IDENTIFICATION SYSTEM FOR PERNICIOUS CODE ON ANDROID USING STATIC TECHNIQUES

OBAYI ADAORA ANGELA.
Department of Computer Science University of Nigeria, Nsukka (UNN),

AGBOKE ADEOLA J.
Department of computer science, Cabel University, imota, lagos state.

AJAYI, TAIWO DAVID
Department of computer science, Cabel University, imota, lagos state.

UZO BLESSING CHIMEZIE
Department of Computer Science, University of Nigeria, Nsukka (UNN),

IKEDILO, OBIORA EMEKA
Department of Computer Science, Akanu Ibaim Federal Polytechnic, Uwana.

UGWU, MAUREEN. K
Department of Computer Science, Enugu State polytechnic, Iwollo.

NLEBEDIM, EMMANUEL ARINZE.
Real brain point schools

ABSTRACT

Android Operating System is widely used mobile OS in the world. There is a high increment in pernicious apps in android phones. This paper is gear towards detecting malware application and proposes a technique that can detect any malware application in android phone using static techniques. It analyzes system calls' logs and also the conduct of an app and afterward produces signatures for malware conduct. This research work provides an effective and efficient technique to detect malicious code in Android Application. The system was developed using Android Studio, Android SDK written with Java and XML. The Object Oriented Analysis and Design Methodology (OOADM) were used for the analysis, design and development of the system using Unified Modelling Language (UML) to model the system.

Keywords: Android application, Malware detection, static techniques.

1.0 INTRODUCTION

Smart devices is the current dominating personal computing device with so many features that are almost equivalent to mini computers such as phone calls, short message service, multimedia messaging service, email, video calling, voice dictation, mobile banking, file exchange, internet browsing etc. As indicated by Pew Research Center in 2015, about 43% of the worldwide populace utilizes a cell phone gadget. Android is an open source working framework for cell phones and tablets. It was dispatched by Google and Open Handset Alliance in September 23, 2008. Android has experience a tremendous development since its initiation in view of its ease of use, open source, simplicity of creating and distributing applications. Android has turned into the most broadly utilized working framework on Smartphones with an expected piece of the pie of 81% in 2015 [1]. The universal use of Android OS has actuated the explosion of versatile application market. Google Play is the biggest application store followed by Apple's App store. Android Applications are accessible for download in Google PlayStore and outsider specialists. The fast development of cell phone advances and their far reaching client acknowledgment came all the while with an expansion in the number and complexity of malevolent programming focusing on famous stages. Malware has been on the increment because of the prominence of Android telephones and the open nature that empowers engineer to foster an application and distribute it in the Android market. Toward the finish of first quarter of 2016, the security firm G Data expressed that a sum of 440,267 new malwares were identified on Android telephone [2]. Malware creators have assorted impetuses for making versatile malware going from exploration, curiosity and entertainment, monetary benefit, conservative, political, selling client data, taking client qualifications, website streamlining, settling on premium-rate decisions, sending premium rate SMSs, SMS spam and so forth Exploration shows that the vast majority of the applications on the outsider application store are repackaged renditions of the first applications present in Google PlayStore. Android utilizes consent framework to limit priviledges allowed to applications during introduce time, yet is has demonstrated deficient up until now. Android takes on a go big or go home consent conceding strategy, in that clients are left with either allowing all the authorization it demands or denying it which implies the applications won't be introduced. In view of the need of the applications, a few clients just don't focus on the authorization framework while others don't completely get what every consent implies.

2.0 Theoretical Background

Android is an open-source operating system for mobile phones, tablets etc. It was built based on Linux kernel, developed by Google and released on September 23, 2008 [3]. Android offers a friendly development environment through a variety of tools such as Android Software Development Kit (SDK), Android Native Development Kit (NDK), Android Debug Bridge (ADB), Android Developer Tools (Eclipse). Google PlayStore is the official distribution center for Android Apps which are developed by Google or third-parties. It allows Android users to browse, install, and update the apps.

