the ground truth of our predicted label will have 5 classes. Level 1 being Danger to Level 5 being Perfect category. A snapshot of our dataset is attached below.

A	B	C	D	E	F	G	H	I
Screen Lock	OS Version	Verify Apps	Potential Harmful Apps	Developer option	Basic Integrity	Cts	Score	Security Level
0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	0	1	1
0	1	0	0	0	0	0	1	1
0	0	0	1	0	0	0	1	1
1	0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	1	1
0	0	1	0	0	0	0	1	1
0	2	0	0	0	0	0	2	2
0	0	0	0	1	0	1	2	2
0	0	0	2	0	0	0	2	2
0	1	0	0	1	0	0	2	2
0	0	1	1	0	0	0	2	2
1	0	0	1	0	0	0	2	2
1	0	0	0	0	1	0	2	2
1	0	0	0	0	0	1	2	2
0	1	1	0	0	0	0	2	2
1	0	0	0	1	0	0	2	2
0	1	0	0	0	0	1	2	2
0	1	0	0	0	1	0	2	2
0	1	0	1	0	0	0	2	2
0	0	0	0	0	1	1	2	2
0	0	0	1	0	0	1	2	2
1	0	1	0	0	0	0	2	2
0	0	0	0	1	1	0	2	2
0	0	0	1	1	0	0	2	2
0	0	1	0	1	0	0	2	2
0	0	1	0	0	0	1	2	2
0	0	1	0	0	1	0	2	2
0	0	0	1	0	1	0	2	2
1	0	1	0	0	0	1	3	2
0	2	0	0	0	1	0	3	2
0	0	0	1	0	1	1	3	2
1	2	0	0	0	0	0	3	2
0	0	1	0	1	1	0	3	2
0	1	1	0	1	0	0	3	2
1	0	0	0	1	1	0	3	2
0	0	0	1	1	0	1	3	2
0	0	1	2	0	0	0	3	2
0	0	1	1	0	0	1	3	2

Values of each metric:

Screen Lock – 0 and 1

OS Version – 0, 1 and 2

Verify Apps – 0 and 1

Potential Harmful Apps – 0, 1 and 2

Developer option – 0 and 1

Basic Integrity – 0 and 1

Compatibility – 0 and 1

Security Level (To be predicted) – 1, 2, 3, 4 and 5

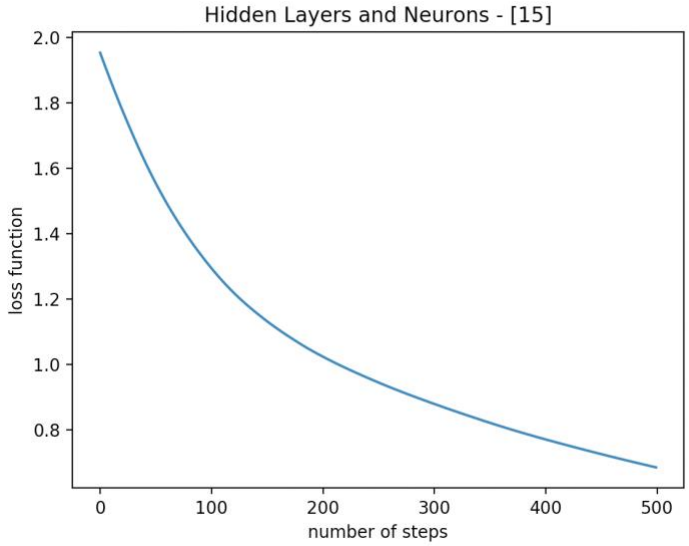
2. This task involves developing a neural network model using the above generated dataset. We used python language and packages such as Sci-kit learn to build our neural network model. We used multilayer perceptron with some initial number of neurons and increased the number of neurons and the number of hidden layer.

   We split the dataset into 67% training data and 33% testing dataset. This can be seen in the screenshots below.

```
No. of records in dataset: 279
No.of Training records: 186
No.of Testing records: 93
/Users/abdulhakim/Documents/CS/Courses/FIS/Project/Data
********** Confusion Matrix **********
  % self.max_iter, ConvergenceWarning)
[[ 0  3  0  0  0]
 [ 0 13  9  0  0]
 [ 0  2 40  1  0]
 [ 0  0  7 14  0]
 [ 0  0  0  4  0]]
********** ANN model report **********
/Users/abdulhakim/Documents/CS/Courses/FIS/Project/Data
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.72      0.59      0.65        22
           3       0.71      0.93      0.81        43
           4       0.74      0.67      0.70        21
           5       0.00      0.00      0.00         4
```

```
********** ANN model report **********
/Users/abdulhakim/Documents/CS/Courses/FIS/Project/Dataset/Code/venv/lib/py
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.72      0.59      0.65        22
           3       0.71      0.93      0.81        43
           4       0.74      0.67      0.70        21
           5       0.00      0.00      0.00         4

    accuracy                           0.72        93
   macro avg       0.43      0.44      0.43        93
weighted avg       0.67      0.72      0.69        93

         267022 function calls (265103 primitive calls) in 9.914 seconds
```

The above two screenshots have 15 neurons with max iterations of 500. As it can be seen that the model accuracy is only 72% and it still has not converged even after it's max iterations. The loss curve of neural network model with one hidden layer and 15 neurons can be seen in the plot.




```
Line #    Mem usage    Increment   Line Contents
================================================
    24  115.121 MiB  115.121 MiB   @profile
    25                             def main():
    26  117.223 MiB    2.102 MiB       df = pd.read_excel('ES_DATA_CLEAN.xlsx', sheet_name='Sheet1')
    27  117.418 MiB    0.195 MiB       df_actual = df.drop('Score', axis=1)
    28  117.418 MiB    0.000 MiB       print(df_actual.columns)
    29  117.418 MiB    0.000 MiB       X = df_actual.drop('Security Level', axis=1)
    30  117.453 MiB    0.035 MiB       Y = df_actual[['Security Level']]
    31
    32  117.547 MiB    0.094 MiB       X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.33, random_state=1)
    33
    34  117.547 MiB    0.000 MiB       print('No. of records in dataset: 279')
    35  117.547 MiB    0.000 MiB       print('No.of Training records: ' + str(len(X_train)))
    36  117.547 MiB    0.000 MiB       print('No.of Testing records: ' + str(len(X_test)))
    37
    38  117.547 MiB    0.000 MiB       mlp = MLPClassifier(hidden_layer_sizes=(15), max_iter=500, random_state=1)
    39                                 # mlp.learning_rate_init = 1
    40
    41                                 # mlp = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
    42                                 #               beta_2=0.999, early_stopping=False, epsilon=1e-08,
    43                                 #               hidden_layer_sizes=(3,), learning_rate='constant',
    44                                 #               learning_rate_init=0.001, max_iter=500, momentum=0.9,
    45                                 #               nesterovs_momentum=True, power_t=0.5, random_state=None,
    46                                 #               shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
    47                                 #               verbose=False, warm_start=False)
    48
    49  118.543 MiB    0.996 MiB       mlp.fit(X_train, Y_train.values.ravel())
    50  118.547 MiB    0.004 MiB       predictions = mlp.predict(X_test)
    51  118.547 MiB    0.000 MiB       print('********** Confusion Matrix **********')
    52  118.582 MiB    0.035 MiB       print(confusion_matrix(Y_test, predictions))
    53  118.582 MiB    0.000 MiB       print('********** ANN model report **********')
    54  118.781 MiB    0.199 MiB       print(classification_report(Y_test, predictions))
    55  161.188 MiB   42.406 MiB       plotting(mlp, [15])
```

Above screenshot is the memory usage of the python script I ran. As it can be seen that there is a rise in usage when fitting the model, however it is not significantly high.

