Design and Implementation of a Malware Detection System on Smartphones

Atanda Aminat Oluchi[1], Obi Adaobi Maria[2], Anyaorah Chukwuka Charles[3], Idoko Nnamdi A[4], Udechukwu Precious Emeka[5], Anusiobi Chinenye Loveline[6], Asogwa Samuel[7], Senu Jephthah Folarin[8]

[1,2,5,6]*Department of Computer Science, University of Nigeria, Nsukka, Nigeria*
[3]*Department Of Computer Science, Federal College of Education Eha-Amufu, Nigeria*
[4]*Department of Computer Science, Renaissance University, Ugbawka, Enugu, Nigeria*
[7]*Department of Computer Science, Michael Okpara University of Agriculture, Umudike, Nigeria*
[8]*Department of computer science, Federal university of tech, Minna, Nigeria*

***Abstract*:-Due to innovative advancement, Smart phones developed into in fact and practically refined gadgets called mobile phones. Giving far reaching abilities, Smart phones are getting progressively well known for the focused on clients as well as all. Malware has been a significant issue on cell phones. General countermeasures to Smart phone malwares are at present restricted to signature-based enemy of infection scanners which proficiently identify known malwares, yet they have genuine inadequacies with new and obscure malwares making a lucky opening for assailants. As Smart phones become a host for delicate information and applications, broadened malware recognition instruments not basing on marks are important com-utilizing with the asset limitations of current cell phones. In this work, we tackle the field of cell phone malware. We give a reasonable clarification on what a cell phone really is. Dynamic and static investigation was utilized in the proposed framework. In the field of dynamic investigation, an observing framework is presented assembling conduct and framework based data that are handled by a distant framework utilizing AI for oddity discovery. In the field of static investigation, we examine its pertinence to the space of various cell phone stages, in particular Symbian OS and Android. This paper adopted the object oriented analysis and design method (OOADM), and utilizations the way to deal with model true cycles, activities and information in an all the more deftly, productively and sensibly way.**

***Keywords:* Malware intrusion detection, Smartphones, Dynamic and Static analysis.**

I. INTRODUCTION

Android has become the most mainstream open source working framework for smart phones and tablets with an expected piece of the overall industry of 70% to 80%. A shipment of one billion Android gadgets has been gauge in 2017; more than 50 billion applications have been downloaded since the first Android telephone was delivered in 2008. The normal number of utilizations per gadget expanded from 32 to 41 and the extent of time spent by clients on smart phone applications nearly approaches the time spent on the Web (73% versus 81%) [1].

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 huge development since its origin due to 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 overall industry of 81% in 2015 [2]. As indicated by report around 432 million smart 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 pie [3].

The fast development of smart phone advancements and their far reaching client acknowledgment came all the while with an expansion in the number and refinement of malignant delicate product focusing on mainstream stages. Malware (short for noxious programming) created for early cell phones, for example, Palm stages and highlighted cell phones was identified preceding 2004. The presence of malware specifically produced for them (generally Symbian OS) developed exponentially with the expansion of smart phones with in excess of 400 cases somewhere in the range of 2004 and 2007 [4]. The iPhone and Android OS were delivered later on and turned out to be quickly the transcendent stages. This offered ascend to a disturbing acceleration in the number and advancement of vindictive programming focusing on these stages, especially Android OS. Over 250.000 Android clients have been survivor of an extraordinary portable assault when they downloaded pernicious programming camouflaged as genuine applications from the Android Market.A comparative report by F-Secure uncovers that the quantity of vindictive Android OS applications got during the first quarter of 2012 expanded from 139 to 3063 contrasted with the first quarter of 2011 and it as of now speaks to 97% of the absolute portable malware before the finish of 2012 [5].A few strategies have been proposed and actualized to recognize, forestall and diminish malware assaults on Android telephones. Procedures utilized for recognizing Android malware can be grouped into Static, dynamic, mixture and AI methods. Static investigation doesn't execute an application yet statically review application and dismantling their code. Static investigation includes checking different boundaries like mark confirmation, network addresses, API calls, authorization examination and so forth. Dynamic investigation which includes identification of malware at run-time, screens application powerfully during their execution. During dynamic examination, a few highlights like framework calls, API following, of an application are removed to see whether that application is amiable or malevolent.

