Jordanian Journal of Computers and Information Technology (JJCIT), Vol. 05, No. 03, December 2019.

BENCHMARKING MACHINE LEARNING ALGORITHMS FOR ANDROID MALWARE DETECTION

Somayyeh Fallah[1] and Amir Jalaly Bidgoly[2]

(Received: 3-Jun.-2019, Revised: 31-Jul.,-2019 and 12-Aug.-2019, Accepted: 4-Sep.-2019)

ABSTRACT

Nowadays, smartphones have captured a significant part of human life and has led to an increasing number of users involved with this technology. The rising number of users has encouraged hackers to generate malicious applications. Identifying these malwares is critical for preserving the security and privacy of users. The recent trend of cyber security shows that threats can be effectively identified using network-based detection techniques and machine learning methods. In this paper, several well-known methods of machine learning were investigated for smartphone malware detection using network traffic. A wide range of malware families are used in the investigations, including Adware, Ransomware, Scareware and SMS Malware. Also, the most used and famous supervised and unsupervised machine learning methods are considered. This article benchmarked the methods from different points of view, such as the required features count, the recorded traffic volume, the ability of malware family identification and the ability of a new malware family detection. The results showed that using these methods with appropriate features and traffic volume would achieve the F1-measure of malware detection by a percentage of about 90%. However, these methods did not show acceptable results in detecting malicious as well as new families of malware. The paper also explained some of the challenges and potential research problems in this context which can be used by researchers interested in this field.

KEYWORDS

Android malware, Malware detection, Network traffic, Machine learning.

1. INTRODUCTION

With the enormous growth of smartphones around the world, the number of malware attacking mobile applications has also witnessed an exponential increase. Due to their wide variety and open source platform, these phones have become an attractive domain for hackers to penetrate. According to a report by F-Secure Corporation in 2017 [1], more than 99 percent of all malware designed for smartphones target Android devices. There are over 19 million malware programs developed particularly for Android, making Google's mobile operating system the main target for mobile malware. The reason for this is the vast distribution of Android devices, as well as the relatively open system for the distribution of apps. Many malware programs use the Internet in order to communicate with the initiator of the attack in order to receive new tasks and software updates or to leak collected data. Yet, when such malware tries to communicate with its Command and Control (C&C) server, it most likely uses a common and known network protocol to pass through firewalls [2]. By studying, capturing and analyzing the flow of information between two hosts, network administrators are able to provide a basic behavioral pattern. When they are familiar with network behavior, they can catch anomalies, such as significant increases in bandwidth usage, distribution of DDOS attacks and other unauthorized occurrences. By analyzing network traffic and identifying suspicious domains, network administrators can detect malware infections months before the actual malware would be discovered. This could be indicative of the fact that malicious attackers need to communicate with their command and control unit of computers, creating network traffic that can be identified and analyzed. Having a previous warning of developing malware infections can enable faster responses and reduce the impact of attacks. Network traffic classification is the first step in analyzing and identifying different types of programs running on a network. Through this method, Internet service providers or network operators can manage the overall network performance. There are three main approaches to traffic classification: 1) port-based approaches, 2) payload-based approaches and 3) machine learning approaches [3]-[4]. The most common and promising approach in the field of traffic classification is the use of machine learning methods. These methods, which are also able to overcome the constraints of both port-based and

S. Fallah and A. Jalaly are with Department of Computer Engineering, University of Qom, Iran. Emails: [1]`s.fallah@stu.qom.ac.ir` and [2]`jalaly@qom.ac.ir`

payload-based methods, assumed that applications send data with a regular pattern. These patterns can be used as a means to identify traffic categories. To find these patterns, flow statistics (such as the average packet size, flow lengths and total number of packets) and just the use of the TCP protocol can be effective in the classification process [5].

Machine learning is a data analysis method that automatically performs analytical modeling. This method is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention. Factors, such as increasing growth in data types, powerful and inexpensive computing process and the storage of cost-effective data have led to rapid and automatic development of models that are capable of analyzing large and complex data, delivering faster and more accurate results and causing a wave of interest and popularity.

In this paper, with the aim of identifying Android malware, we examine the impact of machine learning methods on more accurate detection of the presence of malware in smartphones. For this reason, we present a comprehensive evaluation of all aspects that are effective in achieving an acceptable result. In this evaluation, it is attempted to take a deep look into various aspects of machine learning methods in order to present their strengths and weaknesses. These evaluations are important in providing researchers with useful and comprehensive information for conducting research work in this field. It matters to other researchers in large scale as well. These evaluations include:

1.1 Benchmarking in Terms of the Number of Extracted Features and the Selection of High-grade Features

The purpose of this evaluation is to provide a subset of features that, in addition to feature reduction and optimality of the problem, can identify normal samples from malicious samples with a very high performance.

1.2 Benchmarking in Terms of Time Duration of Capturing Traffic

Considering the fact that the shorter time duration of detecting malware samples, the lesser impact of infections. This evaluation comparatively analyzes the ability of machine learning methods to detect malware in a shorter time duration from recorded traffic. A method that can detect infection more accurately in the shortest possible time can actually be considered more practical.

1.3 Benchmarking in Terms of the Ability to Identify Malware Families

It is very important that when a malicious program infects the network and the first signals of its presence are observed, the type of malware family should be identified. This makes it possible to prevent any damage by studying the behavior patterns of each family for preventive measures.

1.4 Benchmarking in Terms of the Ability to Identify Unknown Malware Types

In this evaluation, the performance of machine learning methods is examined in the detection of unknown malware types about which no training sample has been given to the system.

In evaluations, all of the features studied in recent research works have been used. This set of selective features provided acceptable results in previous studies. Hence, after the creation of dataset, a model is designed which can provide the best prediction for identifying malicious traffic patterns from normal traffic patterns. The results of algorithms act as a criterion to measure the performance of machine learning algorithms.

The paper continues as follows: Section 2 reviews the related works on malware detection using network traffic. Section 3 describes the general explanation of how to perform the benchmarks. The benchmarks are discussed in section 4. Section 5 discusses the limitations and threats to the validity of the results and finally, the paper ends in Section 6 with the research conclusions.

