INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN 2278 – 0211 (Online)

Anomaly Network Based Malware Detection System Using Hybrid Techniques

Onyedeke Obinna Cyril
Researcher, Department of Computer Science,
University of Kaironau, Tunisia
Agubata Immaculate Chidimma
Researcher, Department of Computer Science,
University of Kairouan, Tunisia
Ihedioha Uchechi Michael
Lecturer, Department of Computer Science
University of Nigeria, Nsukka, Nigeria
Dr. Ezema Modesta
Lecturer, Department of Computer Science,
University of Nigeria, Nsukka, Nigeria

Abstract:
Android OS is one of the widely used mobile operating systems. There is a huge increment in malware applications in android phones. Smart android malware detection system (SAMDS) is an effort gear towards detecting malware application or malicious activities. This paper proposes a technique that can detect any malware application in android phone using anomaly based. It analyzes system calls' logs and also the conduct of an app and afterward produces signatures for malware conduct .This paper adopted the object oriented analysis and design method (OOADM), and uses the approach to model real world processes, operations and data in a more flexibly, efficiently and realistically manner. The paper goal is to raise user awareness of the permission-based system of Android phones and the threat it pose and provide counter measure system for more secured operations.

Keywords: *Malware detection techniques, Anomaly based, Smart android malware detection system (SAMDS)*

1. Introduction

Smart phones, tablets, and other versatile stages are quickly developing as famous apparatuses with progressively powerful computing, networking, and sensing capabilities. Smart device is the current overwhelming individualized computing gadget with such a significant number of features that are practically equal to smaller than normal PCs, for example, calls, short message administration, mixed media informing administration, email, video calling, voice correspondence, versatile banking, record trade, web perusing and so forth. As indicated by Pew Research Center in 2015, about 43% of the worldwide populace utilizes a cell phone gadget [1].

Android is an open source working framework for cell phones and tablets. It was propelled by Google and Open Handset Alliance in September 23, 2008. Android has experience a huge development since its initiation in view of its ease of use, open source, simplicity of creating and distributing applications. Android has become the most broadly utilized working framework on Smartphones with an expected piece of the pie of 81% in 2015 [2]. As indicated by report around 432 million cell phones were sold with Google's Android OS making up 81.7% of the market followed by Apple's iOS with 17.9% of the general piece of the overall industry [3].

The ubiquitous utilization of Android OS has actuated the explosion of versatile application advertise. Google Play is the biggest application store followed by Apple's App store. Google play has encountered a fast development from 16 thousand toward the finish of 2009 to about 2.8 million Applications (Apps) as at March, 2017 [4]. Android Applications are accessible for download in Google Play Store and outsider specialists.

The fast development of cell phone advancements and their far reaching client acknowledgment came at the same time with an expansion in the number and modernity of malignant programming focusing on mainstream stages. Malware has been on the expansion because of the ubiquity of Android telephones and the open nature that empowers designer to build up an application and distribute it in the Android advertise. Toward the finish of first quarter of 2016, the security firm G Data expressed that a sum of 440,267 new malwares were distinguished on Android telephone [5]. Malware creators have assorted motivating forces for making portable malware extending from research, oddity and diversion, monetary benefit, efficient, political, selling client data, taking client qualifications, site design improvement, making premium-rate calls, sending premium rate SMSs, SMS spam and so on.

A few systems have been proposed and executed to recognize, forestall and diminish malware assaults on Android telephones. Systems utilized for recognizing Android malware can be grouped into Static, dynamic, half and half and AI strategies. Static examination doesn't execute an application yet statically assess application and dismantling their code. Static examination includes checking different parameters like mark confirmation, arrange addresses, API calls, consent investigation and so on. Dynamic examination which includes location of malware at run-time, screens application powerfully during their execution. During dynamic investigation, a few highlights like framework calls, API following, of an application are separated to see whether that application is benevolent or malevolent.

