International Journal of Computer Networking,
Wireless and Mobile Communications (IJCNWMC)
ISSN (P): 2250–1568; ISSN (E): 2278–9448
Vol. 11, Issue1, Jun 2021, 1 – 12

NTDROID: ANDROID MALWARE DETECTION USING NETWORK TRAFFIC FEATURES

DEEPTI SINGH, KAPIL GUPTA, ANKUR SINGH & ANSHUL ARORA

Delhi Technological University, Delhi, India

ABSTRACT

Mobile malware samples damage the system and can lead to sensitive information leakage and huge monetary loss. Withthe increasing popularity of Android smart phones, the number of malware attacks on the Android platform has alsoincreased. In the first quarter of 2020, around 0.5 million new malware samples were detected per month. Keeping theirthreats in mind, in this paper, we propose a novel network traffic-based Android malware detection model namedNTDroid. The model is trained on network traffic files by selecting the best set of features using a mutual-informationbased feature selection algorithm. Further, the model employs machine learning classifiers to detect malicious apps. Theexperimental results demonstrate that the proposed model can efficiently distinguish between normal traffic and malicioustraffic with an accuracy of 99.99%.

KEYWORDS: Mobile Network, Mobile Security, Malware Detection, Intrusion Detection

Received: Apr 07, 2021; **Accepted**: Apr 27, 2021; **Published**: May 07, 2021; **Paper Id.**: IJCNWMCJUN20211

Original Article

1. INTRODUCTION

According to a recent report [1], 48.41% of the world population has access to smart phones now, and it's expected to become 62.21% by the end of 2021. This fact highlights the up surging popularity of smart phones in the world. Smartphone's provide a simple way to connect to the World Wide Web, i.e., the Internet. There are a lot of mobile operating systems in the market like Android, iOS, KaiOS, etc. Out of all these operating systems, Android is the most popular choice, with a market share of 71.9% [2].

The growing number of Smartphone users across the globe has sparked interest in attackers as well. The total number of new malware samples by March 2020 was around 0.5 million per month [3]. Recently a new malware [4] has been identified that disguises itself as a system update and steals sensitive information in the background. This example points to the security vulnerabilities in smart phones and how easy it is for attackers to steal sensitive information. Trojans, ransomware, mobile bots are some common portable malware that can damage the operating system, leak the data or cause huge monetary loss, etc.

1.1. Motivation

Due to the increasing popularity of Android smart phones, it has become a major target for attackers. According to a mobile security firm [5], there are over 29,000 malicious applications for Android in active use as of 2020. Whereas, there were only 14500 malicious applications in 2019. The obvious reason for increasing attacks on Android OS is the high number of users, the open architecture of Android, and the availability of third-party applications. Due to these vulnerabilities and the presence of a large number of malicious applications in the market

we need a highly accurate mechanism to detect such malicious apps. Several detection mechanisms have been proposed in the literature for Android malware detection. Some of the works have designed static defense solutions that aim to analyze static components of apps like manifest file, Java code, etc, and do not involve the execution of apps. Hence, stealthier samples that can download malicious components at update time evade static detection. Therefore, dynamic solutions are preferred over static ones to detect stealthier samples. In dynamic features, system calls and network traffic features have been used in Android malware detection. However, the research has reported that system calls give relatively lower accuracy in malicious apps detection on Android. Hence, in this work, we have aimed to analyze network traffic features for effective Android malware detection.

1.2. Contributions

To address the above-mentioned problems, we propose a novel network traffic-based malware detection system on Android that can detect malicious applications using machine learning classifiers. The key contributions made in this paper are summarized below:

- We captured the network traces of malware and normal dataset using the packet capturing apps such as tpacketCapture.
- After collecting the network traces, we extracted 21 network traffic features from both malware and normal dataset. Further, we applied a mutual information-based feature selection algorithm to select the best set of features, and remove redundant or irrelevant features.
- Thereafter, to identify the malicious network behaviour, we proposed a novel malware detection model based on the selected features using machine learning algorithms.

