Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume12, Issue 10, December 2021: 6203-6211

Machine Learning Based Android Malware Detection: A comprehensive Survey

Ravneet Singh Bedi[1] and Prashant Singh Rana[2]

[1]Computer Science and Engineering Department, Mewar University, Gangarar, Chittorgarh, Rajasthan, India.
[2]Computer Science and Engineering Department,
Thapar Institute of Engineering and Technology, Patiala, Punjab, India.
Email: ravneetsinghbedi@gmail.com

Abstract

Android Operating System's open-source nature has attracted a broader range of developers which has resulted in an unprecedented proliferation of Android-based devices in various sectors of the economy. However, despite the fact that this growth has resulted in significant technological advances and the ease of conducting business (e-commerce) and social interactions, they have also become powerful platforms for unregulated cyber-attacks and espionage against business infrastructures and individual users of these mobile devices. Malicious application attacks, as opposed to other attack strategies such as social engineering, have taken the lead when it comes to cyber-attack tactics. Android malware has advanced in complexity and intelligence to the point where it is now highly resistant to existing detection systems, especially signature-based systems. Machine Learning techniques have emerged as a more capable choice for countering the complexity and innovation used by emerging Android malware. Machine Learning models operate by first learning existing patterns of malware behavior and then using that information to isolate or classify any related behavior from unknown sources. As found in recent literature, this paper presents a thorough analysis of Machine Learning techniques and their applications in Android malware detection.

Keywords: Machine Learning, Android, Malware, Static Analysis, Dynamic Analysis, An- droid Malware Detection

1 Introduction

Android malware detection[1] can be performed in three different ways, according to various re- searches. The first approach entails deploying static and dynamic investigation of application code in order to detect malicious components before loading the software onto any system. The second method entails modifying the Android system to include modules for monitoring and intercepting abnormal behaviors that may occur on the device, while the third method entails using virtual- ization to implement domain separation ranging from lightweight application isolation to running several instances of Android OS on the same device [2][3].

Machine Learning or "anomaly detection" approaches have now emerged as a leading and more successful method for overcoming Android malware, according to various reports [4][5]. Unlike static analysis, which entails manually inspecting files such as the AndroidManifest.xml file, source files, and Dalvik byte code; dynamic analysis entails running an app in a controlled environment to observe its behavior while the Machine Learning approach entails learning general rules and patterns from benign and malicious app samples and then allowing data to be collected and clas- sified [6][7]. The majority of Machine Learning algorithms depend on static attributes extracted from an application[8]. The static components of an Android application serve as the foundation for Machine Learning approaches and these static features are

meticulously obtained by reverse engineering.

Machine Learning methods have been commonly used for application classification with an em- phasis on generic malware detection. The use of Machine Learning in Android malware detection reduces the time and effort required to manually create and update detection patterns. Machine Learning, as shown in Figure 1, is a procedure for analyzing data using software techniques i.e algorithms to construct a model which is useful for identifying patterns and regularities in data sets [9]. It is a method of teaching machines to learn from previous experiences i.e data or in- formation in order to make decisions about future events or data instances. Feature vectors are critical components of Machine Learning and they are typically created for the particular task that the Machine Algorithm is attempting to complete. Machine Learning is based on the concept of obtaining the probability distribution of data.

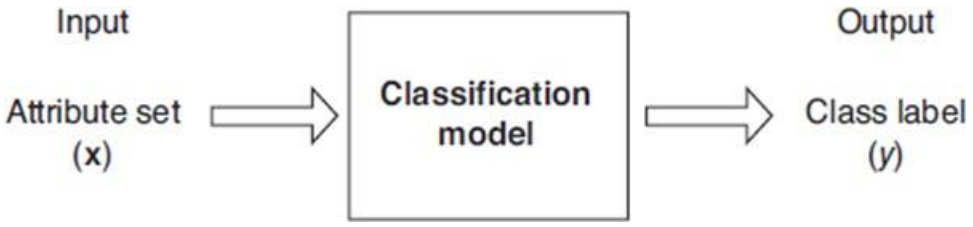

Fig. 1: Classification process in Machine Learning [10]

