ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

A Survey on Machine Learning Techniques for Android Malware Detection and Categorization

Neha Sharma [1], Pooja Yadav [2]

P.G. Student, Department of Computer Science & Engineering, SRCEM at Palwal, Haryana, India[1]

Assistant Professor, Department of Computer Science & Engineering, SRCEM at Palwal, Haryana, India [2]

ABSTRACT: Malware identification is an essential factor in the security of the android frameworks. Nonetheless, as of now used mark based strategies can't give precise identification of zero-day assaults and polymorphic infections, malwares and DDOS (Distributed Denial of Services). That is the reason the requirement for machine learning-based identification emerges. The motivation behind this work was to decide the best component extraction, include portrayal, and characterization techniques that outcome in the best exactness when utilized on the highest point of Dalvik Sandbox (Android Framework). In particular, K-Nearest-Neighbors, Decision Trees, Support Vector Machines, Naive Bayes and Random Forest classifiers were assessed and studied to propose the new amalgamated technique for more accurate and effective results. This study presents suggested strategies for machine learning based malware grouping and location and in addition the rules for its execution. In addition, the investigation will be performed based on strategy for further valuable research in the field of malware examination for android framework with machine learning and its classifications.

KEYWORDS: Learning, Classification, Malware Detection, Support Vector Machines, Random Forest, K-Nearest Neighbours, Naive Bayes, Decision Tree.

I. INTRODUCTION

A Survey on Machine Learning Techniques for Android Malware Detection and Categorization With the quick improvement and immense growth of the Internet, malware wound up one of the major digital dangers these days. Any product performing vindictive activities, including data taking, secret activities, and so on can be alluded to as malware. Kaspersky Labs (2017) characterize malware as a kind of algorithm or program intended to taint a real clients phone or computer and cause hurt on it in different ways. While the decent variety of malware is expanding, hostile to infection scanners which can not satisfy the necessities of insurance, bringing about a large number of hosts and smart phones being assaulted. As indicated by Kaspersky Labs (2016), 6563145 distinct hosts and Smartphone's were assaulted and 4000 one of a kind malware objects were recognized in 2015. Thusly, Juniper Research (2016) predicts the expense of information breaks to increment to $2.1 trillion all inclusive by this year i.e. 2019. Subsequently, malware attack security to mobile frameworks is a standout amongst the most imperative cyber security undertakings for single client and groups too, since even a solitary assault can result in traded off information and adequate misfortunes. Mammoth venerable data loss and continuous assaults direct the requirement for exact and auspicious recognition strategies. Current static and dynamic techniques don't give productive discovery, particularly when administration zero-day attacks and assaults. Thus, machine learning-based strategies can be utilized. This scheme or scenarios talks about the primary concerns and worries of machine learning-based malware recognition, and additionally searches for the best element portrayal and grouping techniques.

The objective of this study is to build up the confirmation of idea and strategy to produce the effective amalgamated malware detector for mobile phones specially using android framework using machine learning and extract the same

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

with respect to android sandbox. This sandbox will be used for the extraction of the conduct of the malware tests, which will be utilized as a contribution to the machine learning calculations. The objective is to decide the best element portrayal strategy and how to highlights the category of malware and to establish the most precise calculation that can recognize the malware families with the least blunder rate. The precision will be estimated both for the instance of identification, whereas the record is malevolent and for the instance in respect to the malware family. The exactness or accuracy to be proposed with come with appropriate, effective outcomes which will likewise be evaluated in connection to current scoring actualized in existing android sandbox whereas the choice of proposed technique in future will performs better solution from previous developed solutions.

II. RELATED WORK

Justin Sahs ; Latifur Khan [1], depicts that, the recent emergence of mobile platforms capable of executing increasingly complex software and the rising ubiquity of using mobile platforms in sensitive applications such as banking, there is a rising danger associated with malware targeted at mobile devices. The problem of detecting such malware presents unique challenges due to the limited resources available and limited privileges granted to the user, but also present unique opportunity in the required metadata attached to each application. In this article, we present a machine learning-based system for the detection of malware on Android devices. Our system extracts a number of features and trains a One-Class Support Vector Machine in an offline (off-device) manner, in order to leverage the higher computing power of a server or cluster of servers.