3. The final task involves on deciding which is the best choice of number of neurons and hidden layers. I ran my python script for various choices of number of neurons and hidden layers and also checked the time and memory usage of every simulation I ran just like above. For our dataset the best and stable one we found was two hidden layers with 15, 10 neurons in each layer. There was also one choice with three layers where we got the same accuracy, but the model consumes less time and memory in the 2 layer with almost similar accuracy.

User Guide

In Pycharm to run the code. Before running the code, install the below packages,

1. Pandas
2. Sklearn
3. Matplotlib
4. cProfile

The above packages can be installed using pip in the terminal window in Pycharm.

Paste this in the Interpreter options: -m memory_profiler in Pycharm. Interpreter options can be found in edit configurations.

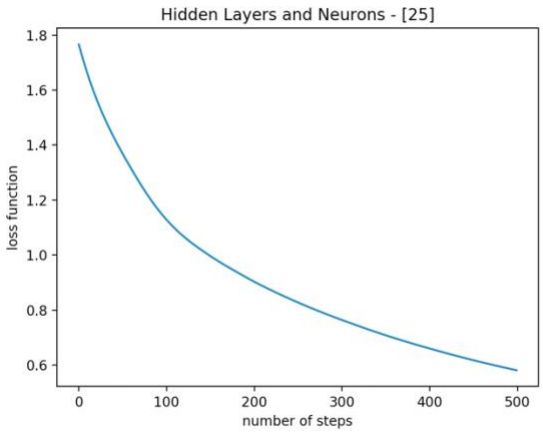
Change the number of neurons by editing in this line of code. Hidden layers can be added by specifying the number of neurons in each layer in comma separated in the below line of code in the argument hidden_layer_sizes=(15). Max iterations can also be changed from the below line of code.

```
mlp = MLPClassifier(hidden_layer_sizes=(15), max_iter=500, random_state=1)
```

Software Requirements: Pycharm professional edition, Python 3.x interpreter.

Experiments and Results

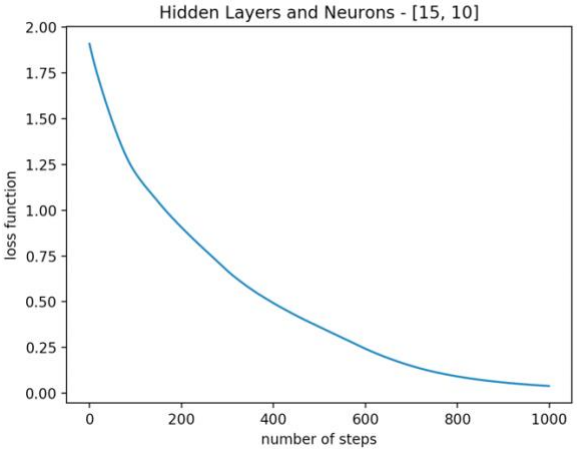
```
********** Confusion Matrix **********
[[ 0  3  0  0  0]
 [ 0 14  8  0  0]
 [ 0  0 41  2  0]
 [ 0  0  7 14  0]
 [ 0  0  0  4  0]]
********** ANN model report **********
/Users/abdulhakim/Documents/CS/Courses/FIS/Project/Data
  'precision', 'predicted', average, warn_for)
              precision    recall  f1-score   support

           1       0.00      0.00      0.00         3
           2       0.82      0.64      0.72        22
           3       0.73      0.95      0.83        43
           4       0.70      0.67      0.68        21
           5       0.00      0.00      0.00         4

    accuracy                           0.74        93
   macro avg       0.45      0.45      0.45        93
weighted avg       0.69      0.74      0.71        93
```




```
         281403 function calls (279400 primitive calls) in 42.163 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     17/1    0.000    0.000   42.163   42.163 {built-in method builtins.exec}
        1    0.000    0.000   42.163   42.163 <string>:1(<module>)
        1    0.000    0.000   42.163   42.163 memory_profiler.py:657(f)
        1    0.002    0.002   42.163   42.163 ann.py:24(main)
        1    0.000    0.000   40.783   40.783 ann.py:17(plotting)
        1    0.000    0.000   40.583   40.583 pyplot.py:245(show)
        1    0.000    0.000   40.583   40.583 deprecation.py:404(wrapper)
        1    0.000    0.000   40.583   40.583 backend_bases.py:3269(show)
        1    0.000    0.000   40.578   40.578 backend_macosx.py:191(mainloop)
        1   40.043   40.043   40.578   40.578 {built-in method matplotlib.backends._macosx.show}
        1    0.000    0.000    0.776    0.776 multilayer_perceptron.py:965(fit)
        1    0.000    0.000    0.776    0.776 multilayer_perceptron.py:310(_fit)
        1    0.056    0.056    0.767    0.767 multilayer_perceptron.py:469(_fit_stochastic)
      2/1    0.000    0.000    0.552    0.552 _decorators.py:146(wrapper)
```

As it can be seen that, increasing the number of neurons from 15 to 25 does not have a big impact on the accuracy of the model, however, the time taken is high. So instead of giving all 25 neurons to single hidden layer, we created two hidden layer and split the number of neurons.