2. Literature Review

The malware detection method in Android platform is in two broad categories: static analysis and dynamic analysis [6]. In static analysis, no application is executed; only the code and other components, like manifest files, are analyzed. Therefore, it is a quick and inexpensive approach, whereas in dynamic analysis, the applications are executed on actual or virtual environments and normal programs are distinguished from malicious programs using event logs, memory, processor and network usage and analysis. The use of dynamic methods to examine network behavior and its evaluation by machine learning methods has become a popular approach to identify and categorize the malware family. A lot of related work has been focused on this topic, which was discussed in [7]-[9]. In a work by Arora in 2014 [10], the network traffic features are analyzed and a rule-based classification is created to detect Android malware. Then, it provides a list of features that can distinguish between malware traffic and normal traffic patterns and a rule-based classification is built on the acquired features. The results of this study indicated that this approach is significantly correct and identifies more than 90% of the traffic samples. In another work [11], an unsupervised machine learning approach is used for Internet traffic identification and the results are compared with a supervised machine learning approach. The unsupervised approach uses a clustering algorithm and the supervised approach uses the Naive Bayes classifier. Finally, it is concluded that the unsupervised clustering technique has an accuracy up to 91% and performed up to 9% better than the supervised technique. The authors of the previous paper, in another study [12], evaluated network traffic with three unsupervised algorithms: K-Means, DBSCAN and Auto Class. The experimental results showed that the Auto Class algorithm produces the best overall accuracy and with a very small difference, the accuracy of the K-Means algorithm is less than that of the Auto Class algorithm, but, similar to the DBSCAN algorithm, it has a high speed in designing the model. In another work [14], to overcome the drawbacks of existing methods for traffic classification, usage of C5.0 Machine Learning Algorithm (MLA) was proposed. Based on traffic statistics, an advanced classifier was constructed which was able to distinguish between 7 different applications in the test set of 76,632–1,622,710 unknown cases with an average accuracy of 99.3%–99.9%. In a paper by Alhawi [15], a machine learning evaluation study for consistent detection of windows ransomware network traffic was introduced. Using a dataset created from conversation-based network traffic features, a True Positive Rate (TPR) of 97.1% was achieved by the Decision Tree (J48) classifier. The training set included 75618 samples and a test set consisting of 45526 samples. In this experiment, six classifiers of the *Bayesian* network, random forest, KNN, J48, multilayer perceptron and logistic model tree (LMT) were used.

In an article written by Pendlebury et al. [16], it is shown that the results of malware classification can be affected for two reasons: spatial bias caused by distributions of training and testing data that are not representative of a real-world deployment and temporal bias caused by incorrect time splits of training and testing sets, leading to impossible configurations. Hence, a set of space and time constraints for an experiment design that eliminates both sources of bias was proposed. A new metric that summarizes the expected robustness of a classifier in a real-world setting was introduced and provided an algorithm for its performance. An open source evaluation framework called TESSERACT was used to compare the three malware classifiers (decision tree, SVM and deep learning), finally shown that TESSERACT is fundamental to accurate evaluation and comparison of different solutions, especially when considering mitigation strategies for time decay. The dataset used consisted of 129K applications. In another work [17], the network traffic was used as a dynamic feature Android malware detection. A set including 16 features which can distinguish between normal and malicious traffic was determined. Decision tree classifier was built on top of these distinguishing features only and a set of 217 samples was given as input to the classifier. The results of this study indicated that this approach identifies more than 90% of the traffic samples. In a work by Chen and his colleagues [18], statistical features of mobile traffic are utilized to identify malicious traffic flows. After analyzing traffic flow statistics, the data imbalance issue is detected that significantly affects the performance of Android malicious flow detection. Based on six network flow features extracted from the flow data set, several classification algorithms; namely, Bayes Net, SVM, C4.5, Grading, Ada boost and Naïve Bayes, were implemented and experiments were performed on various imbalanced datasets with different imbalance ratio (IR) values. The results show that most of the commonly-used classifiers achieve reasonable performance, which confirms that machine learning algorithms are effective in identifying malicious mobile traffic. Then, the performance of the IDGC model is examined in addressing the data imbalance issue. After testing the IDGC model on the same traffic flow dataset, it was shown that the IDGC is significantly more stable than other classifiers. By increasing the IR value, the performance of the IDGC classification is maintained for the AUC and GM range between 0.8 and 1.0. But, the IDGC classification process is very time-consuming, which makes real-time detection impractical. To improve the performance of the IDGC model, the authors proposed a novel S-IDGC model, which optimizes the weight coefficients using an efficient simplex method. The evaluation results showed that S-IDGC inherits the stability characteristic of the IDGC model while drastically reducing time consumption (with approximately 17 times improvement compared with IDGC on time efficiency). Another study [19] demonstrated a behavioral detection method for detecting mobile malware that can communicate with blacklisted domains and pass sensitive personal / financial information. First, an App-URL table is created that logs all attempts made by all applications to communicate with remote servers. Each entry in this log preserves the application id and the URL, so that the application is contacted. From this log, with the help of a reliable and comprehensive domain blacklist, malicious applications that communicate with malicious domains can be detected. In [20], the validation of machine learning malware detection is discussed with in the lab and in the wild scenarios. At first, a feature set for building classifiers that yields high performance measures in lab evaluation scenarios is tested in comparison with state-of-the-art approaches. To this end, several Machine Learning classifiers that rely on a set of features built from applications' CFGs are devised. They used a sizeable dataset of over 50 000 Android applications collected from sources where state-of-the-art approaches have selected the data. Finally, the authors showed that, in the lab, the proposed approach outperforms existing machine learning-based approaches. However, this high performance does not translate in high performance in the wild. The performance gap was observed, F-measures dropping from over 0.9 in the lab to below 0.1 in the wild. In the work by Chen [21], a framework of utilizing model-based semi-supervised (MBSS) classification on the dynamic behavior data for Android malware detection is proposed. In this paper, focus was on detecting malicious behavior at runtime by using dynamic behavior data for analysis. The main advantage of semi-supervised classification is the strong robustness in performance for out-of-sample testing. The model-based semi-supervised classification uses both labeled and unlabeled data to estimate the parameters, since unlabeled data usually carries valuable information on the model fitting procedure. Specifically, MBSS is compared with the popular malware detection classifiers, such as support vector machine (SVM), k-nearest neighbor (KNN) and linear discriminant analysis (LDA). Finally, it is demonstrated that MBSS has a competitive performance for in-sample classification and maintains strong robustness when applied for out-of-sample testing. Under the ideal classification setting, MBSS has a competitive performance with 98% accuracy and very low false positive rate for in-sample classification.