Ivan Firdausi, Charles lim, Alva Erwin [2] depicts that increase of malware that are exploiting the Internet daily has become a serious threat. The manual heuristic inspection of malware analysis is no longer considered effective and efficient compared against the high spreading rate of malware. Hence, automated behavior-based malware detection using machine learning techniques is considered a profound solution. The behavior of each malware on an emulated (sandbox) environment will be automatically analyzed and will generate behavior reports. These reports will be preprocessed into sparse vector models for further machine learning (classification). The classifiers used in this research are k-Nearest Neighbors (kNN), Naïve Bayes, J48 Decision Tree, Support Vector Machine (SVM), and Multilayer Perceptron Neural Network (MlP). Based on the analysis of the tests and experimental results of all the 5 classifiers, the overall best performance was achieved by J48 decision tree with a recall of 95.9%, a false positive rate of 2.4%, a precision of 97.3%, and an accuracy of 96.8%. In summary, it can be concluded that a proof-of-concept based on automatic behavior-based malware analysis and the use of machine learning techniques could detect malware quite effectively and efficiently.

Wen-Chieh Wu, Shih-Hao Hung [3] depicts that, the proposed DroidDolphin, a dynamic malware analysis framework which leverages the technologies of GUI-based testing, big data analysis, and machine learning to detect malicious Android applications. Based on our automatic testing tools, we were able to extract useful static and dynamic features from a training dataset composed with 32,000 benign and 32,000 malicious applications. Our preliminary results showed that the prediction accuracy reaches 86.1% and F-score reaches 0.857. As the dataset increases, the accuracy of detection increases significantly, which makes this methodology promising.

Ying-Chih Shen, Roger Chien, and Shih-Hao Hung [4] Nowadays, Android-based mobile devices, such as smart phones and tablets, have become increasingly popular, and the number of Android applications is growing dramatically. To examine and validate such a high volume of applications, an automated testing and analysis environment is needed. Such an environment is particularly useful for the detection of malicious applications which steal the users' personal information and incur additional charges. In this paper, we present a testing and analysis framework for detecting such malicious applications. Our framework provides an automatic testing flow with minimal user interventions and is enhanced with heuristics to generate stimuli for speeding up the testing process. Compared to the built-in Monkey Runner toolkit provided by Google, our framework delivered better efficiency in testing and detected more malicious applications with the added heuristics, according to our experimental results.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

III. MALWARE TYPES AND DETECTION METHODS

To have a superior comprehension of the strategies and rationale behind the malware, it is valuable to group it. Malware can be partitioned into a few classes depending on its motivation. The classes are as per the following:

1. **Adware or Admobs:** The main motivation behind this malware type is showing promotions on the Android Phones or Smart Phones. Regularly adware can be viewed as a subclass of spyware and it will far-fetched lead to emotional outcomes.
2. **Spyware:** As it suggests from the name, the malware that performs reconnaissance can be alluded to as spyware. Regular activities of spyware incorporate following pursuit you confidential information and venerable information to the outsiders (sniffers) along-with application program interface references.
3. **Trojan or DDOS:** The fundamental inspiration driving this malware type is appearing on the phones and machines is to embark the denial of services and hardware abstract layers. Consistently Trojan malware can be seen as a subclass of spyware and it will implausible This malware class is utilized to characterize the malware types that mean to show up as authentic programming. Along these lines, the general spreading vector used in this class is social designing, i.e. making individuals imagine that they are downloading the real programming (Moffie, et al. 2006).
4. **Rootkits :** Its usefulness empowers the aggressor to get to the information with higher authorizations than is permitted by the respected operating system or framework. For instance, it very well may be utilized to give an unapproved client managerial access. Rootkits dependably shroud its reality and frequently are unnoticeable on the framework, making the discovery and along these lines expulsion unimaginably hard. (Chuvakin 2003).

