www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 5 May 2017, Page No. 21378-21389
Index Copernicus value (2015): 58.10 DOI: 10.18535/ijecs/v6i5.32

Behavioural Analysis of Android Malware using Machine Learning

Lokesh Vaishanav[1],Shanu Chauhan[1], Hrithik Vaishanav[2], Mahipal Singh Sankhla[3], Dr. Rajeev Kumar[4]

[1]Students of Bachelor of Technology in Computer Science, School of Computer Science Engineering, Galgotias University, Greater Noida.

[2]Student of Bachelor of Technology in Information Technology, Techno India NJR, Collage, Udaipur

[3]Student of M.Sc. Forensic Science, [4]Assistant Professor, Division of Forensic Science, School of Basic and Applied Sciences, Galgotias University, Greater Noida.

Abstract

The arrival of android platforms with increased storage capabilities, better visualisations, computing competencies and the ubiquitous use of these platforms in the field of e-banking, online banking, business, and the storage of sensitive information on these devices, android platform is becoming the most targeted platform by malwares. They are primarily spread via repackaged apps to piggyback payload, update attacks, and drive by downloads. Malware constitutes a severe menace to user privacy, money, device and file veracity. It is posing benevolence challenges and difficulties to detect such malwares as signature based detection techniques available today are becoming inefficient in sensing new and anonymous malware. Hence we presents machine learning as an emerging era of modified and latest detection techniques. In this paper we will present various machine learning solutions to counter android malwares that analyse features from malicious application and use those features to classify and detect unknown malicious applications. This paper summarizes the evolution of malware detection techniques based on machine learning algorithms focused on the android OS.

Key-Words: Android, Platforms, Malware, Machine, Detection, Malicious.

Introduction

The convenience of interactive mobile devices has enticed their users, who now carry a wealth of sensitive information around with them: personal data, bank information and account details, GPS location, contacts, text messages and emails [1-5]. The value of these data has attracted cyber-criminals who invest time and money in exploiting vulnerable mobile platforms, commonly through malware. Google's Android platform has become the most targeted mobile operating system, likely for two key reasons [6]. On one hand, Android is a ubiquitous platform, with more than 1.9 billion installed-base devices [7]. On the other hand, Android applications are easy to reverse-engineer and can be readily modified or repackaged. Since attackers focus their energy on targets that have the highest return on investment, popular platforms like Android with accessible inner workings are doomed to attract special attention from cyber-criminals [8]. Mobile malware may perform malicious activities like steal data, send credentials to attackers, send premium SMSs [9]. Expecting a shipment of 1 billion Android devices in 2017 and with over 50 billion total app downloads since the first Android phone was released in 2008, cyber criminals naturally expanded their vicious activities towards Google’s mobile platform. Mobile threat researchers indeed recognize an alarming increase of Android malware from2012 to 2013 and estimate that the number of detected malicious apps is now in the range of 120,000 to 718,000 [10-13].

Based on the current attack trends and analysis of the present literatures there are following types of malwares

1. **Information Extraction** Compromises the device and steals personal information such as IMEI number, user’s personal information, etc.

2. **Automatic Calls and SMS** User’s phone bill is increased by making calls and sending SMS to some premium numbers

3. **Root Exploits** The malware will gain system root privileges and takes control of the system and modifies the information.

4. **Search Engine Optimizations** Artificially search for a term and simulates clicks on targeted websites in order to increase the revenue of a search engineer increase the traffic on a website.

5. **Dynamically Downloaded code**an installed benign application downloads a malicious code and deploys it in the mobile devices.

6. **Covert channel**a vulnerability in the devices that facilitates the information leak between the processes that are not supposed to share the information.

7. **Botnets** A network of compromised mobile devices with a Bot Master which is controlled by Command and Control servers (C&C). Carry out Spam delivery, DDDos attacks on the host devices. [14]

An important concern on the growing Android platform is malware detection. Malware detection techniques on the Android platform are similar to techniques used on any platform. Detection is fundamentally broken into static analysis, by analysing a compiled file; dynamic analysis, by analysing the runtime behaviour, such as battery, memory, and network utilization of the device; or hybrid analysis, by combining static and dynamic techniques [15]. Static analysis is advantageous on memory-limited Android devices because the malware is not executed, only analysed. However, dynamic analysis provides additional protection, particularly against polymorphic malware that change form during execution. To use the advantages from both static and dynamic analysis, desktop vendors such as AVG [16] employ hybrid techniques [17].

