Hierarchical expert system for security evaluation and its implementation on an Android smartphone

Sarang Narkhede, Ayush Soni, Hitesh Vaidya
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
{sn7330, as2425, hv8322}@rit.edu

***Abstract*—The second decade of the 21st century has seen a tremendous surge in the production and usage of smartphones. Due to the increasing demand, popularity and competition among smartphone manufacturers, mobile phones are being packed with great number of advanced features in incredibly small form factor. This exponential growth of technology in smartphones has also given rise to various security threats.**

In this paper, we first analyze the related literature and draw insights about various security factors discussed in these studies, while also briefly comparing these studies with the others. A list of security metrics is then derived from these, which are to be used in the project.

Finally, the implementation details of the project are discussed. The project is divided into two phases, the first being a rule-based Expert System approach for security evaluation. Since the compatibility of rule-engines for developing Expert Systems for Android is very limited, a better and faster Machine Learning based approach is used, which uses an Artificial Neural Network that is trained to effectively approximate the previously stated rule-based expert system.

Keywords—artificial intelligence; android smartphones; android security; hierarchical expert system; mobile security; machine learning.

I. INTRODUCTION

With the increased globalization and availability of internet, smartphone market has penetrated worldwide. The total number of mobile users worldwide has reached the mark of 2.71 billion and is expected to reach 2.87 billion by 2020 [5]. With the increasing innovation, mobiles companies have been offering users with features like Wi-Fi, Bluetooth, location services, camera, sensors and various mobile software applications. These features have opened tremendous possibilities for smartphone users. Many companies like Google, Microsoft, Apple, Ubuntu, Mozilla, Nokia, Samsung tried offering their OS for mobile devices. However, by 2019 android has captured 75% of the international mobile market [6]. With this surge in the number of smartphones, there has also been an increase in the number of cyber-attacks on these devices. Hence, it is important to pay utmost attention towards the security of mobile phones.

The majority of work done so far on the security analysis of Android devices focuses mainly on the lines of mobile application analysis, host-based intrusion detection, encryption, malware detection etc. Right from application level to hardware level, there have been efforts to maintain the security of android devices.

In this paper, we propose a rule based and a machine learning based hierarchical expert system to evaluate security of android devices. Within this system we consider different metrics that could be used to form a general picture of how secure a respective mobile device is. Using various metrics, we calculate a numerical score that estimates how secure a mobile device is. The idea behind using various metrics is to develop a solution that covers all the aspects of system security. Attackers constantly try to attack a device using novel techniques. It is therefore important to constantly keep on updating the security solution. The use of aforementioned

hierarchical expert system would make it possible to update the security system, thus making it more robust.

The rest of the paper is organized as follows. In the next section, an analysis of important metrics described in related literature is discussed and their potential in evaluating the security of the Android OS is analyzed. Section III introduces the final list of metrics to be used with the Expert System and the Machine Learning model. It also describes the various APIs of Android SDK or Google that can be used for the acquisition of these metrics. Section IV mentions the implementation details of Phase 1 and Phase 2 of the projects. Finally, the paper concludes with section V.

II. Literature Review

In their paper[5], A. Hayran et al. discuss the factors that influence security within iOS and Android. This evaluation covers many factors that are relevant to the Android OS, and are important in determining the security of the platform. They discuss the importance of *Application Distribution Channels*, that are used by consumers to download or install third party applications on their devices. For Android, there are two possibilities: Google Play Store and Other Sources. Other sources may include installation from device's storage, emails, web browsers, etc. The Applications published on Google Play Store are verified using Google Play Protect and every application publisher is required to go through the registration and vetting process. This ensures higher security for the application users, as compared to apps installed from other sources. The vulnerability due to *Attack Surfaces*, is also discussed in the study. Attack Surfaces are the pieces of code in apps that can be used by attacker to remotely exploit these apps and inject malicious code. Hence, the goal is to minimize such attack surfaces. Detection of such vulnerabilities however can be tricky and requires analysis of decompiled application code, and has to be repeated for all the application installed on the device, making it hard to be used as a metric for security evaluation. The use of *Privilege Separation* is also discussed. Due to Privilege Separation in Android, the third part applications have limited permissions, and cannot modify system files, while the important system processes run with root permissions. However, there are exploits available to bypass this mechanism, allowing applications to gain root access to the environment they are running on. This poses serious threats to the user and device, making it an important factor to consider for security evaluation. The authors also discuss *Permission-Based Access Control*. After API level 23 in Android, a user is prompted to allow individual permissions to the app, allowing a more fine grained user control over what the app can access. Geo-location access, discussed later in the paper also falls in this category, and can be used by malicious apps or web browser scripts to access private information about user's location. Such location access should only be given to trusted apps. Detection of malicious permission requests requires us to know whether an app genuinely requires such permission for its functioning, or is it a dangerous request. This is a challenging task, and requires a database of already known malicious apps for detection, again making it a hard metric to evaluate. *Data Encryption* is another factor considered by the authors. In Android, on API level 24 and above, File-based encryption is available as an in-built feature. Using such encryption on file system greatly decreases the possibility of spoofing attacks, making it a good metric for evaluation. Finally, the study discusses the detection of *Malware-ridden Malicious Apps*, that perform harmful tasks on device without user consent. Such apps again require matching against a database of known harmful apps for detection. Thus, overall, this study gives us some metrics that are difficult and complex to evaluate, but also some that are simple yet very effective. For our use, we focus more on the the later metrics, which are listed in the Section III.