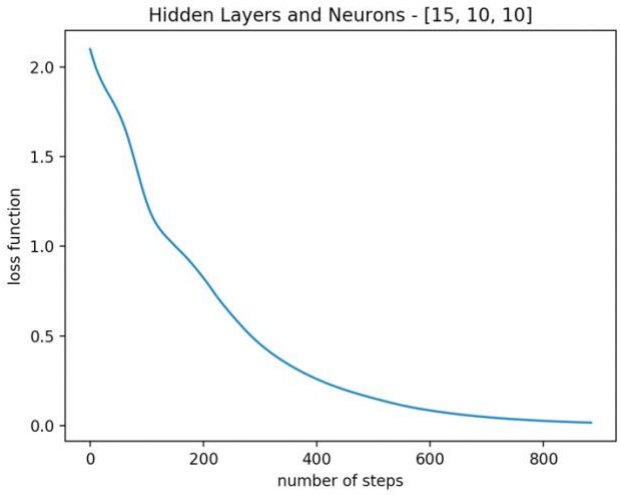
```
********** Confusion Matrix **********
[[ 3  0  0  0  0]
 [ 0 22  0  0  0]
 [ 0  0 43  0  0]
 [ 0  0  0 21  0]
 [ 0  0  0  3  1]]
********** ANN model report ***********
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00        22
           3       1.00      1.00      1.00        43
           4       0.88      1.00      0.93        21
           5       1.00      0.25      0.40         4

    accuracy                           0.97        93
   macro avg       0.97      0.85      0.87        93
weighted avg       0.97      0.97      0.96        93
```




```
396886 function calls (394936 primitive calls) in 40.488 seconds

Ordered by: cumulative time

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
  17/1    0.000    0.000   40.488   40.488 {built-in method builtins.exec}
     1    0.000    0.000   40.488   40.488 <string>:1(<module>)
     1    0.000    0.000   40.488   40.488 memory_profiler.py:657(f)
     1    0.002    0.002   40.488   40.488 ann.py:24(main)
     1    0.000    0.000   38.399   38.399 ann.py:17(plotting)
     1    0.000    0.000   38.206   38.206 pyplot.py:245(show)
     1    0.000    0.000   38.206   38.206 deprecation.py:404(wrapper)
     1    0.000    0.000   38.206   38.206 backend_bases.py:3269(show)
     1    0.000    0.000   38.202   38.202 backend_macosx.py:191(mainloop)
     1   37.793   37.793   38.202   38.202 {built-in method matplotlib.backends._macosx.show}
     1    0.000    0.000    1.526    1.526 multilayer_perceptron.py:965(fit)
     1    0.000    0.000    1.526    1.526 multilayer_perceptron.py:310(_fit)
     1    0.094    0.094    1.517    1.517 multilayer_perceptron.py:469(_fit_stochastic)
  1000    0.093    0.000    0.808    0.001 multilayer_perceptron.py:179(_backprop)
```

As it can be seen from above, after splitting the number of neurons (25) between two hidden layers, the accuracy has spiked to 97% while the time consumed is almost same actually lower. From the confusion matrix, security level 5 has not been picked up well by the model. Also, we increased the number of max iterations to 1000, therefore the warning of convergence has disappeared.

```
No. of records in dataset: 279
No.of Training records: 186
No.of Testing records: 93
*********** Confusion Matrix ***********
[[ 2  1  0  0  0]
 [ 0 22  0  0  0]
 [ 0  0 43  0  0]
 [ 0  0  0 21  0]
 [ 0  0  0  1  3]]
*********** ANN model report ***********
              precision    recall  f1-score   support

           1       1.00      0.67      0.80         3
           2       0.96      1.00      0.98        22
           3       1.00      1.00      1.00        43
           4       0.95      1.00      0.98        21
           5       1.00      0.75      0.86         4

    accuracy                           0.98        93
   macro avg       0.98      0.88      0.92        93
weighted avg       0.98      0.98      0.98        93
```




```
         421213 function calls (419313 primitive calls) in 78.549 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     17/1    0.000    0.000   78.549   78.549 {built-in method builtins.exec}
        1    0.000    0.000   78.549   78.549 <string>:1(<module>)
        1    0.000    0.000   78.549   78.549 memory_profiler.py:657(f)
        1    0.002    0.002   78.549   78.549 ann.py:24(main)
        1    0.000    0.000   76.335   76.335 ann.py:17(plotting)
        1    0.000    0.000   76.116   76.116 pyplot.py:245(show)
        1    0.000    0.000   76.116   76.116 deprecation.py:404(wrapper)
        1    0.000    0.000   76.116   76.116 backend_bases.py:3269(show)
        1    0.000    0.000   76.111   76.111 backend_macosx.py:191(mainloop)
        1   75.556   75.556   76.111   76.111 {built-in method matplotlib.backends._macosx.show}
        1    0.000    0.000    1.627    1.627 multilayer_perceptron.py:965(fit)
        1    0.000    0.000    1.627    1.627 multilayer_perceptron.py:310(_fit)
        1    0.092    0.092    1.618    1.618 multilayer_perceptron.py:469(_fit_stochastic)
```

```
Line #    Mem usage    Increment   Line Contents
================================================
    24  115.082 MiB  115.082 MiB   @profile
    25                             def main():
    26  117.145 MiB    2.062 MiB       df = pd.read_excel('ES_DATA_CLEAN.xlsx', sheet_name='Sheet1')
    27  117.336 MiB    0.191 MiB       df_actual = df.drop('Score', axis=1)
    28  117.336 MiB    0.000 MiB       print(df_actual.columns)
    29  117.340 MiB    0.004 MiB       X = df_actual.drop('Security Level', axis=1)
    30  117.383 MiB    0.043 MiB       Y = df_actual[['Security Level']]
    31
    32  117.469 MiB    0.086 MiB       X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.33, random_state=1)
    33
    34  117.469 MiB    0.000 MiB       print('No. of records in dataset: 279')
    35  117.469 MiB    0.000 MiB       print('No.of Training records: ' + str(len(X_train)))
    36  117.469 MiB    0.000 MiB       print('No.of Testing records: ' + str(len(X_test)))
    37
    38  117.473 MiB    0.004 MiB       mlp = MLPClassifier(hidden_layer_sizes=(15,10,10), max_iter=1000, random_state=1)
    39                                 # mlp.learning_rate_init = 1
    40
    41                                 # mlp = MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
    42                                 #               beta_2=0.999, early_stopping=False, epsilon=1e-08,
    43                                 #               hidden_layer_sizes=(3,), learning_rate='constant',
    44                                 #               learning_rate_init=0.001, max_iter=500, momentum=0.9,
    45                                 #               nesterovs_momentum=True, power_t=0.5, random_state=None,
    46                                 #               shuffle=True, solver='adam', tol=0.0001, validation_fraction=0.1,
    47                                 #               verbose=False, warm_start=False)
    48
    49  119.336 MiB    1.863 MiB       mlp.fit(X_train, Y_train.values.ravel())
    50  119.340 MiB    0.004 MiB       predictions = mlp.predict(X_test)
```

The three layer model has definitely better accuracy (though not significant difference) and more stable compared to the two layered model. However, the time and memory consumed by this model is almost double than the 2 layered model. Therefore, the 2 hidden layer with 15,10 neurons in each layer neural network model is better.

Part – B

Executive Summary

In this project, primary aim is to find how the attackers attack android OS such as in what ways and how to keep the user aware of such threats, educate the user by providing a way to evaluate the security status of their mobile device. Furthermore, since a lot of users are not tech savvy, this project aims to find the metrics to collect data automatically and intelligently reaching a conclusion regarding the security status of the device and suggesting or directing the user to make some measures from their end to keep them protected.

In the best interests of covering all the user base, we aim to provide flexibility in our android app to give the control to user to select any particular metrics they want to test or a complete checkup of the device. The apps user interface has to be kept simple and easy to understand or use at the same time give out the maximum information in layman terms and guide the user to better protect their device. In other words, the app has to be smart enough to do most of the work without depending much from the user input.

The above problems are identified, and a lot of time has been invested in researching and finding the best way to develop an android app to enhance the security of android OS.

Specification

Building an Android Application:

Android app has a checkbox for each metric that the user wants to be tested and additionally another checkbox to check all the metrics that is complete scan. There is a score label (textview) assigned in correspondence to each metric checkbox and also another button to check the details of the metric. The details button explains the output and gives suggestions or directions to user to improve their device security.