2.1 Android Permission Model

The permissions model is the primary security idea that Android security depends on. Android runs applications independently in sandboxes. Each application is run in a secluded climate where it has no admittance to the framework's assets. The consents should be given to the application to have the option to access and utilize framework assets that are needed for its usefulness. Every one of the authorizations that should be announced in the AndroidManifest.xml document by the engineer in the advancement stage. These authorizations additionally should be conceded by the framework or the client at the establishment time; whenever they are allowed they couldn't be changed except if the application is uninstalled by the client. The consents can be pronounced in at least one <permission> labels in the

AndroidManifest.xml document; they additionally should be characterized with required and discretionary qualities. The <label>and <description>attributes are the strings that are shown to the client at establishment time. These credits should be unmistakably characterized to help the client in understanding the advantages that the consent demonstrates. The <permissionGroup> quality is discretionary and utilized by the framework to show the classification of the authorization to the client. The <protectionLevel> quality is needed to distinguish the security level of the authorization; it characterizes the criticality of the application advantages. The authorizations model uses the two <uses-permission>and <permission> labels to oversee applications' admittance to the framework assets and other application's information, separately. The <uses-permission> tag characterizes the consents that the application needs to get to explicit information, equipment, programming and other framework assets. Then again, the <permission>tag characterizes the authorizations that other applications need to approach the application's information and parts. As such, it characterizes how the application's parts can be open by the other applications.

2.1.1 Android Malware

A pernicious application or malware refers to an application that can be used to compromise an operation of a device, steal data, bypass access controls or otherwise cause damage on the host terminal [9]. Below are Android malware threats:

1. **Spyware**
2. **Grayware**
3. **Fraudware**
4. **Trojan**

1. Spyware

Spyware is extremely normal in the Android stage, it is intended to get delicate data from a casualty's framework and move this data to the aggressor. Spyware can be business and noxious. Business spyware are applications introduced on the client's handset physically by someone else specifically to keep an eye on the client, while noxious spyware secretively take information and communicate it to an outsider.

2. **Grayware**

The main purpose of grayware is to keep an eye on clients who introduced the product all alone in light of the fact that they felt that it is authentic programming. This is incompletely right in light of the fact that the creators incorporate genuine usefulness as publicized. All things considered, they likewise gather data from the framework, for example, the client's location book or his perusing history.

3. Fraudware

Fraudware-based applications are installed by tricking the user to install a legitimately looking application, which will gain full functionality after sending several premium-rated SMS messages.

4. Trojan

Trojans are applications that claim to be helpful, however perform noxious activities behind the scenes, for example, downloading extra malware, adjusting framework settings, or tainting other files on the framework. Android malware is generally Trojans. The assault vectors utilized by infections and worms are generally inaccessible to malware engineers in light of the sandboxing model. The malignant code is typically included into authentic applications, which are then rearranged as the first application.

Applications utilized for this reason for existing are regularly paid applications reallocated as free applications on outsider business sectors.

2.2 Review of Related Literature

In this section, we survey some of the previous approaches used by researchers for detecting pernicious applications. Various approaches have been used to detect malicious applications.