Supervised Machine Learning, Unsupervised Machine Learning, and Reinforcement Machine Learning are the three major types of machine learning [11]. In addition, each Learning Cate- gory is correlated with three simple Learning Methods: classifications, clustering, and regression. Classification is a Supervised Learning process in which data sets are well labeled into groups or classes. Clustering is an Unsupervised Learning process for unlabeled data sets while Regression is best associated with Reinforcement Learning in which the predicted end result is rated, graded, or estimated. The name of the specific class or category to which a particular data instance belongs is called a label. Data is characterized in Machine Learning by a fixed number of features that can be categorical, nominal, or continuous [12]. This paper provides a comprehensive analysis of the current literature in the field of Machine Learning techniques being used to detect Android malware.

2 Attack Trends: Android Malware

Malware coders use a variety of methods to avoid detection. Various examples of such concealment techniques are code obfuscation, encryption, asking for inappropriate and unwarranted permissions to access unauthorized hardware, and lastly update attacks in which a benign application updates itself or another application with malicious payload [13]. Malware attack methods can be classified as follows:

1. **Information Extraction**: Malware in this category infects a computer and then extracts personal information such as the device's IMEI number, the user's personal information, andso on.
2. **Root Exploits**: This type of malware tries to gain root privileges on the device in order to take control of it and change its configuration and other data.
3. **Automatic Calls and SMS**: This form of malware inflates a user's phone bill by making automated calls and sending SMS to premium numbers.
4. **Dynamically Downloaded Code**: This technique allows a benign program to download malicious code and deploy it in mobile devices without the user's knowledge or consent.
5. **Search Engine Optimisations (SEO)**: The malware in this case artificially searches for a word and simulates clicks on targeted websites in order to boost a search engine's revenue or increase website traffic.

6. **Covert and Overt Communication Channels**: This is a system flaw that allows informa- tion to leak between processes that aren't supposed to share it. This is regarded as a highly advanced technique.
7. **Botnets**: A botnet is a network of infected mobile devices managed by command and con- trol (C&C) servers via a Bot-Master. It sends spam and targets the host devices with DDoS (Distributed Denial of Service) attacks.

3 Literature Review

DroidFusion [14] is one of the most proposed and sought-after novel classifier fusion methods based on a multilevel architecture that enables efficient Machine Learning algorithms for improved accuracy. DroidFusion works by first training the base classifiers at a lower level to construct a model and then applying a series of ranking-based algorithms to their predictive accuracies at a higher level to produce combination schemes, one of which is chosen to develop a final classification model. The training set used in training the base classifiers at the lower level is subjected to a stratified 10-fold cross-validation technique in order to estimate their relative prediction accuracies. Various authors have used J48, REPTree, Random Tree-100, Random Tree-9, and Voted perceptron as their base classifiers. On all of the data sets given, DroidFusion outperformed all of the base classifiers as well as Stacked Generalisation.

Another research [15] suggested a system for detecting Android malicious applications based on Support Vector Machine (SVM) and Active Learning technologies. The authors used dynamic data feature extraction and timestamps as attachments to some of the features in order to use novel time-dependent behavior monitoring in order to significantly improve malware detection accuracy. The authors used an intended error reduction query strategy to combine new insightful instances of Android malware and retrain the model to be able to do adaptive online learning in order to create an active learning model. The authors used the DREBIN benchmark malware dataset in a series of experiments to test their model and the results showed that their system could detect new malware more accurately.

Idrees et al. (2017) proposed PIndroid which is a novel Android malware app detection system that uses permissions and intent features for model training and classifier fusion in order to combine classifiers for better results. To conduct the experiments, the authors used 1745 app samples, starting with a performance comparison of six classifiers: Multi-Lateral Perceptron (MLP), Decision Table, Decision Tree, Random Forest, Nave Bayesian, and Support Vector Machine (SVM) using Sequential Minimal Optimization (SMO). They used three fusion schemes: Average of Probabilities, Product of Probabilities, and Majority Voting to merge the Decision Table, MLP and Decision Tree classifiers. Through the use of the Product of Probability Combination approach, the system provided a 99.8% True Positive detection accuracy rate, 1.1 percent False Positive detection rate,and a 99.7% F-measure.