1.3. Organization

The rest of the paper is structured as follows. We summarize the related works in the field of Android malware detection in Section 2. The detailed methodology of the model *NTDroid* is explained in Section 3. We discuss the results obtained from the proposed approach in Section 4 and conclude with future work directions in Section 5.

2. RELATED WORK

Several studies have been carried out on Android malware detection using machine learning techniques. Because the proposed model aims to detect Android malware based upon dynamic traffic features, hence, we review the similar related works focusing on dynamic Android malware detection. The authors in [6] employed packet classification and counting techniques and detected malware by monitoring traffic to and from a network-connected host. The authors in [7] also proposed a network-based system for the detection of malware that protects users from network attacks. Yujie et al. [8] used traffic analysis and deep learning and introduced a rapid system to detect malware. They first collected the traffic generated by the Android apps and preprocessed it with third-party applications to remove impure traffic data. The authors in [9], [10], [11], and [12] also used network behavior analysis to detect Android malware. The authors in [13] incorporated host-based information along with network features to detect malware. Furthermore, Zhou et al. [14] proposed the construction of a traffic fingerprint that combines ML algorithms and can be used for encrypted traffic. The

authors in [15] suggested Hybrid Malware Detection Approach (HDMA) which takes into account features that display a similar pattern in network traffic. They used ensemble learners along with the XGBoost algorithm to get high accuracy. The authors in [16] proposed a malware detection system on the server-side, which consumed minimum mobile resources and combined ML algorithms with network traffic analysis. Wu et al. [17] applied the Bayesian classifier model for malware detection on network traffic data. Conti et al. [18] considered attacks that do not directly interact with the device but through the network side like Wi-Fi and used advanced ML techniques to detect malware. The authors in [19] proposed Locker-ransomware detection using comprehensive analysis of the transactions as well as ML algorithms. The authors in [20] suggested monitoring the application's interactions and behavior to derive statistical features, which are later fed to ML classifiers for malware detection. The authors in [21] introduced "Mystique-S" for malware detection at runtime and under different user conditions, it automatically selects attack features. Moreover, the authors in [22] used conversation-level features and proposed an Extra-tree classifier that outperforms other classifiers in terms of accuracy. The authors in [23] used an integrated two-layer detection system based primarily on neural networks to detect android malware accurately. Wang et al. [24] also proposed a multi-view neural network for malware detection. The authors in [25] applied NLP on HTTP flow text documents to detect Android malware. The authors in [26] discovered that the application layer traffic is dominated by HTTP and DNS traffic, accounting for more than 99% of all traffic. The malware samples were generated in a real internet environment. Manzano et al. [27] considered 10 ransomware families and 9 features related to time. The authors compared 3 ML algorithms- Random Forest, Decision Tree, and KNN for classifying ransomware. Pang et al. [28] identified four imbalanced algorithms to detect malware by using an imbalanced traffic dataset. The authors in [29] proposed a malware detection system with just 9 traffic feature measurements. The authors in [30] analyzed k-means and mini-batch k-means clustering algorithms to detect malware.

Apart from analyzing network traffic, many authors have used Android OS's dynamic features for malware detection. Ribeiro et al. [31] introduced an IDPS, HIDROID, that analyses CPU usage, memory usage, battery, bandwidth, etc. The model is trained on benign data only and it issues an alert to the end-user whenever it detects an anomaly. The authors in [32] and [33] also proposed a similar model where the analysis is done on different levels. Moreover, Cai et al. [34] proposed "DroitCat", which complements the existing malware detection as well as classification apps with the help of app-level profiling. The authors in [35] and [36] suggested the use of call graphs and API calls for detecting malware. Among various dynamic features for Android malware detection, network traffic gives relatively better accuracy as compared to other features. Hence, in this work, we aim to analyze network traffic for effective Android malware detection.

3. METHODOLOGY

In this section, we discuss the proposed *NTDroid* model to detect Android malware. Our approach can be broadly divided into four phases, as summarized in Figure 1, namely, 1) Data Collection, 2) Data Pre-processing, 3) Classifier Training and 4) Predicting Anomalies. We discuss all the phases in detail in the following subsections.