Behaviour of both malicious and benign applications are profiled with a set of feature vectors, which are snapshots of system state information, such as memory utilization and power consumption. Machine learning algorithms are trained with known feature vectors to attempt to predict the classification of unknown feature vectors. Due to the range of hardware configurations a very large number of feature vectors from a diverse set of hardware are needed to effectively train machine learning algorithms. Future work will explore machine-invariant hardware metrics [17].

CHALLENGES OF EVALUATING MOBILE MALWARE

Classifiers while malware classifiers have the potential to detect and proactively prevent the spread of malware on mobile devices, there are a number of challenges to determining which techniques are most effective at detecting malware. A critical challenge is the need for the collection and experimentation with a large dataset for training malware classifiers, typically spanning hundreds of applications and thousands of feature vectors. These datasets can be difficult to collect accurately, as there is an inherent trade-off between profiling malware operating maliciously, such as gaining network access on a mobile device, and ensuring that both the malware remains within its sandbox and the malware profile remains accurate. Moreover, malware classifiers must be trained and evaluated in a repeatable and consistent manner with large-scale experimentation and automation infrastructure.

- **Stream:** STREAM, a System for automatically Training and Evaluating Android Malware classifiers and provides an effective method of rapidly profiling malware and training machine learning classifiers. STREAM can run on a single server or distributed across a grid of remote servers. Figure 1 shows a high– level operational overview of STREAM. The master server distributes profiling jobs to the worker nodes. The node servers then distribute the jobs between devices or emulators in parallel. Inside each device or emulator, STREAM manages the applications, drives the feature vector collection, and manages classifier training and evaluation. The parallelization addresses the size of the problem domain and the STREAM framework provides a combination of accuracy and scalability [17].

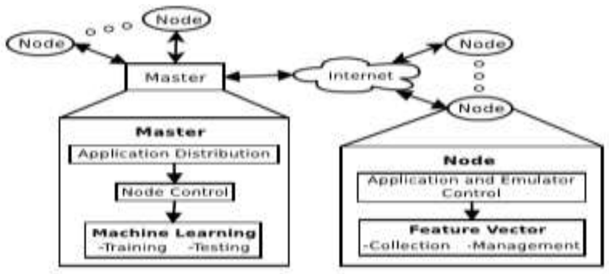

Figure 1. High-level overview of STREAM distributed across multiple nodes [17].

DOI: 10.18535/ijecs/v6i5.32

MADAM (Multi-Level Anomaly Detector for Android Malware

MADAM monitors the device actions, its interaction with the user and the running apps, by retrieving five groups of features at four different levels of abstraction, namely the kernel level, application-level, user-level and package-level. For some groups of features MADAM applies an anomaly based approach, for other groups it implements a signature based approach that considers behavioural patterns that we have derived from known malware misbehaviours. In fact, MADAM has been designed to detect malicious behavioural patterns extracted from several categories of malware. This multi-level behavioural analysis allows MADAM to detect misbehaviours typical of almost all malware which can be found in the wild. MADAM also has shown efficient detection capabilities as it introduces a 1.4% performance overhead and a 4% battery depletion. Finally, MADAM is usable because it both requires little-to-none user interaction and does not impact the user experience due to its efficiency. MADAM achieves the above goals as follows:

(i) It monitors five groups of Android features, among which system calls (type and amount) globally issued on the device, the security relevant API calls, and the user activity, to detect unusual user and device behavioural patterns; to this end, it exploits two cooperating proximity-based classifiers to detect and alert anomalies

(ii) It intercepts and blocks dangerous actions by detecting specific behavioural patterns which take into account a set of known security hazard for the user and the device;

(iii) Every time a new app is installed, MADAM assesses its security risk by analysing the requested permissions and reputation metadata, such as user scores and download number, and it inserts the app in a suspicious list if evaluated as risky.