2. Related Works

This chapter clarifies the advances and apparatuses utilized in building up the framework. We likewise present a far reaching investigation of the development and momentum condition of malware for cell phones and past methodologies utilized by analysts for distinguishing noxious applications. We give a short survey of the past methodologies that have been utilized. The past methodologies utilized can be assembled into Static and Dynamic Analysis.

2.1. Theoretical Background

Android is an open-source working framework for cell phones, tablets and so forth. It was fabricated dependent on Linux piece, created by Google and discharged on September 23, 2008 [6]. Android offers a neighborly improvement condition through an assortment of devices, for example, Android Software Development Kit (SDK), Android Native Development Kit (NDK), Android Debug Bridge (ADB), Android Developer Tools (Eclipse). Google PlayStore is the official appropriation community for Android Apps which are created by Google or outsiders. It permits Android clients to peruse, introduce, and update the applications.

2.1.1. Android Architecture

Android Stack depends on Linux portion and it comprises of four layers that deal with the entire framework beginning from equipment sensors to the client's elevated level applications. It comprises of different layers running on one another, the lower ones offering types of assistance to the upper level layers.

The main layer, the Linux Kernel layer is the most significant layer and situated at the base; it speaks to the core of Android framework. It gives the OS benefits and deals with the equipment's capacities, for example, memory, power, and drivers, arrange stack, security settings, shared libraries and equipment reflection.

The subsequent layer, the local library layer, gives local libraries which are a lot of guidelines that oversee information handling. The local layer gives the open source libraries, for example, surface administrator, media structure, SQLite, Webkit, OpenGL/ES, FreeType, and SSL.

The third layer, the Application Framework Layer, incorporates the Android APIs. The APIs are classes and interfaces for Android applications' improvement. This layer communicates with the running applications and deals with the fundamental capacities on the gadget. The most significant projects right now action supervisor, content supplier, communication administrator, area chief, and assets director.

The fourth layer, the Application Layer which is the highest layer where the telephone's capacities are given to the end-client and it gives functionalities that incorporate making calls, overseeing contacts, sending messages, and perusing web. Right now, gives a lot of center applications, for example, email customer, schedule, program, maps, contacts, SMS program, display, and so forth.

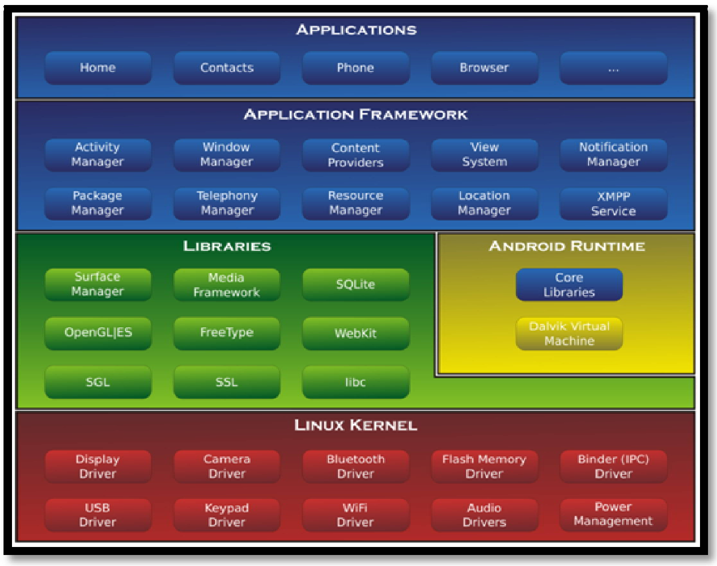

Figure 1: Android's Stack Structure (Adapted From [7])

2.1.2. Android Application Components

In the advancement stage, an Android application comprises of two envelopes and one record: Class, Resources and AndroidManifest.xml, separately. The Class envelope contains the applications' source codes; the Resources organizer contains the application's sight and sound; AndroidManifest.xml is the application's arrangement record that rundowns basic data about the application for the Android framework. The data that is recorded in the AndroidManifest.xml document is as per the following: Java bundle name, the application's segments, facilitated forms, authorizations, instrumentation classes, and different libraries. During the ordering stage, the two envelopes and the AndroidManifest.xml document are packaged together to create the executable record of the application in .apk group. The structure of Android application comprises of four segments: Activity, Service, BroadCastReceiver, and ContentProvider [8].