Dong (2017) developed a novel detection method for Android malware using permissions as his primary function. A web crawler was used to gather samples of both benign and malicious Apps. To complete the reverse engineering process, the author created a tool that automatically decompiles the Apps to source code and manifest files. One of the results was that the distribution of App permissions differed between the malware and benign datasets. To find patterns and more useful knowledge, the author combined Machine Learning algorithms such as Logistic Regression Model, Tree Model with Ensemble techniques, Neural Network, and finally an ensemble model.

The authors have also suggested two methods for static analysis of Android malware that are focused on Machine Learning [16]. The first method relied on Android authorization features such as manifest analysis, while the second relied on a bag-of-words representation model to analyze Android source code. The aim of the project was to show how effective Machine Learning methods are for static analysis of Android malware. The Authors used two Machine Learning approaches classification and clustering. The classification method was primarily used for malware detection

while the clustering method was only used to infer the class of unlabeled data when the data set had few labels and the labels obtained from clustering were then used to retrain the classification model with more data. The authors used a fusion approach called Majority voting to run the experiment with ensembles that contained unusual combinations of three and five classifiers. Permission-based clustering, permission-based classification, source code-based clustering, and source code-based classification were the four experiments conducted in the study. For the training and testing of the models, 200 malicious and 200 benign applications were used.

Support Vector Machine (SVM), Decision Tree, C4.5, JRIP (Rulebased), Random Forest, Linear regression, and Random Tree are some of the classification algorithms which were used in the model building while Farthest First, Simple K-means, and Expectation-Maximization (EM) were the clustering algorithms used during the model building. The permission-based model was found to be computationally less costly than the source code analysis model but the source code analysis model had an FMeasure of 95.1% compared to 89 percent for the permission-based model.

The authors have also suggested an anomaly-based malware detection system for the Android platform [17]. They used a behavioral analysis technique to study the behavioral habits of both malicious and benevolent applications by running them on a physical Android computer. Decision Tree, K Nearest Neighbor, Logistic Regression, Multilayer Perceptron Neural Network, Nave Bayes, Random Forest, and Support Vector Machine are some of the Machine Learning algorithms used to classify malware. An output metric was used to evaluate each algorithm. Support Vector Machine and Random Forest got the best results for malware detection according to their findings.

In another research, the authors proposed a composite classification model for Android mal- ware detection based on a parallel combination of heterogeneous classifiers [18]. The algorithms were trained using static features extracted from 6,863 app samples and it used Decision Tree (tree-based), Naive Bayes (probabilistic), Simple Logistics (function-based), PART (rule-based), and RIDOR as classifiers. The classification algorithms were used to compare four classifier hybrid approaches: Average of Probability, Maximum Probability, Product of Probability, and Majority Vote. The composite model was created with the aim of enabling a more accurate early detec- tion model for Android malware as well as a faster white-box analysis through the use of more interpretable constituent classifiers. The approach's goal was to combine the advantages of various supervised learning algorithms to generate a single classification judgment for a new application. Static features i.e permissions, APIs, and Android system commands were collected from the An- droid application using reverse engineering which used a custom APK analysis tool written in Java and was used to detect malware using Machine Learning. Their findings revealed that PART worked best for individual classifiers: products of probabilities mixture schemes had the best ac- curacy while TPR had the best TPR (True Positive Rate).

DynaLog, a dynamic Android malware analysis framework that characterizes Android apps was proposed in another work [19] as a way to combat malware that uses obfuscation concealment methodologies to evade most state-of-the-art static-based detection and analysis systems. It is a behavioral-based analysis system that mostly relies on the number of comprehensive dynamic features in a given application. It provides an automated environment that can evaluate and characterize a large number of applications, quickly detecting and isolating malicious ones. The framework is capable of accepting a large number of applications, serially starting them in an emulator, logging the various dynamic behaviors, and then removing them for further processing. DynaLog is based on existing open-source software, providing it with a broad range of capabilitiesfor the dynamic analysis of Android apps.