This [4] study presented a smartphone double guard insurance structure that permits official and elective Android Markets to recognize pernicious applications among those new applications that are submitted for public delivery. This system comprises of workers running on mists where engineers who wish to deliver their new applications can transfer their product for check reason. The confirmation worker first uses framework call measurements to recognize expected noxious applications. After check, in the event that the product is spotless, the application will be delivered to the pertinent business sectors. The exploratory outcomes utilizing 120 test applications (which comprise of 50 malware and 70 ordinary applications) show that we can accomplish 94.2% and 99.2% exactness with J.48 and Random timberland classifier separately utilizing this system.
According to [5], proposed another structure to get and dissect cell phone application action. They found that checking framework calls is perhaps the most dependable methods for deciding the conduct of Android application. The creator fostered a lightweight customer called Crowdroid. This application utilizes publicly supporting way of thinking where a client sends non individual however conduct related information of every application they use to the worker. This is trailed by malware identification dependent on the call vectors by the worker. The exploratory outcomes completed by the writer had 100% identification rate for self composed malware.
In [6], presented TaintDroid, a proficient, framework wide data stream following apparatus that can all the while track numerous wellsprings of delicate information. We additionally utilized our TaintDroid execution to concentrate on the conduct of 30 well known outsider applications, picked indiscriminately from the Android Marketplace. Our review uncovered that 66% of the applications in our review show dubious treatment of delicate information, and that 15 of the 30 applications revealed clients' areas to remote promoting workers.
In [7], a simple, and yet highly effective technique for detecting malicious Android applications on a repository level was proposed. The technique performs automatic classification based on tracking system calls while applications are executed in a sandbox environment. The technique was implemented in a tool called MALINE, and performed extensive empirical evaluation on a suite of around 12,000 applications.
This [8] paper eliminates six sorts of information Permission, Intent channel, Intent channel, Process name, Intent channel, number of renamed approval from show records and uses them to perceive Android malware. Results show that the strategy can distinguish dark malware tests that are indistinct by a direct imprint based technique. This philosophy is unassuming to execute considering the way that single the show archive is analyzed.

According to [9], presented a fast, versatile, and exact system for Android malware acknowledgment and family recognizing evidence reliant upon lightweight static examination. DroidSieve uses significant survey of Android malware to build convincing and solid features sensible for computational learning. Their revelations show that static examination for Android can prevail regardless, when gone facing with indefinite quality frameworks, for instance, reflection, encryption and continuously stacked nearby code. While significant changes in characteristics of malware stay a generally open issue, DroidSieve stays adaptable against top tier jumbling frameworks which can be used to quickly deduce new and syntactically interesting malware varieties.

In [10], Presents an assent based Android malware recognizable proof system, APK Auditor that uses static assessment to depict and arrange Android applications as considerate or malignant. APK Auditor contains three portions: An imprint information base to store removed information about applications and assessment results, an Android client which is used by endusers to give application examination requests, and a central worker at risk for talking with both imprint data set and wireless client. 8762 applications were used to test structure execution. Result shows that APK Auditor can perceive most striking malwares and elements the ones with a potential in around 88% accuracy with a 0.925 specificity.

This [11] study presented a mobile phone twofold gatekeeper protection framework that licenses official and elective Android Markets to perceive malignant applications among those new applications that are submitted for open release. This construction contains workers running on fogs where creators who wish to release their new applications can move their item for affirmation reason. The affirmation worker first uses system call estimations to perceive likely vindictive applications. After affirmation, in case the item is great, the application will by then be released to the huge business sectors. The exploratory results using 120 test applications (which involve 50 malware and 70 average applications) show that we can achieve 94.2% and 99.2% precision with J.48 and Random forest classifier exclusively using this framework.

[12] In this paper we have proposed one more framework to procure and look at phone application development. They tracked down that noticing structure calls is perhaps the most exact methodologies for choosing the lead of Android application. The maker developed a lightweight client called Crowdroid. This application uses openly supporting perspective where a customer sends non individual yet lead related data of each application they use to the worker. This is followed by malware acknowledgment reliant upon the call vectors by the worker. The exploratory results finished by the author had 100% area rate for self-formed malware

.In [13], Presented TaintDroid, a capable, structure wide information stream following gadget that can meanwhile follow different wellsprings of fragile data. We also used our TaintDroid execution to inspect the lead of 30 notable pariah applications, picked erratically from the Android Marketplace. Our examination uncovered that 66% of the applications in our assessment show dubious treatment of tricky data, and that 15 of the 30 applications definite customers' regions to remote advancing workers.