2.1.3. Android Permission Model

The consents model is the principle security idea that Android security depends on. Android runs applications independently in sandboxes. Each application is run in a disengaged domain where it has no entrance to the framework's assets. The authorizations must be given to the application to have the option to access and use framework assets that are required for its usefulness. All the consents that must be announced in the AndroidManifest.xml document by the engineer in the improvement stage. These consents additionally should be conceded by the framework or the client at the establishment time; when they are allowed they couldn't be changed except if the application is uninstalled by the client. The authorizations can be proclaimed in at least one <permission> labels in the AndroidManifest.xml document; they additionally should be characterized with required and discretionary properties. The <label>and <description>attributes are the strings that are shown to the client at establishment time. These properties must be obviously characterized to help the client in understanding the benefits that the authorization shows. The <permission Group> quality is discretionary and utilized by the framework to show the classification of the authorization to the client. The <protectionLevel> credit is required to distinguish the security level of the authorization; it characterizes the criticality of the application benefits. The authorizations model uses the two <uses-permission>and <permission> labels to administer applications' entrance to the framework assets and other application's information, individually. The <uses-permission> tag characterizes the authorizations that the application needs to get to explicit information, equipment, programming and other framework assets. Then again, the <permission>tag characterizes the authorizations that different applications need to approach the application's information and segments. As it were, it characterizes how the application's parts can be open by the different applications.

2.1.4. Android Application Development

Android applications are basically written in Java with assistance for their own neighborhood libraries written in C. The Java source code gets requested to a Dalvik Virtual Machine (DVM) executable byte code, which is taken care of in a DEX file. The DEX file, the Manifest, all benefits, the certificates and have libraries for the application are packaged to a ZIP record file with the APK suffix. This APK file is given through the Google Play to the customers. The source code of an Android application isn't available in clear substance while emptying an APK file.

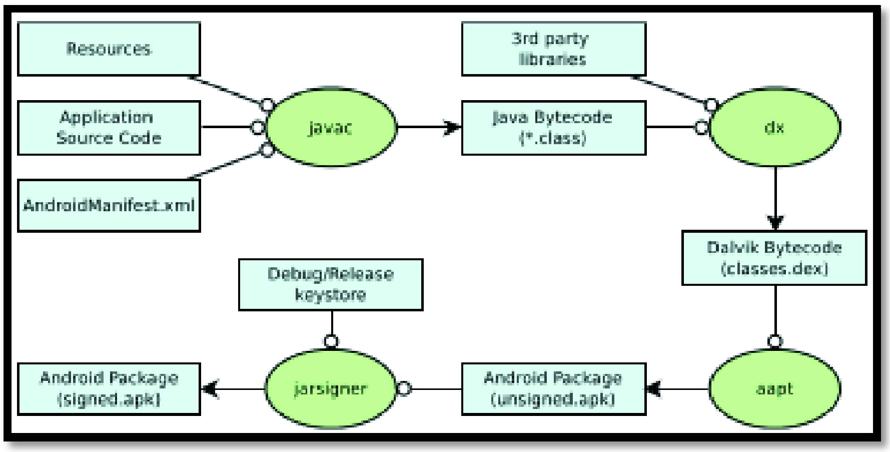

Figure 2: Build Process of an Android Application (Adapted From [9])

2.1.5. Android Malware

A vindictive application alludes to an app that can be used to bargain an activity of a gadget, take information, sidestep get to controls or in any case cause harm on the host terminal [9]. This area presents the Android malware risk and development.