Some of the work relates to the Intrusion Detection System (IDS), which defines and describes the taxonomies of such systems [22]. In another work by Homoliak [23], the taxonomy of intrusion detection methods is presented. Machine learning-based intrusion detection systems as well as network traffic classification ones are employed. Also, the study described datasets utilized for evaluation of Intrusion Detection Systems (IDSs) and finally provided an overview of obfuscation and evasion approaches in ADS and IDS, supplemented by several prevention techniques against obfuscations and evasions.

The system's features denoted as Advanced Security Network Metrics (ASNMs) are designed and defined and there was a performance improvement of detection by ASNM features and a supervised classifier, resulting in two categories of proposed obfuscation techniques. The first category of obfuscation techniques is called tunneling obfuscation and the second category is called non-payload-based obfuscations, which were evaluated and reviewed.

Considering the importance of traffic analysis and the benefits of machine learning methods to early detection of malware, many recent research works have considered these techniques for malware detection. However, as far as we know, none of these works have yet reviewed the comprehensive aspects of machine learning performance and different goals in malware detection. For example, no research work has discussed whether it is possible to identify the type of malware family or not; nor any of them has ever investigated the timing of the apps and the balance of the classes as effective parameters in the final results. Meanwhile, because the data collection and sampling method is different in each work, it has not been possible to compare the results with each other.

3. METHODOLOGY

The main purpose of this work is to evaluate the effective factors in identifying malware in smartphones, especially Android operating systems. To collect and capture traffic, it is necessary to check the flow of information between the points of connection carefully. In other words, these flows are known as a connection when there is a two-way exchange between two nodes. In fact, a flow is defined by a sequence of packets with the same values for five attributes; namely, Source IP, Destination IP, Source Port, Destination Port and Protocol [24]. More useful information is obtained by checking traffic associated with TCP flows. Hence, only TCP-related packets are considered in this work. In this article, in order to have a complete and comprehensive look at the pattern of behavior of malwares, it has been tried to collect a complete collection of all types of malware families and their behavioral differences are carefully evaluated. Malware samples include four different types of families extracted from the CICAndMal2017 dataset and UNB website [25]. Malware samples in the CICAndMal2017 dataset are classified into four categories: Adware, Ransomware, Scareware and SMS Malware. Our samples come from 42 unique malware families. A collection of benign applications from the Google Play Market was collected in the period from 2015 to 2017. These apps were collected based on their popularity (i.e., top free popular and top free new) for each category available on the market. In order to improve the quality of the surveys, four criteria were used: F1-measure (1), Precision (2). Recall (3) and false positive. Precision is known as positive predictive value **(**PPV**)** and recall is known as sensitivity**,** hit rate and true positive rate **(**TPR**).** F1-measure is a balanced mean between precision and recall. *TP* is the number of positive instances classified correctly; *FP* is the number of negative instances misclassified; *FN* is the number of positive instances misclassified; and *TN* is the number of negative instances classified correctly.

$$F1 - measure = \frac{2Precision * Recall}{Precision + \ Recall} \tag{1}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{2}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{3}$$

Since the normal distribution of benign and malware apps in the real world is unbalanced, most machine learning algorithms do not work well with the unbalanced dataset. Unbalanced datasets are a special case of classification problem, where the class distribution is not uniform among the classes. Typically, they are composed of two classes: the majority class and the minority class. This type of sets supposes a new challenging problem for data mining, because standard classification algorithms usually consider a balanced training set and this supposes a bias towards the majority class. To solve the problem of imbalance of data, a stratified sampling method was used. Stratified sampling builds random subsets and ensures that the class distribution in the subsets is the same as in the whole dataset.

The dataset contains 600 samples in the benign class and 400 samples in the malware class. After that, stratified 10-fold cross validation was implemented in our experiment and the average recall per class, average precision per class, FP and F1-measure were used for measuring performance.

To collect the feature set, all of the features used in recent works were studied. In many cases, up to 250 features per flow were extracted. These features are among the most important features of articles [6], [13] and [24]. Ideally, it is best to explore all feature combinations to select the one which gives the best result; however, in practice, this is problematic. For example, having n features, the number of experiments required to conduct training/testing on an algorithm is calculated as follows:

$$\sum_{k=0}^{n} \binom{n}{k} = 2^n \tag{4}$$

Therefore, the possible combinations are 2^n, where for n more than 40 the problem becomes unmanageable. For this reason, the forward feature selection algorithm was used to select features in five subsets. In order to have the best feature set, common features between this algorithm and the mentioned articles were chosen. These features are based on five characteristics: behavior-based, byte-based, packet-based, time-based and flow-based. The full list of features is presented in Table 1.

4. Benchmarking and Comparison Results

In this section, different classification methods have been comparatively analyzed on the collected data in order to evaluate their strengths.

Table 1. List of network features.

Feature	**Description**
Behavior-based	
F1	The duration of the flow
Byte-based	
F2	Average number of bytes sent
F3	Average number of bytes received
F4	The total number of bytes used for headers sent
F5	The total number of bytes used for headers received
F6	Ratio of number of incoming bytes to number of outgoing bytes
F7	Average number of bytes per second
Packet-based	
F8	Total number of packets sent
F9	Total number of packets received
F10	Total length of packets sent
F11	Total length of packets received
F12	Average number of packets per second
F13	Average number of packets sent per second
F14	Average number of packets received per second
F15,F16,F17,F18	Min, Mean, Max and standard deviation of the size of packet
F19,F20,F21,F22	Min, Mean, Max and standard deviation of the size of packet sent
F23,F24,F25,F26	Min, Mean, Max and standard deviation of the size of packet received
F27	Average number of packets sent/bulk
F28	Average number of packets received /bulk
F29	Subflow packets sent
F30	Subflow packets received
F31	Ratio of number of incoming packets to number of outgoing packets
Time-based	
F32,F33,F34,F35	Min, Mean, Max and standard deviation time between two packets sent in the forward direction
F36,F37,F38,F39	Min, Mean, Max and standard deviation time between two packets sent in the backward direction
F40,F41,F42,F43	Min, Mean, Max and standard deviation time when a flow was idle before becoming active
F44,F45,F46,F47	Min, Mean, Max and standard deviation time when a flow was active before becoming idle
Flow-based	
F48,F49,F50, F51	Min, Mean, Max and standard deviation of the length of a flow
F52	Average number of packets per flow
F53	Average number of packets sent per flow
F54	Average number of packets received per flow
F55	Average number of bytes sent per flow
F56	Average number of bytes received per flow
F57	The average number of bytes in a subflow in the forward direction
F58	The average number of bytes in a subflow in the backward direction
F59	Variance of total number of bytes used in the backward direction
F60	Variance of total number of bytes used in the forward direction
F61,F62,F63,F64,F65,F66,F67	Number of packets with FIN,SYN,RST,PSH,ACK,URG,CWE
F68	The total number of bytes sent in the initial window in the forward direction
F69	The total number of bytes sent in the initial window in the backward direction
F70,F71,F72	Ave, Max/Min segment size observed in the forward direction
F73,F74,F75	Ave ,Max/Min segment size observed in the backward direction

