International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2015): 78.96 | Impact Factor (2015): 6.391

NASAM: Novel Approach to Secure Android Devices from Malware based on Apps Behaviour

Sagar Vitthal Shinde[1], Amrita A. Manjrekar[2]

[1]M.Tech Computer Science & Technology, Department of Technology, Shivaji University, Kolhapur, Maharashtra, India

[2]Assistant Professor, Department of Technology, Shivaji University, Kolhapur, Maharashtra, India

Abstract: ***Android is preferred platform for mobile devices. Smartphone's and mobile tablets are speedily indispensable in way of life. Android has been the most widespread open sources mobile OS. On the one aspect android users are increasing, but other side malicious activity also at the same time increasing. The risk of malware (Malicious apps) is sharply increasing in android platform, android mobile malware detection and prevention has become a very important research topic. Some malware attacks will build the phone partially or totally unusable, cause unwanted SMS/MMS (short message service/multimedia messaging service) charge, money, or expose personal data various applications contain wrong or incorrect info conduct code, however those don't seem to be really malicious apps. Present system categories such apps as malware apps, which may create problems in a system. The more accurate/proper system is required to classify malware apps. This NASAM system classifies android applications with the help of recent feature extraction algorithm. In this system android features are taken from feature set to detect malware on four phases: package, user, application, and validation phase. The malware detection is based on behavioural and classified according to their risk (High, Medium, and Low). This is useful for the user to handle the system (Application) very smoothly & will provide more secure system.***

Keywords: Android Security, Android Permissions, Android malware detection and prevention, feature extraction, behavioral base, Risk analysis.

1. Introduction

Today's world is mobile (Smartphone) world. Due to low prices of Smartphone's is available in 3G and 4G networks. Smartphone's and tablets have become extremely popular in the last years. At the end of 2014, the number of active mobile devices worldwide was almost 7 billion, and in developed nations the ratio between mobile devices and people is estimated as 120.8 % [22]. Most of these many are common peoples who don't know what is the android structure and how android system work. Due to well interface and openness android is very popular day by day. There are much more application available which increase performances of system as well as reduce time and cost. The user is unaware of which application is good and which are harmful. Many apps are stealing user personal data. More than 1 millions of malicious apps are currently available in the world [23]. Many apps looks like as normal and useful apps, but they hide treacherous code which performs actions in the background that threatens the user privacy, the device integrity, or even user's credit [37]

Android is an open-source mobile operation system. It is developed & Maintained by Google and is based on a Linux kernel. Currently it is become very popular because of its user friendly interface & many other features. Numbers of users with android phone are increased exponentially in recent years. As there is increased in number of users thread to android mobile is also increased over the year. Mobile malware app's that perform malicious activity like misusing user's important data & personal info by sending messages etc. Malware may be a kind of malicious software package that interrupt the various operations on mobile system, crashes the necessary info or personal info of the mobile system. In different words malicious software, malware refers to software programs designed to break or do alternative unwanted actions on a mobile system.

Furthermore, all these misbehaviors may be performed on android devices in background while many times user is unaware about it. (Or once it's too late). It's been recently reported that nearly 60 % of existing malware send stealthy premium rate SMS messages. Most of those behaviors are exhibited by a category of apps known as Trojanized that may be found in online marketplaces not controlled by Google. However, additionally Google Play, the official marketplace for android apps, has hosted apps that are found to be malicious [1] [21].

Existing system consist of some limited features of android app, malware detection is based on behavioral base. The malware detection and prevention method is static that produce some issues such as it increase false positive rate. Malicious apps (generically known as malware) represent the most vectors for security attacks against mobile devices. Disguised as traditional and useful apps, they hide treacherous code that performs actions inside the background that threatens the user privacy, the device integrity, or even user's credit. Some common samples of attacks performed by android malicious apps are stealing contacts, login credentials, text messages, or maliciously subscribing the user to costly premium services.

All private companies and government organizations moved their work to android application. The chances of information leakage and theft of personal data is increased. As existing system focus on limited features of android application to detect malware. This Proposed NASAM (Novel Approach to Secure Android Mobile) system will detection malware dynamically and more accurate in android device with

variable no of feature extraction [37].