Ransomware: This kind of malware means to encode every one of the information on the machine and request that an injured individual exchange some cash to get the decoding key. More often than not, a machine contaminated by ransomware is "solidified" as the client can't open any document, and the work area picture is utilized to give data on aggressor's requests. (Savage, Coogan and Lau 2015).

Malware Espionage or Detection Techniques

All malware recognition procedures can be isolated into mark based and demeanor based techniques. Prior to going into these techniques, it is fundamental to comprehend the nuts and bolts of two malware examination approaches: static and dynamic malware investigation. As it suggests from the name, static investigation is performed statically, i.e. without execution of the record. Conversely, dynamic examination is directed on the record while it is being executed for instance in the virtual machine i.e. dalvik virtual machine.

1. **Fingerprinting:** this incorporates cryptographic hash calculation, finding the ecological ancient rarities, for example, hardcoded username, filename, library strings.
2. **AV scanning:** in the event that the examined record is a notable malware, doubtlessly all enemy of infection scanners will have the capacity to identify it. In spite of the fact that it may appear to be immaterial, along these lines of location is regularly utilized by AV sellers or sandboxes to "affirm" their outcomes.
3. **String Extraction:** this alludes to the examination of the product yield (e.g. status or mistake messages) and deriving data about the malware activity.
4. **Disassembly:** this refers to reversing the machine code to assembly language and inferring the software logic and intentions. This is the most common and reliable method of static analysis.
5. **File Format Inspection:** recorded metadata can give valuable data. For instance, file system or category of file or fire pertaining records (POSIX) can give much data on order time, imported and sent out capacities, and so on.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

IV. MACHINE LEARNING

The below context will explore the foundation that is basic to comprehend the malware recognition and the requirement for machine learning strategies to espionage and remit such malwares from android framework. The malware types applicable to the investigation are portrayed first and further trailed by the standard malware location techniques. In view of the information act as ready reference for categorizations and develop the amalgamated technique to be proposed in future. However, the fundamental thought of any machine learning errand is to prepare the model, in light of some calculation, to play out a specific assignment: characterization, clusterisation, relapse or regression, and so on. Preparing is done dependent on the information dataset, and the model that is fabricated is consequently used to make forecasts. The yield of such model relies upon the underlying assignment and the usage. Conceivable applications are: given information about house properties, for example, room number, size, and cost, anticipate the cost of the beforehand obscure house; in light of two datasets with solid therapeutic pictures and the ones with tumor, order a pool of new pictures; group pictures of creatures to a few bunches from an unsorted pool. To develop a deeper understanding, it is worth going through the general workflow of the machine learning process, which is shown in Figure 1.

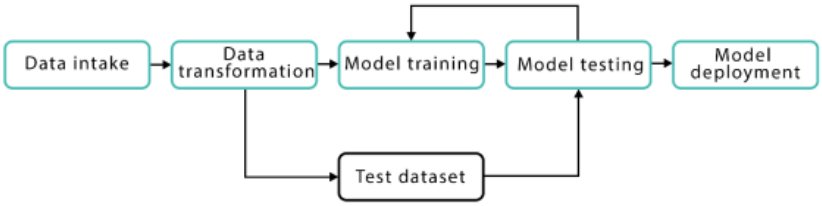

Figure 1. General workflow process

Supervised and Unsupervised Learning: In Supervised Learning, learning depends on named information. For this situation, we have an underlying dataset, where information tests are mapped to the right result. The lodging costs case is a case of managed learning: here we have an underlying dataset with houses, its traits, and its costs. The model is prepared on this dataset, where it "knows" the right outcomes. Instances of managed learning are regression and classification issues:

1. **Regression:** Anticipate the esteem dependent on past perceptions, i.e. estimations of the examples from the preparation set. Normally, we can say that on the off chance that the yield is a genuine number/is persistent, it is a regression issue.
2. **Classification:** In light of the arrangement of marked information, where each name characterizes a class, that the example has a place within context, we need to foresee the class for the existing obscure mechanism. The arrangement of conceivable yields is limited and generally little. By and large, we can say that on the off chance that the yield is a discrete/all out factor, it is an order issue for prediction appropriate results therein.