The authors in another work published a study on the use of a Machine Learning algorithm to detect Android malware [20]. The aim of the study was to compare various static features of applications in order to find the best static feature combination that would result in the most efficient Machine Learning Detection Model. The study's main goal was to evaluate the effectiveness of some feature sets familiar from desktop systems (x86 domain) but never before used in the Android ecosystem in aiding Android malware detection through Machine Learning. The entropy

of files in the resource folder, Graphical User Interface (GUI) Layout, Number of methods, and the number of instructions per process are among the new proposed features. To effectively classify the different features, the authors used WEKA Data Mining Software. They randomly divided their samples into training and test sets in the proportion of 80 to 20 for the device evaluation. There are no samples from the malware family used in the training set in the test set. The objective of the comparative analysis was to assess the features' individual and collective ability. Their findings showed that, on an individual basis, Android permissions are the strongest single indicator of an app's malignancy, with an accuracy of around 96 percent. Opcode frequency, opcode sequences, and app components are among others with an accuracy of about 90%. Random Forest was found to be the most effective classifier. The accuracy of features extracted from the Android Manifest which contains permissions, features, intents, and device components was over 97 percent for the attributes or function combination test. As a result, rather than the app's actual code, the best- performing attribute can be extracted from the app's metadata contained in the Android Manifest.

The authors of another work concentrated on identifying and defeating a form of unknown malware known as "0-day" malware [21]. They created the SherlockDroid system. The Sherlock- Droid works by filtering a large number of applications and keeping only the ones that are most likely to be malicious for potential inspections. Apart from crawling marketplaces for apps, Sher- lockDroid extracts code-level features and then uses Alligator to classify unknown apps. Alligator is a classification method that integrates many classification algorithms effectively and automat- ically. SherlockDroid looked at characteristics of Android apps gleaned from a static code study. A lightweight technique for Android malware called DREBIN was also proposed, which allows for direct malicious application detection on the Smartphone. DREBIN overcomes the disadvantages of a shortage of resources which prevents applications from being monitored in real-time. It works by conducting a thorough static review of an application and collecting as many features as possi- ble like permissions, API calls, hardware resources, app components, filtered intents, and network addresses. These features were embedded in joint vector space so that common malware patterns could be automatically identified. The malware classification was done using a Machine Learning technique called Support Vector Machine. Using Sandboxing Systems, the work provided a detailed description of the various research techniques, including Static Analysis, Behavioural Analysis, and Dynamic Analysis. Strings are an important part of any static malware study, according to the researchers, as they can include clues about malware construction, functionality, authorship, and Command & Control (C & C). The research discovered that when an app is unpacked, the most relevant strings are contained in classes.dex, the source code of the app.

The Android operating system is an open platform-based Operating System with many new mobile device hardware manufacturers embracing it as their primary operating system. This has resulted in an ever-increasing number of new applications being created for the platform. Most mobile applications present in today's world lack a central control system for integrity check which can certify whether they are fit and safe for the general public. Hence, many applications which were created with good intentions end up behaving maliciously as a result of bad design by novice developers who are either not careful about removing all bugs in the apps' code or do not follow proper security protocols throughout the development phase. A large number of poorly created apps along with those created with malicious intent pose a serious security hazard for the Android platform. All these security threats aimed at the Android platform will continue to serve as the foundation for a slew of new studies focusing on the identification, review, and remediation of evasive Android malware. Machine Learning, which is a subset of Artificial Intelligence is gaining traction as the best way to fight zero-day malware attacks. As a result, various innovative methods for using Machine Learning methodologies in malware detection will continue to be investigatedand implemented.