Once the required metrics are selected, the user can press the scan button to automatically collect the necessary data (facts) to arrive at a solution. The cumulative score and the level of security of the device will be displayed using a label above scan button. As mentioned earlier, to check a particular metrics information, the user can click the corresponding details button.

Implementation

Database:

We have used a hashset to store all the facts. Choosing hashset data structure for representing database seems a logical choice due to the reasons that, each fact is unique and retrieval from the hashset is O(1) similar to database. Therefore, we decided to represent the facts in hashset.

Knowledge Base:

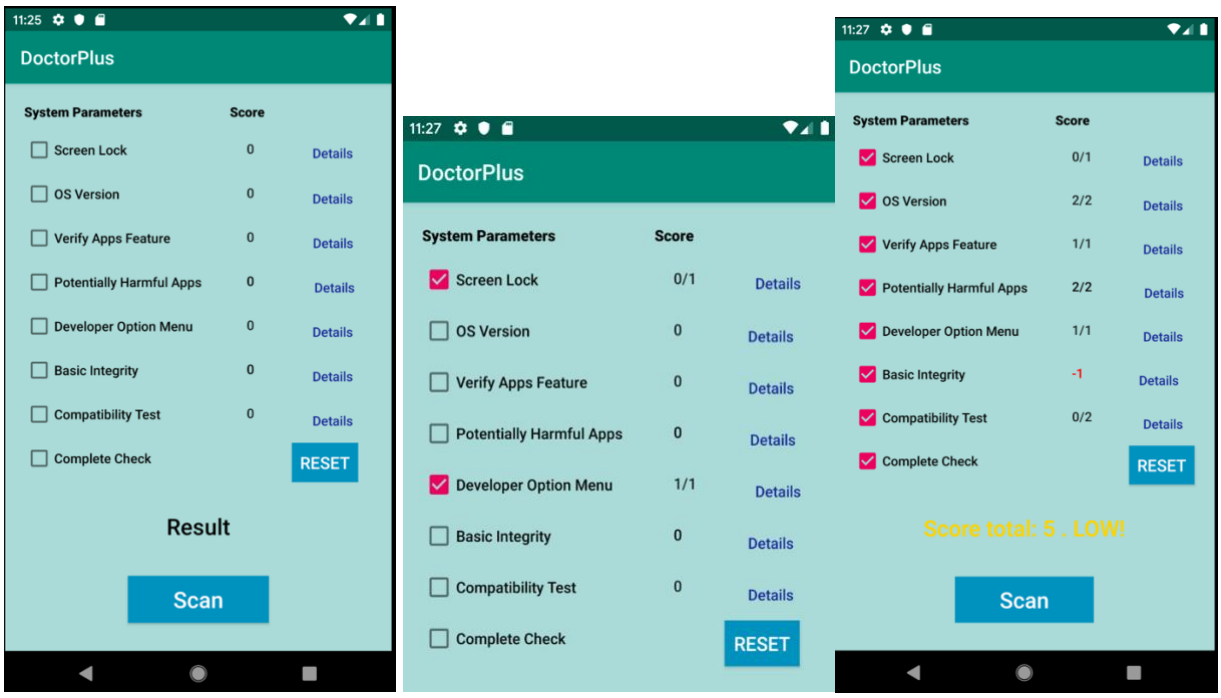
We have created If-Then structure rules which checks for facts and fires the action associated if the fact is present. Some rules create more facts and those facts are used in other rules. Therefore, firing each rule is done is the order of importance. Each rule check for proof of facts present and acts accordingly.

Inference Engine:

Inference engine cumulatively uses all the individual rule's output and uses those facts to decide the security level. Inference Engine uses forward chaining by checking all the rules and creating more facts until it reaches a conclusion. All the facts are represented in string and some have some int parameters (score).

User Interface:

A simple user interface which anyone can use has been developed.



Description of working:

1. DoctorPlus app detects if the device has password, pattern or pin setup. When 'Details' is pressed, the user will be let know that they have not set up password and also gives the option to set the password by redirecting to settings page.
2. Application detects the OS version of the device which can be used to see if that is the latest or an old one.
3. VerifyApps feature enabled makes the device more secure and therefore the setting is checked. From Android Oreo onwards Unknown Source option is removed [5], therefore it does not seem logical to check a metric which is deprecated. Since the app recommends having the latest OS version, there is no need to check for older devices.
4. Application fetches the list of harmful applications and lists the harmful application installed in the device.
5. Application detects if the developer option is enabled which causes to leak many sensitive information.
6. SafetyNet Attestation API fetches the response by checking several factors such as whether the device is rooted, unlocked bootloader etc.
   Sample response of SafetyNet Attestation API:
   {"nonce":"NmY4YWRkNWYtYzdiNS00MGRlLTk3ODQtZjIwYzYzODQzNTJl","timestampMs":1560615095371,"apkPackageName":"com.rit.doctorplus","apkDigestSha256":"QPUho8s7nz+u2T31/ZPKymKMZvOaZefdL3F54YPdxZQ=","ctsProfileMatch":false,"apkCertificateDigestSha256":["LCT2mYJjuzVxlvZQM2yPjbaR1ux0Fe/VxEwt7Ofh3X0="],"basicIntegrity":false,"advice":"LOCK_BOOTLOADER"}

Software and Hardware requirements:

