Tejas Padliya

1Rochester Institute of Technology

April 01, 2024
SecuFone: Android Security Advisor

Tejas Padliya
Rochester Institute of Technology
tsp3859@rit.edu

ABSTRACT

Android is one of the most popular mobile operating systems (OS). Android is currently developed by the Google and other developers. It is an open source OS, based on the Linux kernel. Android can be used on smartphones, tablets, smartwatches and other devices. Android supports different types of applications, hardwares and firmwares. The Android operating system has different flavors and versions due to its open nature. This can lead to security and privacy concerns. Google Play Store is the most popular online app store for Android. As of 2015, more than 1.6 million applications have been listed on the Google Play Store [1].

In this project, I develop ‘SecuFone: Android Security Advisor’ an Android application. SecuFone evaluates the security of the overall system and provides advisory (recommendation) to improve their security. It evaluates security by using a hierarchical system. Hierarchical system uses fuzzy logic to provide users with the generalized score based on the metrics. The recommendations and score generated by the application helps users to enhance their privacy and security of their devices.

General Terms:

Experimentation, Measurement, Security, Theory

Keywords:

Android, jFuzzyLogic, Advisory, SecuFone

1. Introduction

Android is the most popular mobile operating systems. Android is used by the different products like smartphones, tablets, cars, TV and wearable devices. Android worldwide market
share in the smartphone segment is 81% as of 2015. As of 2014, 71% of mobile developers create applications for the Android. Android has more than 100 device manufacturers. Wireless service providers and manufacturers customize their operating system and add bloatware, because of which Android software system diversity is ever increasing. Below figure shows the market share of different Android versions as of Jan 2016 and May 2016 respectively [7].

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>0.2%</td>
</tr>
<tr>
<td>2.3.3-2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>3.0%</td>
</tr>
<tr>
<td>4.0.3-4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>2.7%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>9.0%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>12.2%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>3.5%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>36.1%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>16.9%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>15.7%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>0.7%</td>
</tr>
</tbody>
</table>

Figure 1: Android Version, Market Share: Jan 2016

<table>
<thead>
<tr>
<th>Version</th>
<th>Codename</th>
<th>API</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2</td>
<td>Froyo</td>
<td>8</td>
<td>0.1%</td>
</tr>
<tr>
<td>2.3.3-2.3.7</td>
<td>Gingerbread</td>
<td>10</td>
<td>2.2%</td>
</tr>
<tr>
<td>4.0.3-4.0.4</td>
<td>Ice Cream Sandwich</td>
<td>15</td>
<td>2.0%</td>
</tr>
<tr>
<td>4.1.x</td>
<td>Jelly Bean</td>
<td>16</td>
<td>7.2%</td>
</tr>
<tr>
<td>4.2.x</td>
<td></td>
<td>17</td>
<td>10.0%</td>
</tr>
<tr>
<td>4.3</td>
<td></td>
<td>18</td>
<td>2.9%</td>
</tr>
<tr>
<td>4.4</td>
<td>KitKat</td>
<td>19</td>
<td>32.5%</td>
</tr>
<tr>
<td>5.0</td>
<td>Lollipop</td>
<td>21</td>
<td>16.2%</td>
</tr>
<tr>
<td>5.1</td>
<td></td>
<td>22</td>
<td>19.4%</td>
</tr>
<tr>
<td>6.0</td>
<td>Marshmallow</td>
<td>23</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

Figure 2: Android Version, Market Share: May 2016

Android has no centralized update systems to update its operating system. Most devices have a long software update lifecycle. Although Google performs periodic application scanning for malware and potential vulnerabilities, application distribution via Google Play Store is not tightly controlled. Because of which users can be exposed to vulnerabilities. Android devices can be easily infected by the applications sideloaded or downloaded from unregulated third party app stores. This can lead to security and privacy concerns. As per 2010 reports, 15 out of 30 popular apps analyzed by the Intel Labs (Taintdroid) were maliciously sending user’s geographical location to a remote server for advertising without the user being aware [2]. Some other applications were sending information related to the phone such as phone numbers, IMEI number and SIM card serial number to the developers. Certain applications collect user analytics data anonymously, which can accidently leaks user personal information. Some application directs user to infected websites by launching adware. Apps like AVPass, DroidJack, TapSnake and ZergRush are among popular malware for Android. AVPass [4] tries to steal confidential data like SMS messages, contact details, stored photos and can receive additional commands from SMS. TapSnake [5] is an adware which shows alerts that the device is infected
and ask to download fake app to remove virus. ZergRush [6] used a vulnerability in Android devices version from 2.2 to 2.6 to root the phone and hack.