4.1 Benchmarking Based on the Number of Extracted Features

Feature selection is a very important component in data science. When data is presented on a very large scale, the training time increases resulting in that most models do not have proper and optimum output. Also, the larger dimension feature space creates a larger number of parameters that need to be estimated. As a result, the number of parameters increases with the possibility of overfitting in the model. Appropriate feature selection on one hand leads to a new set of features that are more compact and have more distinctive properties and on the other hand, unrelated and repetitive features are eliminated, which increases the F1-measure of the evaluation. Depending on the type of feature selection algorithm, there are three general approaches to feature selection which include filters, wrappers and embedded [26]-[28]. The wrapper method is widely used for classification issues; therefore, it was used here in this evaluation for feature selection. Features were examined in five subsets:

The first set consisted of 75 features and the second, the third, the fourth and the last sets contained 45, 25, 15 and 9 features, respectively (Table 2). In choosing these sets, packet-based features and then flow-based features were given special priority. These feature categories were included in all the subsets. In packet-based features [29], all received packets are processed; thus producing low false alarms, which makes this method very time-consuming. Flow-based features have an overall lower amount of data to be processed; therefore, this method is the logical choice to work with in high-speed networks. But it has less input information available to detect attacks and suffers from producing high false alarms. The main advantage of packet-based approach is that all common kinds of known attacks and intrusions can be detected if the data source delivers the entire network packet for analysis. On the other hand, performance issues in flow-based method are not a primary concern and therefore it is the logical choice for high-speed networks. Thus, the combination of both features is the best option, as it can detect a wide range of attacks with lowest error and highest speed.

To select the optimal feature set, three feature selection approaches from the wrapper method were investigated. These approaches were: forward selection, backward elimination and optimized selection. For this purpose, in the Rapid Miner tool, these three operators were compared. The survey was carried out each time with several classifications. The results of the survey showed that the optimized selection operator performed the best in all classifiers. Therefore, each feature set was selected by optimized selection.

Since the dataset contains a variety of heterogeneous features, these features are used in various algorithms that require a measure of distance/ similarity. Therefore, the data is normalized before it is used. Normalization is important when dealing with features and data with different scales.

In this evaluation, it has been attempted to use several well-known methods of ML supervised and unsupervised machine learning algorithms, in order to examine the performance of each one accurately. Supervised learning algorithms made use of a decision tree (with the maximal depth of the decision tree being 10 and the minimal leaf size of the tree being 4. Also, the minimal size for split is 4), random forest, KNN, Gaussian Naïve Bayes, SVM, MLP (multilayer perceptron) and Linear Regression; whereas unsupervised learning algorithms applied K-means and DBSCAN algorithms. The test results for the supervised algorithms were as follows: decision tree with high F1-measure of more than 90% and with average precision, average recall and FP rate at respectively 90.92%, 90.5% and 0.062%, followed by the KNN algorithm with F1-measure of 89.04% and average precision, average recall and FP rate at respectively 90.09%, 89% and 0.0955%, had the best performance. The SVM algorithm with F1-measure of 74.5% and average precision, average recall and FP rate at respectively 76.92%, 70% and 0.117% had the worst performance. Also, the F1-measure of the results for the two unsupervised K-means and DBSCAN algorithms was obtained at around more than 50% (Figure 1). According to the results of this evaluation (Figure 2), the reduction in data scale led to a change in the behavior of the classifiers, so that the reduction of the property gradually increased the F1-measure of all classifiers. This upward trend in F1-measure continued up to 25 features, but after a certain value, the increasing trend stopped and the values of F1-measure decreased in some classes and remained fixed in some others. Two algorithms (KNN and decision tree) are among the most widely used models that will have different F1-measure values depending on the input data. In other words, the quality of data, scales and classes used in a dataset can affect the performance of each of the categories. In a dataset with a low number of inputs, KNN can increase the F1-measure of prediction due to using methods such as linear least squares approximation. This change in classifier behavior is clearly visible in the peaking phenomenon. According to this phenomenon, as the number of features increases from one point to the next, the classification error also increases. This phenomenon *per se* can indicate using proper value, for features (not too little and not too much). In this study, the appropriate number of features was 25, for which most methods yielded acceptable results.

The results of the study showed that unsupervised methods have poor performance for malware detection compared to supervised methods. This is not a good result to identify Android malware that is usually unknown. Although supervised learning is most often used, it requires that the outputs of the algorithm be already known and the data used to train the algorithm be already labeled with the correct answer. On the other hand, unsupervised machine learning is closer to what is called real artificial intelligence. Given the problem under investigation, which is the identification of malware on smartphones with increasing emergence of malicious samples, labeling all the data is not practically possible. Thus, it is better to use a method that can accurately identify unlabeled data, or, in other words, identify new ones. Hence, semi-supervised classification that has been on the path of development over the past few years is a good choice to identify unlabeled data, or in other words unknown malware, with high performance. In these methods, labeled data is trained by supervised methods and unlabeled data is grouped using a labeled dataset and labeled with the highest degree of assurance. Finally, all the labeled data is trained and evaluated using one of the classifiers.

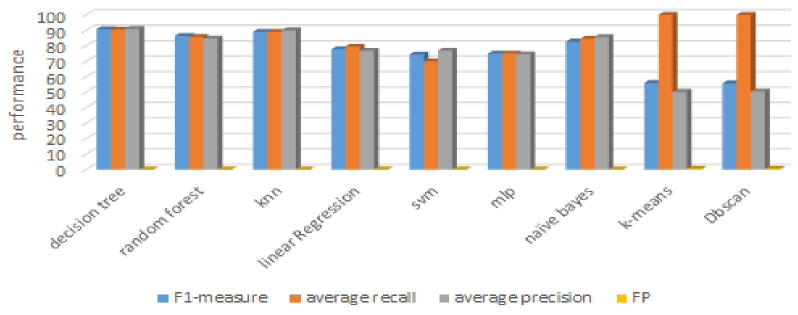

Figure 1. Results of the testing process with four criteria: F1-measure, average recall, average precision and FP.