2.1.5.1. Malware

Assailants need to access a gadget by introducing malware on it. The reason for existing is to take information or harm the gadget. Malware is introduced by deceiving the client to introduce a really glancing application or by and large to misuse powerlessness on the gadget, e.g., a security flaw in the Web program.

2.1.5.2. Spyware

Spyware is exceptionally normal in the Android stage, it is intended to get delicate data from an injured individual's framework and move this data to the assailant. Spyware can be business and malevolent. 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 transmit it to an outsider.

2.1.5.3. Grayware

The fundamental reason for grayware is to keep an eye on clients who introduced the product all alone in light of the fact that they believed that it is real programming. This is mostly right in light of the fact that the creators incorporate genuine usefulness as promoted. By the by, they likewise gather data from the framework, for example, the client's location book or his perusing history.

2.1.5.4. Fraudware

Fraudware-based applications are introduced by deceiving the client to introduce a genuinely looking application, which will increase full usefulness subsequent to sending a few premium-appraised SMS messages.

2.1.5.5. Trojan

Trojans are applications that claim to be valuable, however perform malevolent activities out of sight, for example, downloading extra malware, changing framework settings, or contaminating different files on the framework. Android malware is for the most part Trojans. The assault vectors utilized by infections and worms are generally inaccessible to malware designers in light of the sandboxing model. The malevolent code is normally included into real applications, which are then redistributed as the first application. Applications utilized for this reason for existing are regularly paid applications redistributed as free applications on outsider markets.

2.1.5.6. Root abuse

Root abuses are conceivable on Android so as to deal with the gadget, yet are viewed as a twofold edged sword among the security network. Establishing can give the client authority over a gadget and furthermore gives a similar measure of control to any application, which accesses the root rights. Root benefits given to a malignant application can totally bargain the gadget, as the application can hypothetically expel the root benefits from the client. The noxious application professes to be typical until it is introduced on the client's gadget, as most Trojans. It endeavors, when introduced, to utilize at least one root adventures to pick up root access to the gadget. An application with root access can supplant, alter and introduce applications as it wishes.

2.1.5.7. Bot

Bots are a developing pattern in versatile malware that speak with and get directions from at least one Command and Control (C&C) servers, giving the malware essayist remote authority over completely contaminated gadgets. Malware engineers muddle their code and use encryption to shroud basic data helpless to distinguish them. Also, a bot can introduce extra applications with no client information or intercession.

2.2. Review of Related Literature

Right now, review a portion of the past methodologies utilized by analysts for distinguishing vindictive applications. Different methodologies have been utilized to identify malevolent applications and they can be generally assembled into static, dynamic and anomaly based approach. Underneath, we give a concise audit of research considers that have been directed utilizing static and dynamic investigation.

2.2.1. Android Malware Techniques

- Static approach removes features from the application's document without executing the application to identify malevolent examples. From the application's source code, numerous highlights are separated, for example, authorizations, communicate recipients, APIs, purposes, information stream, control stream, equipment segments and so forth. The most generally utilized static highlights are the Permission and API calls. Since these are removed from the application AndroidManifest.xml and impact the malware recognition rate to a high degree.
- Dynamic approach investigates the conduct of the application in a run time condition and screens the application's dynamic conduct and framework reactions. It executes the suspicious application inside a controlled situation frequently called sandbox. The dynamic features checked are organize associations, work calls, assets utilization, framework calls and so on.
- Anomaly based approach is based with respect to viewing the conduct of the gadget by monitoring various parameters and the status of the segments of the gadget. Andromly is a conduct based malware location system and it consistently screens the various highlights of the gadget state, for example, battery level, CPU use, arrange traffic, and so on. Estimations are taken during running and are then provided to a calculation that groups them likewise. CrowDroid and AntiMalDroid are two unique oddities based instruments utilized for malware discovery in Android gadgets. The first relies upon breaking down framework calls' logs while the last examines the conduct of an application and afterward creates marks for malware conduct. SMS Profiler and iDMA are two apparatuses used to identify ill-conceived utilization of framework benefits in iOS.[10]