In [14], a basic, however at that point significantly fruitful system for perceiving malicious Android applications on a store level was proposed. The technique performs modified classification subject to following structure calls while applications are executed in a sandbox circumstance. The strategy was completed in a contraption called MALINE, and performed wide definite evaluation on a set-up of around 12,000 applications.

3.0 **Static Techniques**

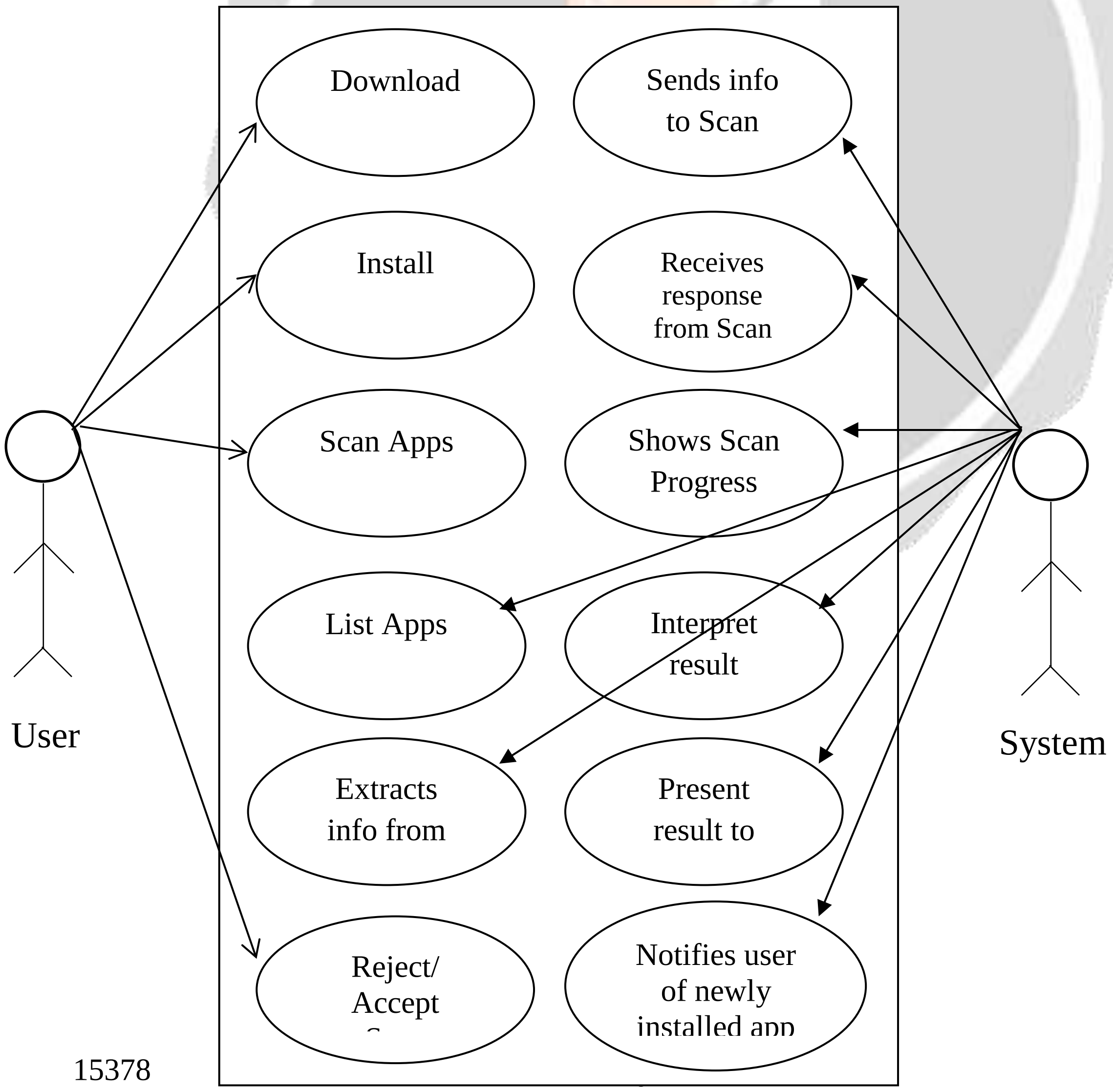
Static techniques extract features from the application's record without executing the application to distinguish malevolent examples. From the application's source code, many provisions are extricated like consents, broadcast collectors, APIs, aims, information stream, control stream, equipment parts and so on The most generally utilized static elements are the Permission and API calls. Since these are removed from the application AndroidManifest.xml and impact the malware recognition rate to a serious degree.