Rather than Supervised Learning, in Unsupervised Learning, there is no underlying marking of information. Here the objective is to discover some example in the arrangement of unsorted information, rather than foreseeing some esteem. A typical subclass of Unsupervised Learning is Clustering.

3. **Clustering**: Locate the shrouded examples in the unlabeled information and separate it into bunches as indicated by likeness. A model can be the revelation of various clients or contextual bunches inside the client base of the online shop as exemplary model.

Classification Methods: From machine learning point of view, malware discovery can be viewed as an issue of grouping or clusterization: obscure malware types ought to be clusterized into a few bunches, in light of specific properties, distinguished by the calculation. Then again, having prepared a model on the wide dataset of vindictive and kind records, we can diminish this issue to grouping. For known malware families, this issue can be limited to characterization just – having a constrained arrangement of classes, to one of which malware test absolutely has a

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

place, it is simpler to distinguish the best possible class, and the outcome would be more precise than with clusterization calculations. In this area, the hypothetical foundation is given on every one of the strategies utilized in this task by using various classification methods.

K-Nearest Neighbors: (KNN) is one of the least complex, however, precise machine learning calculations. KNN is a non-parametric calculation, implying that it doesn't make any presumptions about the information structure. In true issues, information once in a while complies with the general hypothetical suppositions, making non-parametric calculations a decent answer for such issues. KNN display portrayal is as straightforward as the dataset – there is no learning required, the whole preparing set is put away. KNN can be utilized for both grouping/classification and relapse/regression issues. In the two issues, the expectation depends on the k preparing occurrences that are nearest to the information occasion. In the KNN order issue, the yield would be a class, to which the information occurrence has a place, anticipated by the greater part vote of the k nearest neighbors. In the relapse issue, the yield would be the property estimation, which is commonly a mean estimation of the k closest neighbors. The schematic example is outlined in Figure 2.

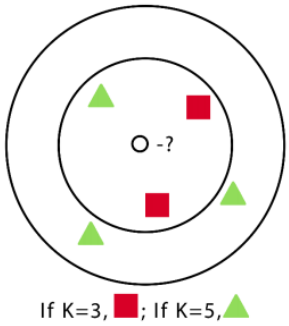

Figure 2. KNN example

Diverse separation estimation techniques are utilized for finding the nearest neighbors. The well known ones incorporate Hamming Distance, Manhattan Distance, Minkowski distance:-

$$Hamming\ Distance:\ d_{ij} = \sum_{k=1}^{r} |x_{ik} - x_{jk}| \quad [1]$$

$$Manhattan\ Distance:\ d_1(p,q) = ||p-q||_1 = \sum_{i=1}^{n} |p_i - q_i| \quad [2]$$

$$Minkowski\ Distance = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p} \quad [3]$$

Support Vector Machines : Support Vector Machines (SVM) is another machine learning calculation that is commonly utilized for characterization issues. The principle thought depends on finding such a hyperplane, that would isolate the classes in the most ideal way. The term ‘support vectors' alludes to the focuses lying nearest to the hyperplane, that would change the hyperplane position whenever expelled. The separation between the help vector and the hyperplane is alluded to as edge. Naturally, we comprehend that the further from the hyperplane our classes lie, the more precise forecasts we can make. That is the reason, albeit different hyperplanes can be found per issue, the objective of the SVM calculation is to discover such a hyperplane, to the point that would result in the most extreme edges.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