Another study[6] by D. Vecchiato et al. proposes an approach for assessment of Android OS security using user defined system configurations or settings. This paper provides a long list of 41 security configurations/settings that are available for users to change, divided into 7 distinct categories: Application, Browser, Content, Network, Password, Permission and System. These configurations and their security impact were selected after rigorous consultation form eight security experts from academia and industry. Based on the potential of these configurations to be useful as security metrics, and the feasibility of implementation, we select a subset of eight most important configurations, as given in table 1. As can be observed from the table, some of these metrics- Rooted Device, Unknown Sources and Device Encryption- overlap with the ones discussed in [5], reaffirming their usefulness

TABLE I. A LIST OF USEFUL SECURITY METRICS AND THEIR POSSIBLE VALUES, DERIVED FROM [6]

Metric	Possible Values
Screen Lock	Yes, No
Rooted Device	Yes, No
Unknown Sources	Enabled, Disabled
Device Encryption	Enabled, Disabled
Developer Options	Enabled, Disabled
SIM PIN Lock	Enabled, Disabled
Bluetooth Discoverable	Yes, No
Mock Location	Enabled, Disabled
ADB over USB	Enabled, Disabled

for security evaluation. There are also some newer metrics, that are discussed here. *Screen Lock* is the most primitive but effective factor to block complete access of device from unauthorized people. The *Developer Options* should only be enabled for application or platform developers who require access to some advanced in-built features required for development and testing of their code. For all other common users, this setting should be disabled, as intruders can use some of this sensitive features to perform harmful actions on device. The SIM card should preferably be locked using a *SIM PIN Lock* code, as it can have information related to user's phone number, security data, billing information, and some other bits of sensitive data. Whenever not needed, *Bluetooth Discoverability* should be disabled, to protect the device from attacks such as Bluejacking, Bluesnarfing, and Bluebugging. These attacks allow intruders to send unsolicited messages from device, access information such as emails, contacts, text messages, and even take control over the device. *Android Debug Bridge(ADB) over USB* is used by developers to have access to device and perform actions via a PC without using the device. As powerful as this tool is for developers, it is equally dangerous when an intruder somehow gains physical access to device. Thus, ADB over USB should be disabled when not needed. Thus, in summary, this paper provides significantly more metrics, which are also easier to implement and still play a key role in determining system security, in comparison to [5].

To protect devices from OS and other suspicious modifications/tempering, Android uses Secure Boot that uses a chain of bootloader certificates, where the first(root) bootloader verifies the second, the second verifies the third, and so on till the execution reaches Applications Bootloader(ABOOT). This is discussed elaborately in [7] by R. Hay. Apart from this normal chained bootloader execution, ABOOT can also be executed using the Fastboot interface, available in most Android devices with few exceptions where the OEMs implement their own interface. This interface allows manufacturers to implement features such as Bootloader Unlocking, Locking and ROM flashing. Since ABOOT is a critical stage of Android booting mechanism and executes before the Operating System, Fastboot interface usually requires user interference. However, this interface is vulnerable, can be exploited and attacked as described in [7]. The most important consequence of this is that the attacker can breach the Bootloader lock- which ensures the integrity of Device and OS using Secure Boot. Hence, bootloader lock status gives crucial information about system integrity or device tampering. Thus, in comparison to [1] and [2], this study focuses only on the vulnerabilities in Application Bootloader and Fastboot interface, rather than focusing on multiple security parameters, and effectively implies the high significance of Bootloader Lock status as a metric in system security evaluation.

An important factor that is at the root of android security in general is the current OS version or API level. One of the easiest steps a user can take to maintain device's security is keeping the device up-to-date with the latest On-The-Air OS version upgrades and Security Patches. An older OS version in an Android device can have many vulnerabilities. This is because all the bugs and loopholes of an older OS become well known to attackers. The patches to these loopholes are most of the times released in the newer version upgrades or security patches. Unfortunately, not all the manufacturers provide regular OS or software updates. Despite many mobile phones getting sold on 12-24 month contracts, they happen to receive just 1.26 updates per year on an average [4]. This leaves a device unpatched for a long period of time. Therefore, there is a direct correlation

between the current OS version or API level of an android device to its security. Finally, in comparison to previously discussed papers and similar to [3], this paper specifically draws attention towards one important factor- regular OS updates- rather than discussing multiple factors, giving us the OS version or API level as an important metric to consider while evaluating the device's security.