3.1. Data Collection

Data collection is the first step in building malware detection models. Here we have used tpacketCapture app to capture normal traffic data from trending apps on the play store like WhatsApp, Pinterest, Maps, Uber, etc. For malware traffic

files we used an Android emulator and infected it with known malicious apps. In this way, we collected around 1 million packets each for malware and normal Android traffic in the form of pcap files.

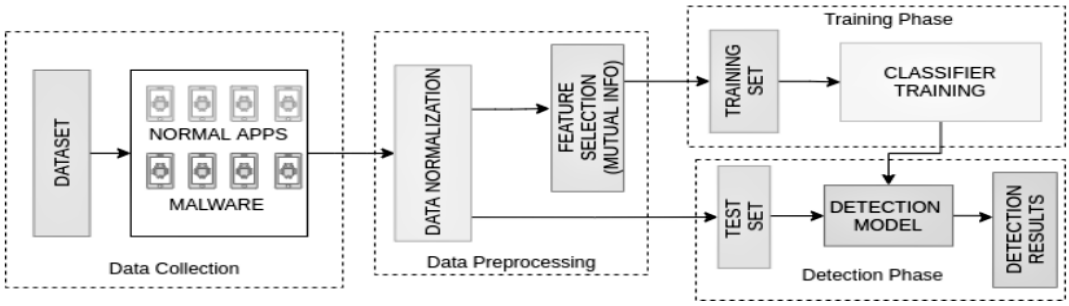

Figure 1: Proposed NTDroid Model System Design

3.2. Data pre-processing

The pcap files collected for malware and benign dataset are processed using Wireshark to extract traffic features as summarized in Table1.

Table 1: List of Extracted Features

Total Packet transferred	Total Bytes transferred	Packets Sent
Bytes sent	Packets received	Bytes received
Rel Start	Duration	Bits sent per second
Bits received per second	Bytes transferred per second	Packets transmitted per second
Packets sent per second	Packets received per second	Average packet size
Average packet size sent	Average packet size Received	Ratio of incoming to outgoing packets
Ratio of incoming to outgoing Bytes	Average time between a packet sent	Average time between a packet received

This phase is further divided into two sub-phases namely, (1) Data Normalization and (2) Feature Selection. We discuss them in detail in the next subsections.

3.2.1. Data Normalization

Normalization is a technique to scale all data on one common scale without losing information about the distribution of the data. Many machine learning algorithms require data to be normalized to give a better prediction. Here we've used the Z-score normalization, the z-score is calculated as

$$z = \frac{x - \mu}{\sigma}$$

Where x is the data point, μ is the mean and σ is the standard deviation of data. So, we calculated the Z score for each of the features.

3.2.2. Feature Selection

We have a set of 21 traffic features in the dataset, but not all features are required to train our detection model. Hence, we use a mutual information-based feature selection algorithm to select relevant and most important features. We use mutual information because it is a good estimator to check the dependency of the features. It is a non-negative value where a higher number indicates a strong dependency of the random variables while zero indicates that the variables are mutually independent. To calculate the mutual information, we have used the entropy-based k nearest neighbour algorithm as proposed by authors in [37]. We then use the following algorithm suggested by Ambusaidi et al. [38] to obtain the best set of features with minimum mutual redundancy. The mutual redundancy between two features f_i and the feature set S can be calculated as shown in Equation 1.

$$MR = \frac{1}{|S|}\sum_{fs}\frac{\mathrm{MI(fi,fs)}}{\mathrm{MI(C,fi)}} \quad (1)$$

Where MI(f_i, f_s) represents the mutual information between f_i and f_s and C is the target class variable here. We calculate GMI scores for all the features that intend to maximize MI(f_i, C) while minimizing the mutual redundancies, MR. We use f_i to indicate features taken from the original feature set, and f_s for features from the selected features subset, S.

$$G_{MI}(f_i) = MI(C,f_i) - MR \quad (2)$$

In Equation (2), if $G_{MI}(f_i)$ is less than zero, then it means that the feature f_i is not providing any relevant information for the target variable C, and therefore we don't add it to our selected feature subset. On the other hand, a positive value of $G_{MI}(f_i)$ implies that feature f_i is important and relevant to C. The complete feature selection and the ranking procedure is described in Algorithm 1. We use this Algorithm to select the relevant features to be used in the Detection phase.