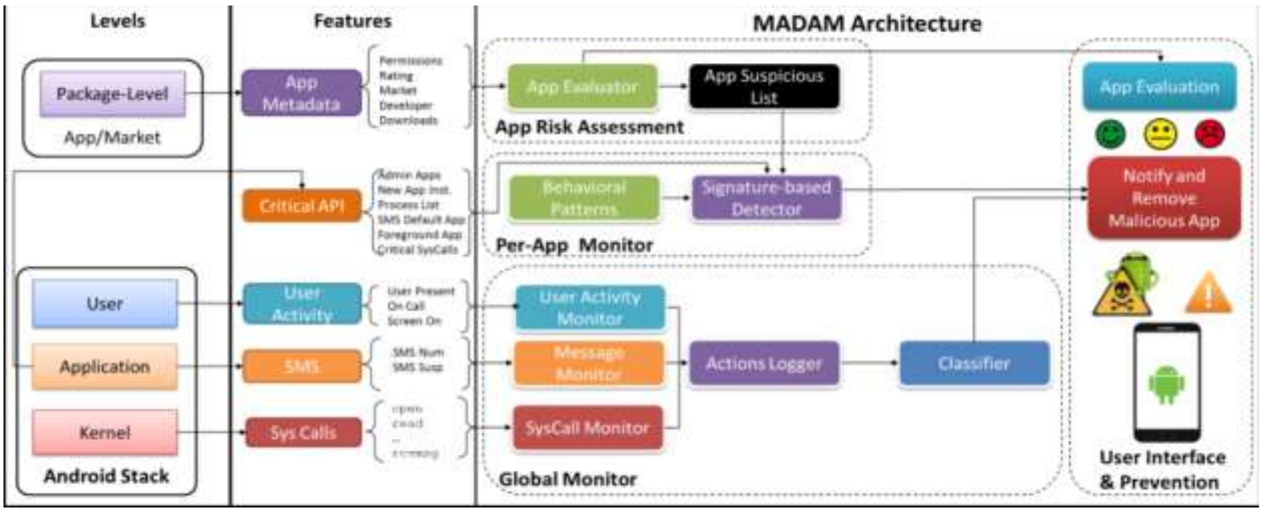

Figure 2. Architecture of MADAM [18].

To derive the features at the four system levels, and to detect and prevent a misbehaviour, MADAM can be logically decomposed into four main architectural blocks, which are depicted in Fig. 2 (in particular, see

“Madam Architecture”). The first one is the App Risk Assessment, which includes the App Evaluator that implements an analysis of metadata of an app package (apk) (permission and market data), before the app is installed on the device. This evaluation computes the app’s risk score, i.e. the likelihood that the app is a malware. Based on this risk evaluation, this component populates a set of suspicious apps (App Suspicious List), which will be monitored at run-time. The second block is the Global Monitor, which monitors the device and OS features at three levels, i.e. kernel (SysCall Monitor), user (User Activity Monitor) and application (Message Monitor). These features are monitored regardless of the specific app or system components generating them, and are used to shape the current behaviour of the device itself. Then, these behaviours are classified as genuine (normal) or malicious (anomalous) by the Classifier component. The third block is the Per-App Monitor, which implements a set of known behavioural patterns to monitor the actions performed by the set of suspicious apps(App Suspicious List), generated by the App Risk Assessment, through the Signature-Based Detector. Finally, the User Interface & Prevention component includes the Prevention module, which stops malicious actions [18].

- **Crowdroid** used a machine learning-based framework that recognizes Trojan like malware on Android Smartphones, it monitored the number of times a particular system call was issued by an application during the execution of an action that requires user interaction. Crowdroid used about 100 system calls with only two trepanised applications tested. [19].

- **Andromaly** is an intrusion detection system that relies on machine learning techniques. It monitors both the Smartphone and user's behaviours by observing several parameters, spanning from sensor activities to CPU usage. Andromaly used 88 features to describe system behaviours besides rooting the device and the use of external Linux server; the features are then pre-processed by feature selection algorithms [20].

- **VMM** approach to malware detection in their design of Paranoid Android system where researchers can perform a complete malware analysis in the cloud using mobile phone replicas. In their work, the phone replicas are executed in a secure virtual environment, limiting their system to no more than 105 replicas running concurrently [21].