[11] Proposed a proactive plan to spot zero-day Android Malware without depending on malware tests and their marks to spot potential security dangers presented by untrusted applications. They created RiskRanker, a robotized framework that scalably examine applications whether they show perilous practices. They performed static investigation on the figured out Dalvik bytecode contained in each application by removing the information stream and control stream from the code way. They gathered 118,318 applications from different Android advertises and handled it inside four days. From their investigation they detailed 3281 dangerous applications. [12] Used both static and dynamic investigation to recognize malware in android applications. They consolidated the static investigation (consent) and dynamic examination (System call following) with AI. They performed static investigation by removing authorizations from the Android's manifest.xml record and examined the contrast between the quantity of consents mentioned by favorable and vindictive applications. They understood that the quantity of authorizations mentioned by kindhearted and malignant application is marginally the equivalent. This strategy was tried on different benevolent and malignant applications.

In [13], Kirin security administration which performs lightweight affirmation of uses was proposed for Android to moderate malware at introduce time. Kirin pronounces that a mix of authorizations could be perilous. Kirin comprises of three parts, installer, security administration and database of security rules. The installer removes security design from the AndroidManifest.xml record. Their outcome shows that affirmation strategy falls flat for just 1.6% of utilizations in their dataset thus Kirin can be sensible for all intents and purposes relieve malware.

[14] Proposed DREBIN, a lightweight technique for identification of Android malware that empowers recognizing pernicious applications straightforwardly on the cell phone. DREBIN plays out a wide static investigation, extricates a lot of highlights from the application's AndroidManifest.xml (equipment segments, mentioned authorizations, App parts, and separated expectations) and dismantled code (limited API calls, utilized consents, confined API calls, arrange addresses) to create a joint vector space. At the point when tried with 123,453 generous applications and 5,560 malware tests, DREBIN effectively identified 94% of the malware with a bogus positive pace of 1%.

This [15] paper removes six kinds of data Permission, Intent channel, Intent channel, Process name, Intent channel, number of reclassified authorization from show documents and uses them to recognize Android malware. Results show that the technique can identify obscure malware tests that are imperceptible by a straightforward mark based methodology. This methodology is modest to execute in light of the fact that solitary the show document is examined.

[16] Presented a quick, adaptable, and precise framework for Android malware recognition and family distinguishing proof dependent on lightweight static investigation. DroidSieve utilizes profound review of Android malware to construct compelling and strong highlights reasonable for computational learning. Their discoveries show that static investigation for Android can succeed in any event, when gone up against with obscurity systems, for example, reflection, encryption and progressively stacked local code. While major changes in qualities of malware stay a to a great extent open issue, DroidSieve stays flexible against best in class muddling systems which can be utilized to rapidly infer new and grammatically unique malware variations.

[17] Presents a consent based Android malware identification framework, APK Auditor that utilizes static examination to describe and order Android applications as benevolent or vindictive. APK Auditor comprises of three segments: A mark database to store extricated data about applications and examination results, an Android customer which is utilized by endusers to give application investigation demands, and a focal server liable for speaking with both mark database and cell phone customer. 8762 applications were utilized to test framework execution. Result shows that APK Auditor can recognize most notable malwares and features the ones with a potential in around 88% precision with a 0.925 specificity.

This [18] study introduced a cell phone double guard insurance system that permits official and elective Android Markets to recognize vindictive applications among those new applications that are submitted for open discharge. This structure comprises of servers running on mists where designers who wish to discharge their new applications can transfer their product for confirmation reason. The confirmation server first uses framework call measurements to recognize potential malevolent applications. After confirmation, on the off chance that the product is perfect, the application will at that point be discharged to the significant markets. The exploratory outcomes utilizing 120 test applications (which comprise of 50 malware and 70 typical applications) show that we can accomplish 94.2% and 99.2% exactness with J.48 and Random woodland classifier individually utilizing this system.

