MACHINE LEARNING BASED ON MALWARE DETECTION IN MOBILE NETWORKS

Deemah A Alotaibi[1], Malak F Aldakheel[2], Norah S Al-Serhani[3], Rachid Zagrouba[4]
College of Computer Science and Information Technology, Imam Abdulrahman bin Faisal University Dammam Kingdom of Saudi Arabia
2150004996@iau.edu.sa, 2160007006@iau.edu.sa, 2150000744@iau.edu.sa, rmzagrouba@iau.edu.sa
(Presented at ICSC, 2019, KSA)

ABSTRACT*—The rapid growth of Android malware is noticeable with its serious threat to Android users. Such mobile malware usually uses network communication to steal users' data and launch different attacks. Therefore, there is a need to protect Android users from malware and preserve their privacy. Accordingly, many researchers proposed network-based detection mechanisms that utilize machine learning techniques for detecting mobiles malware. This paper reviews malware detection techniques that rely on analyzing network traffic. Then, it compares and analyzes them based on specified criteria, which are the type of malware to be detected, the detection rate, the used machine learning algorithm, resource consumption rate, the extracted traffic features, the ability to detect encrypted traffic, and the ability to detect unknown apps. The result shows that TextDriod mechanism is the best solution since it comprehensively satisfies the required criteria.*

Keywords—Malware detection, network-based, Android, Botnet, Machine Learning

INTRODUCTION

NOWADAYS, smartphones play a major role in people's life since it gives its users an opportunity to do everything in only a few minutes. It facilitates people's life by providing easily nailed educational tutorials, communication, sharing media, money transaction, and many others. These activities are done by using the appropriate application that provides the wanted activity. Besides, the smartphone is not limited to only applications, it also used to store information including the confidential information. Therefore, smartphones are considered as an ultimate goal that can be used to harm the user. Accordingly, the number of malware that targets the smartphone has been increased, such malware is, ransomware, spyware, malicious apps, etc.

Since smartphones are goldmine that contains lots of information about the user, there is a need to protect it from any type of attacks. Although all mobile devices are targeted to be attacked, Android devices are the most attacked ones due to its popularity and its open architecture [1]. The variations among malware behaviors in Android mobile devices results in some deficiencies in the existing security solutions. Some security solutions are designed to detect malware types that present its maliciousness in its codes, accordingly, can be detected by analyzing the applications' codes. Others are designed to examine the changes in the operating system functionalities or device resources such as memory and CPUs for malware detection. However, there is some malware that covers its maliciousness in term of code and OS functionalities, such as botnets and self-updating malware but represents it through the generated network traffic. Since the detection of such malware depends on network monitoring instead of static approaches, employing the science of Artificial Intelligence and Machine Learning in the security solutions designed to detect such types of malware considered as an effective approach. Accordingly, this paper aims to solve this problem by analyzing different network-based detection solutions that employ machine learning techniques and propose enhancements to effectively detect such malware.

The paper is organized as followed. Section II states the problem statement, Section III addresses the background. Section IV is a literature review that discusses several machine learning malware detection methods. Section V illustrates a comparison between the detection methods in section IV. Finally, Section VI is the conclusion.

I. LITERATURE REVIEW

In this section, several network-based malware detection techniques are discussed. Starting with the botnets detection techniques, followed by self-updating malware detection techniques, and concluding with some generic malware detection techniques.

A. Botnet detection techniques

Garg, Peddoju, and Sarje [2]. proposed a network-based app model to detect malicious apps, specifically botnets, in Android-based mobile devices. The main objective of such a proposed model is to identify malicious apps from benign ones. It consists of three main modules, namely, Network Traces Collection, Network Feature Extraction, and Detection module, as shown in