- **Neural network** approach this proved effective in detecting fraud calls and imposters. The disadvantage of this method is that the process is relatively slow and this method classifies applications into groups having same behaviours and hence, there will be lot of false positives [22].

This technique focused on viruses that are transmitted through SMS messages and other communication interfaces like Bluetooth and Infrared. But they did not concentrate on worms that will automatically make high rate calls from the mobile device which will incur loss to the user as they are only collecting and monitoring SMS traces [23].

- **HOSBAD** is a Host-based Anomaly Detection System targeted at Android Malware propagated via SMSs and calls. HOSBAD integrates data mining with supervised machine learning techniques in its implementation. It is designed to monitor and extract Android's applications data at the application layer and using these data to detect malware infections using a supervised machine learning approach to differentiate between normal and malicious behaviours of applications.[24]. In fact, the problem of anomaly detection can be seen as a problem of binary classification, in which each normal behaviour is classified as "Standard", whereas abnormal ones are classified as "Suspicious" [25]. The supervised machine learning model could be viewed as a black box having as input sets of behaviours formatted into sets of feature vectors in ARFF and the output is a flag of "Normal" or "Malicious". The supervised machine learning model which in this case is the K-NN classifier is trained using a normality model on how to classify correctly each element of the feature vector. The training of the classifier takes place at the point called the training phase. This phase is critical because the accuracy of the classifier is dependent on the training phase hence a good training set must be supplied to the classifier [24]. To generate a good feature vector that represents typical Android device behaviour HOSBAD utilize features that represents behaviours when the device is active and when it is inactive. However, our training set also contains some malicious behaviour, which strongly differs from the normal ones. Choosing the right features to best represent the device behaviours is a critical task, since their number and correlation determine the quality of the training set [26].

Malware Detection Process

HOSBAD combines features extracted from different categories of the device functionality as given in Fig 3. First, it monitors the device activities and extracts the features associated with these activities. Secondly, it observes correlation among the features derived from the events belonging to the different activities. In order to extract features first, HOSBAD monitors the incoming and out-going SMSs. An application may send SMS during its execution and for this to happen, it must contain the SEND_SMS permission in its manifest file otherwise the application will crash as soon as it tries to send SMS message by invoking the SEND_SMS function. The Permissions required by an application are displayed to the user during installation time and the user must either accept all or forfeit the installation of the application that is, the user must agree with all the stated permissions for the application to get installed. This mechanism provides a rough control that can be effective especially to new Android users. An application that gets the

SEND_SMS permission could be harmful, since it is able to send SMSs including premium SMSs without the knowledge of the user [24].

S/No.	Features
1.	In/Out SMSs
2.	In/Out Calls
3.	Device Status
4.	Running Applications/Processes

Table 1[24].

To monitor and extract the stated features from the device, the design includes three monitors; the call monitor, the SMS monitor and the device status monitor see Figure 4. A collector receives these features from all the monitors and then builds the vectors in .csv format. These vectors are parsed and converted to arff; the format acceptable by Weka machine learning tool and stored in local files on the SD card in .arff using a logger module so that they can be used as test set.