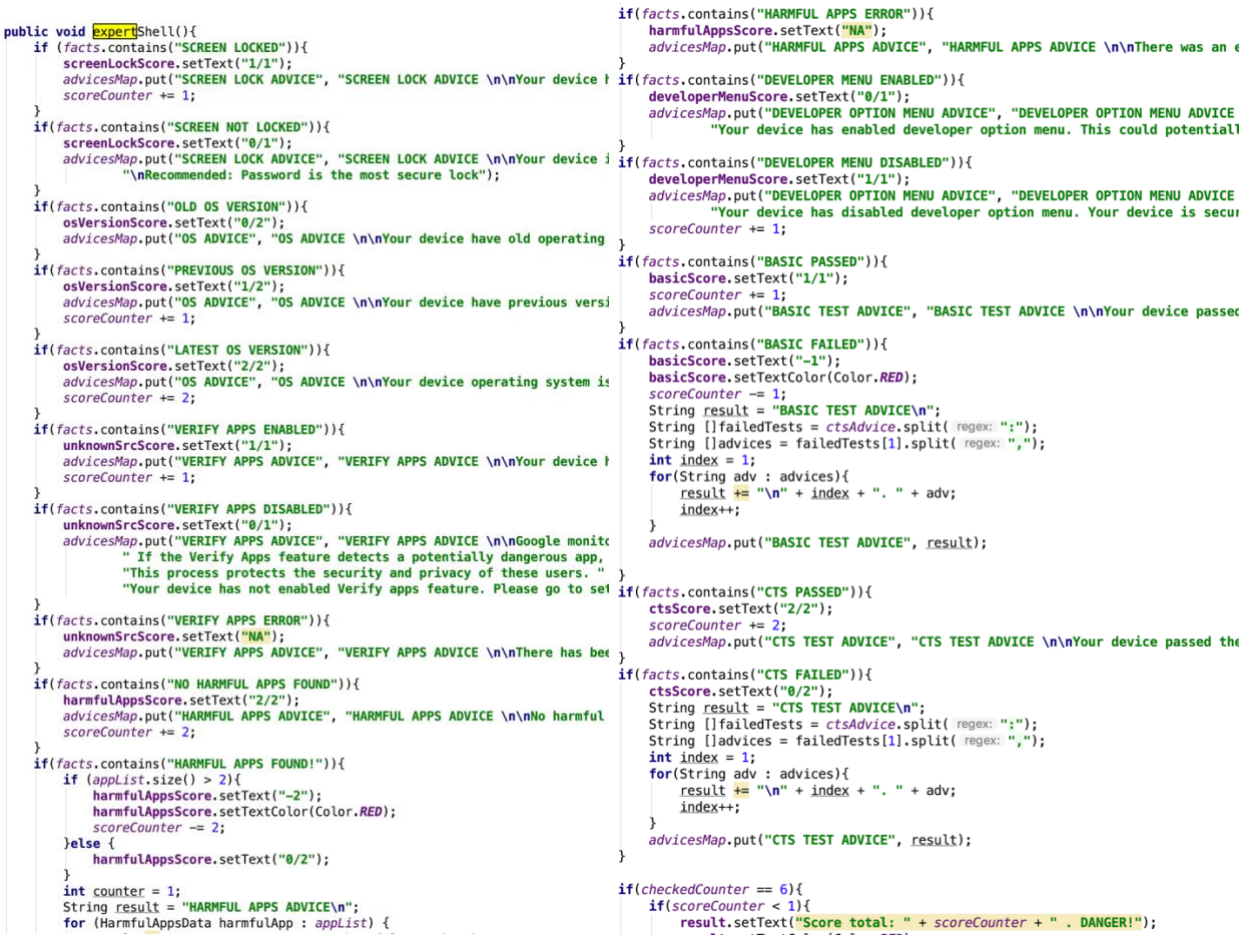
Minimum API required to run application is 23. Mobile device must have internet connection as the API hits the Google backend service to get results. Mobile device should have minimum 512 mb ram.

Classes used:

```
androidx.annotation.NonNull;
androidx.appcompat.app.AppCompatActivity;

android.app.AlertDialog;
android.app.KeyguardManager;
```

```
android.content.Context;
android.content.DialogInterface;
android.graphics.Color;
android.net.ConnectivityManager;
android.os.Build;
android.os.Bundle;
android.provider.Settings;
android.util.Base64;
android.util.Log;
android.view.View;
android.widget.Button;
android.widget.CheckBox;
android.widget.CompoundButton;
android.widget.TextView;

com.google.android.gms.common.ConnectionResult;
com.google.android.gms.common.GoogleApiAvailability;
com.google.android.gms.common.api.ApiException;
com.google.android.gms.safetynet.HarmfulAppsData;
com.google.android.gms.safetynet.SafetyNet;
com.google.android.gms.safetynet.SafetyNetApi;
com.google.android.gms.tasks.OnCompleteListener;
com.google.android.gms.tasks.OnFailureListener;
com.google.android.gms.tasks.OnSuccessListener;
com.google.android.gms.tasks.Task;

java.net.InetAddress;
java.util.HashSet;
java.util.List;
java.util.UUID;
```