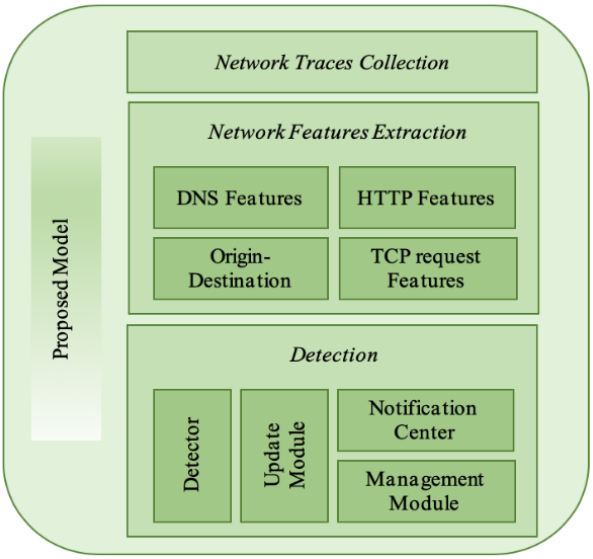

Figure 1.

Figure 1 Proposed Detection System

The first module, Network Traces Collection, is responsible for monitoring the network traffic generated from all the running apps in the mobile device, whether it is encrypted or plain. This is done periodically, every t seconds. Then, it records the headers only instead of the entire packets. Such collection contributes in making the model lightweight since it records a small amount of data which suites the limited memory, CPU processing of mobile devices.

The second module, Network Feature Extraction, is responsible for extracting selected network features that is generated from short intervals and can be extracted without a need to record the entire packets. To meet such conditions, four different categories of network traffic features are used, DNS-based features, HTTP-based features, Origin-Destination-based features, and TCP-based features.

The third module, Detection, is responsible for detecting malicious apps. In this module, Random Forest Machine learning classifier is used. This module consists of four components, namely, Detector, Update Module, Notification Center and Management Module. First, the detector takes an input from the previous module, Network Feature Extraction, in a feature vector form. Based on that, the detector classifies the app as malicious or benign. The detector triggers the Notification Center in case of malicious app detection which in turn transmits a warning alert. Such a warning alert is then shown in the notification bar of the mobile device. The Management Module is used for management purposes to set different parameters such as log records, alert mode, and alert threshold.

An experiment is performed to test the proposed model on two different mobile devices. The experiment followed four phases. First, traffic from malicious and normal apps was collected to learn its behavior. Second, this traffic is used to test different machine learning algorithms against accuracy, complexity, and stability. Then, the optimal algorithm, which is Random Forest, is chosen as a detector in the proposed app after training. Lastly, a complete mobile app results that can be downloaded from a server. Four experiments were implemented to test the detection accuracy, the device dependency, unknown apps detection and the detection of malicious apps with encrypted communication.

The results showed that the proposed model is capable of detecting malicious apps using network traffic with 98% accuracy. Moreover, Random Forest classifier showed a lesser detection time, higher accuracy and stability. Also, the experiment shows that the model is device independent with an average detection rate of 95.4%. In addition to detecting the plaintext malicious communications, the experiment showed an average detection rate of 93% of malicious apps that are communicated in an encrypted format. Lastly, it is shown that the model is capable of detecting the unknown apps that have similar behavioral apps in the training with a 99% rate. However, it reached an average rate of 86% for the whole tested classifiers.

Narudin, Feizollah, Anuar and Gani [3] proposed a solution to evaluate malware detection that integrates anomaly-based approach with machine learning classifiers. The main objective of this work is to evaluate various machine learning classifiers for Android malware detection using network traffic. Specifically, the used machine learning classifiers are decision tree, Bayes network, multi-layer perceptron, k-nearest neighbors and random forest.

The evaluation process basically consists of two stages, the first stage is performing experiments and the second is analysis. The experiment carried out in three substages, data collection, feature selection and extraction and machine learning classifiers, as shown in Figure 2.

In the data collection stage, network traffic for malicious and benign applications was captured. The selected benign traffic was captured from running 20 different trusted applications such as Twitter and Facebook, each one was captured separately to preserve the isolation of the traffic. On the other hand, malicious traffic was collected by utilizing two malware sets, public and private. The former is MalGenome, which is a public dataset consist of 1260 malware categorized into 49 different families, the experiment utilized 1000 malware samples from such families. The latter is a self-collected dataset from the latest mobile malware traffic consists of 30 samples.

In the feature selection and extraction stage, the traffic generated from the previous stage was utilized as an input to select the intended features. First, the TCP traffic was filtered. Then, four features were selected, namely, basic, content-based, time-based and connection-based features. After that, the extracted features were labelled and stored in a database to constitute the input of the next stage.