[19] In this paper we have proposed another system to acquire and examine cell phone application movement. They found that observing framework calls is one of the most precise strategies for deciding the conduct of Android applications. The creator built up a lightweight customer called Crowdroid. This application utilizes publicly supporting way of thinking where a client sends non individual yet conduct related information of every application they use to the server. This is trailed by malware recognition dependent on the call vectors by the server. The exploratory outcomes completed by the writer had 100% location rate for self-composed malware.

[20] Presented TaintDroid, a proficient, framework wide data stream following device that can all the while track various wellsprings of delicate information. We additionally utilized our TaintDroid execution to examine the conduct of 30 well known outsider applications, picked aimlessly from the Android Marketplace. Our investigation uncovered that 66% of the applications in our examination show suspicious treatment of touchy information, and that 15 of the 30 applications detailed clients' areas to remote promoting servers.

In [21], a basic, but then profoundly successful strategy for recognizing malevolent Android applications on a store level was proposed. The method performs programmed classification dependent on following framework calls while applications are executed in a sandbox situation. The method was actualized in an apparatus called MALINE, and performed broad exact assessment on a suite of around 12,000 applications.

3. Hybrid Analysis

Hybrid Analysis is a mix of static and dynamic examination. It is an innovation or strategy that can incorporate run-time information extricated from dynamic investigation into a static examination calculation to recognize conduct or noxious usefulness in the applications. The half and half examination technique includes joining static highlights acquired while investigating the application and dynamic highlights and data removed while the application is executed. In spite of the fact that it could expand the exactness of the recognition rate, it makes the framework awkward and the examination procedure is tedious. [22]

3.1. Structure of the Proposed Framework

The proposed Smart Android Malware discovery framework (SAMDS) is an application that sudden spikes in demand for an Android OS, and can distinguishes noxious application precisely. The principle Objectives are as per the following:

- Develop a crossover malware discovery system that can check android applications.
- Scan existing applications and recognize noxious and considerate application.
- Scan applications during establishment time and report to clients whether to proceed or not.
- Generate an outcome after output.

3.2. System Architecture

The framework will be planned dependent on a 3-tier framework architecture. The presentation tier, the pre-processing tierand the Result tier. The presentation tier is the environment that shows the graphical User Interface (GUI) and application-explicit passage structures or intelligent windows. The pre-processing tier runs the application from the presentation level to distinguish if the application is a malware application or normal flow. Result tier is the result that tells if a malware is recognized. It filters the application from the android phone and if malware is detected, it automatically uninstall (for already installed app), however for a newly application undergoing installation, it checks and if malware is detected it blocks the app from installing.

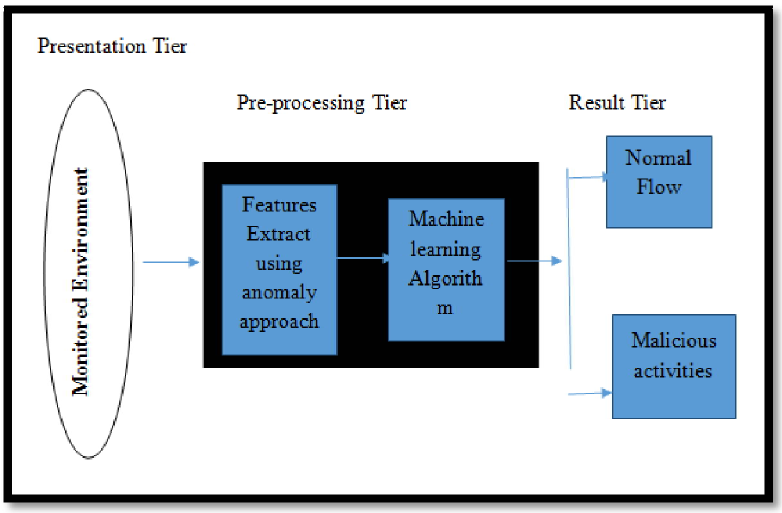

Figure 3: Architecture of the Proposed System

4. System Implementation

The system implementation is the development of the new system or application following the laid plans from analysis and design stage. This chapter depicts how the plan from the previous section is executed with the aim of providing a proficient system to detection of intrusion on Android phone. Apparatuses and techniques used to actualize are presented in this section.