3.1 Analysis of the proposed system

The proposed system is an application that runs on an Android OS. We propose an Android malware location framework that distinguishes malignant applications precisely. The proposed application is utilized by Android telephone clients, specialists and the overall Smartphone clients. Thinking about the constraint of the current framework, the proposed framework looks to address a portion of the inborn issues distinguished by adequately checking applications and identifying malignant applications. The proposed framework plays out the accompanying movement to check and distinguish malware in Android Apps. Get the list of already installed applications.

1. Extract the source_dir of the application
2. Upload the source_dir on VirusTotal database
3. Receive response from VirusTotal database
4. Interpret the result to user/researcher
5. Present result to user

Apart from the ability of the system to scan application, it also educates the end user on necessary security tips to keep their device and file safe. Below is the flow of the system.



3.2 System Architecture

The system will be designed based on a typical 3-tier system architecture. The presentation tier, the middle tier and the data tier. The presentation tier shows the programming that provides the graphical user interface (GUI) and application-specific entry forms or interactive windows. The middle tier is tier that performs the runs the code which acts as an intermediary between the presentation and data tier. The data tier is the repository for date needed to be presented to the user.

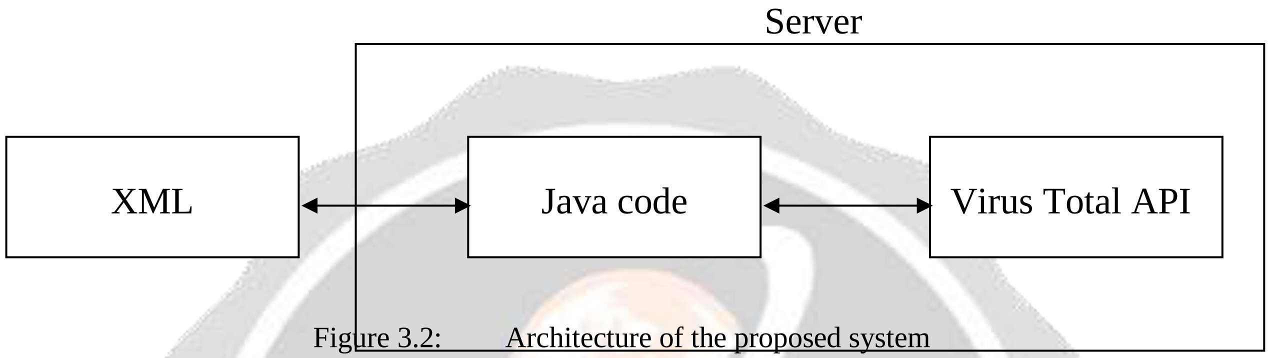


Figure 3.2: Architecture of the proposed system

4.0 Result and Discussions

This paper portrays how the construction from the main area is completed with focus to give a powerful system to recognize vindictive Android applications. The framework utilization uses the construction delivered during building plan identified with the delayed consequences of structure examination to foster a system that meets the end-customers' pre-necessities. Gadgets and methodology used to complete will be introduced at this moment. The framework clearly shows that the introduction of the system is exact and it shows promising results similar to low computational expense and high result. It is a convenient application created utilizing Java. After the application has been sent, you select application to examine, by then eliminate information from the application and send data to channel. You will get a responses result and if the application is pernicious it will uninstall subsequently. These modules realize a part that isn't seen in past work. After a customer actually take a look at an application, it makes a result that contains, the idea of the application – liberal or pernicious, the gathering of the malware, the damage it might cause to the contraption/archives and an option to uninstall the application.