The paper is organized as follows; Section 2 shows information on related research. Section 3 gives an executive summary of the research. It explains about the metrics used to calculate scores and provide recommendations. Section 4 briefly explains about design and overall system architecture. Section 5 describes the implementation and jFuzzyLogic in detail. It also lists rules used in evaluation. Section 6 provides information on testing methodology adopted. Section 7 shows test results. Section 8 provides conclusion to this research.

2. Related Work

The evaluation of Android security is challenging as there is diversity in the form of hardware to the operating system. Each device has some unique attributes which are hard to measure or identify based on the model or operating system. A different approach has been used to evaluate the security of Android devices. In the research paper Yao et al [7], the authors measure battery parameters like voltage, temperature and battery power while evaluating encryption algorithms. In the research paper Weiss et al [8] measured system parameters, battery voltage, CPU utilization, network communications for individual apps to detect malicious code. They measured network communication and analyses using tcpdump whether communication performed is secure or not, ratio of incoming traffic to outgoing traffic and third party traffic. In the research paper Sarma et al [9], the author’s analyses permissions used by the applications. They identify the permissions which are often used in malicious applications and have high occurrence among those applications. The authors in the research paper Wei et al [10], implemented a novel approach to analyze the security of Android devices. They combine several approaches to evaluate certain apps from the Google Play Store. They perform static analysis by identifying sensors used by those applications with permissions or by intent. They also account user interaction by measuring touch events intensity and swipe/press ratio to classify applications. In the research paper, Zheng et al [11], the authors use static and dynamic analysis to analyze firmwares and pre-installed applications for vulnerabilities and malware.

3. Executive Summary

SecuFone analyzes overall security and privacy of the devices, especially focused on the metrics obtained from devices and app stores. Metrics are based on the applications installed on the device, user settings, hardware configurations and on-board sensors. The metrics used in this research are as follows 1) Screen Lock, 2) SIM Lock, 3) Root Access, 4) Storage
Encryption, 5) Malware Protection, 6) API Level and 7) Installed Application. ‘Screen Lock’ is an important feature of smartphone providing various ways to lock and unlock the screen. It is always advisable to set a screen lock. Screen lock protects user from unauthorized access to the devices. If the device is in a locked state, There are certain system activities which are not allowed to execute in order to protect the user from unknown behavior. It is advised to use strong screen lock like PIN/Password or biometrics. ‘SIM Lock’ is a feature found on all Android devices which ask the user the password whenever new SIM (Subscriber Identity Module) is detected or device is rebooted. This prevents unauthorized use of the network to perform call, send SMS or obtain SIM data like IMEI number, modify PIN or read/modify SIM contacts which can cause financial loss to the user. Rooting is the process of allowing users to get root access over various Android subsystems. As Android uses the Linux kernel, rooting an Android device gives access to any files or folders in the system. This is dangerous as it can give hackers access to the whole file system and can potentially damage software. ‘Storage Encryption’ is the feature provided on Android devices to encrypt the device. Full device encryption is to protect phone data from unauthorized access. But it has limitation due to hardware design and available processing power. Due to this reason certain devices enable it by default or provide an option to the user to enable it. ‘Malware protection’ is provided by various anti-malware softwares which detects applications for malware signatures and abnormal behavior. In this application, I detect the presence of anti-malware application on the device. Android OS updates are provided by the Google or Manufacturer but not all devices are compatible or receives latest update. So there could be vulnerability related to important security bugs in the device which are not yet updated on the user device. So the OS version of the devices can be found out using the ‘API level’.

Google Play Store provides more than 1.6 billion applications. All the applications are scanned at the Play store level for malware. Applications installed on the phone plays a major role in protecting user device from malicious applications and private data. But there are also other app store which are not so effective in detecting the malicious app. I use application related data like ‘ratings of application’, ‘last update status’ and ‘source of application’ as derived parameters to identify the unreliable or potentially dangerous application. I am considering two app distribution services viz. 1) Google Play Store and 2) Amazon App Store. There is no official API available to get data from Google Play Store and Amazon App Store because of which I need to obtain data by performing web scraping.

4. Design

The application is designed as a dynamic web application using the MVC pattern. It is designed to work on Android powered smartphones and tablets having an API level greater
than 14. Application supports multiple simultaneous request, user authentication and multiple test. A user can have multiple devices of same or different types. Client devices sends test data to the server, the server then processes parameters, calculates scores using fuzzy logic and generates recommendations which are provided to the client. The below figure 1 shows the architecture of the system and interaction among components. Test data, user related information (user login details, user device details and user-device model details) and score data are stored in the MySQL server. Device information like unique device name (as seen in the bottom left area of figure 3), android ID (id used by Android to uniquely identify device) and carrier information are also stored in the SQL server. The server uses an ‘Expert System’ to compute scores. The expert system is a knowledge based system which uses inferences to solve a problem of a particular application domain.