In the machine learning classifier stage, five different machine learning classifiers were used to train on the information stored in the database to eventually produce an effective detection model.

The assessment of the ideal classifier involved two main experiments, one utilized MalGenome and the second utilized the private dataset. The evaluation measures were True Positive rate, False Positive rate, Precision, Recall, and F-measure.

- True Positive rate: the rate of predicted malware classified correctly
- False Positive rate: the rate of benign apps classified as malware
- Precision: the rate of relevant results
- Recall: the sensitivity for the most relevant result
- F-measure: a combination of precision and recall to estimate the entire system performance

The result of each experiment was analyzed and compared. The final result showed that random forest machine learning classifier achieved 99.99% for MalGenome detection. On the other hand, KNN went better over random forest by a difference of 10.42% with the latest malware detection and achieved 84.57%.

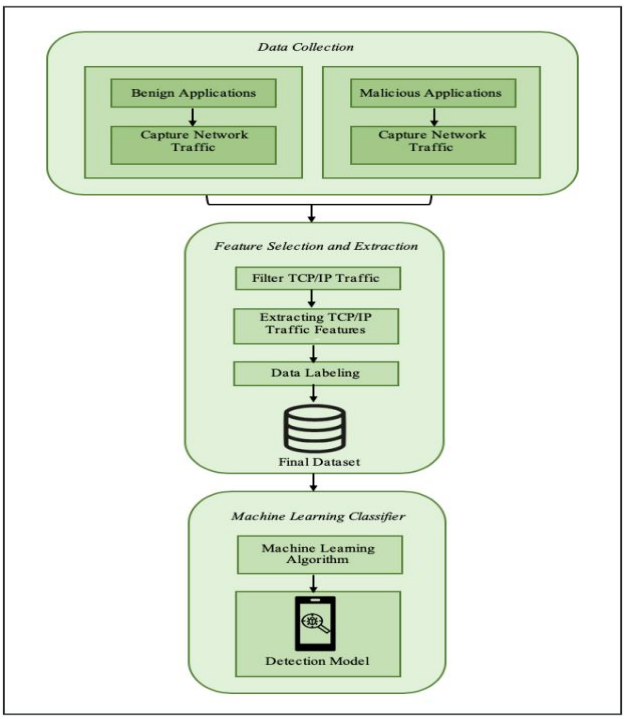

Figure 2 experiment structure

B. Self-updating malware detection techniques

Shabati *et al.* [4] have proposed a host-based method for detecting mobile malware, especially, the self-updating malware, in the paper entitled by "Mobile malware detection through analysis of deviations in application network behavior" [4]. The detection method relies on learning the usual network traffic caused by mobile applications. Once this traffic is known, it becomes possible to detect any deviations on the application's traffic. This method uses machine learning algorithms on the host machine, Android mobiles, to detect malware. Since this method solely depends on network traffic, it can detect any malware that causes a moderate to high network traffic even if the malware is new or encrypted. This is due to the fact that self-updating malware, and some other types, change the network traffic and make it distinguishable from the regular one. The malware detection system in this solution has four main components, which are Features Extraction, Features Aggregation, Local Learner, and Anomaly Detector. Table 1 summarizes these components.

The main component for detecting malware is the Anomaly Detector. This component relies on predicted values assigned by the Local Learner for all the chosen features (i.e. the aggregated features during the Features Aggregation process). The Anomaly Detector continually monitors the network traffic of an application to observe the values of the selected features and compare them with the predicted ones. The probability of classifying an event as abnormal and consequently a malware is found by multiplying the evaluated differences between the predicted and observed values of all the features. Equation 3 calculates the probability of classifying a feature x as abnormal, where A(x) is the predicted, B(x) is the observed value, and dist denotes the distance between the two values.

Table 1 The main components of Shabati's malware detection system [4].

Features Extraction	Extracts certain information from the network traffic of a running application.
Features Aggregation	Aggregates the extracted data to represent the application's traffic.
Local Learner	Learns the application network pattern by adapting the cross-feature analysis approach using Decision table and REPTree algorithms. It takes place whenever an application is installed or updated.
Anomaly Detector	Analyzes the network behavior for detecting any deviations from the regular traffic.