2. Literature Survey

There are various methods available for android malware detection, classification & prevention in literature. It has been observed that mainly three approaches were considered which are as follows:

- **Signature-based detection:** may be a widespread technique supported searching for antecedently outlined virus signatures in input files [23]. Signature detection has the advantage of detection malicious activity before the system is infected by the malicious code.
- **Behavior checking:** is another standard technique supported a behavior checker that resides inside the memory looking for uncommon behavior [23]. Throughout this case, the user is alerted. Behavior checker encompasses a drawback that by the time a malicious activity is detected, some changes have already been done to the system.
- **Integrity Checker:** is the technique that maintains a log of all the files that unit of measurement gift inside the system. The log may contain characteristics of files similar to the file size, date/time stamp and substantiation. Whenever associate degree integrity checker is run, it will check the files on the system and compares with the characteristics it had saved earlier[23]

Depending upon types of malware detection method/technique the existing approaches are classified and listed below

Host based malware detection

Andrea Saracino at. [1] Presented MADAM, a novel host-based malware detection system for android devices this at the same time analyzes and correlates features at four levels: kernel, application, user and package, to sight and stop malicious behaviors. MADAM has been designed to require under consideration those behaviors characteristics of almost each real malware which may be found within the wild. MADAM detects and effectively blocks more than of 96 % malicious apps, that return from 3 massive datasets with regarding 2,800 apps, by exploiting the cooperation of parallel classifiers and a behavioural based mostly detector.

Inter-App permission Leakage

Hamid Bagheri at [2], presents COVERT, a tool for integrative analysis of android inter-app vulnerabilities. COVERT's analysis is standard to *alter incremental* analysis of applications as they're put in, updated, and removed. It statically analyzes the reverse *built source* code of every individual app, and extracts relevant security specifications during a format appropriate for formal verification. Given a group of specifications extracted during this means, a proper analysis engine (e.g., model checker) is then wont to verify whether or not it's safe for a combination of applications holding certain permissions and doubtless interacting with one another to be put in along.

ICC Detector trained model for malware detection

Author Proposed [4] a new malware detection technique, named inter-component communication Detector. Inter component communication Detector outputs a detection model once coaching with a collection of benign apps and a collection of malwares, and employs the trained model for malware detection. The performance of inter component communication Detector is evaluated with 5264 malwares, and 12026 benign apps. Compared with their benchmark, that may be a permission-based technique proposed by Peng et al. in 2012 with associate accuracy up to 88.02%

ALTERDROID, a dynamic analysis approach for detecting such hidden or obfuscated malware

In this paper [5] they had describe ALTERDROID, a run-time analysis approach for detection such as hidden or obfuscated malware elements distributed as parts of an app package. The key plan in ALTERDROID consists of analyzing the behavioural variations between the original app and variety of automatically generated versions of it, wherever a number of modifications (faults) are rigorously injected. Noticeable differences in terms of activities that seem or vanish within the changed app are recorded

Exchanging data using two detection methods

The Author have aim [6] to spot malware covertly exchanging knowledge exploitation two detection ways supported artificial intelligence tools, like neural networks and decision trees. To verify their effectiveness, seven covert channels are enforced and tested over measure framework exploitation android devices. Experimental results show the practicable ness and effectiveness to notice the hidden knowledge exchange between colluding applications.

Privacy preserving data-leak detection

In this technique author presented [10] a privacy conserving information/data -leak detection (DLD) resolution to resolve the difficulty wherever a special set of sensitive data digests is used in detection. The advantage of our methodology is that it permits *the information* owner to securely delegate the detection operation to a semi honest provider while not revealing the sensitive data to the provider. They describe however web service providers can give their customers DLD as associate add-on service with robust privacy guarantees.

Permission - induced risk in Android apps

In this paper [12], they explore the permission-induced risk in android apps on 3 levels during a systematic manner. First, they completely analyze the danger of a personal permission and also the risk of a bunch of collaborative permissions. They employ 3 feature ranking methods, namely, mutual info, correlation coefficient, and T-test to rank android individual permissions with relevancy their risk. They have a tendency to then use sequent forward choice furthermore as principal element analysis to spot risky permission subsets.

VetDroid, a dynamic analysis platform

In This paper [13] they have presented VetDroid, a run-time analysis platform for typically analyzing sensitive behaviors in android apps from a unique permission use perspective.