II. THE ORETICAL 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 [6]. 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 Malware Detection Approaches and Coun-Ter Measures

Countermeasures, which help to secure a system, taken byinstalling certain hard- or software. Three main systems for computers can identify: firewalls, antivirus software and intrusion detection systems

Firewalls are purported "white list" - based frameworks, which implies that there is an uncommon rundown of rules expressly permitting certain ports to speak with inward or outside peers13. On the off chance that noxious programming can take on the appearance of believed programming utilizing a confided in port, an essential firewall will permit all correspondence exercises [7]. Antivirus scanners use "boycotts" so as to distinguish certain dangers remembered for the boycott. An infection scanner can impede infections, worms, and Trojan ponies with continuous checking or manual examining. Malware is recognized by filtering for and finding a specific string or example, additionally called signature. There-front, the malware must be known by the scanner. Infection scanners typically incorporate a particular cleansing schedules relating to the recognized marks [8]. Interruption Detection Systems (IDS) once were frameworks that observed organization traffic. Logged traffic was utilized by network directors so as to identify irregular conduct [9]. Countermeasures like shutting ports or bolting frameworks could be taken by the overseers. IDS advanced into interruption anticipation frameworks (IPS) which can recognize certain irregular practices and take preventive measures naturally. Base on anomalous practices, interruption discovery and avoidance frameworks (IDPS) are fundamentally ready to recognize malware movement while they do not have the expulsion schedules known from infection scanners. Infection scanners and Intrusion Detection Systems present the premise of our methodologies. While firewalls center on confining organization traffic, infection scanners and Intrusion Detection Systems attempt to identify vindictive programming and exercises utilizing static and dynamic examination.

2.1.2 Static Analysis and Dynamic Analysis

Infection scanners and Intrusion Detection Systems attempt to distinguish noxious programming and exercises utilizing static and dynamic examination. The significant distinction among static and dynamic analysis is the way the information is procured. Strategies that are utilized to analyze the checked information can be the equivalent for the two methodologies.

Static Analysis Static analysis speaks to a methodology of checking source code or ordered code of uses before it gets executed. Static examination can utilize basic example search activity or marginally more mind boggling AI approaches so as to identify blemishes and powerless nesses in the code of programming. A basic inquiry may target finding in-secure capacity brings in C programs. A more mind boggling approach may be the use of factual techniques so as to decide events of specific calls.

Dynamic analysis examines the conduct of the application in a run time condition and screens the application's dynamic conduct and framework reactions. It executes the dubious application inside a controlled domain regularly called sandbox. The dynamic highlights observed are network associations, work calls, assets utilization, framework calls and so on.

The significant distinction among static and dynamic analysis is that dynamic examination alludes to information obtained on runtime while static analysis doesn't. Static examination can exclusively depend on information removed from pairs in a static way. Strategies being applied to the obtained information for distinguishing malware can essentially be the equivalent to for the two variations [10].

2.2 Review of Related Literature

At the present time, survey a part of the past techniques used by investigators for recognizing malicious applications. Various strategies have been used to distinguish malicious applications and they can be by and large amassed into static, dynamic and peculiarity based methodology. Underneath, we give a succinct review of exploration thinks about that have been coordinated using static and dynamic examination.

According to [11], present a standard based framework so as to demonstrate pernicious capability of Android applications. Hence, they gathered the top311 applications from android market and checked them for events of certain authorization set in an arrangement document of each. This

Check demonstrated that five of these applications executed perilous functionalities. Another five additionally indicated perilous authorizations however these could be contended through gave usefulness of those applications.

In [12], study introduced a cell phone double safeguard insurance system that permits official and elective Android

Markets to identify malignant applications among those new applications that are submitted for open delivery. This structure 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, on the off chance that the product is perfect, the application will at that point be delivered to the significant business sectors. The test results utilizing 120 test applications (which comprise of 50 malware and 70 typical applications) demonstrate that we can accomplish 94.2% and 99.2% exactness with J.48 and Random woodland classifier separately utilizing this structure.