P(B(x) is abnormal) = 1- dist (A(x), B(x))

The researchers conducted several experiments with benign, self-written malware, and real malware applications. The benign applications encompassed Gmail, WhatsApp, Facebook. The self-written malware applications were created to imitate the behavior of self-updating malware. The real malware were normal applications injected by real malware such as PJApps trojan. After conducting these experiments, the researchers found that this mechanism gives the best results when extracting a specific set of features, such as the average of the sent and received bytes and data as well as using the Decision Table and REPTree algorithms. Using these specified features and algorithms provides a high detection rate of malware, especially the self-updating malware where the detection rate is between 90-100% and the false positive rate doesn't exceed 10%. In addition to the high detection rate, the resources consumption in this mechanism is acceptable where it consumes while learning applications' traffic less than 2% of the mobile's memory and about 14% of the CPU.

C. Generic malware detection techniques

Wang et al [1] [5] proposed an Android malware detection platform based on text semantics of the collected traffic, namely TextDriod. It analyzes the request header of HTTP/HTTPS traffic since it contains lots of text semantics. It is deployed in the server and this server has a connection with the gateway to collect the traffic, see Figure 3.

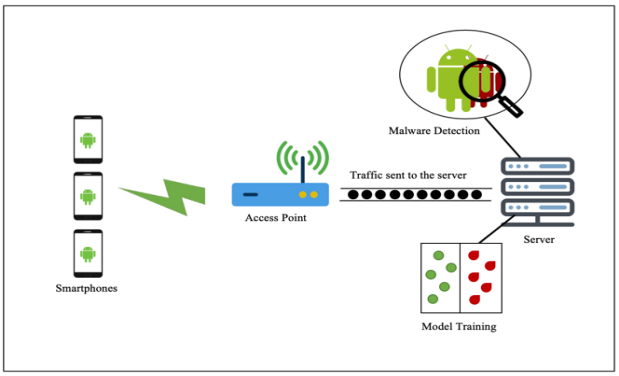

Figure 3 TextDriod Detection Model

Figure 4 shows the TextDriod detection process that goes through five different stages, which are:

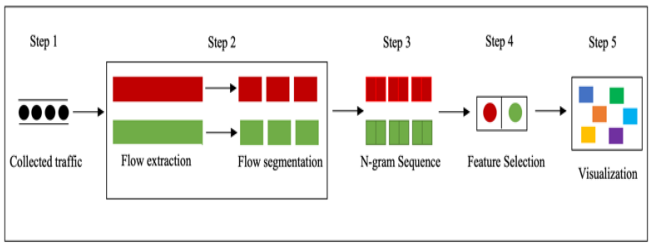

Figure 4 TextDriod's process

1. Traffic collection

A traffic collection platform consists of Con2trol Center, App & Traffic Storage server, and traffic collection server is used to collect all the mobile traffic. The collected traffic contains a different type of traffic which are the execution and data of the app that is generated by the apps. The data traffic is a set of HTTP/HTTPS flows that are mixed of all apps' data traffic.

2. Traffic Preprocessing

Once the data traffic is received from the collection platform, firstly, the flow extraction process takes a place which is an algorithm that takes the mixed traffic as an input and extracts a collection of documents, each document contains the request header of a single HTTP/HTTPS flow.

Secondly, flow segmentation process that splits the document's strings into single words by using special characters which are (, : ; &) . Additionally, it removes the meaningless words such as the .jpg, .png, the, is, were, content-length, and en-us.

3. N-Gram Generation

N-gram method is one of the natural language processing (NLP) methods that is used to identify the sequence of words in the flow in order to give semantic information about the flow. That is done by observing the occurrence of the n-th word if the occurrence of each word is depending upon the previews words (n-1) or not. For example, if the words are independent of each other, then we substitute the letter N to be 1 (1-gram), if two words are depended in each other, then we substitute the letter N to be 2 (2-gram). This step provides contextual information in order to know if there is a relationship between multiple words. The outputted meaningful word sets called features.