Rules Design

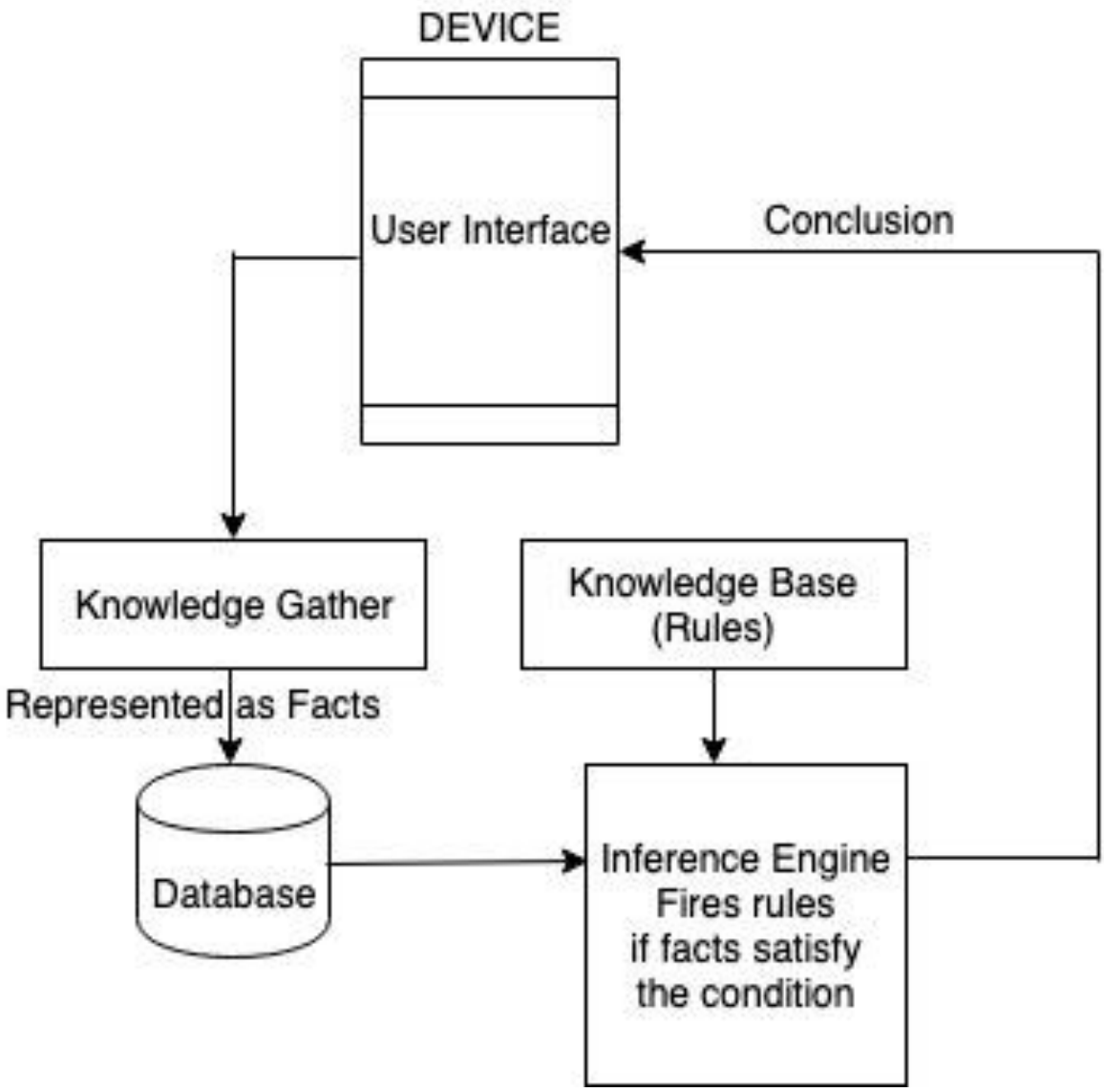


```
public void expertShell(){
    if (facts.contains("SCREEN LOCKED")){
        screenLockScore.setText("1/1");
        advicesMap.put("SCREEN LOCK ADVICE", "SCREEN LOCK ADVICE \n\nYour device h
        scoreCounter += 1;
    }
    if(facts.contains("SCREEN NOT LOCKED")){
        screenLockScore.setText("0/1");
        advicesMap.put("SCREEN LOCK ADVICE", "SCREEN LOCK ADVICE \n\nYour device i
                "\nRecommended: Password is the most secure lock");
    }
    if(facts.contains("OLD OS VERSION")){
        osVersionScore.setText("0/2");
        advicesMap.put("OS ADVICE", "OS ADVICE \n\nYour device have old operating
    }
    if(facts.contains("PREVIOUS OS VERSION")){
        osVersionScore.setText("1/2");
        advicesMap.put("OS ADVICE", "OS ADVICE \n\nYour device have previous versi
        scoreCounter += 1;
    }
    if(facts.contains("LATEST OS VERSION")){
        osVersionScore.setText("2/2");
        advicesMap.put("OS ADVICE", "OS ADVICE \n\nYour device operating system is
        scoreCounter += 2;
    }
    if(facts.contains("VERIFY APPS ENABLED")){
        unknownSrcScore.setText("1/1");
        advicesMap.put("VERIFY APPS ADVICE", "VERIFY APPS ADVICE \n\nYour device h
        scoreCounter += 1;
    }
    if(facts.contains("VERIFY APPS DISABLED")){
        unknownSrcScore.setText("0/1");
        advicesMap.put("VERIFY APPS ADVICE", "VERIFY APPS ADVICE \n\nGoogle monito
                " If the Verify Apps feature detects a potentially dangerous app,
                "This process protects the security and privacy of these users. "
                "Your device has not enabled Verify apps feature. Please go to set
    }
    if(facts.contains("VERIFY APPS ERROR")){
        unknownSrcScore.setText("NA");
        advicesMap.put("VERIFY APPS ADVICE", "VERIFY APPS ADVICE \n\nThere has bee
    }
    if(facts.contains("NO HARMFUL APPS FOUND")){
        harmfulAppsScore.setText("2/2");
        advicesMap.put("HARMFUL APPS ADVICE", "HARMFUL APPS ADVICE \n\nNo harmful
        scoreCounter += 2;
    }
    if(facts.contains("HARMFUL APPS FOUND!")){
        if (appList.size() > 2){
            harmfulAppsScore.setText("-2");
            harmfulAppsScore.setTextColor(Color.RED);
            scoreCounter -= 2;
        }else {
            harmfulAppsScore.setText("0/2");
        }
        int counter = 1;
        String result = "HARMFUL APPS ADVICE\n";
        for (HarmfulAppsData harmfulApp : appList) {
```

```
if(facts.contains("HARMFUL APPS ERROR")){
    harmfulAppsScore.setText("NA");
    advicesMap.put("HARMFUL APPS ADVICE", "HARMFUL APPS ADVICE \n\nThere was an e
}
if(facts.contains("DEVELOPER MENU ENABLED")){
    developerMenuScore.setText("0/1");
    advicesMap.put("DEVELOPER OPTION MENU ADVICE", "DEVELOPER OPTION MENU ADVICE
            "Your device has enabled developer option menu. This could potentiall
}
if(facts.contains("DEVELOPER MENU DISABLED")){
    developerMenuScore.setText("1/1");
    advicesMap.put("DEVELOPER OPTION MENU ADVICE", "DEVELOPER OPTION MENU ADVICE
            "Your device has disabled developer option menu. Your device is secur
    scoreCounter += 1;
}
if(facts.contains("BASIC PASSED")){
    basicScore.setText("1/1");
    scoreCounter += 1;
    advicesMap.put("BASIC TEST ADVICE", "BASIC TEST ADVICE \n\nYour device passed
}
if(facts.contains("BASIC FAILED")){
    basicScore.setText("-1");
    basicScore.setTextColor(Color.RED);
    scoreCounter -= 1;
    String result = "BASIC TEST ADVICE\n";
    String []failedTests = ctsAdvice.split( regex: ":");
    String []advices = failedTests[1].split( regex: ",");
    int index = 1;
    for(String adv : advices){
        result += "\n" + index + ". " + adv;
        index++;
    }
    advicesMap.put("BASIC TEST ADVICE", result);

}
if(facts.contains("CTS PASSED")){
    ctsScore.setText("2/2");
    scoreCounter += 2;
    advicesMap.put("CTS TEST ADVICE", "CTS TEST ADVICE \n\nYour device passed the
}
if(facts.contains("CTS FAILED")){
    ctsScore.setText("0/2");
    String result = "CTS TEST ADVICE\n";
    String []failedTests = ctsAdvice.split( regex: ":");
    String []advices = failedTests[1].split( regex: ",");
    int index = 1;
    for(String adv : advices){
        result += "\n" + index + ". " + adv;
        index++;
    }
    advicesMap.put("CTS TEST ADVICE", result);
}

if(checkedCounter == 6){
    if(scoreCounter < 1){
        result.setText("Score total: " + scoreCounter + " . DANGER!");
```

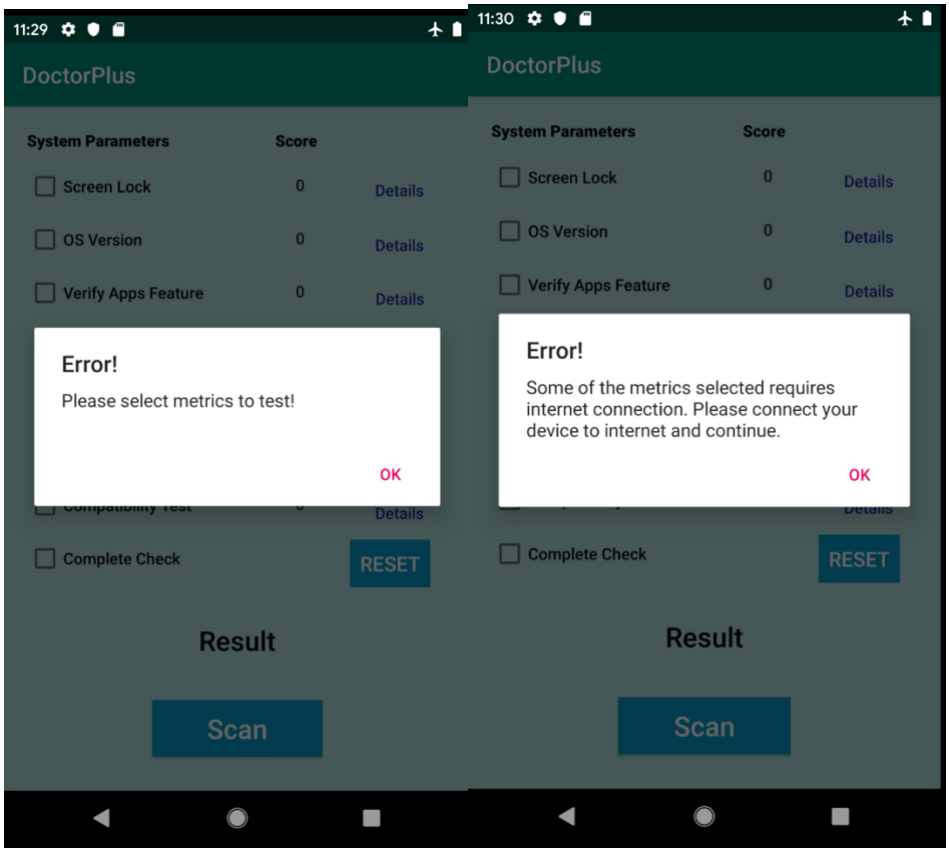
```
if(checkedCounter == 6){
    if(scoreCounter < 1){
        result.setText("Score total: " + scoreCounter + " . DANGER!");
        result.setTextColor(Color.RED);
    }
    else if(scoreCounter > 0 && scoreCounter < 3){
        result.setText("Score total: " + scoreCounter + " . CRITICAL!");
        result.setTextColor(Color.parseColor( colorString: "#FFA500"));
    }
    else if(scoreCounter > 2 && scoreCounter < 6){
        result.setText("Score total: " + scoreCounter + " . LOW!");
        result.setTextColor(Color.parseColor( colorString: "#FFD300"));
    }
    else if(scoreCounter > 5 && scoreCounter < 9){
        result.setText("Score total: " + scoreCounter + " . MEDIUM!");
        result.setTextColor(Color.parseColor( colorString: "#FDFF00"));
    }
    else if(scoreCounter > 8 && scoreCounter < 11){
        result.setText("Score total: " + scoreCounter + " . PERFECT!");
        result.setTextColor(Color.GREEN);
    }
}else {
    result.setText("Done! Click details to know more");
}


resetBtn.setEnabled(true);
```