As indicated by [13], proposed another system to get and examine cell phone application movement. They found that observing framework calls is one of the most precise procedures 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 however conduct related information of every application they use to the worker. This is trailed by malware recognition dependent on the call vectors by the worker. The exploratory outcomes did by the writer had 100% identification rate for self-composed malware.

In [14], express that implanted gadgets, as mobile phones, shrewd cards or installed network sensors are generally compact, convey remote and are battery controlled or if nothing else vitality restricted. The plan of security for implanted frameworks contrasts from customary security plan, as various attributes can be found for every sort. There are two principle gatherings of qualities that separate the security engineering from Embedded System from that of workstations and workers: asset impediments and physical openness. In their work, Hwang et al. guarantee that inserted security can't be comprehended at single security deliberation layer and subsequently present safety efforts for all reflection layers.

In [15], introduced TaintDroid, a productive, framework wide data stream following apparatus that can all the while track different wellsprings of delicate information. We additionally utilized our TaintDroid usage to consider the conduct of 30 famous outsider applications, picked indiscriminately from the Android Marketplace. Our investigation uncovered that 66% of the applications in our examination display dubious treatment of touchy information, and that 15 of the 30 applications revealed clients' areas to far off promoting workers.

In [16], a basic, but then profoundly compelling procedure for identifying malevolent Android applications on an archive level was proposed. The procedure performs programmed classification dependent on global positioning framework calls while applications are executed in a sandbox domain. The method was actualized in an apparatus called MALINE, and performed broad experimental assessment on a set-up of around 12,000 applications.

According to [17], proposed a proactive plan to spot zero-day Android Malware without depending on malware tests and their marks to spot potential security hazards presented by untrusted applications. They created Risk Ranker, a robotized framework that scalably dissect applications whether they display perilous practices. They performed static investigation on the figured out Dalvik bytecode contained in each application by separating 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 examination they revealed 3281 hazardous applications.

In [18], utilized both static and dynamic examination to distinguish malware in android applications. They consolidated the static investigation (consent) and dynamic examination (System call following) with AI. They performed static examination by separating authorizations from the Android's manifest.xml document and contemplated the distinction between the quantity of consents mentioned by kind and noxious applications. They understood that the quantity of consents mentioned by favorable and pernicious application is marginally the equivalent. This strategy was tried on different amiable and noxious applications.

In [19], Kirin security administration which performs lightweight affirmation of utilizations was proposed for Android to alleviate malware at introduce time. Kirin proclaims that a mix of consents could be risky. Kirin comprises of three parts, installer, and security administration and information base of security rules. The installer separates security arrangement from the AndroidManifest.xml document. Their outcome shows that affirmation method fizzles for just 1.6% of uses in their dataset subsequently Kirin can be sensible for essentially moderate malware.

According to [20], proposed DREBIN, a lightweight strategy for recognition of Android malware that empowers recognizing noxious applications legitimately on the cell phone. DREBIN plays out a wide static examination, removes a lot of highlights from the application's AndroidManifest.xml (equipment segments, mentioned authorizations, App segments, and separated plans) and dismantled code (limited API calls, utilized consents, confined API calls, network addresses) to create a joint vector space. At the point when tried with 123,453 considerate applications and 5,560 malware tests, DREBIN effectively identified 94% of the malware with a bogus positive pace of 1%.

According to [21], extricates six sorts of data Permission, Intent channel, Intent channel, Process name, Intent channel, number of re-imagined consent from show records and uses them to identify Android malware. Results show that the strategy 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 record is examined.

In [22], introduced a quick, versatile, and exact framework for Android malware location and family distinguishing proof dependent on lightweight static investigation. DroidSieve utilizes profound review of Android malware to assemble successful and strong highlights reasonable for computational learning. Their discoveries show that static examination for Android can succeed in any event, when gone up against with obscurity methods, for example, reflection, encryption and progressively stacked local code. While essential changes in attributes of malware stay a generally open issue, DroidSieve stays tough against cutting edge obscurity methods which can be utilized to rapidly determine new and grammatically extraordinary malware variations.