![Figure 1: Overall architecture of the system](image)

5. Implementation

The Application client interface is implemented using Android SDK Tools 25.0 and Android Studio. Apache Tomcat is used as a web server. The application starts with a login screen as seen in figure 2 for the first time user. The user is provided with an option to register. The session is maintained using ‘SharedPreferences’. The client performs login using Android’s
‘AsyncTask’. User after a successful login, can perform the test by clicking on ‘Perform Test’ button as shown in figure 4. Alternatively the user can click on ‘Check Scores’ and ‘Check Advisory’ to view if the results from previous test are available, otherwise it does new test and show its score and advisory.

Clients connect to the Apache Tomcat using HTTP protocol. The clients send evaluation data as JSON (JavaScript Object Notation). It is generated using Google Gson. Gson is a popular library to serialize and deserialize Java objects. jsoup [12] is used to perform web scraping of application data from the app stores using ‘application package name’ to find ‘app rating’, ‘last_updated_date’ and ‘install_count’. jsoup provides a well developed methods to scrape and parse data using HTML elements, DOM traversal and CSS selectors. jFuzzyLogic’ [14] is an open source rule engine with fuzzy package. It is used in my system, to calculate scores based on rules and metric values on the server. jFuzzyLogic implements Fuzzy Control Language (FCL) [15]. It allows to optimize parameters. It supports input variable of type ‘REAL’ only. It allows to build custom membership functions. FCL elements used in jFuzzyLogic are as follows viz., 1) Function Block, 2) Variables Declaration, 3) Fuzzification of Inputs, 4) Defuzzification of Output, 5) Rule Block, 6) Rule Setting and 7) Rules.

‘Function Block’ is a top level component. It includes variable declarations, fuzzification, defuzzification and atleast one rule block. ‘Variable Declaration’ consist of input and output variables. I have used 10 input variables and one output variable. My output_variable is ‘safety_level’. ‘Fuzzification of Inputs’ is a process in which input values are converted into
partial degrees of membership using the membership functions (MF). A membership function defines how each point in the input space is mapped to degree of membership [16]. The input space is referred as the ‘universe of discourse’. jFuzzyLogic provides support for linear, triangular, Gaussian and singleton membership functions. I have used linear and singleton membership function. A singleton is a binary membership function having two values, i.e. either the element is present or absent. ‘Defuzzification of Outputs’ is a process in which conversion of linguistic value of a variable of a numeric value takes place. ‘Rule Block’ consist of ‘Rules’ and ‘Rule Setting’. jFuzzyLogic provides rule settings such as Activation and Accumulation. jFuzzyLogic allows to specify 10 rules per rule block and maximum 15 rules per file. My output variable represents a security score in a form of number between 0 and 100. There are 4 classes to describe the safety_level viz. 1) danger, 2) medium_risk, 3) low_risk and 4) safe.

Rules used by me are based on user settings, device configurations and applications. Below are my rules, used currently.

<table>
<thead>
<tr>
<th>Rule No.</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IF (root_status IS NOT unsafe OR sim_lock IS safe) AND antivirus IS present THEN safety_level IS low_risk;</td>
</tr>
<tr>
<td>2</td>
<td>IF screen_lock IS unsafe AND sim_lock IS safe AND storage_encrypt IS good THEN safety_level IS safe;</td>
</tr>
<tr>
<td>3</td>
<td>IF screen_lock IS safe OR root_status IS unsafe THEN safety_level IS medium_risk;</td>
</tr>
<tr>
<td>4</td>
<td>IF (api_level IS old) AND ((antivirus IS absent) AND (sim_lock IS unsafe)) THEN safety_level IS danger;</td>
</tr>
<tr>
<td>5</td>
<td>IF (avg_app_rating IS good) AND (avg_app_update IS good) THEN safety_level IS low_risk;</td>
</tr>
<tr>
<td>6</td>
<td>IF (avg_app_rating IS good) AND ((antivirus IS present) AND (root_status IS safe)) THEN safety_level IS safe;</td>
</tr>
<tr>
<td>7</td>
<td>IF avg_app_rating IS poor AND avg_app_source_null IS NOT low THEN safety_level IS danger;</td>
</tr>
<tr>
<td>8</td>
<td>IF (avg_app_rating IS good) AND ((avg_app_source_null IS low) and (avg_app_update IS poor)) THEN safety_level IS medium_risk;</td>
</tr>
<tr>
<td>9</td>
<td>IF ((api_level IS good) OR (api_level IS latest)) AND (avg_app_rating IS NOT poor) THEN safety_level IS safe;</td>
</tr>
<tr>
<td>10</td>
<td>IF ((battery_health IS NOT bad) OR (storage_encrypt IS good)) AND (avg_app_source_null IS low) THEN safety_level IS low_risk;</td>
</tr>
</tbody>
</table>

Table 1: Rules
User can view their scores by clicking on the ‘Check Scores’ button, in case it is not shown directly after performing a test. Scores have a range from 0 to 100, 100 being better. Scores are relative to each other. Using test data, the mean score is also provided for the device model and for global data as seen in the below figure 5. Based on the scores, the user is provided with the advisory to improve the scores as seen in the below figure 6. Advisory are generated based on parameters and score. Advisory can be broadly classified as application based advisory and setting based advisory. Application based advisory considers the source and the region of the application. These gives users an overview of the security and privacy of their devices.