VetDroid proposes a scientific permission use analysis technique to effectively construct permission use behaviors, i.e., however applications use permissions to access (sensitive) system resources, and the way these acquired permission-sensitive resources area unit additional utilized by the application. With permission use behaviors, security analysts will easily examine the internal sensitive behaviors of an app.

Solution that leverages a method to assign a risk score to each app

Christopher S. Gates [15] had proposed a solution that leverages a way to assign a risk score to every app and show a outline of that data to users. Results from four experiments are reportable during which they examine the effects of introducing summary risk data and the way best it is.

3. Android Malware Overview

There are attack vectors exist that compromises security of mobile devices [24]. 3 main classes of attacks may be carried over mobile devices that includes- malware attacks, grayware attacks and spyware attacks represented as:-

3.1 Malware

These quite attacks steal personal information from mobile devices and injury devices [24]. With device vulnerabilities and luring user to put in extra apps, attacker will gain unauthorized root access to devices. a number of the malware attacks are listed as:-

- Bluetooth attacks: With Bluetooth attacks; attacker may insert contacts or SMS messages, steals victim's information from their devices and might track user's mobile location. Blue-bugging is quite blue-tooth attack through that attacker may listen conversations by activating software together with malicious activities [24].
- SMS attacks: Through SMS attacks, attacker will advertise and spread phishing links. SMS messages may also be utilized by attackers to exploit vulnerabilities [24].
- GPS/Location attacks: User's current location and movement is accessed with global positioning system (GPS) hardware then data is sold-out to different companies concerned in advertising [24].
- Phone jail-breaking: With jail-breaking, AN attacker will remove security implications of software system like it permits OS to install extra and unsigned applications. Users are interested in install them as they might get extra functionality [24].
- Premium rate attacks: They posed serious security considerations as a result of premium rate SMS messages might go unnoticed till attacker faces thousands or dollars of bill on his device as they do not need permissions to send SMS on premium rated numbers [24].

3.2 Grayware

Grayware include applications that collects the information from mobile devices for marketing purposes. Their intention is build no hurt to users however annoy them. consumes resources that may be used for reporting suspicious behaviour of application to android market [24].

3.3 Spyware

Spyware collects personal info from user's phone like contacts, call history and location. Personal spyware are ready to gain physical access of the device by putting in software system without user's consent. By assembling info regarding victim's phone, they send it to attacker who installed the app rather than the author of the application [24]

Detection Methods of Android Malware

Presently, there are two Methods to detect Android Malware: static behavioral detection method and dynamic behavioral detection method.

A. Static analysis

Many researchers recommend exploitation static techniques for detecting attainable malicious behavior while not really executing the application. These will include extracting permissions requested from the Manifest file likewise as analyzing info more experienced Intents, Inter-Component Communication and API calls. DroidMat, a tool proposed in [25] extracts these info from the byte-code and applies K-means & EM cluster algorithms to classify the app as malware or benign. Another recent paper [26] discusses associate app referred to as stowaway that calculates the permissions that an app actually uses supported its API calls and compares it with the permissions requested by the app from the manifest file to find malicious behaviour.

- **Packages imported by the app:** This is considered by zhou et al.[27] in their projected app Droid Ranger. It uses a heuristic primarily based approach for detection unknown malwares. This involves longing for dynamic loading of untrusted code (for e.g., use of Dex Class Loader) likewise as looking for suspicious native code placed in non-standard locations.
- **Data flow policies via app manifest and content providers**: Fuchs et al. [28] proposed Scan Droid as a tool that performs information flow analysis for generating automatic security certification for android applications. It detects intra component flows by analyzing Uri-based addressing present in calls to content suppliers. It conjointly detects inter-component flows by analyzing intent-based addressing present within the manifest file.
- **Message passing through Intents**: Chin et al. [29] proposed ComDroid, a tool that analyses android applications to observe communication based mostly vulnerabilities. The tool statically analyses Dalvik executable files, performs flow sensitive intra procedural analysis, and examines the permissions outlined by the app, Intents sent by the app also as elements that receive Intents. Warnings are issued on detecting potential vulnerabilities.

B. Dynamic analysis

Dynamic or behaviour primarily based analysis techniques involve running the app during a controlled atmosphere, monitoring and analyzing the actions performed by the

app.[34] provides a comprehensive summary of various automated dynamic analysis techniques. Whereas considered more effective against various polymorphic and metamorphic malwares that evade static analysis, dynamic analysis suffers from being extremely resource intensive.