III. Proposed Metrics And Acquisition Methods

A. Proposed Metrics:

In this section, we propose the final list of metrics for use with the AI-based project. These metrics are selected after analyzing the various metrics discussed in the literature survey for their potential in determining the system security and also their feasibility and ease of implementation in Android devices. The metrics are described in Table II.

TABLE II. Table of Metric

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

These metrics can be put into clusters, depending on which system component's security they address. This gives us a hierarchical collection of metrics as shown in figure 1.

B. Metrics Acquisition Methods:

The metrics mentioned in Table II can be acquired using various APIs in Android SDK and Google's APIs for Android. Whether Installation from Unknown sources is enabled or disbaled can be extracted using *android.provider.Settings* API with the key *Settings.Secure.INSTALL_NON_MARKET_APPS*. Whether device filesystem is encrypted or not can be checked using the *android.app.DevicePolicyManager* API's *getStorageEncryptionStatus()* method. The list and number of Harmful Applications can be obtained using Google's *SafetyNet Verify Apps* API. To check whether Screen Lock is present, the *android.app.KeyguardManager* class can be used. The Developer Options state can be found similar to Unknown Sources settings using the key *Settings.Global.DEVELOPMENT_SETTINGS_ ENABLED*. The status of SIM PIN Lock can be obtained from the *android.telephony.TelephonyManager* class using the *SIM_STATE_NETWORK_LOCKED* constant. Bluetooth Discoverability can be determined using the SCAN_MODE_CONNECTABLE_ DISCOVERABLE constant in *android.bluetooth. BluetoothAdapter.* The status of ADB over USB(or USB Debugging) can be determined using the Settings API's *Settings.Secure.ADB_ENABLED* constant. The API level can be found from the static class *android.os.Build.VERSION*, using *SDK_INT* constant. Google Play Protect status can again be found using the *SafetyNet Verify Apps.* Finally, to determine the integrity of the device, considering factors such as Root Access(presence of /system/bin/su or /system/xbin/su), SELinux status(whether /sys/fs/selinux/enforce is present), and Bootloader Lock status, the *SafetyNet Attestation* API can be used.