According to [23], presents an authorization based Android malware recognition framework, APK Auditor that utilizes static investigation to describe and order Android applications as benevolent or malevolent. APK Auditor comprises of three parts: A mark data set to store removed data about applications and examination results, an Android customer which is utilized by endusers to give application investigation demands, and a focal worker answerable for speaking with both mark information base 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 roughly 88% exactness with a 0.925 specificity.

III. ANALYSIS OF THE PROPOSED SYSTEM

The proposed framework is an application that sudden spikes in demand for an Android OS. We propose an Android malware discovery framework that recognizes malevolent applications precisely. The proposed application is utilized by Android telephone clients, analysts and the general Smartphone clients. Thinking about the impediment of the current framework, the proposed framework looks to address a portion of the innate issues recognized by viably filtering applications and distinguishing noxious applications. The proposed framework plays out the accompanying movement to examine and identify malware in Android Apps.

- Get the list of already installed applications.
- Extract the source_dir of the application
- Upload the source_dir on VirusTotal database
- Receive response from VirusTotal database
- Interpret the result to user/researcher
- Present result to user

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

3.1 Design of the Proposed System

The proposed system's design is described with the following operations:

- Application Listing: system fetches all the applications installed on the device and display the list to the user to scan.
- Install-Time Scanning: System provide the functionality to scan an application that is being installed automatically at install time. This module listens to messages broadcasted by Android by default signaling an installation. When this module receives a message indicating the addition of a new application to the system, it notifies the user to scan the application before use.
- Static Analysis: System scrutinize the application to be scanned and extracts the source-dir of the application.
- Real-Time Scanning: After extracting the source_dir of a given application, System sends it to VirusTotal's aggregated data which comprises heuristic engines, known-bad signatures, metadata extraction, identification of malicious signals, etc.
- Instruction Module: Based on previous work, it was ascertained that majority of Smartphone users do not know the necessary security precautions to reduce the widespread of malware. This module adds the functionality of educating user about the permission system, importance of checking the user review section and need to download applications from official android market.
- Display of Result: This module implement a feature that is not seen in previous work. After a user scan an application, System generates a result that contains, the nature of the application – benign or malicious, the family of the malware, the damage it might cause to the device/files and an option to uninstall the app.
- Implementation Architecture of System

IV. WALWARE DETECTION TECHNIQUES

Techniques utilized for identifying malware can be classified comprehensively into two classifications: abnormality based recognition and mark based discovery. An inconsistency based identification strategy utilizes its information on what comprises ordinary conduct to choose the noxiousness of a program under examination. A unique sort of inconsistency based discovery is alluded to as determination based recognition. Determination based procedures influence some detail or rule set of what is substantial conduct so as to choose the perniciousness of a program under investigation. Projects abusing the determination are viewed as bizarre and typically, malignant. Mark based location utilizes its portrayal of what is known to be malignant to choose the noxiousness of a supportive of gram under assessment. As one may envision this portrayal or mark of the malevolent conduct is the way in to a mark based discovery technique's adequacy. Every one of the recognition procedures can utilize one of three unique methodologies: static, dynamic, or mixture. The particular methodology or examination of an abnormality based or signature-based procedure is controlled by how the strategy assembles data to identify malware. Static investigation utilizes grammar or auxiliary properties of the program (static)/measure (dynamic) under examination (PUI) to decide its perniciousness. For instance, a static way to deal with signature-based identification would just use basic data (for example succession of bytes) to decide the noxiousness, though a powerful methodology will use runtime data (for example frameworks seen on the runtime pile) of the PUI. When all is said in done, a static methodology endeavors to distinguish malware before the program under assessment executes. Alternately, a powerful methodology endeavors to distinguish noxious conduct during program execution or after supportive of gram execution.

4.1 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.