- **Data and control flow analysis:** *TaintDroid* proposed by Enck et al. [30] provides system-wide dynamic taint tracking for android. TaintDroid marks knowledge originating from sensitive sources like GPS, camera, microphone and alternative phone identifiers and monitors all network interfaces (taint sinks) for potentially sensitive information leaks.
- **Emulation based analysis:** *DroidScope* proposed by Yan et al. [31] uses instruction traces, profile API-level activity etc uses virtual machine self-examination to mirror the 3 levels of an android device: hardware, OS and Dalvik Virtual Machine facilitating collection of detailed native and Dalvik instruction traces, profile API-level activity etc. *AASandbox* proposed by Blasing, Thomas et al. [32] executes the app in AN isolated sandbox setting to analyze low level interactions with the system.
- **Logged behaviour sequence:** Zhao et al. [33] proposed Anti MalDroid to check android malware that use logged behavior sequence because the feature, and construct the models for further detection of malware and its variants effectively in execution time.

4. NASAM

This NASAM (Novel Approach to Secure Android Mobile) System consists of new feature extraction algorithm [2] to extract appropriate feature of android application. The malware detection is based on behavioral. This system classifies android apps into two type's i.e. Benign app or malicious app (not malware or harmful). The system will also give risk rank (High Medium low risk involved) of that particular app. This is helpful for the user to handle various applications in his android devices very smoothly.

This System consists of three modules which are as follows:

1) Pre-App Monitor
2) Risk Analysis
3) User Interface

I. Pre-App Monitor

To derive the features at the four system levels, and to detect and prevent a misbehavior, NASAM can be logically decomposed into three main architectural blocks, which are depicted in Fig. 1 (in particular, see "NASAM Architecture"). The first one is the Pre App Monitor, which includes Android application, feature set and Extracted features. The modem extractor algorithm is used to extract the features of selected application. These features are stored in feature set for malware checking. The following algorithm [2] is used to extract the application features.

Extraction of features of android app [2]

Algorithm 1. Model Extractor

Input: app: Android App
Output: A: App's Extracted Model

```
1 A ← < {}; {}; {}; {}; {}>
2 ICFG ← {}
3 summaries ← {}
//►Entity Extraction cf. Sec. 5.1
4 A:C ← extractManifestComponents (app)
5 A:P ← extractManifestPermissions(app)
6 A:F ← extractManifestFilters(app)
7 IFEntities ← {}
8 foreach method € app do
9 IFEntities ← identifyIFEntity(method; summaries)
10 end
11 resolveIFEntityAttr(IFEntities)
12 A:I ← getIntents(IFEntities)
13 A:F ← getIntentFilters(IFEntities) Ư A:F
//►ICFG Augmentation - cf. Sec. 5.2
14 G ← constructICFG(app)
15 E ← extractImplicitCallBacks(app)
16 ICFG ← augmentICFG(G;E)
//►Vul. Paths Identification - cf. Sec. 5.3
17 A:S← findVulPaths(A:C; ICFG)
```

II. Risk Analysis

The second block is the Risk Analysis, which detect malware in extracted feature and classify the application into two categories (Malware app & Benign App). These features are monitored regardless of the specific app or system components generating them, and are used to shape the current behavior of the device itself. Then, these behaviors are classified as genuine (normal) or malicious (anomalous) by the Classifier component. This model include an analysis of metadata of an app package (apk) (permission and market data), after the app is installed on the device [36]. This evaluation computes the app's risk score, i.e. High, Medium, and Low. The computed risk score will be displayed on user screen.

III. User Interface

The third block is the User Interface & Prevention component includes the Prevention module, which stops malicious actions and, in case a malware is found, handles the procedure for removing malicious apps using the User Interface (UI). The UI handles notifications to device user, in particular: (i) the evaluation of the risk score of newly-downloaded apps (ii) Based on this risk evaluation, user will take decision wither to keep app or uninstall app. The activity monitor continuously monitors behavior of application at run-time.