System Hierarchy

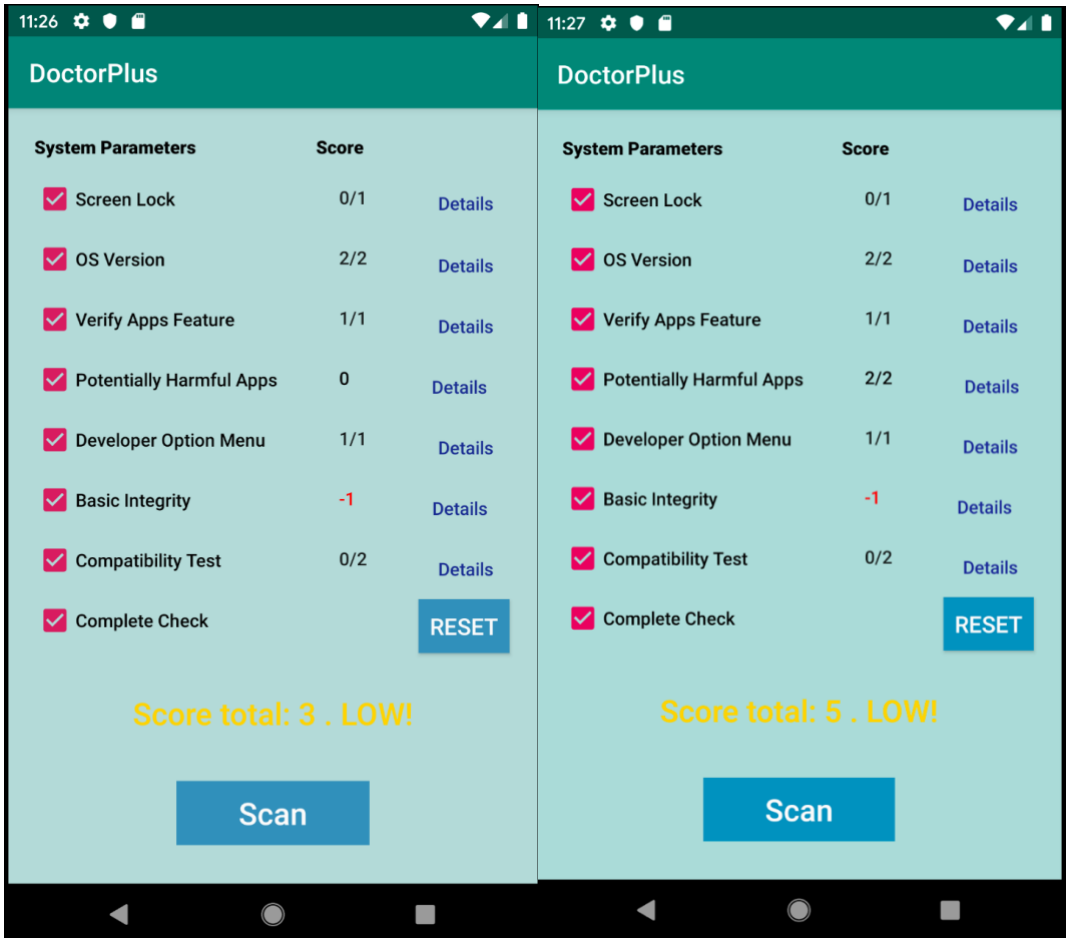

DEVICE
User Interface
Conclusion
Knowledge Gather
Knowledge Base
(Rules)
Represented as Facts
Database
Inference Engine
Fires rules
if facts satisfy
the condition

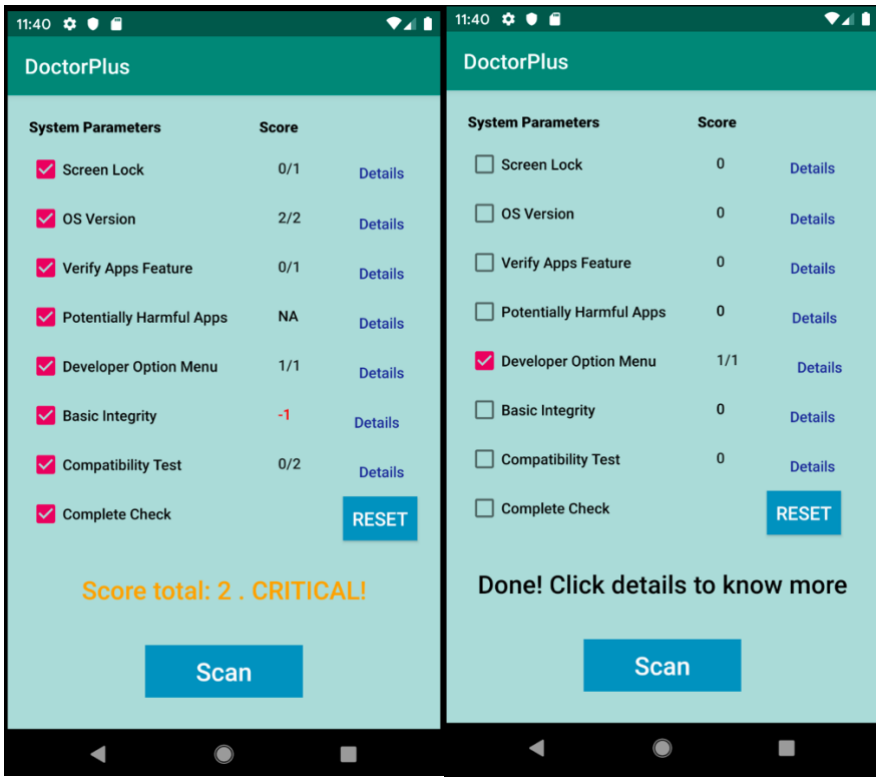
Testing

Doctor Plus android app was tested using virtual device of Nexus 3 5'' screen. Did not test the app with any other devices. Latest API (29) was used.