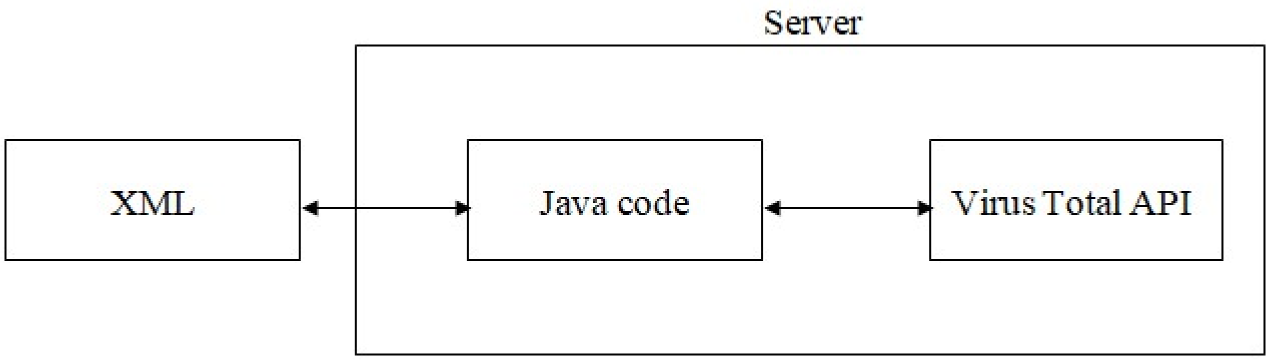

Figure 4.1: Architecture of the proposed system

4.2 Implementation Architecture

The proposed system is a lightweight and flexible system that scans applications to detect malicious code. The architecture of the system 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 4.1 shows the logical view of the architecture for implementing the system.