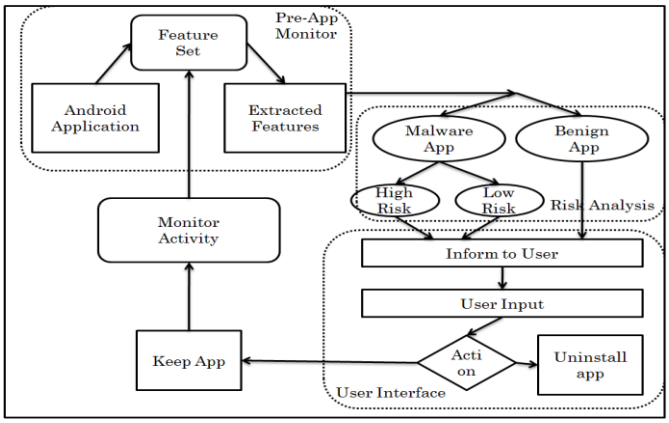

Figure 1: NASAM Architecture

5. Result & Analysis

This NASAM system is compared with existing malware detection system MADAM (Multi-Level Anomaly Detector for Android Malware) [1]. The following table shows comparative overview of these two systems based on selected attributes

Table 1: Comparison between Existing & Proposed System

Sr. No	Attributes	MADAM	NASAM
1	Feature Extracted	Limited	Full
2	User Interaction	No	Yes
3	False Positive Rate	High	Reduced
4	Speed of Operation	High	Low

Here we have tested and analyzed the implemented NASAM system on different configuration mobile devices and also considers various factors to analyze system performances. The CPU load of every application is analyzed on four different mobile devices with different configuration. The device (Mobile 1) has very low configuration, device (Mobile 2) has low configuration, the device (Mobile 3) has medium configuration, while mobile device 4 has highest configuration. The details of these devices are mentioned in table 2.

Table 2: Mobile Devices Used for Testing

Sr No	Mobile Device Names	Operating System	Processor	RAM
1	Micromax Canvas A110 *(Very Low Configuration)*	Android v4.0.4	1 GHz	512 MB
2	Lenovo A319 *(Low Configuration)*	Android v4.4.2 (KitKat)	Dual-core 1 GHz	512 MB
3	Intex Aqua Star *(Medium Configuration)*	Android v4.4.2 (KitKat)	Dual-core 1 GHz	1 GB
4	Panasonic ELUGA *(High Configuration)*	Android v4.4.2 (KitKat)	Quad-Core 1.2 GHz	2 GB

The test is done on four levels which is started with simple small benign app. In Level 2 we have selected another benign app with large size. In level 3 we have selected the malware app with small size. In level 4 we have selected malware app with large size. The same procedure is executed on remaining three mobile devices with configuration mentioned in table 2. The result graph of this analysis is given in figure 2.

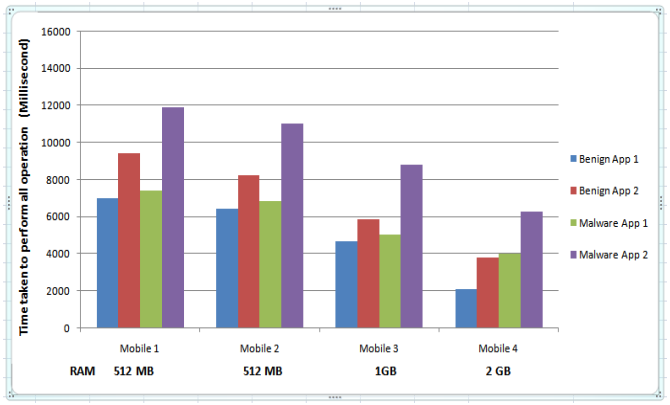

Figure 2: CPU time taken by each application on four devices

Fig 2: shows CPU time taken in millisecond by four different applications on four different android devices which indicates that CPU time depends on application size

NASAM is based on feature extraction algorithm of android apps. Each time depending upon app it extracts variable number of features of the particular app. Fig 3 shows number of feature extracted by NASAM

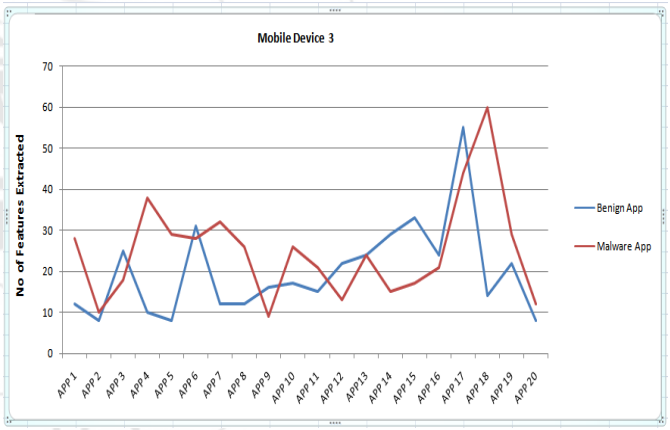

Figure 3: No of Feature extracted by NASAM

Fig 3: shows number of feature extracted for each application by the NASAM on mobile devices 3(Medium Configuration).