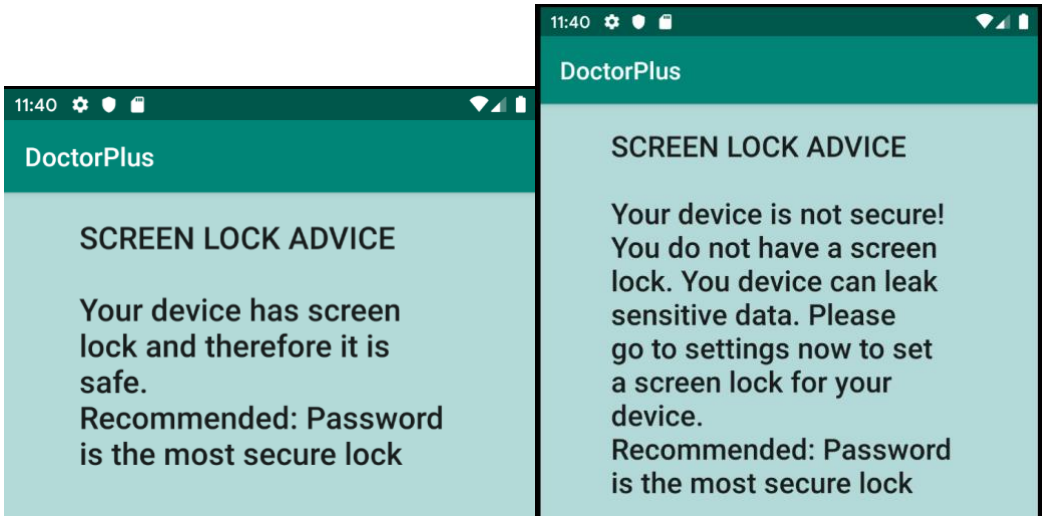
The above two screenshots show that if the scan button is pressed without selecting any metrics, it tells the user to select atleast one metric to proceed. And since some of the metrics need internet connection, the app detects if those metrics are selected and checks for internet connection and tells the user to connect to internet. While doing so, the app resets the app.

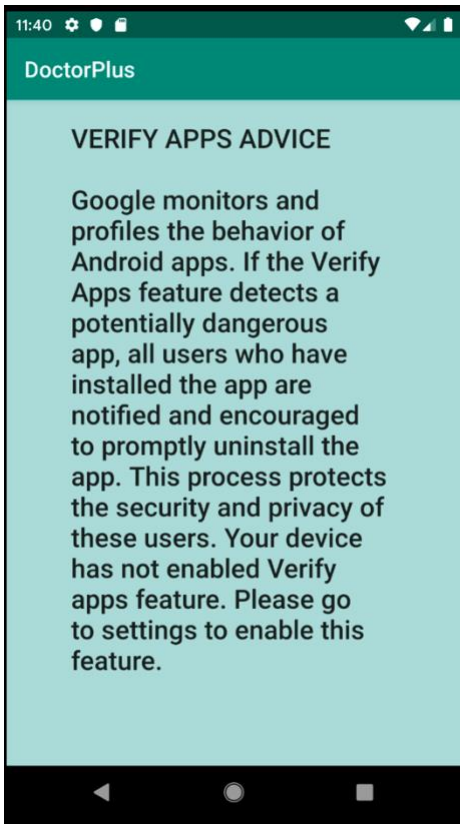

11:26
DoctorPlus
System Parameters
Score
Screen Lock 0/1 Details
OS Version 2/2 Details
Verify Apps Feature 1/1 Details
Potentially Harmful Apps 0 Details
Developer Option Menu 1/1 Details
Basic Integrity -1 Details
Compatibility Test 0/2 Details
Complete Check
RESET
Score total: 3 . LOW!
Scan
11:27
DoctorPlus
System Parameters
Score
Screen Lock 0/1 Details
OS Version 2/2 Details
Verify Apps Feature 1/1 Details
Potentially Harmful Apps 2/2 Details
Developer Option Menu 1/1 Details
Basic Integrity -1 Details
Compatibility Test 0/2 Details
Complete Check
RESET
Score total: 5 . LOW!
Scan

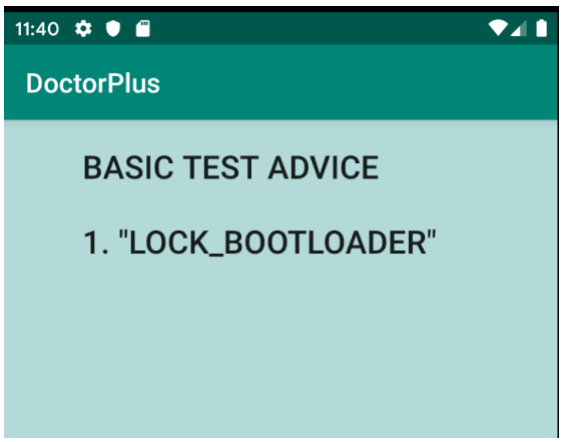


If the user does not want a complete check or scan of the device, the user has the liberty to choose only the certain metric evaluation and will only get to know the score for that metric and suggestions.

By clicking the details button, user gets advice on how to improve their device security against the specific metric.







The entire system is evaluated based on the values assigned to each metric.
The max score is 10 points. There is a possibility that the system can end up with negative scores.

Based on the score, the system is categorized as shown below

Score	Category
Less than 1	Danger
1 – 2	Critical
3 – 5	Low
6 – 8	Medium
9 – 10	Perfect!

If complete scan is not done, then total score and category will not be displayed.

User Guide

Doctor Plus is a rule based expert system android app which evaluates the security aspects of the device and suggests the user to follow certain directives to better protect the device.

1. Open the android project using Android Studio.
2. Select a virtual device based on the size and API. If not, can be downloaded from Android Studio.
3. Run the application by selecting the virtual device needed.
4. Select the metrics needed or select complete check to select all metrics.
5. Make sure the device is connected to internet.
6. Press scan button to see the result of the device security.
7. Change the settings to see the change.
8. Alternately, you can also install the DoctorPlus.apk file in your android device and launch the app from your device.

References

1. "Worldwide mobile operating system market share. Android has approximately 75% market share." [Online]. **http://gs.statcounter.com/os-market-share/mobile/worldwide**

2. SafetyNet VerifyApps API [Online**].** **https://developer.android.com/training/safetynet/verify-apps**

3. SafetyNet attestation API [online]. **https://developer.android.com/training/safetynet/attestation.html**

4. Android Oreo remove allow unknown source [Online]. **https://www.slashgear.com/android-oreo-removes-allow-unknown-sources-setting-but-dont-panic-22496285/**