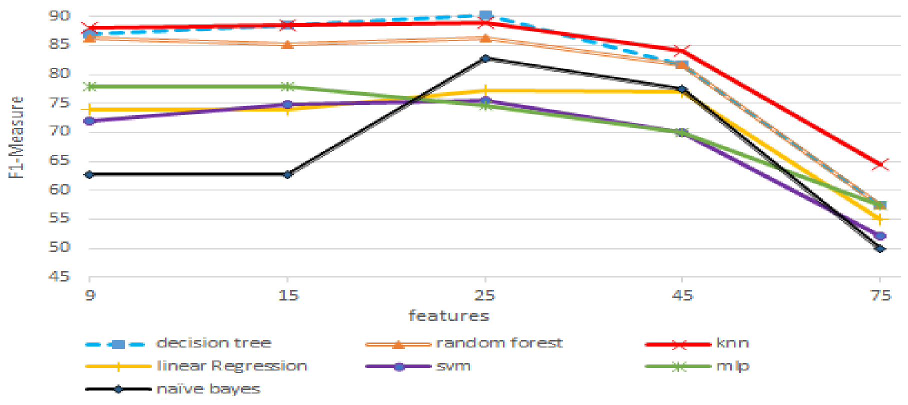

Figure 2. Benchmarking result in terms of the number of extracted features.

In the subsequent evaluations, we examined all experiments with 25 features as well as only the supervised algorithms.

Table 2. List of feature subsets.

Subsets	Features
9 features	F10, F11, F15, F16, F18, F42, F61, F69, F72
15 features	F10, F11, F13, F14, F15, F16, F18, F31, F42, F61, F69, F70, F71, F72, F74
25 features	F4, F5, F8, F9, F10, F11, F12, F13, F14, F15, F17, F18, F21, F25, F29, F30, F31, F42, F52, F61, F69, F70, F71, F72, F74
45 features	F1, F2, F3, F4, F5, F6, F8, F9, F10, F11, F12, F13, F14, F15, F16, F17, F18, F21, F25, F29, F30, F31, F32, F33, F34, F35, F36, F37, F38, F39, F42, F48, F49, F50, F51, F52, F53, F54, F55, F61, F69, F70, F71, F72, F74
75 features	F1,……, F75

4.2 Benchmarking in Terms of Time Duration of Capturing Traffic

The purpose of this evaluation is to assess the ability of various methods to detect malware in the shortest possible time. This issue is important, because in practice, if a system is supposed to detect malware, methods should be able to detect infections in a shorter time, as Internet access of phones to a connection point may not be possible for long. An applicable method should be able to detect malware as soon as possible by monitoring the traffic. In this benchmark, network traffic recording is marked every 15 minutes and classifiers are benchmarked using recorded traffic of 15, 30, 45 and 60 minutes, respectively. The results are shown in Figure 3. The F1-measure values of all methods increase with giving more traffic sampling time. The decision tree has a significant performance. Although KNN and decision tree are the best in 60 minutes of recorded traffic data, KNN cannot keep this performance for lesser time, while decision tree maintains this performance for 15 minutes of network traffic. The low level of F1-measure of other classifiers in the traffic volume of 15 minutes is due to the fact that many malwares have been widely generating traffic in more than 15 minutes of network presence. However, the decision tree has identified the presence of the first signals in the network.

This makes the decision tree the best choice in practice which can even detect a malware by recording smartphone traffic for a couple of minutes. The lowest F1-measure belongs to the Naïve Bayes method. This method had the least F1-measure in traffic for 15 minutes, while its F1-measure has improved significantly in traffic for 60 minutes compared to 15 minutes. Considering the need to rapid malware detection and the probability that a mobile phone would not be permanently connected to a long-term connectivity point, it seems that more research is needed to identify the appropriate features for high F1-measure detection in time durations as short as possible.

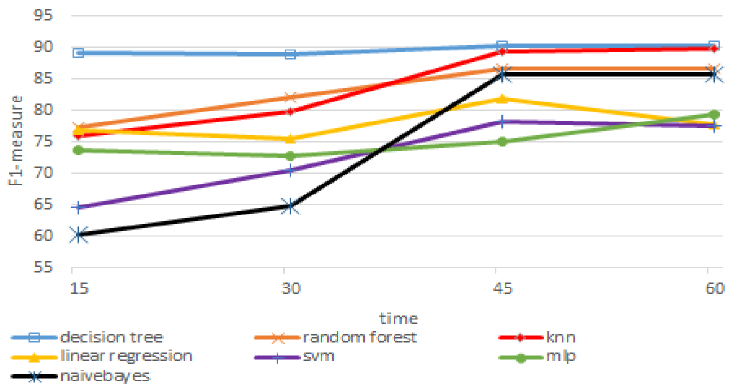

Figure 3. Benchmarking result in terms of the timing of program execution.

4.3 Benchmarking in Terms of the Ability to Detect Malware Families

Since malicious software can be categorized into the cybercrime group, it's important to identify types of malware families for proper response and defense after the attack. Identifying the behavioral patterns of traffic in each family can be very useful as a preventive measure of serious damage to Android devices. Therefore, in order to study the performance of machine learning classifications in identifying the type of malware family, two types of evaluation were performed. In the first one, the performance of each classifier was examined for identifying the type of malware family, while the second assessment evaluates the ability of each classifier to identify a new family of any malware that is separately evaluated in the next section.

In order to achieve better results, the experiment was performed with 25 attributes and a duration of 60 minutes. These were the best obtained adjustments in previous evaluations. In the first evaluation, four classes of malware families and 100 samples of them were tested. In this experiment, Naïve Bayes classification had a better performance with F1-measure of 74.83% compared to other classifiers. Decision tree classification had the lowest F1-measure (57.27%), (Figures 4-7).

Despite the best test conditions, the results were not satisfactory. The amount of data training is one of the important reasons for classification performance. In the case of the high number of classes, due to the lower number of data to be taught to each class, the classifier has less ability to distinguish between each class's patterns, which reduces the F1-measure of class identification. Other reasons include the type of input feature. Some of the features are noisy and they are not used in separating classes or some features cause incorrect recognition. The charts of F1-measure, average recall, average precision and FP values for each family are separately given in Figures 4-7. The high FP value indicates that the error rate of classification in identifying families from each other is significantly high. This can be confirmed for the F1-measure of this evaluation, based on the poor performance of classification in identifying families from one another. In this test, the features used to separate samples from each malware family have a lower ability of detaching malicious samples from benign samples.