As this system extract different features depending upon android application .time required to extract these features also various

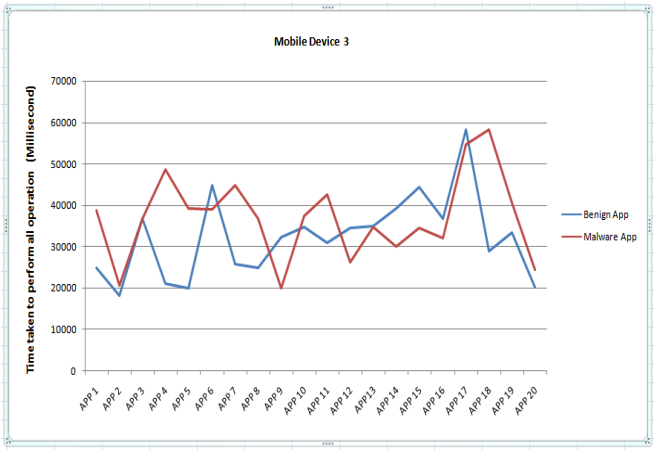

Figure 4: CPU time taken by each application

Fig 4: shows CPU time taken in millisecond by 20 different applications on mobile device 3(Medium configuration) which indicates that this system runs very smoothly on any configuration device with any application.

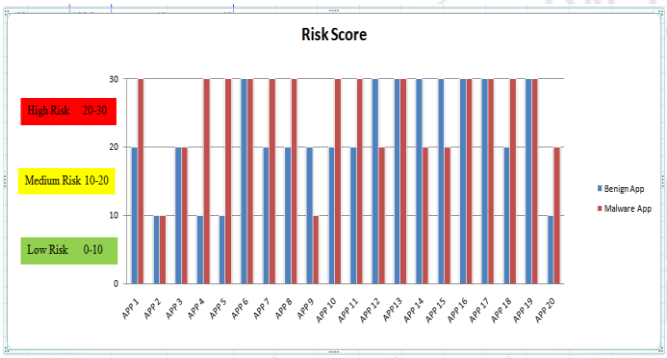

Figure 5: Risk Score of each application

Fig 5: shows risk score (High Medium Low) millisecond of 20 different applications on mobile device 3 (Medium configuration).

6. Conclusion

This System is consisting of new feature extraction algorithm to extract appropriate features of android application. The malware detection is based on behavioral pattern. This System classifies apps into two type's malicious or benign app (not malware present). It also gives risk involved with that app (high medium and low risk).This is helpful for the user to handle the Android devices as well as Application very smoothly.

Experimental result shows that for checking the malware in every (Benign & Malware) application CPU time depends on application size. RAM Usage indicates that this system run very smoothly on any configuration device. If the security problems of Android platform and the approval mechanism are not improved in the process of the future development, the security problems of Android platform would become serious thread.

This system focused on detection of misbehavioural app using model extractor algorithm in future more detail information about malware can be provided to user, also updated dataset can be used to test accuracy of malware detection system.

7. Acknowledgement

I express my sincere thanks to Ms. A. A. Manjrekar whose supervision, inspiration and valuable guidance helped me a lot to complete my Project. Her guidance proved very valuable for me to overcome all the hurdles in the fulfillment of this Project.

I would also express my gratitude towards my colleagues and friends for the moral and technical support throughout the duration of my design paper. Also I am thankful to all those who have helped me in the completion of Project work.