4.1. Choice of Development Environment

The integrated development environment (IDE) used in the development of this work is the Android studio v2, on which the source codes are written, compiled and uploaded on Google Play Store. Android Studio offers numerous features that enhances profitability when building Android applications, for example, Gradle-based system which is use to manage all dependencies (to build, test, run and package your app), Android Virtual Device (Emulator) also helps run and debug apps in the Android studio. The programming languages employed in this project are Java while Realm database management system was used [23].

4.2. Implementation Architecture

SAMDS is a lightweight and flexible system that scans applications to detect malicious code. The architecture of SAMDS is based on five modules, the module of retrieving application, the module of analysis, the module of interpretation of results, the module of presentation of results and the module of preferences. Figure 4 shows the logical view of the architecture for implementing the system.

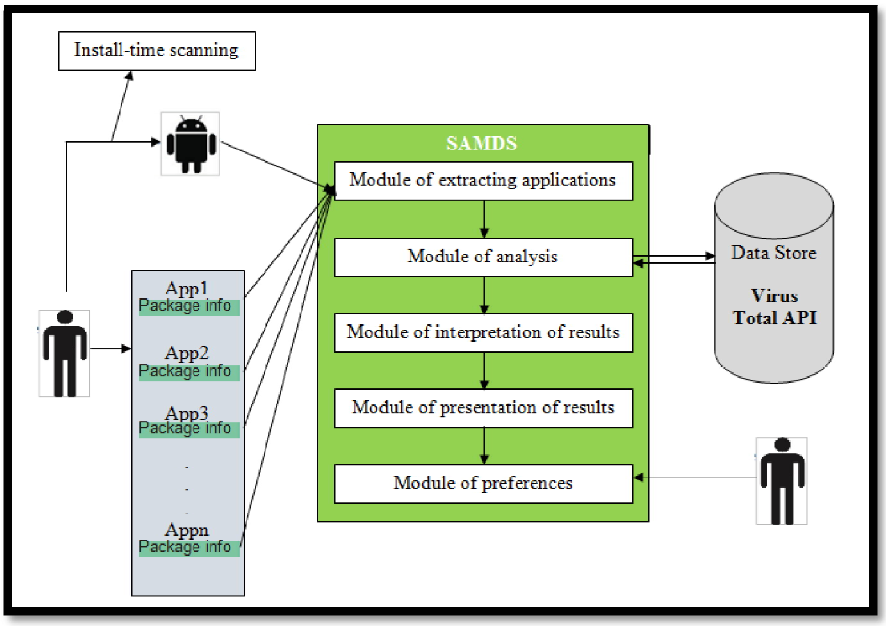

Figure 4: Implementation Architecture of SAMDS

5. Result and Discussions

The framework execution includes the improvement of the new framework or application following the laid plans from investigation and configuration stage. This paper depicts how the structure from the first section is actualized with center to give an effective procedure to identify malicious Android applications. The framework usage stage utilizes the structure produced during building plan related to the aftereffects of framework investigation to develop a framework that meets the end-clients' pre-requisites. Devices and procedures used to actualize will be presented right now.
Smart Android Malware Detection System (SAMD) obviously shows that the presentation of the framework is precise and it shows promising outcomes as far as low computational cost and high outcome. SAMDS is a portable application developed using Java. After the application has been sent, you select application to scan, at that point remove data from the application and send information to filter. You will get a reactions result and if the application is malicious it will uninstall consequently.