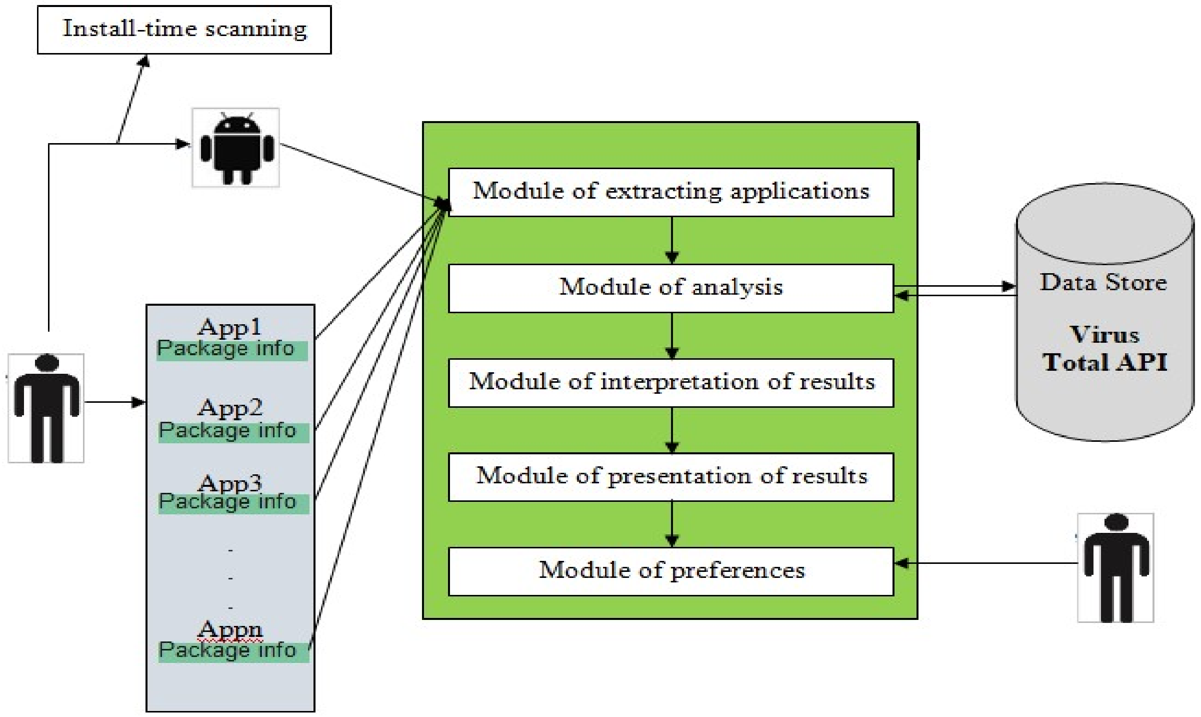

Figure 4.2: Implementation Architecture

V. RESULTS AND DISCUSSION

Android malware growth has been increasing drastically along with increasing the diversity and complicity of their developing techniques. This research work provides an effective and efficient technique to detect malicious code in Android Application. We have been able to design and develop an application that can scan already installed Android applications and newly installed applications to detect whether they are Benign or Malware. 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.Malware Detection System clearly shows that the introduction of the system is exact and it shows promising results similar to low computational expense and high result. This 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 vindictive it will uninstall thusly. This module realize a segment that isn't seen in past work. After a customer check an application, makes a result that contains, the idea of the application – liberal or vindictive, the gathering of the malware, the mischief it might cause to the device/archives and an option to uninstall the application.

5.1 Conclusion

Mobile devices mostly running Android OS have become the new personal computer, storing as much data as a PC but providing greater flexibility and portability. Smart devices equipped with powerful computing, sensing, and networking capabilities have increasingly become the platform of choice for many users, outselling the number of PCs worldwide. Online banking, commerce, and other business applications put daily business and financial transactions at user fingertips. Users are at every turn stipulated to download applications for further increasing the value of their mobile devices. As mobile devices grow in popularity, so do the incentives for attackers. Mobile malware is clearly on the rise, as attackers experiment with new business models by targeting mobile phones. This increase is in some cases accompanied by sophisticated techniques purposely designed to overcome security architectures and detection mechanisms.

This research work studies various methods and approaches that have addressed Android Malware, designed, analyzed and developed an efficient and effective mechanism to detect Android malware.

REFERENCES

[1]. Canalys. " Small tablets drives big share gains for Android". press release. Aug 2013

[2]. 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].

[3]. 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].

[4]. Guillermo suarez- T, et al. "Evolution , Detection and analysis of malware for smart devices". IEEE communication Survey and tutorials. Issue 2, Vol 16. Oct 2013

[5]. Sven Dietrich. " Detection of intrusion and malware, and vulnerability assessment" Egham, UK. July 2014.

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

[7]. C.L. Lodin, S.W.; Schuba. Firewalls fend off invasions from the net.Spectrum, IEEE, 35(2):26–34, Feb 1998.

[8]. Peter Szor.Virus Research and Defense, chapter 11 Antivirus DefenseTechniques, pages 425–491. Symantec Press, 2005.

[9]. R.A. Kemmerer and G. Vigna. Intrusion detection: A brief historyand overview.Computer, 35(4):27–30, Apr 2002.

[10]. Onyedeke Obinna C, et al, " Anomaly network based intrusion detection system using hybrid techniques" international journal of innovative research and development. vol 8, issue 3, 2020.

[11]. William Enck, Machigar Ongtang, and Patrick Drew McDaniel. Onlightweight mobile phone application certification. InACM Con-ference on Computer and Communications Security, pages 235–245,2009.

[12]. 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.

[13]. 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.

[14]. David D. Hwang, Patrick Schaumont, Kris Tiri, and Ingrid Ver-bauwhede. Securing embedded systems.Security & Privacy Magazine,IEEE, 4(2):40–49, 2006.

[15]. 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.

[16]. 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.

[17]. 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.

[18]. P. Kaushik and A. Jain, "Malware Detection Techniques in Android," *Int. J. Comput. Appl.*, vol. 122, no. 17, pp. 22–26, 2015.

[19]. W. Enck, M. Ongtang, and P. Mcdaniel, "On Lightweight Mobile Phone Application Certification," *ACM Conf. Comput. Commun. Secur.*, 2009.

[20]. 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.

[21]. 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.

[22]. 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.

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