Table 1: Frameworks Comparison of Machine Learning-Based Malware Detection

Ref.	Main Approach	Algorithm	Accuracy	False Positive Rate	Detection Rate
[6].	mAIS produces two independent detector sets for "non-self" and "self" detections. The "non-self" detector sets will detect the non-self app instances and the other detects "self" app instances. The outputs from both detector sets are used in the classifying and detecting Android malware [6].	XOR.	88.33%	0.00%	*Not Mentioned*
		GEFeS and SDM.	93.33%	0.00%	*Not Mentioned*
[7].	The proposed system contains two main parts, client and server. In the client side, the user interface will alert the user in case if the application MD5 value matches one of the stored malicious applications MD5 values. Otherwise, the calculated MD5 of the installed application will be submitted to the server [7].	SVM and PCA-RELIEF algorithms.	95%	13.3%	*Not Mentioned*
[8].	It is a combination of Application Permissions Gatherer, Permissions Analyzer, and Keyloggers Detector components. The first component will gather the permissions of the applications using API and it will be stored in SQLite database. The second component will use SVM algorithm to analyse different permissions and recognize its patterns. The last component is the keylogger detector, which will detect the keyloggers applications [8].	SVM	*Not Mentioned*	*Not Mentioned.*	*Not Mentioned*
[9].	It uses *Androguard* project to extract features APKs. The used dataset contains 91 malicious and 2081 benign applications; however, one class *SVM* classifier has been trained on benign applications only using Scikit-learn framework [9].	SVM.	*Not Mentioned.*	*Not Mentioned.*	*Not Mentioned.*
[10]	Using real malware and benign application samples. The developed detector is a parallel combination of different heterogeneous classifiers which are *Decision Tree, Simple Logistic, Naïve Bayes, PART, and RIDOR* algorithms. Combining different machine learning algorithms demonstrates algorithms' efficacy and improve the detection accuracy [10].	DT	95.4%	4%	96.4%
		NB	86.7%	8.7%	91.5%
		SL	93.2%	4.6%	97.7%
		RIDOR	95%	5.8%	94.9%
		PART	96.3%	3.3%	97%
		AvgProb	96.3%	3.1%	98.8%
		ProdProb	97.2%	3%	95.3%
		MaxProb	95.2%	7.2%	98.6%
		Mvote	96.3%	3.1%	96.3%
[11].	It detects new and unpredictable malicious applications with three different algorithms which are: *RIPPER, Naive Bayes, and a Multi Naive Bayes Classifier.* The framework has been trained on the collected dataset to find patterns and detect new and unseen malicious applications [11].	RIPPER	89.36%	7.77%	71.05%
		NB	97.11%	3.80%	97.43%
		Multi-NB	96.88%	6.01%	97.76%
[12].	It is designed to operate in the cloud environment to support multiple devices simultaneously. The proposed platform contains Security Web Server, Analysis Module, Android application, and Google Cloud Messaging (GCM) [12].	ZeroR	49.7 %,	0.00%	45%
		OneR	100%	0.00%	83%
		NB	*Not Mentioned*	20%	94.59%
		J48	100%	0.00%	90%
[13].	It classifies malicious and benign applications. The framework is based on multi-stage combination "*cascade*" of perceptron algorithm [13]. *"One-sided perceptrons", and a "kernelized one-sided perceptrons"* have been used in the framework. The *"one-sided perceptrons"* performs the training on only one label either malicious or benign application, while *"kernelized one-sided perceptrons"* it trains the system based on the *Polynomial Kernel Function* [13].	"One-sided perceptron (OSP)"	74.6%	0.83%	68.73%
		OSP F1	85.54%	0.55%	83.76%
		OSP F2	85.17%	0.55%	83.22%
		kernelized OSP - Polynomial function"	88.84%	3.9%	89.96%
		kernelized OSP - Kernal function"	57.08%	3.6%	50.97%
[14].	TSDNN is end-to-end trainable which trains all networks' nodes simultaneously and it classifies the data in layer-wise manner to learn better from minor classess. it outperforms the multi-layer perceptron framework by overcomes the difference between imbalance classes, and adjusting the sensitivity toward each class using QDBP which is based on backpropagation which is a "gradient-based method to train neural networks" [14].	TSDNN and QDBP.	99.63%	*Not Mentioned*	85.4%

4 Comparison of different Frameworks for Malware Detection usingMachine Learning

This section compares the malware detection frameworks proposed in the paper under study. Table 1 compares different techniques available for detecting Android malware based on machine learning techniques algorithms. Different machine learning techniques have been used in the development of malicious software detection systems such as SVM, NB, Perceptron, and DNN algorithms. In Table 1, if the study authors did not mention accuracy, false-positive rate, detection rate, or algorithm used in a specific setting, "Not mentioned" will be used.