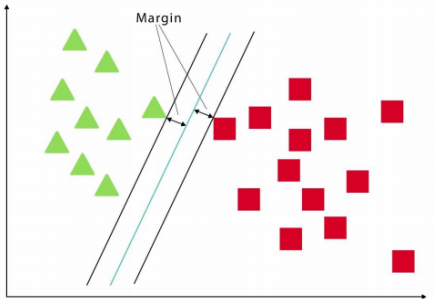

Figure 3. SVM example

Naive Bayes: Naive Bayes] is a simple probabilistic classifier based on Bayes' rule. The naive Bayes algorithm builds a probabilistic model by learning the conditional probabilities of each input attribute given a possible value taken by the output attribute. This model is then used to predict an output value when we are given a set of inputs. This is done by applying Bayes' rule on the conditional probability of seeing a possible output value when the attribute values in the given instance are seen together. Before describing the algorithm we first define the Bayes' rule.

Bayes' rule states that $P(A \mid B) = \dfrac{P(B \mid A)P(A)}{P(B)}$,

where *P(A|B)* is defined as the probability of observing *A* given that *B* occurs. *P(A|B)* is called posterior probability, and *P(B|A)*, *P(A)* and *P(B)* are called prior probabilities. Bayes' theorem gives a relationship between the posterior probability and the prior probability. It allows one to find the probability of observing *A* given *B* when the individual probabilities of *A* and *B* are known, and the probability of observing *B* given *A* is also known.

J48 Decision Tree: As it suggests from the name, decision trees are information structures that have a structure of the tree. The preparation dataset is utilized for the production of the tree, that is along these lines utilized for making forecasts on the test information. In this calculation, the objective is to accomplish the most exact outcome with minimal number of the choices that must be made. Choice trees can be utilized for both characterization and relapse issues. A model can be found in Table 1:

Predictors				Target
Outlook	Temperature	Humidity	Windy	Play tennis
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Overcast	Cool	High	True	No
Rainy	Cool	High	True	Yes
Rainy	Mild	High	False	No
Sunny	Cool	Normal	False	Yes

Table 1. Decision tree example dataset

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

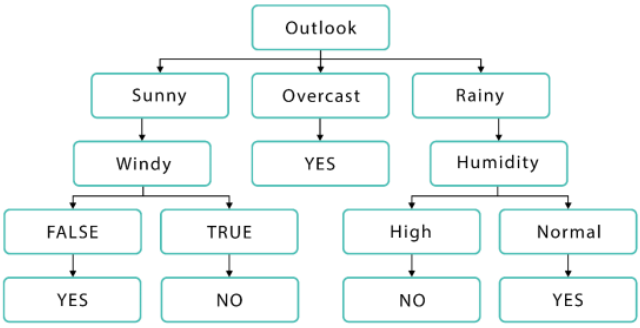

Figure 4. Decision tree example

As it very well may be found in Figure 4, the model was prepared dependent on the dataset and would now be able to group the tennis playing choice to "yes" or "no". Here, the tree comprises of the decision nodes and leaf nodes. Decision nodes have a few branches prompting leaf nodes. Leaf nodes speak to the choices or characterizations. The highest introductory nodes is alluded to as root node. The basic calculation for decision trees is ID3 (Iterative Dichotomiser 3). It depends on the ideas of the Entropy and Information Gain. Entropy here alludes to the dimension of vulnerability in the information content. For instance, the entropy of the coin hurl would be inconclusive, since there is no real way to make certain in the outcome. Oppositely, a coin flip of the coin with two heads on the two sides would result in zero entropy, since we can anticipate the result with 100% likelihood before each hurl. In basic words, the ID3 calculation can be portrayed as pursues: beginning from the root node, at each stage we need to segment the information into homogenous (comparative in their structure) dataset. All the more explicitly, we need to discover the quality that would result in the most elevated data gain, i.e. restore the most homogenous branches. Evaluate the entropy of the objective as below:-

$$E(T,X) = \sum_{c \in X} P(c)E(c) \quad E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

Thereafter divide the dataset and evaluate the entropy of apiece branch. Subsequently compute in sequence about achieved values and the divided values that is with variation in the preliminary entropy and the comparative summation of the entropies of the branches as $Gain(T, X) = Entropy(T) - Entropy(T, X)$ means The attribute with the highest Gain value is selected as the decision node otherwise If one of the branches of the selected decision node has an entropy of 0, it becomes the leaf node. Other branches require further splitting. Consequently, the algorithm is run recursively until there is nothing to split anymore.