5.0 Conclusion

This research gives a viable and proficient strategy to recognize malevolent code in Android Application. The framework was created utilizing Android Studio, Android SDK composed with Java and XML. Versatile malware is obviously on the ascent, as assailants explore different avenues regarding new plans of action by focusing on cell phones. This increment is now and again joined by refined procedures deliberately intended to beat security structures and location components. This examination work concentrates on different techniques and approaches that have tended to Android Malware, planned, dissected and fostered a proficient and successful component to identify Android malware with a part to teach clients on the vital security tips to protect their gadget and documents from assault.

REFERENCE

[1] M. (Business I. Rosoff, “IDC smartphone OS market share - Business Insider,” 2015. [Online]. Available: http://www.businessinsider.com/idc-smartphone-os-market-share-2015-12?IR=T. [Accessed: 10-Aug-2017].

[2] A. Stevenson, “Over 50 percent of Android malware aims to steal money - Business Insider,” 2015.

[3] H. A. Alatwi, “Android Malware Detection Using Category-Based Machine Learning Classifiers,” Rochester Institute of Technology, 2016.

[4] X. Su, M. Chuah, and G. Tan, “Smartphone Dual Defense Protection Framework : Detecting Malicious Applications in Android Markets,” *Mob. Ad-hoc Sens. Networks (MSN), 2012 Eighth Int. Conf.*, pp. 153–160, 2012.

[5] I. Burguera and U. Zurutuza, “Crowdroid : Behavior-Based Malware Detection System for Android,” *Proc. 1st ACM Work. Secur. Priv. Smartphones Mob. devices (SPSM ’11). ACM, New York, NY*, pp. 15–26, 2011.

[6] W. Enck, L. P. Cox, P. Gilbert, and P. Mcdaniel, “TaintDroid : An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones,” *ACM Trans. Comput. Syst.*, p. 32(2):5, 2014.

[7] M. Dimjaˇ, S. Atzeni, I. Ugrina, Z. Rakamari, and M. Dimjaˇ, “Android Malware Detection Based on System Calls Android Malware Detection Based on System Calls,” *J. Comput. Secur.*, 2015.

[8] R. Sato, D. Chiba, and S. Goto, “Detecting Android Malware by Analyzing Manifest Files,” *Proc. Asia-Pacific Adv. Netw.*, vol. 36, pp. 23–31, 2013.

[9] G. Suarez-tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, “DroidSieve : Fast and Accurate Classification of Obfuscated Android Malware,” *ACM Conf. Comput. Commun. Secur.*, 2017.

[10] K. Abdullah, D. Ibrahim, and C. Aydin, “APK Auditor : Permission-based Android malware detection system,” vol. 13, pp. 13–15, 2015.

[11] X. Su, M. Chuah, and G. Tan, “Smartphone Dual Defense Protection Framework : Detecting Malicious Applications in Android Markets,” *Mob. Ad-hoc Sens. Networks (MSN), 2012 Eighth Int. Conf.*, pp. 153–160, 2012.

[12] I. Burguera and U. Zurutuza, “Crowdroid : Behavior-Based Malware Detection System for Android,” *Proc. 1st ACM Work. Secur. Priv. Smartphones Mob. devices (SPSM ’11). ACM, New York, NY*, pp. 15–26, 2011.

[13] W. Enck, L. P. Cox, P. Gilbert, and P. Mcdaniel, “TaintDroid : An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones,” *ACM Trans. Comput. Syst.*, p. 32(2):5, 2014.

[14] M. Dimjaˇ, S. Atzeni, I. Ugrina, Z. Rakamari, and M. Dimjaˇ, “Android Malware Detection Based on System Calls Android Malware Detection Based on System Calls,” *J. Comput. Secur.*, 2015.