This module actualizes a component that isn't seen in past work. After a client check an application, SAMDS creates an outcome that contains, the nature of the application – generous or malevolent, the group of the malware, the harm it may cause to the gadget/documents and an alternative to uninstall the application.

6. Conclusion

This paper outlines ongoing advancements in android malware discovery utilizing AI calculations. The proposed framework, Smart Android Malware location framework (SAMDS) is an application that continues running on an Android OS. We propose a Smart android malware recognition framework that identifies pernicious application accurately and the application is used by Android phone users. Location strategies and frameworks that utilizes static, dynamic and abnormality approaches are talked about and featured. A technique that could prompt potential neutralizing the update assault is discussed.

7. References

i. Drake Bear (Tech Insider), 'How many people own smartphones around the world - Business Insider,' 2016. [Online]. Available: http://www.businessinsider.com/how-many-people-own-smartphones-around-the-world-2016-2?IR=T. [Accessed: 10-Aug-2017].

ii. 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].

iii. R. Price, 'BlackBerry global smartphone market share is 0,' 2017. [Online]. Available: http://www.businessinsider.com/blackberry-smartphone-marketshare-zero-percent-gartner-q4-2016-2017-2?IR=T. [Accessed: 10-Aug-2017].

iv. (Statista), 'App stores number of apps in leading app stores 2017 _ Statista,' 2017. [Online]. Available: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/. [Accessed: 10-Aug-2017].

v. A. Stevenson, 'Over 50 percent of Android malware aims to steal money - Business Insider,' 2015. .

vi. H. A. Alatwi, 'Android Malware Detection Using Category-Based Machine Learning Classifiers,' Rochester Institute of Technology, 2016.

vii. V. Grampurohit, 'Android App Malware Detection,' International Institute of Information Technology, India, 2016.

viii. 'Application fundamentals -android developers.' [Online]. Available: http://developer.android.com/%0Aguide/components/fundamentals.html. [Accessed: 20-Aug-2017].

ix. Franklin Tchakounte, 'A Malware Detection System for Android,' Universitat Bremen, 2015.

x. Belal Amro, College of Information Technology, Hebron University, 'malware detection techniques for mobile devices'. International Journal of Mobile Network Communications & Telematics (IJMNCT) Vol.7, No.4/5/6, December 2017

xi. M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, 'RiskRanker : Scalable and Accurate Zero-day Android Malware Detection Categories and Subject Descriptors,' Int. Conf. Mob. Syst. Appl. Serv., 2012.

xii. P. Kaushik and A. Jain, 'Malware Detection Techniques in Android,'Int. J. Comput. Appl., vol. 122, no. 17, pp. 22–26, 2015.

xiii. W. Enck, M. Ongtang, and P. Mcdaniel, 'On Lightweight Mobile Phone Application Certification,' ACM Conf. Comput. Commun. Secur., 2009.

xiv. D. Arp, M. Spreitzenbarth, H. Malte, H. Gascon, and K. Rieck, 'Drebin : Effective and Explainable Detection of Android Malware in Your Pocket,'Proc. 17th Netw. Distrib. Syst. Secur. Symp., pp. 23–26, 2014.

xv. 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.

xvi. 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.

xvii. K. Abdullah, D. Ibrahim, and C. Aydin, 'APK Auditor : Permission-based Android malware detection system,' vol. 13, pp. 13–15, 2015.

xviii. 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.

xix. 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.

xx. 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.

xxi. 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.

xxii. 1Prof. Vidhya Rao, Khushboo Hande, ' A comparative study of static, dynamic and hybrid analysis techniques for android malware detection', international journal of Engineering Development and Research (IJEDR) volume 5, issue 2, 2017

xxiii. Okoronkwo M.C and Onyedeke O.C, 'An intrusion detection system(IDS) on and roidphones using a filter base feature selection algorithm,' Int. journal of innovative research and developmenyt., vol.8, issue 11, pp. 101, 2019.