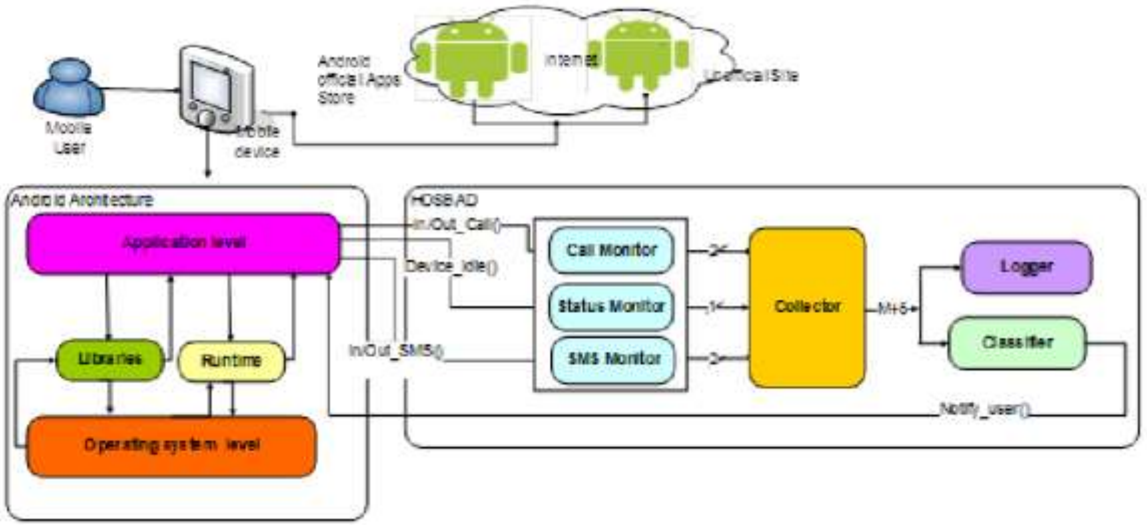

Figure 4: Architecture of the Host-based Anomaly Malware Detection System (HOSBAD) [24].

The classifier module is responsible for performing behaviour-based analysis in which Android applications are classified as either Normal or Malicious. This is done by employing the trained K-NN classifier. A key process of the system is the training phase which identifies the behaviour of the applications. It identifies Android applications into two classifications namely: Normal and Malicious. Figure 4 gives a complete representation of the processes involved in the different phases of the malware detection [24].

MAMA (Manifest Analysis for Malware detection in Android)

A new technique for the detection of malicious Android executable. This approach employs several features extracted by analysing the Manifest file of the Android applications. In particular, we use the permissions and the feature tags within the manifest file. These features are then used to build well-known supervised machine-learning algorithms to detect malicious applications. In summary, our main findings are:

(i) A new method for representing Android applications, based on the permissions and the features from the Manifest file.

(ii) Adoption of well-known machine learning classifiers to provide detection of malicious applications in Android. We found out that machine-learning algorithms can provide detection of malicious applications in Android and that the best representation of executable is the combination of both permissions and features from the Manifest file.

Feature sets that are used in order to detect Android malware are gathered from the AndroidManifest.xml file that is within each Android application. These feature sets are: (i) the permissions required for the application, under the uses-permission tag and (ii) the features under the uses-features group in the Android Manifest File. In order to obtain these features, we first extracted the permissions used by each of the applications. To this extent, we employed the aapt tool (Android Asset Packaging Tool), available within the set of tools provided by the Android SDK [27].

Machine learning algorithms

Machine learning is an area within Artificial Intelligence that develops and designs new algorithms to generalize behaviours using data [28]. Traditionally, these algorithms are categorized with regards to the availability of labelled instances in the training dataset. We have used supervised algorithms, which are the ones that employ a dataset that has been previously labelled (in our case, into malware and benign software). In this section we detail the algorithms employed [27].

- **K-Nearest Neighbours** The K-Nearest Neighbour (KNN) classifier is one of the simplest supervised machine learning models. This method classifies an unknown specimen based on the class of the instances closest to it in the training space by measuring the distance between the training instances and the unknown instance. Even though several methods exist in order to choose the class of the unknown sample, the most common technique is to simply classify the unknown instance as the most common class amongst the K-nearest neighbours [29].
- **Decision Trees Decision** Tree classifiers are a type of machine-learning classifiers that are graphically represented as trees. Internal nodes denote conditions regarding the variables of a problem, whereas final nodes or leaves represent the ultimate decision of the algorithm [30]. Different training methods are typically used for learning the graph structure of these models from a labelled dataset. We used Random

Forest, an ensemble (i.e., combination of classifiers) of different randomly-built decision trees [31, 32, and 30].

- **Bayesian networks** Bayesian Networks [33], which are based on the Bayes Theorem, are defined as graphical probabilistic models for multivariate analysis. Specifically, they are directed acyclic graphs that have an associated probability distribution function [34]. Nodes within the directed graph represent problem variables (they can be either a premise or a conclusion) and the edges represent conditional dependencies between such variables. Moreover, the probability function illustrates the strength of these relationships in the graph [34]. The most important capability of Bayesian Networks is their ability to determine the probability that a certain hypothesis is true (e.g., the probability of an application of being malware) [27].
- **Support Vector Machines(SVM)** SVM algorithms divide the n-dimensional space representation of the data into two regions using a hyperplane. This hyperplane always maximizes the margin between those two regions or classes. The margin is defined by the farthest distance between the examples of the two classes and computed based on the distance between the closest instances of both classes, which are called supporting vectors [35]. Instead of using linear hyperplanes, it is commonto use the so-called kernel functions. These kernel functions lead to non-linear classification surfaces, such as polynomial, radial or sigmoid surfaces [36].