4. Feature Selection

The previous stage generates lots of features that are not that much significant for the malware detection model, thus, the chi-square test is used to filter them out. the chi-square test is a statistical test used to determine if the observed distribution of categorical variables significantly differs from the expected one or not, meaning, to distinguish if the observed distribution is malware traffic or benign traffic. That is done by using a measurement method that assigns each feature chi-squared test value which is the feature's frequency in the specific traffic. The chi-squared test value formula is:

$$\chi^2(t,c) = \frac{(N_{t,c} - E_{t,c})^2}{E_{t,c}}$$

If the result of this formula exceeds the threshold, that means the selected feature and the specified category are dependent on each other. If they are dependent that means this feature is significant to this category, otherwise, the feature is insignificant to this category. The resulted sets of the selected and weighted N-gram sequences is called bag-of-word.

5. Flow Header Visualization

The last step of the proposed platform is to visualize the detected traffic by representing it in a word cloud format. That is done by using the word and its weight. The researchers use the font size and the font color to represent the words, meaning, the word with high frequency will be larger and more vivid than the word with low frequency.

The used machine learning is a Support Vector Machine (SVM) algorithm. Since it should be trained to detect the malware, the conducted consists of two steps which are :

- Word Vectorization:

It converts the bag-of-words into numeric vectors, that is done by replacing each N-gram sequences that exist in the flow with binary values. If the N-gram sequence exists, then the corresponding binary value is "1", otherwise, the binary value is "0". Such an encoding process, each HTTP flow transformed into a numeric vector.

- Model Building:

It is a process of identifying the malicious traffic by finding out the source of this traffic by using Support Vector Machine (SVM) algorithm. The SVM algorithm works by learning the features of the malicious traffic and benign traffic, then constructs multiple hyperplanes which is a line that separates the two classes. Those multiple hyperplanes are compared and select the one that has a maximum margin between the two classes. After that, the detection model calculates the detection function which is a combination of the weight vector and the feature vector, as equation below shows, if the F(x) > 0 that indicates malicious activity, while F(x) < 0 indicates a benign activity.

$$F_{(x)} = \langle W, V \rangle$$

This proposed detection model is applied in the test set, it detects and identifies 99.15% of the malware and its misjudgment rate is 0.45%, in case that the N value of N-gram sequence is 1 and the number of features is 600, which is the optimal level. On the other hand, once the detection model is applied in a real-world environment, it detects 76.99% of the all malware application. Additionally, the model has the ability to detect 54.81% of the new malware types that are not be detected by anti-virus.

In 2019, Wang et al. [6] have proposed another mechanism that is similar to the previous one, however, the features set has been extended to include features from the TCP flow. These features are the uploaded/downloaded bytes, the number of uploaded/downloaded packets per one session, and their average bytes. Including TCP flow features assists in detecting malware that uses HTTPs instead of HTTP. The learning-based for the detection in this mechanism uses the C4.5 decision tree algorithm where a tuple of tree nodes, branches, and leaf nodes represent the feature's name, value, and class respectively.

The researchers did an experiment with more than 10,000 tuples of HTTP and 24,000 for TCP. Based on this experiment, it is found that the detection rate of malware varies based on the malware family and the number of its

initial training samples, and the used model (i.e. HTTP or TCP). By that, a malware from an unknown family type that had no training samples is hard to be detected by this mechanism unless it has similar features with a malware family that had many samples. Therefore, this mechanism needs to be continually trained with samples of different malware types. As for the model type, when binding both of the models, the detection rate increases approximately by 50%. However, when using one model only, then relying on HTTP model would give a higher detection rate by 5%.

As for the resources' consumption, this mechanism is considered to be a lightweight detection mechanism since the learning and detecting processes are done on a server that receives the network traffic from the access point. In addition to the process of detecting malware, the server provides a meaningful discerption once it detects an application as malware to the mobiles. Such messages clarify the feature that led to classifying the detected application as malware.