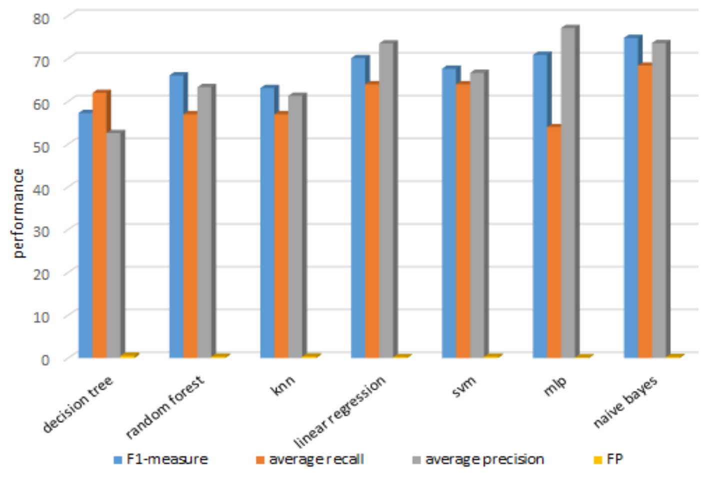

Figure 4. Benchmarking result in terms of the ability to identify Adware families.

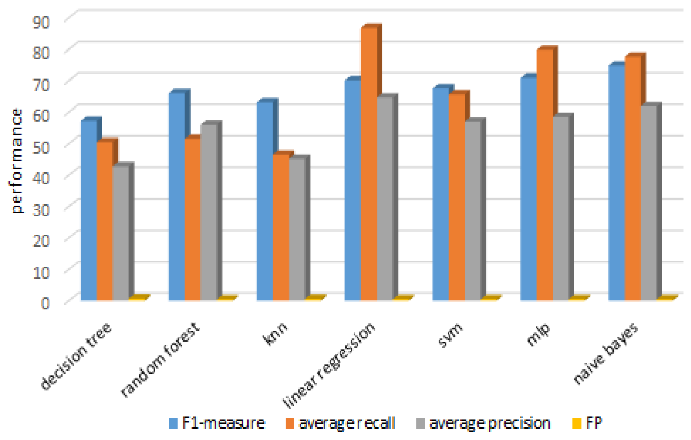

Figure 5. Benchmarking result in terms of the ability to identify Ransomware families.

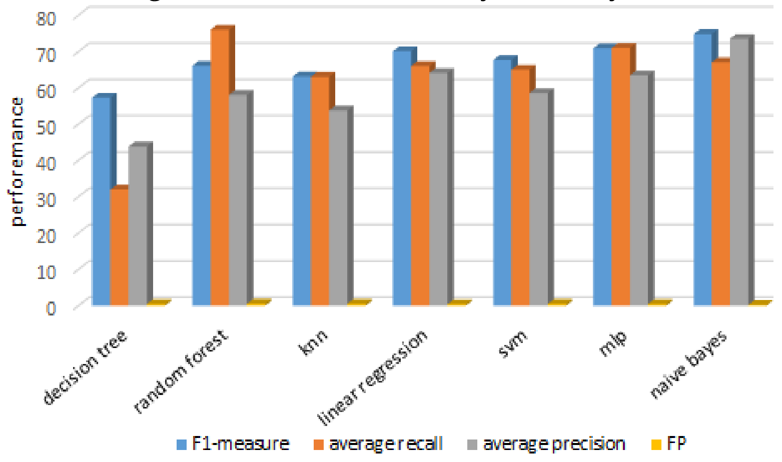

Figure 6. Benchmarking result in terms of the ability to identify Scareware families.

4.4 Benchmarking in Terms of the Ability to Identify Unknown Malware Types

In this evaluation, each new sample of families was separately examined. In this way, from each category of malware, a limited number of families were assigned to the test data and the rest of the families were used as the training data to the algorithm given. The purpose of this study is to measure the performance of machine learning classifications when new types of families are produced. These samples are initially unknown and each sample may have a different pattern. Therefore, the extraction of undiscovered patterns from unknowns and the association of these patterns with known samples can be easily accomplished by new methods such as machine learning. The results of the study showed that the F1-measure values obtained ranged from 21 to 80 percent, according to Figure 8. The SVM classifier achieved an F1-measure of over 80% and has been able to identify new samples of the Ransomware family. Also, the Naive Bayes classifier with the same F1-measure has been able to identify the family of SMS Malwares. However, the Random forest classifier with an average F1-measure of 66.49% had the best performance in identifying the four families and unexpectedly the KNN classifier with an average F1-measure of 54.92% in identifying the four families had the lowest level of detection F1-measure. Generally, classifier performance was better in identifying new samples in the Ransomware and SMS Malware families than in the families of Adware and Scareware. Since new families of

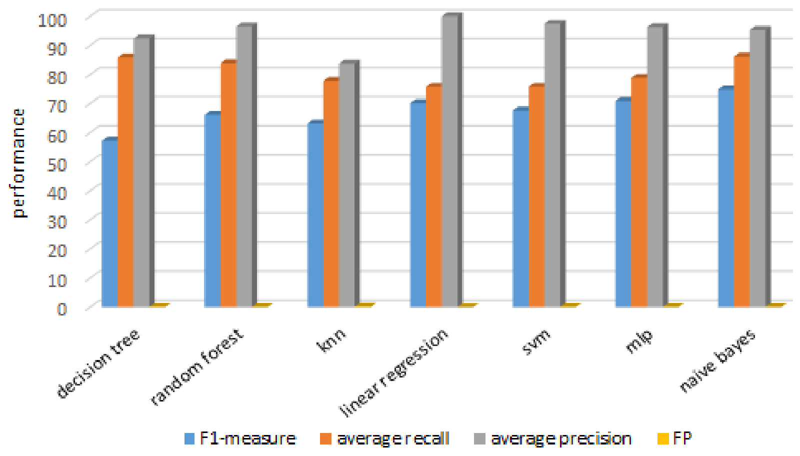

Figure 7. Benchmarking result in terms of the ability to identify SMS Malware families.