References

[1] Andrea Saracino, Daniele Sgandurra, Gianluca Dini and Fabio Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention", IEEE Transactions on Dependable and Secure Computing , 2016.
[2] Hamid Bagheri, Member, IEEE, Alireza Sadeghi, Joshua Garcia, and Sam Malek, Member, IEEE, "COVERT: Compositional Analysis of Android Inter-App Permission Leakage" IEEE Transactiton on software engineering,2015
[3] Shancang Li, Theo Tryfonas, Gordon Russell, and Panagiotis Andriotis, "Risk Assessment for Mobile Systems Through a Multilayered Hierarchical Bayesian Network", IEEE TRANSACTIONS ON CYBERNETICS, VOL. 46, NO. 8, AUGUST 2016
[4] Ke Xu, Yingjiu Li, and Robert H. Deng "ICCDetector: ICC-Based Malware Detection on Android", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 11, NO. 6, JUNE 2016
[5] Guillermo Suarez-Tangil, Juan E. Tapiador, Flavio Lombardi, and Roberto Di Pietro, "ALTERDROID: Differential Fault Analysis of Obfuscated Smartphone Malware" IEEE TRANSACTIONS ON MOBILE COMPUTING, VOL. 15, NO. 4, APRIL 2016.
[6] Luca Caviglione, Mauro Gaggero, Jean-François Lalande, Wojciech Mazurczyk, Senior Member, IEEE,and Marcin Urba´nski "Seeing the Unseen: Revealing Mobile Malware Hidden Communications via Energy Consumption and Artificial Intelligence", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 11, NO. 4, APRIL 2016
[7] Xiaokui Shu, Jing Zhang, Danfeng (Daphne) Yao, Senior Member, IEEE, and Wu-Chun Feng, Senior Member IEEE, "Fast Detection of Transformed Data Leaks", IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 11, NO. 3, MARCH 2016
[8] Chunjie Zhou, Shuang Huang, Naixue Xiong, Senior Member, IEEE, Shuang-Hua Yang, Senior Member, IEEE,Huiyun Li, Yuanqing Qin, and Xuan Li, "Design and Analysis of Multimodel-Based Anomaly Intrusion Detection Systems in Industrial Process Automation" IEEE TRANSACTIONS ON SYSTEMS, MAN, AND

CYBERNETICS: SYSTEMS, VOL. 45, NO. 10, OCTOBER 2015

[9] Jemal Abawajy, Senior Member, IEEE, Morshed Chowdhury and Andrei Kelarev, "Hybrid Consensus Pruning of Ensemble Classifiers for Big Data Malware Detection", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 3, NO. 2, OCTOBER 2015

[10] Xiaokui Shu, Danfeng Yao, *Member, IEEE*, and Elisa Bertino, *Fellow, IEEE* "Privacy-Preserving Detection of Sensitive Data Exposure" , IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 10, NO. 5, MAY 2015

[11] Parvez Faruki, Ammar Bharmal, Vijay Laxmi, Vijay Ganmoor, Manoj Singh Gaur,Mauro Conti, Senior Member, IEEE, and Muttukrishnan Rajarajan "Android Security: A Survey of Issues, Malware Penetration, and Defenses" IEEE COMMUNICATION SURVEYS & TUTORIALS, VOL. 17, NO. 2, SECOND QUARTER 2015

[12] Wei Wang, Xing Wang, Dawei Feng, Jiqiang Liu, Zhen Han, and Xiangliang Zhang, *Member, IEEE,* "Exploring Permission-Induced Risk in Android Applications for Malicious application Detection" IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 11, NOVEMBER 2014

[13] Yuan Zhang, Min Yang, Zhemin Yang, Guofei Gu, Peng Ning, and Binyu Zang, "Permission Use Analysis for Vetting Undesirable Behaviors in Android Apps" IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 11, NOVEMBER 2014

[14] Silvio Cesare, Member, IEEE, Yang Xiang, Senior Member, IEEE , and Wanlei Zhou, Senior Member, IEEE, "Control Flow-Based Malware Variant Detection" IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 4, JULY/AUGUST 2014

[15] Christopher S. Gates, Jing Chen, Ninghui Li, Senior Member, IEEE, and Robert W. Proctor, "Effective Risk Communication for Android Apps" IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 3, MAY-JUNE 2014

[16] Zhiyong Shan and Xin Wang "Growing Grapes in Your Computer to Defend Against Malware" IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 2, FEBRUARY 2014

[17] Vaibhav Rastogi, Yan Chen, and Xuxian Jiang ,"Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation Attacks" IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 1, JANUARY 2014