Nacy, and Sharma [7] proposed a detection technique that uses a decision tree model which is a supervised classification learning technique that is widely used to differentiate each traffic from the other. That is done by relaying upon various features that are used to distinguish each traffic. Firstly, the researchers conducted an experiment in order to find out the distinguishing features which are features that can clearly differentiate between malware traffic and benign traffic. As a result of that, there are 8 distinguishing features which are:

- Average Packet Size
- Average No of Packets Sent per Flow
- Average No of Packets Received per Flow
- Average No of Bytes Sent per Flow
- Average No of Bytes Received per Flow
- The ratio of Incoming to Outgoing Bytes
- Average No of Bytes Received per Second
- The ratio of Number of Connections to Number of Destination IPs.

In all those features, the malicious traffic has a small value compared to the benign one. Thus, if the received traffic has a small value of all those features, that means the traffic is from malicious application.

Secondly, the researchers build a detection model which is Decision Trees model. The tree is built on the 8 distinguishing features that are specified above. Therefore, once the traffic of an application is forwarded to the Decision Tree classifier, the traffic will go through the tree and the last node that is reached will give the conclusion about the traffic whether it is malicious or not. This detection technique detects 90% of the samples, which means it provides high accuracy. In contrast, it lacks the opportunity to detect encrypted malware traffic.

II. · COMPARISON AND ANALYSIS

This section compares and analyzes the discussed malware detection techniques based on seven different criteria as shown in Error! Reference source not found.. These criteria are Malware Detection type, Malware Detection rate, the used Machine Learning classifiers (ML technique), Resources Consumption rate, the Extracted Features to be analyzed, the ability of the mechanism to detect malware which communicate in encrypted format, and lastly, the rate of detecting unknown malware that were not part of its training set.

The following is the analysis results for each criterion:

- Detected Malware Type

As shown in Error! Reference source not found., [2, 3] proposed a solution that is capable of detecting botnets. On the other hand, [4] proposed a solution to detect different type of malware which is self-updating. As for [1, 6, 7], they proposed a solution that deals with general malware types and not specific for a certain malware. The detection scope of the security solution considered a vital element that affects the idealness of such solution.

- Detection rate

As shown in Error! Reference source not found., most of the solutions achieved high detection rates which is not less than 90%. Although the majority of the proposed solutions achieved high rates, the classifier detection ability depends mainly on the training. If the classifier trained on specific applications and tested on applications that have similar behavior, the detection rate will be high. On the other hand, if the classifier tested with applications that behave differently, the detection rate will be

decreased. Thus, the detection rate criterion cannot be a definite factor to identify the ideal solution since each solution has its own dataset with certain samples. It needs to be combined with other factors to get a clear result.

- Machine learning technique

As shown in Error! Reference source not found., each proposed solution uses different machine learnings. Each solution uses an appropriate classifier for it. Therefore, researchers won't use this criterion for the best proposed solution decision.

- Extracted traffic features

This criterion is various from one solution to the other since it depends upon the proposed technique and the type of malware that the detection technique is designed to detect it. For example, the botnet detection techniques are depended upon the TCP packets for the connection between the attacker and zombie, in contrast, self-updating detection technique depends on the applications' behavior, and other detection techniques rely on another aspect. Therefore, each proposed solution uses different features based on the type of malware it detects.

- Resources consumption

An important criterion when comparing between malware detection mechanisms is the resources consumption to not affect the mobile performance. In [2], the consumed resources are low since the mechanism relies on extracting the headers only to be aggregated and analyzed. As for [4], the learning process consumes high CPU. However, since the learning process only takes place when an application is installed or updated, then this consumption rate is considered to be acceptable. Unlike the aforementioned solutions, the proposed mechanisms in [1] and [6] have the lowest rate in resources consumption. This is due to the architecture of the mechanisms themselves where the learning and detection processes are done in a server instead of the mobiles. This leads to lower consumption of resources and a better analysis

of network traffic since the AP will aggregate all the traffic from different mobiles and send them to one server. By that, the detection mechanism will be enhanced.