Discussion & Future Scope

The sudden growth of the Android mobile platform has made it a main target of cyber-criminals. Mobile malware specifically aiming Android has gushed and grown in tandem with the rising popularity of the platform. This paper recapitulates recent developments in android malware detection using machine learning algorithms. Detection techniques and systems that uses static, dynamic and hybrid tactics are discussed and highlighted. We must propose new techniques which have the ability to detect advanced malware attacks such as Zero-day attack. Because miscellaneous variants and new types of mobile malware are on the rise, further study on a technique that could detect future malware should be planned Implementation of Behavior-based analysis with permission-based can also be done to determine malicious Android applications. Administrative User interface and an AMDA Android Application will allow easier analysis and access of the system.

Conclusion

In the world of machines everything is becoming smart and number of users connected to internet and smartphones is increasing on vast scale. As huge users are interconnected with internet by smartphones cyber terrorist are aiming to disrupt the whole network of advanced users. They are aiming at the confidential data of business companies, they are stealing abstract information and murdering the cyber world through their actions. Thus it is necessary to halt their activities to large extent so that modern era of internet can be protected. This paper is based on the behavioural analysis of android malwares using various machine learning algorithms already being developed. This paper majorly focus on STREAM, a System for automatically Training and Evaluating Android Malware classifiers and provides an effective method of rapidly profiling malware and training machine learning classifiers, MADAM (Multi-Level Anomaly Detector for Android Malware, (HOSBAD) Host-based Anomaly Detection System, and MAMA (Manifest Analysis for Malware detection in Android) and last various machine learning algorithms are being discussed.

References

1. Leavitt, Neal: Malicious code moves to mobile devices. IEEE Computer, vol. 33, num. 12, pp. 16–19. IEEE (2000).
2. Foley, Simon N, Dumigan, Robert: Are handheld viruses a significant threat? Communications of the ACM, vol. 44, num. 1, pp. 105–107. ACM (2001).
3. Dagon, David, Martin, Tom, Starner, Thad: Mobile Phones as Computing Devices: The Viruses are Coming! Pervasive Computing, IEEE, vol. 3, num. 4, pp. 11–15. IEEE (2004).
4. Hypponen, Mikko: State of cell phone malware in 2007. USENIX (2007). http://www.usenix.org/events/sec07/tech/hypponen.pdf
5. Lawton, George: Is it finally time to worry about mobile malware? Computer, vol. 41, num. 5, pp. 12–14. IEEE (2008).
6. Zhou, Yajin, Jiang, Xuxian: Dissecting Android Malware: Characterization and Evolution. IEEE Symposium on Security and Privacy (SP), pp. 95–109. IEEE (2012). http://www.malgenomeproject.org
7. Strategy Analytics: Global Smartphone Installed Base by Operating System for 88 Countries: 2007 to 2017 (2012). http://www.strategyanalytics. com/default.aspx?mod=reportabstractviewer&a0=7834
8. Paolo Rovelli et al. “PMDS: Permission-based Malware Detection System” International conference on information system security ICISS: 2014information system security pp 338-357
9. Srikanth Ramu “Mobile Malware Evolution, Detection and Defense”,EECE 571B, TERM SURVEY PAPER, APRIL 2012
10. Kindsight Security Labs Malware Report - Q2 2013. Alcatel-Lucent, Jul. 2013.
11. FortiGuard Midyear Threat Report. Fortinet, Aug. 2013.
12. Third Annual Mobile Threats Report. Juniper Networks, Jun. 2013.
13. TrendLabs 2Q 2013 Security Roundup. Trend Micro, Aug. 2013.

14. Raveendranath, V. Rajamani, A.J. Babu, and S.K. Datta. Androidmalwareattacksandcountermeasures:Currentand future directions. In Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on, pages 137–143, July 2014. doi: 10.1109/ICCICCT.2014.6992944.