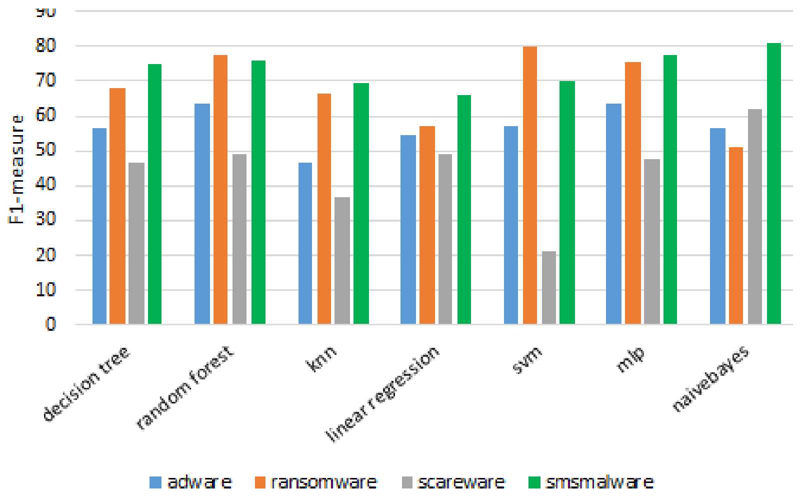

Figure 8. Benchmarking result in terms of the ability to identify new samples from each family.

Ransomware appear almost daily and traditional signature-based detection methods cannot be used to detect them, it is possible to combine deep system monitoring with machine learning, resulting in a system that can detect new families of Ransomware in real time by searching for certain behavioral patterns. Despite the fact that multiple families of Ransomware use different approaches to obfuscation, encryption and demand for ransom, the majority of them display similar behavioral patterns that can be detected. It can be argued that the traffic patterns of each family of malware are almost identical and that algorithms currently do not have the ability to identify each group of malware families. This low performance means the inability of the current algorithms to detect Android malware.

5. LIMITATIONS AND THREATS TO VALIDITY

The validity of benchmarking results may be threatened from several points which are mentioned and discussed below:

- Despite the title of the paper that is to benchmark Android malware detection machine learning algorithms, the studied methods are only able to detect those malwares which need to send packets over the network. These methods cannot be applicable if a malware does not send any packets on the network.
- The dataset studied in this paper has four categories of malware families; namely, Adware, Ransomware, Scareware and SMS Malware families. The authors do not claim that these results are valid for all malware families. Other case studies on malware detection may yield other results.
- The results represented in this paper are dedicated to CICAndMal2017 dataset which is a recently published one. The results for another dataset may be slightly different from those achieved in our study. For example, the current dataset includes the most popular applications. Obviously, any other sampling of applications, such as rarely used ones, requires a dedicated research.
- Although the dataset used in this paper is fresh, it is clear that changes in malware behavior in the future may lead to changes in the results.
- These results are in a situation where malwares are not aware of the existence of a malware detection system through network traffic.
- Malwares may bypass a detection mechanism by hiding or obfuscating their communications or by deceiving the machine learning algorithm through creating "adversarial examples" if they are aware of the existence of such a system.

6. CONCLUSIONS

One of the advantages of malware detection by analyzing network traffic is that through network traffic behavior, malicious samples can be identified before causing serious damage. In this paper, machine learning methods have been investigated in several aspects. These evaluations include the number of features needed to learn, the type of machine learning, the recorded traffic volume, the ability to identify the type of malware family and the ability to identify a new type of malware family. The investigated methods; namely, decision tree, Random forest, KNN, Linear Regression, SVM, MLP and Gaussian Naïve Bayes, were investigated in terms of sensitivity to the number of features examined. The results were as follows:

- The number of features should not be greater or less than a certain amount. Choosing the proper number of features and optimizing the size of the data lead to acceptable results.
- Almost all methods of increasing the time or increasing the volume of traffic recorded improved F1-measure. Only the decision tree algorithm was able to detect the presence of malware in a small volume of traffic with highly accurate prediction.
- In benchmarking the type of malware family, the performance of the algorithms was very low, considering the best conditions for the previous evaluations. Machine learning algorithms had a low ability to identify the behavior patterns of each family of malware.
- In terms of the last aspect benchmark to examine the performance of classification in identifying new or unknown samples. The results were also very low and not acceptable. It seems that further research is needed to identify the proper features for faster detection of malwares and their type at shorter time points.
- In all of the benchmarks, the type of data, the number and volume of data, as well as the number of noise features and the use of their proper number, have a great influence on the performance of all algorithms.

One of the main constraints of machine learning is the lack of prediction of points that are considered as noise. If a new data is received that belongs to the noise part, it can't be classified using these methods.

Therefore, deep learning approaches at a deeper level than machine learning, inspired by the performance of the human brain and complex calculations on a large volume of data, solve issues thoroughly, having far better outcomes than machine learning approaches.

REFERENCES

[1] T. T. Mikko Hypponen, "F-Secure 2017 State of Cybersecurity Report," F-Secure, Tech. Rep., 2017.

[2] S. -H. Seo, A. Gupta, A. M. Sallam, E. Bertino and K. Yim, "Detecting Mobile Malware Threats to Homeland Security through Static Analysis," Journal of Network and Computer Applications, vol. 38, pp. 43-53, 2014.

[3] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller and K. Hanssgen, "A Survey of Payload-based Traffic Classification Approaches," IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1135-1156, 2013.

[4] H. Singh, "Performance Analysis of Unsupervised Machine Learning Techniques for Network Traffic Classification," Proc. of the 5th IEEE International Conference on Advanced Computing & Communication Technologies, pp. 401-404, , 2015.

[5] S. Zander, T. Nguyen and G. Armitage, "Automated Traffic Classification and Application Identification Using Machine Learning," Proc. of IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05), pp. 250-257, 2005.

[6] F. A. Narudin, A. Feizollah, N. B. Anuar and A. Gani, "Evaluation of Machine Learning Classifiers for Mobile Malware Detection," Soft Computing, vol. 20, no. 1, pp. 343-357, 2016.

[7] S. Garg, S. K. Peddoju and A. K. Sarje, "Network-based Detection of Android Malicious Apps,"International Journal of Information Security, vol. 16, no. 4, pp. 385-400, 2017.

[8] Y. Pang et al., "Finding Android Malware Trace from Highly Imbalanced Network Traffic," Proc. of IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1, pp. 588-595, 2017.

[9] S. Pooryousef and K. Fouladi, "Proposing a New Feature for Structure-Aware Analysis of Android Malwares," Prco. of the 14th IEEE International ISC (Iranian Society of Cryptology) Conference on Information Security and Cryptology (ISCISC), pp. 93-98, 2017.

[10] A. Arora, S. Garg and S. K. Peddoju, "Malware Detection Using Network Traffic Analysis in Android Based Mobile Devices," Proc. of the 8th IEEE International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 66-71, 2014.