Table 2 Comparison Table

Criteria	References					
	[2]	[3]	[8]	[1] [5]	[7]	[6]
Detected Malware Type	Botnets	Botnets	Self-updating	General	General	General
Detection rate	98%	99.99%	90-100%	99.15%	90%	91%
ML technique	Random Forest	Random Forest and KNN	Decision table and REPTree	SVM	Decision Tree	Decision Tree
Resource Consumption	Low	Not specified	Low to moderate	Low	Not specified	Low
Extracted Traffic Features	• DNS-based • TCP-based • Origin-destination based • HTTP-based	• Basic • Content-based • Time-based • Connection-based	• Avg. sent/received bytes • Avg. sent/ received data • Inner and outer avg. sent/received interval	• HTTP Request header	• Eight different features in a HTTP traffic	• HTTP request header • TCP flow features
Encrypted Traffic	Detected	Not specified	Detected	Detected	Not detected	Detected
Detection of Unknown apps	High	High	High	moderate	Not specified	Low

- Encrypted traffic

Network traffic may include encrypted packets, such as HTTPs. Such traffic is challenging for some malware detection mechanisms as in [5]. However, there are mechanisms that use advanced techniques to detect malware when analyzing the network traffic even if it was encrypted such as the solutions proposed in [1, 2, 4, 6]. This was achieved by analyzing the TCP flow. Another approach was to create a profile for each application's regular network traffic and use it as a baseline to detect any deviation that may be evidence of malware activity.

- Detection of unknown apps

As shown in Error! Reference source not found., the detection of unknown apps reached high rates in [2, 3, 4, 6] achieved low detection rate. Moreover, [1] achieved moderate detection rate. These measures constitute a vital factor since it represents the classifier ability to detect malware that behaves differently. Thus, [2, 3, 4] are more qualified in term of new behavioral malware detection than the others.

Although some solutions exceeded the others in specific criteria, the best solution measures constitute in a comprehensive model. The security has a clear trade-off to achieve its goal. Taking these into consideration and based on the conducted analysis, TextDriod [1] is considered as the best solution to detect malware based on network traffic features for different reasons. First, it has a high detection rate. Second, it is general for all types of malware. Third, it can detect most of the unknown malware that it did not train on. Fourth, it can detect encrypted malware traffic. Finally, it consumes low resources.

III. CONCLUSION AND FUTURE WORK

Malware that targeted Android platform is rapidly growing which increases the threats to its users. Such malware may steal users' information or launch an attack. Some types of malware cannot be detected by traditional anti-virus software or other detection methods that rely on analyzing malware's code or access rights. As a remedy, network-based detection techniques are introduced. This paper discussed several machine learning detection techniques, two of them focus on detecting botnets malware, one of them focus on detecting self-updating malware, and the rest focus on detecting the general type of malware. After that, the researchers compared and analyzed the discussed techniques based on specified criteria which are Malware Detection type, Malware Detection rate, the used Machine Learning classifiers, Resources Consumption rate, the Extracted Features to be analyzed, the ability of the mechanism to detect malware which communicate in encrypted format, and lastly, the rate of detecting unknown malware that were not part of its training set. The result OF this comparison and analysis showed that TextDriod detection technique is the most powerful one since it detects most of the malware type and it has a high detection accuracy rate which is 99.15% with 0.45% misjudgment.

REFERENCES

[1] M. Conti, C. Zhao, B. Yang, Z. Chen, Q. Yan and S. Wang, "Detecting Android Malware Leveraging Text Semantics of Network Flows," in *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY*, 2018.

[2] S. Garg, S. Peddoju and A. Sarje, "Network-based detection of Android malicious apps," *International Journal of Information Security*, vol. 16, no. 4, pp. 385-400, 2017.

[3] 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.

[4] A. Shabtai, L. Tenenboim-Chekina, D. Mimran, L. Rockach, B. Shapira and Y. Elovici, "Mobile Malware Detection Through Analysis of Deviations in Application Network Behavior," *Elsevier*, vol. 43, pp. 1-18, 2014.

[5] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao and M. Conti, "TextDroid: Semantics-based Detection of Mobile Malware Using Network Flows," in *IEEE Conference on Computer Communications Workshops*, 2017.

[6] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng and Z. Jia, "A Mobile Malware Detection Method Using Behavior Features in Network Traffic," *Elsevier*, vol. 133, pp. 15-25, 2019.

[7] Nancy and D. Sharma, "Android Malware Detection using Decision Trees and Network Traffic," *International Journal of Computer Science and Information Technologies*, 2016.