[18] Wei Peng, Student Member, IEEE, Feng Li, Member, IEEE,Xukai Zou, Member, IEEE, and Jie Wu, Fellow, IEEE "Behavioral Malware Detection in Delay Tolerant Networks", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 1, JANUARY 2014

[19] Junghwan Rhee, *Member, IEEE*, Ryan Riley, *Member, IEEE*, Zhiqiang Lin, *Member, IEEE*, Xuxian Jiang, and Dongyan Xu, *Member, IEEE*, "Data-Centric OS Kernel Malware Characterization" IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 9, NO. 1, JANUARY 2014

[20] Guillermo Suarez-Tangil, Juan E. Tapiador, Pedro Peris-Lopez, and Arturo Ribagorda, "Evolution, Detection and Analysis of Malware for Smart Devices", IEEE COMMUNICATIONS SURVEYS & TUTORIALS, VOL. 16, NO. 2, SECOND QUARTER 2014

[21] http://www.symantec.com/connect/blogs/another-media-stealing-app-found-google-play

[22] "Global mobile statistics 2014 part a: Mobile subscribers;handset market share; mobile operators," http://mobiforge.com/research-analysis/global-mobile-statistics-2014-part-a-mobile subscribers-handset-market-share-mobile-operators, 2014.

[23] D.Venugopal, "An Efficient Signature Representation and Matching Method fo Mobile Devices," *Proc. 2nd Annual International workshop on Wireless Internet* (WICON '06), Boston, MA, United States, 2006. doi: 10.1145/1234161.1234177.

[24] D.Stites, A.Tadimla "A Survey Of Mobile Device Security: Threats, Vulnerabilities and Defenses./urlhttp://afewguyscoding.com/2011/12/survey-mobile-device security-threats vulnerabilities defenses."

[25] Wu, Dong-Jie et al. "DroidMat: Android Malware Detection through Manifest and API Calls Tracing." *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference* on 9 Aug. 2012: 62-69.

[26] Felt, Adrienne Porter et al. "Android permissions demystified." *Proceedings of the 18th ACM conference on Computer and communications security* 17 Oct. 2011: 627-638.

[27] Russello, Giovanni et al. "Yaase: Yet another android security extension." Privacy, security, risk and trust (passat), *2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)* 9 Oct.2011: 1033-1040.

[28] Fuchs, Adam P, Avik Chaudhuri, and Jeffrey S Foster. "SCanDroid: Automated security certification of Android applications." Manuscript, Univ. of Maryland, http://www. s. umd. edu/~ avik/projects/scandroidascaa (2009).

[29] Chin, Erika et al. "Poster: Analyzing Inter-Application Communication in Android."

[30] Enck, William et al. "TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones." OSDI 4 Oct. 2010: 255-270.

[31] Yan, Lok Kwong, and Heng Yin. "Droidscope: seamlessly reconstructing the os and dalvik semantic views for dynamic android malware analysis." *Proceedings of the 21st USENIX Security Symposium* 8 Aug. 2012.

[32] Blasing, Thomas et al. "An android application sandbox system for suspicious software detection." Malicious and Unwanted Software (MALWARE), 2010 *5th International Conference* on 19 Oct. 2010: 55-62.

[33] Zhao, Min et al. "AntiMalDroid: An Efficient SVM-Based Malware Detection Framework for Android." *Information Computing and Applications (2011):* 158-166.

[34] Rahul Raveendranath, Venkiteswaran Rajamani, Anoop Joseph Babu "Android Malware Attacks and Countermeasures: Current and Future Directions" *International Conference on Control, Instrumentation,*

Communication and Computational Technologies (ICCICCT)

[35] Nitesh Patil, Dr. K. V. Kulhalli, "A Survey: Inter-App Permission Leakage on Android Devices", *IJARCSSE*-Sept 2015.

[36] Nitesh Patil, Dr. K. V. Kulhalli, "Review of Inter-App Permission Leakage and Malware Characterization in Android Operating System", *iCETETA*, March-2017.

[37] Sagar V. Shinde, Ms Amrita A. Manjreka, "A Review Paper on Effective Behavioral Based Malware Detection and Prevention Techniques for Android Platform", *International Journal of Engineering Research and Technology. ISSN 0974-3154 Volume 10, Number 1* (March 2017)