15. N. Idika and A. P. Mathur, "A survey of malware detection techniques," Purdue University, p. 48, 2007.

16. "Malware Detection Methods," URL http://www.avg.com/us-en/ avg-software-technology.

17. Brandon Amos et al. "Applying machine learning classifiers to dynamic Android malware detection at scale",978-1-4673-2480-9/13/$31.00 ©2013 IEEE

18. Andrea Saracino et al. "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention", IEEE Transactions on Dependable and Secure Computing (Volume: PP, Issue: 99) 01 March 2016 **DOI:** 10.1109/TDSC.2016.2536605

19. Burquera I., Zurutuza U. & Nadjm-Tehrani S., (2011). Crowdroid: Behavior-based Malware Detection System for Android. In Proceedings of the 1st ACM workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 15-26

20. Asaf S., Uri K., Yuval E., Chanan G. & Yael W., (2011). Andromaly: A Behavioural Malware Detection Framework for Android Devices. Journal of Intelligent Information Systems, pp. 1-30. doi: 10.1007/s10844-010-0148-x.

21. Portokalidis G., Homburg P., Anagnostakis, K. & Bos, H., (2010). Paranoid Android: Versatile Protection for Smartphones. In Proceedings of the ACM 26th Annual Computer Security Applications Conference, ACSAC'10, ACSAC '10, New York, NY, USA, pp. 347-356.

22. Mirela S.M., Azzedine B., & Annoni N. (2002). Behaviour-based Intrusion Detection in Mobile Phone Systems. Journal of Parallel and Distributed Computing, 62, pp. 1476-1490

23. Jerry C., Starsky H.Y.W., Hao Y., and Songwu L. (2007). SmartSiren: Virus Detection and Alert for Smartphones. In Proceedings of the 5th International Conference on Mobile Systems, Applications and Services (MobiSys' 07), ACM New York, NY, USA, pp. 258-271. doi: 10.1145/1247660.1247690.

24. Joshua Abah et al. "A MACHINE LEARNING APPROACH TO ANOMALY-BASED DETECTION ON ANDROID PLATFORMS",International Journal of Network Security & Its Applications (IJNSA) Vol.7, No.6, November 2015

25. Dini G., Martinelli F., Saracino A. & Sgandurra, A. (2012). MADAM: A Multi-level Anomaly Detector for Android Malware. Computer Network Security, 7531, pp. 240-253. Retrieved from www.links.springer.com/book.

26. Kwak N., & Choi C.H., (2002). Input Feature Selection for Classification Problems. IEEE Transactions on Neural Networks 13(1) January, 2002. pp. 143-159.

27. Borja Sanz *et al*. "MAMA: Manifest Analysis for Malware Detection in Android", Cybernetics & Systems 44:469-488 · October 2013DOI: 10.1080/01969722.2013.803889

28. Bishop, C.M. 2006. Pattern Recognition and Machine Learning. Springer New York.

29. Fix, E., and J.L. Hodges Jr. 1952. Discriminatory Analysis-nonparametric Discrimination: Small Sample Performance. DTIC Document.

30. Quinlan, J.R. 1986. "Induction of Decision Trees." Machine Learning 1 (1): 81–106. 1993. C4. 5: Programs for Machine Learning. Morgan kaufmann.

31. Breiman, L. 2001. "Random Forests." Machine Learning 45 (1): 5–32.

32. Garner, S.R. 1995. "Weka: The Waikato Environment for Knowledge Analysis." In Proceedings of the 1995 New Zealand Computer Science Research Students Conference, 57–64.

33. Pearl, Judea. 1982. "Reverend Bayes on Inference Engines: a Distributed Hierarchical Approach." In Proceedings of the National Conference on Artificial Intelligence, 133–136.

34. Castillo, E., J.M. Gutiérrez, and A.S. Hadi. 1997. Expert Systems and Probabilistic Network Models. Springer Verlag.

35. Vapnik, V.N. 2000. The Nature of Statistical Learning Theory. Springer.

36. Amari, S., and S. Wu. 1999. "Improving Support Vector Machine Classifiers by Modifying Kernel Functions." Neural Networks 12 (6): 783–789.