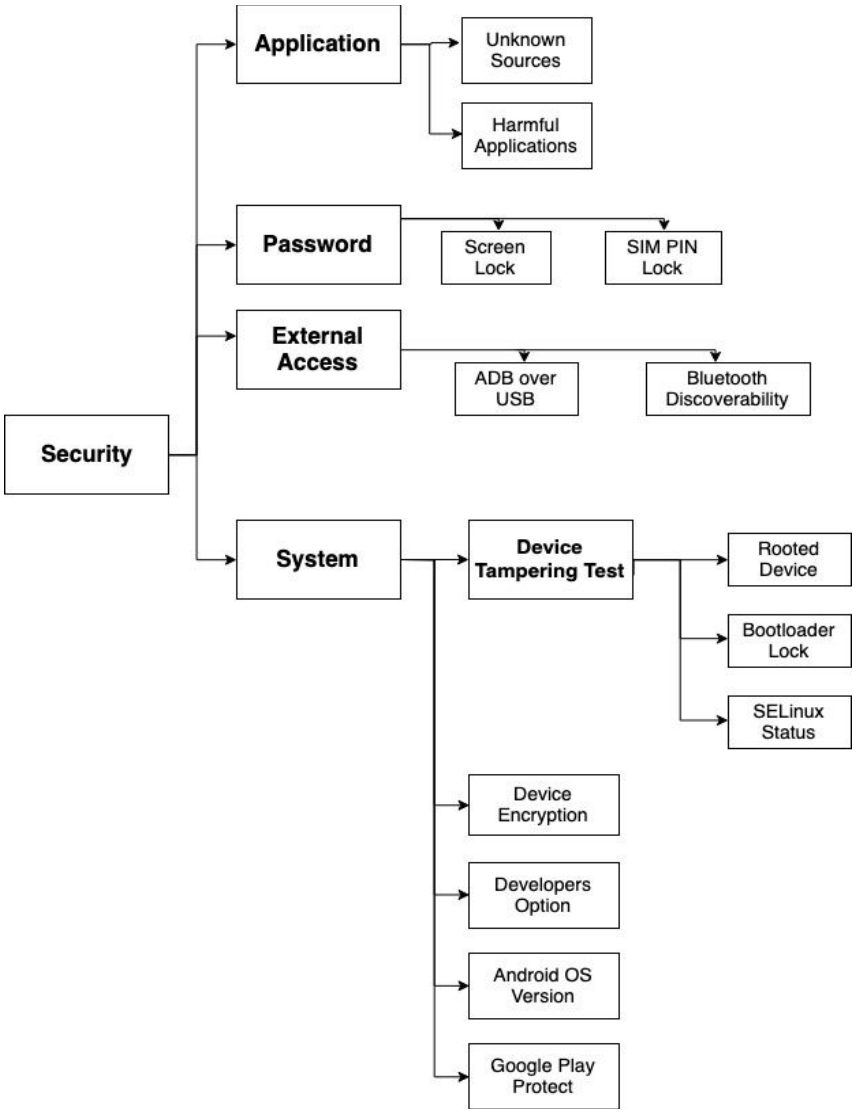

Fig. 1. The hierarchy of metrics to be used in the project.

IV. Project Specification

The project has been divided into two phases. Both phases incorporate different approaches to achieve the end goal.

Phase 1 will consist of designing and development of rule-based Expert System. Whereas, Phase 2 consist of design and development of neural network that approximates the expert system developed in phase 1.

A. Phase I

In phase 1, a rule-based expert system will be developed which is developed using handcrafted rules. The expert system will make use of facts put into the knowledge-base, and rules that will be run using a rule-engine. In total, approximately 12 to 15 rules will be used for expert evaluation system. Some of these rules depend on the outputs of other ones. This dependency will form a chaining mechanism. The final output as a result of the execution of all the rules will be a single evaluation score given to the input metrics.

The rule engine used for this expert system is PyCLIPS. PyCLIPS is an extension module for Python language that encapsulates and embeds complete CLIPS functionality in Python applications.

CLIPS is often taken as "reference implementation" of an expert systems shell by many programmers and groups involved in AI development. It has a forward chaining rule-based inference system, as well as all

imperative and object-oriented constructs that allow full control of the execution flow.

We will develop and run phase 1 of expert system in a linux desktop environment.

B. Phase Ⅱ

In phase 2, the same concept of developing an expert system will be used but instead of using handcrafted facts, statistical machine learning models will be taken into consideration. In this phase a neural network will be built which will approximate the findings developed in the expert system from the phase 1.

The model will be developed using Tensorflow library. The number of layers and neurons in each layer will be iteratively evaluated and improved on a linux machine until the Neural Network reaches the performance similar to that of the expert system. Once model tuning is complete, a frozen computation graph of this model will be exported to a protocol buffer (*pb) file. This graph will then be converted to a Tensorflow Lite model, will be used for performing inference on the android device.

For the deployment of this inference graph, Google's Firebase Machine Learning kit will be used. Firebase API for Android provides useful libraries to develop high-quality applications and solve common application development challenges.

Machine Learning Kit is a mobile Software Development Kit (SDK) that brings Google's machine learning expertise to Android and iOS apps in a powerful yet easy-to-use package. It also provides very convenient Application Programming Interfaces (APIs) that helps to develop custom TensorFlow Lite models in mobile applications. ML Kit comes with a set of ready-to-use APIs for common mobile use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images, and identifying the language of text.

To summarize phase 2, Firebase ML kit will be integrated with an android application and will use frozen computational graph or model to perform on-device inference, on the metrics gathered by the application.

V. Conclusion

In this paper, we discussed and analyzed various security metrics described in related literature, and finalized a subset of them for use with the project. We also described how a rule-based expert system can be developed to run on these metrics, and how can a Neural Network, which is faster and more efficient than a rule-based expert system on the Android device, can be used to approximate the Expert System and finally deployed for security evaluation. Thus, in conclusion, it can be said that by gathering crucial information about the environment from the OS, and by designing a system that can analyze this information, we can effectively provide the users an insight about the overall level of security of their device, and also a breakdown of the security level of individual system components.

VI. References

[1] Hayran, Ahmet & İğdelİ, Muratcan & Yilmaz, Atif & Gemcİ, Cemal. Security Evaluation of IOS and Android. In *International Journal of Applied Mathematics, Electronics and Computers* (2016), pp. 258-258.

[2] D. Vecchiato, M. Vieira and E. Martins, Risk Assessment of User-Defined Security Configurations for Android Devices. In *IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (2016), pp. 467-477.

[3] R. Thomas, Daniel & Beresford, Alastair & Rice, Andrew. Security Metrics for the Android Ecosystem (2015), pp. 87-98.

[4] Hay, Roee. fastboot oem vuln: Android Bootloader Vulnerabilities in Vendor Customizations." *WOOT* (2017).

[5] Number of smartphone users worldwide from 2014 to 2020 (in billions). Retrieved June 01, 2019, from https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

[6] Mobile Operating System Market Share Worldwide, Retrieved from http://gs.statcounter.com/os-market-share/mobile/worldwide

[7] Number of Android apps on Google Play, Retrieved on June 01, 2019, from https://www.appbrain.com/stats/number-of-android-apps