![Figure 5: Score Activity](image1)

![Figure 6: Advisory Activity](image2)

**6. Testing**

The testing consists of generating a score and advisory for different devices and measuring time taken for various functionalities for a given test. I also tested the application for establishing connection with the server in poor Internet connection environments, generating logs and appropriate session. I used three devices viz., 1) LG Optimus Black (LG-P970), 2) Google Nexus 7 (2012) and 3) LG G3 (LG-D850). LG Optimus Black was released in 2011 with API Level 10 (Android Gingerbread). It was running an Android API level 15 while testing and had 15 user-installed applications. Nexus 7 was released in 2012, received positive reviews for its pure Android experience, responsive display and high performance. The device used for testing had only WiFi support and 16 user-installed applications. LG G3 was released in 2014, having Quad
HD display and laser auto-focus system. It was running API Level 22 (Android Lollipop) and had 69 user-installed applications.

During evaluation, I am also measuring the time taken in milliseconds to execute complete tests and certain functionalities like 1) generating parameter values on client, 2) generating scraping, 3) generating score using jFuzzyLogic and 4) time taken on a client to get a response from the server. I also rerun the test to check fuzzy logic component for any random behavior. I also installed/removed some applications to see how it affects final score and advisory generation.

7. Results

The total average time to compute the score and advisory is 15 seconds out of which roughly 12 seconds is for performing web scraping. Scores represented in table 2 below are rounded half_up to two decimal places.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Manufacturer</th>
<th>Model</th>
<th>Api_Level at Test</th>
<th>Test Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LGE</td>
<td>LG-P970</td>
<td>15</td>
<td>32.58</td>
</tr>
<tr>
<td>2</td>
<td>Asus</td>
<td>Nexus 7</td>
<td>23</td>
<td>60.63</td>
</tr>
<tr>
<td>3</td>
<td>LGE</td>
<td>LG-D850</td>
<td>22</td>
<td>64.67</td>
</tr>
</tbody>
</table>

Table 2: Test Results

8. Conclusion

This paper contains detailed information on the metrics involved in this research for analysis of the Android devices. This research helped me to understand Android architecture and the API’s provided by Android SDK. Due to increased data consumption and network bandwidth usage, security has become more important. The client-server distributed architecture helps to process parameter values efficiently by using rule engine on the server. It is ideal for low computing devices. The use of a rule engine is highly scalable and allows to write rules in a plain language which is flexible. It is faster and easier to manage than implementing rules directly in Java.

9. References
   Number of Apps available in leading Play Store
[2] Intel Labs, Penn State University and Duke University
Realtime Privacy Monitoring on Smartphones
DOI= http://appanalysis.org

[3] Data gathered from Google Play Store app
DOI= http://developer.android.com/about/dashboards/index.html

DOI= http://forensics.spreitzenbarth.de/android-malware

DOI= http://www.symantec.com/android-tapsnake

DOI= http://www.microsoft.com/security/ZergRush

[7] H. Yao, L. Lian, Y. Fan, Q. Liang and X. Yan
The Evaluation of Security Algorithms on Mobile Platform
DOI= http://dx.doi.org/10.1109/MSN.2013.64

Trust Evaluation in Mobile Devices: An Empirical Study
DOI= http://dl.acm.org/citation.cfm?id=2847604.2848359

Android Permissions: A Perspective Combining Risks & Benefit

ProfileDroid: Multi-layer Profiling of Android Applications
DOI= http://dl.acm.org/citation.cfm?id=2348563

DroidRay: A Security Evaluation System for Customized Android Firmwares
DOI= http://dl.acm.org/citation.cfm?id=2590313

[12] jsoup Java HTML parser
DOI= https://jsoup.org/apidocs

DOI= https://www.ntu.edu.sg/JavaServlets.html

[14] jFuzzyLogic Design Controllers
DOI= http://jfuzzylogic.sourceforge.net/html
[15] Fuzzy Control Language (IEC 1131-7 CD1)

[16] Design Fuzzy Logic Systems
    DOI=http://www.mathworks.com/help/fuzzy

[17] Android API Guide and Samples

[18] Android Official Supported Devices
    DOI=https://support.google.com/googleplay/1727131