IV. CONCLUSION

In general, the aspiration defined for this study is to achieve knowledge and techniques for the understand and dimensions about malwares along with machine learning techniques to develop an effective and more accurate malware detector using machine learning techniques (amalgamation or composition of two or more) itself. From above scenarios the desired feature extraction and representation methods will be selected and the selected machine learning algorithms were applied and evaluated for such solutions

REFERENCES

1. Justin Sahs ; Latifur Khan, A Machine Learning Approach to Android Malware Detection 2012 European Intelligence and Security Informatics Conference, 22-24 Aug. 2012
2. Ivan Firdausi ; Charles lim ; Alva Erwin ; Anto Satriyo Nugroho Analysis of Machine learning Techniques Used in Behavior-Based Malware Detection 2010 Second International Conference on Advances in Computing, Control, and Telecommunication Technologies.
3. Wen-Chieh Wu, Shih-Hao Hung : DroidDolphin: a dynamic Android malware detection framework using big data and machine learning ACM New York, NY, USA ©2014

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 12, December 2018

4. Ying-Chih Shen1, Roger Chien1, and Shih-Hao Hung1, Toward Efficient Dynamic Analysis and Testing for Android Malware 1Academia Sinica 2016, Taiwan, National Taiwan University, Taiwan
5. D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, K. Rieck, and C. Siemens. Drebin: Effective and explainable detection of android malware in your pocket. 2014.
6. S.-J. Chang. Ape: A smart automatic testing environment for android malware. 2013.
7. S. Y. Yerima, S. Sezer and G. McWilliams. "Analysis of Bayesian Classifcation Aprroaches for Android Malware Detection," IET Information Security, Vol 8, Issue 1, January 2014.
8. S. B. Kotsiantis, "Decision trees: a recent overview," Artificial Intelligence Review (2013) 39: pp 261-283, 2013.
9. I. H. Witten, E. Frank and M. A. Hall, Data Mining: Practical Machine Learning Tools and Techniques, 3rd ed., Morgan Kaufmann, Burlington: MA 01803, USA, 2011.
10. S. Y. Yerima, S. Sezer, G. McWilliams and I. Muttik, "A New Android Malware Detection Approach Using Bayesian Classification," Proc. 27th IEEE int. Conf. on Advanced Inf. Networking and Applications (AINA 2013), Barcelona, Spain, March 2013.
11. J. Sahs and L. Khan, "A Machine Learning approach to Android malware detection," 2012 European Intelligence and Security Informatics Conference, 2012.
12. L. Batyuk, M. Herpich, S. A. Camtepe, K. Raddatz, A. Schmidt, S. Albayrak, "Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications," Proc. 6th International Conference on Malicious and Unwanted Software (MALWARE), 2011.
13. H. Peng, C. Gates, B. Sarma, N. Li, A. Qi, R. Potharaju, C. NitaRotaru and I. Molloy, "Using probabilistic generative models for ranking risks of Android apps. In Proceedings of the 19th ACM Conf on Computer and Comms. Security (CCS 2012), Oct. 2012.
14. G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "MADAM: A Multi-level Anomaly Detector for Android Malware," Proc. 6th Int. Conf. on Mathematical Methods, Models and Architectures for Computer Network Security (MMM-ACNS 2012), Saint Petersburg, Russia.
15. A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss. "Andromaly: A behavioral malware detection framework for Android devices," J. Intell. Inf. Syst., 38(1), pp 161–190, 2012.