5 Discussion and Findings

As shown in Table 1, various machine learning algorithms were used to develop malware detection frameworks [10]. The parallel combination of various heterogeneous classifications such as DT, SL, NB, PART, and RIDOR algorithms worked well. The four different combination schemes AvgProb, ProdProb, MaxProb, and Mvote obtained accuracy 96.3%, 97.2%, 95.2%, and 96.3%, false-positive
rate with 3.1%, 3%, 7.2%, and 3.1%, and detection rate with 98.8%, 95.3%, 98.6%, and 96.3% respectively [10]. “Kernelized One-Side Perceptrons - Polynomial Function" obtained a precision of 88.84%, a detection rate of 89.96%, and a reasonable false positive rate of 3.9%. The highest accuracy was 100%, with a false positive rate of 0.00% from OneR and J48. However, the figures are 83% and 90% respectively[12]. Furthermore, the use of "Quantity Dependent Backpropagation (QDBP)" and an "End-to-end trainable Tree-Shaped Deep Neural Network (TSDNN)" gave good precision and detection results at 99.63% and 85.4% respectively.

Since parallelizing machine learning algorithms showed excellent results in [10], parallelizing schemas with OneR or J48 [12] or parallelizing TSDNN along with QDBP [14] may increase the accuracy and the detection rate of detecting malware on Android devices. In addition, the integration of "Kernelized One-Side Perceptrons -Polynomial Function" with OneR or J48 [13] may increase the precision and detection rate of Android malware detection. The combination of one or more techniques calls for more detailed experiments that focus on the performance of algorithms such as robustness and security.

6 Conclusion and Future work

Smartphone users have expanded rapidly over the years, mostly Android users. This notable growth in Android users has been used by malware authors to target and harm a large number of users in the meantime. The primary concern of the growing Android platform is detecting malware. This document examined existing Machine Learning like "mobile security platform", "versatile machine learning-based framework" and TSDNN as well as QDBP systems and frameworks for malware detection respectively. Most of the articles studied are used SVM and NB algorithms for detecting malicious software. However, the greatest accuracy was 100%, with a false positive rate of 0.00% obtained with the OneR and J48 algorithms. Nonetheless, the detection rate is 83% and 90% respectively. In a critical field like malware detection, we need a good machine algorithm for detecting malware with high precision and high detection rate. In the surveyed papers, TSDNN along with QDBP, SVM, Multi-NB, and parallelizing combination of different heterogeneous using AvgProb scheme obtained accuracy and detection rate higher than 90%. However, incorporating one or two good techniques will improve their efficiency, accuracy, and detection rate.

In future work, there will be a growing body of work dealing with detecting Android malicious software using machine learning techniques. It would be nice to offer new frameworks and algo- rithms that are light, fast, and strong enough to detect malware. Based on the documents reviewed, the parallel or integration of one or more good techniques may result in a precise and effective

mechanism. The use of deep neuron networks in TSDNN with QDBP showed high accuracy with 99.63% and a detection rate of 83%. Parallelization of TSDNN and QDBP will demonstrate ef- fective algorithms and improve detection accuracy. Moreover, the parallelization of a combination of OneR and J48 algorithms, using the AvgProb schema should increase the detection rate and prove the efficiency of the algorithms. Use of NB in showed 86.7% accuracy and 91.5% detection, while the use of Multi NB in yielded good results with 96.88% accuracy and 97.76% detection. For example, test Yerima, SY, and al. experience in, and replacing NB with Multi NB will increase the precision and percentages of detection rates in all combination regimes.