[11] J. Erman, A. Mahanti and M. Arlitt, "Qrp05-4: Internet Traffic Identification Using Machine Learning," IEEE Globecom, pp. 1-6, 2006.

[12] J. Erman, M. Arlitt and A. Mahanti, "Traffic Classification Using Clustering Algorithms," Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, ACM, pp. 281-286, 2006.

[13] A. Arora and S. K. Peddoju, "Minimizing Network Traffic Features for Android Mobile Malware Detection," Proceedings of the 18th International Conference on Distributed Computing and Networking, ACM, p. 32, 2017.

[14] T. Bujlow, T. Riaz and J. M. Pedersen, "Classification of HTTP Traffic Based on C5. 0 Machine Learning Algorithm," Proc. of IEEE Symposium on Computers and Communications (ISCC), pp. 000882-000887, 2012.

[15] O. M. Alhawi, J. Baldwin and A. Dehghantanha, "Leveraging Machine Learning Techniques for Windows Ransomware Network Traffic Detection," Cyber Threat Intelligence, pp. 93-106, 2018.

[16] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder and L. Cavallaro, "TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time," arXiv preprint arXiv:1807.07838, 2018.

[17] D. Nancy and D. Sharma, "Android Malware Detection Using Decision Trees and Network Traffic," International Journal of Computer Science and Information Technologies, vol. 7, no. 4, pp. 1970-1974, 2016.

[18] Z. Chen et al., "Machine Learning Based Mobile Malware Detection Using Highly Imbalanced Network Traffic," Information Sciences, vol. 433, pp. 346-364, 2018.

[19] M. Zaman, T. Siddiqui, M. R. Amin and M. S. Hossain, "Malware Detection in Android by Network Traffic Analysis," Proc. of IEEE International Conference on Networking Systems and Security (NSysS), pp. 1-5, 2015.

[20] K. Allix, T. F. D. A. Bissyande, Q. Jerome, J. Klein and Y. Le Traon, "Empirical Assessment of Machine Learning-based Malware Detectors for Android: Measuring the Gap Between in-the-lab and in-the-wild Validation Scenarios," Empirical Software Engineering, pp. 1-29, 2014.

[21] L. Chen, M. Zhang, C.-Y. Yang and R. Sahita, "Semi-supervised Classification for Dynamic Android Malware Detection," arXiv preprint arXiv:1704.05948, 2017.

[22] H. Debar, M. Dacier and A. Wespi, "A Revised Taxonomy for Intrusion-detection Systems," Annales des Télécommunications, Springer, vol. 55, no. 7-8, pp. 361-378, 2000.

[23] I. Homoliak, Intrusion Detection in Network Traffic, Dissertation, Faculty of Information Technology, University of Technology, 2016.

[24] A. H. Lashkari, A. F. A. Kadir, H. Gonzalez, K. F. Mbah and A. A. Ghorbani, "Towards a Network-based Framework for Android Malware Detection and Characterization," Proc. of the 15th IEEE Annual Conference on Privacy, Security and Trust (PST), pp. 233-239, 2017.

[25] A. H. Lashkari, A. F. A. Kadir, L. Taheri and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Android Malware Datasets and Classification," Proc. of IEEE International Carnahan Conference on Security Technology (ICCST), pp. 1-7, 2018.

[26] A. Jović, K. Brkić and N. Bogunović, "A Review of Feature Selection Methods with Applications," Proc. of the 38th IEEE International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1200-1205, 2015.

[27] S. Maldonado, R. Weber and J. Basak, "Simultaneous Feature Selection and Classification Using Kernel-penalized Support Vector Machines," Information Sciences, vol. 181, no. 1, pp. 115-128, 2011.

[28] M. Q. Nguyen and J. P. Allebach, "Feature Ranking and Selection Used in a Machine Learning Framework for Predicting Uniformity of Printed Pages," Electronic Imaging, vol. 2017, no. 12, pp. 166-173, 2017.

[29] H. Alaidaros, M. Mahmuddin and A. Al Mazari, "An Overview of Flow-based and Packet-based Intrusion Detection Performance in High Speed Networks," Proceedings of the International Arab Conference on Information Technology, pp. 1-9, 2011.

ملخص البحث:

أصبحت الهواتف الذكية تحتل جزءاً أساسياً من حياة الناس هذه الأيام، وقد قاد ذلك الى انخراط أعداد متزايدة من المستخدمين في استخدام هذه التقنية. ولقد شجع ازدياد أعداد المستخدمين القراصنة الإلكترونيين على إنشاء تطبيقات هدفها إلحاق الأذى بمستخدمي الهواتف الذكية. وما من شك في أن تحديد الاختراقات التي يتعرض لها المستخدمون والكشف عنها أمر حاسم من أجل الحفاظ على أمن المستخدمين وخصوصيتهم. وتبين الاتجاهات الحديثة للأمن السيبراني أنه يمكن تحديد تلك التهديدات بصورة فعالة باستخدام تقنيات كشف ترتكز على الشبكة وعدد من طرق تعلم الآلة.

وفي هذه الورقة، تم استقصاء عدد من طرق تعلم الآلة المعروفة جيداًمن أجل الكشف عن الاختراقات التي تتعرض لها الهواتف الذكية باستخدام حركة المرور على الشبكة. واستخدمت عدة عائلات اختراق في الاستقصاء (أدوير "Adware"؛ رنسم وير "Ransomware"؛ سكيروير "Scareware"؛ إس إم إس مالوير "SMS Malware"). من جهة أخرى، جرى استخدام طرق تعلم الآلة الأوسع استخداماً والأكثر شهرة؛ المراقبة وغير المراقبة. وقارنت الدراسة بين هذه الطرق من عدة جوانب، مثل عدد السِّمات المطلوبة، وحركة المرور المسجلة، والقدرة على كشف عائلة الاختراق، والقدرة على كشف عائلة اختراق جديدة.

وبينت النتائج أن استخدام هذه الطرق بالسمات المناسبة وحركة المرور ذات الحجم المناسب من شأنه أن يحقق نسبة كشف اختراقات تقرب من 90%. غير أن هذه الطرق لم تحقق نتائج مقبولة من حيث كشف عائلات الاختراق أو عائلات الاختراق الجديدة غير المعروفة. ومن جهة اخرى، شرحت الدراسة بعض التحديات ووجهت الى مسائل محتملة يمكن للباحثين أن يتناولوها مستقبلاً بناءً على نتائج الدراسة الحالية.