References

1. Android Malware Detection through Machine Learning Techniques: A Review (2020).
2. Andrus, J., Dall, C., Hof, A. V., Laadan, O., and Nieh, J. (2011). Cells. Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles - SOSP '11, 173.
3. Lange, M., Liebergeld, S., Lackorzynski, A., Warg, A., and Peter, M. (2011). L4Android: A Generic Operating System Framework for Secure Smartphones. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, 39–50.
4. Rescuers, V. (2018). How Cybercriminals became 'The New Mafia.' Retrieved January 31, 2018, from http://www.virusrescuers.com/how-cybercriminals-became-the-new-mafia/
5. Ucci, D., Aniello, L., and Baldoni, R. (2018). Survey on the Usage of Machine Learning Tech- niques for Malware Analysis. Computers and Security, 1(1), 1–67.
6. Demontis, A., Melis, M., Biggio, B., Maiorca, D., Arp, D., Rieck, K., Roli, F. (2017). Yes, Machine Learning Can Be More Secure! A Case Study on Android Malware Detection, 1– 14.
7. Arp, D., Spreitzenbarth, M., Malte, H., Gascon, H., and Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In Symposium on Network and Distributed System Security (NDSS) (pp. 23–26).
8. Hahn, S., Protsenko, M., and Müller, T. (2016). Comparative evaluation of Machine Learning- based malware detection on Android. In Sicherheit 2016: 5.-7. April 2016, Bonn (pp. 79–88).
9. Russell, I., and Markov, Z. (2017). An Introduction to the Weka Data Mining System. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education - SIGCSE '17 (pp. 742–742).
10. Tan, P.-N., Steinbach, M., and Kumar, V. (2006). Classification: Basic Concepts , Decision Trees , and Model Evaluation Classification. Introduction to Data Mining, 1, 145–205.
11. Scott, G. (2015). ML 101: Reinforcement Learning. Retrieved November 23, 2017.
12. Guyon, I., and Elisseeff, A. (2006). Feature Extraction, Foundations and Applications: An introduction to feature extraction. Studies in Fuzziness and Soft Computing, 207, 1–25.
13. Baskaran, B., and Ralescu, A. (2016). A Study of Android Malware Detection Techniques and Machine Learning. In Proceedings of the 27th Modern Artificial Intelligence and Cognitive Science Conference 2016, Dayton, OH, USA, April 22-23, 2016. (pp. 15–23).
14. Yerima, Suleiman Y., Sezer, S. and Muttik, I. (2016). Android Malware Detection Using Parallel Machine Learning Classifiers. 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, (Ngmast), 37–42.
15. Rashidi, B., Fung, C., and Bertino, E. (2018). Android malicious application detection using support vector machine and active learning. In 2017 13th International Conference on Network and Service Management, CNSM 2017 (Vol. 2018-Janua).
16. Milosevic, N., Dehghantanha, A., and Choo, K. K. R. (2017). Machine Learning aided Android malware classification. Computers and Electrical Engineering, 61, 266–274.
17. Ali, M. Al, Svetinovic, D., Aung, Z., and Lukman, S. (2017). Malware Detection in Android Mobile Platform using Machine Learning Algorithms. In International Conference on Infocom Technologies and Unmanned Systems (ICTUS'2017) (pp. 4–9). ADET: IEEE.

18. Yerima, Suleiman Y., and Sezer, S. (2018). DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. In IEEE Transactions on Cybernetics (pp. 1–14). IEEE.
19. Alzaylaee, M. K., Yerima, S. Y., and Sezer, S. (2016). DynaLog: An automated dynamic analysis framework for characterizing android applications. In 2016 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2016 (pp. 1–8).
20. Hahn, S., Protsenko, M., and Müller, T. (2016). Comparative evaluation of Machine Learning- based malware detection on Android. 5.-7. April 2016, Bonn (pp. 79–88).
21. Apvrille, L., and Apvrille, A. (2015). Identifying unknown android malware with feature ex- tractions and classification techniques. Proceedings - 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015, 1, 182–189.