Algorithm 1 Feature Selection algorithm based on Mutual information

Input: Feature set F

Output: Subset of selected features - S

Begin

Step 1. Initialize S as an empty set

Step 2. Calculate mutual information, MI(C, f_i) ∀ f_i ∈ F

Step 3. For F_i in F :

For F_k in F:

Calculate MI(F_i, F_k)

Step 4. Select the feature F_i such that

MI(C;F_i) >= MI(C;F_k) where k = 1,.,n

Add F_i in S

Remove F_i from F

Step 5. While set F is not empty:

For f_i in F:

$$MR = \frac{1}{|S|}\sum_{fs}\frac{MI(fi,fs)}{MI(C,fi)}$$

$$G_{MI}(f_i) = MI(C,f_i) - MR$$

Select f_k such that $G_{MI}(f_k) \geq G_{MI}(f_i)\ \forall\ f_i \in F$

Remove f_k from F

If $G_{MI}(f_k) \geq 0$:

Add f_k to S

Step 6. Sort S based on the values of G_{MI} obtained.

return S

3.3 Training

After feature selection, we obtain a subset of selected features. We aim to find the best set of features that could give better accuracy as compared to any other set of features. However, Algorithm 1 does not convey anything about the optimal number of features that could give better detection accuracy. Hence, to identify the optimal number of features, we start with a top-ranked feature and record the training accuracy. Further, we add the second feature with the top-ranked one and again record the training accuracy. Thereafter, we add the third-ranked feature with the top two features and record the training accuracy. On similar lines, iteration by iteration, we add all 18 features and record their training accuracy. In the end, we identify the best set of features that gives better training accuracy than any other set of features. The data set used for training was kept separate from the testing set, which we have used in the next step.

3.4 Anomaly Detection

In this step, we use the saved trained models from the previous step to predict anomalies. From the testing set, we first selected the optimal features as reported in the previous step, and then we fed this set into our trained classifiers to predict the anomalies. We compare the predicted class labels with actual data and record the testing results for all classifiers.

4. RESULTS AND DISCUSSION

In this section, we discuss the results obtained from the proposed *NTDroid* model. First we highlight the selected features based on their G_{MI} score. Table 2 summarizes the list of 18 selected features.

Table 2: List of Selected Features

Packets received	Ratio incoming to outgoing Bytes	Packets sent per second
Packets sent	Total Packets transferred	Duration
Average time between a packet received	Bytes sent	Bytes sent
Average packet size sent	Average packet size	Rel Start
Average packet size received	Total Bytes transferred	Bits sent per second

Ratio incoming to outgoing packets	Packets received per second	Average time between a packet sent

The proposed model now aims to find the best set of features, out of 18 ranked features, that can give better accuracy. Table 3 summarizes the training accuracies from the proposed model. Note that we start with the top-ranked feature to find the training accuracy and keep on adding the next features in the list. Here, Set-1 represents the accuracies with the top ranked traffic feature. Set-2 represents the accuracies with top two ranked features. On the similar lines, Set-n, for any value of n lying in the set [1,18], represents the accuracies with top n-ranked traffic features.

Table 3: Training Accuracies with Different Classifiers

Set of Features	**Training Accuracy (in %) With Different Classifiers**				
	Ridge	**SGD**	**KNN**	**Decision Trees**	**SVM**
Set-1	49.57	50.14	99.72	99.81	82.38
Set-2	52.07	50.47	99.92	99.95	83.55
Set-3	26.18	84.52	99.70	99.97	98.06
Set-4	33.08	47.89	99.85	99.94	98.82
Set-5	33.23	53.36	99.86	99.94	99.26
Set-6	36.43	71.81	99.83	99.95	99.42
Set-7	36.60	44.73	99.86	99.96	99.54
Set-8	36.40	49.72	99.87	99.96	99.48
Set-9	40.16	49.11	99.87	99.97	98.52
Set-10	40.59	48.26	99.88	99.97	98.46
Set-11	37.84	66.91	99.88	99.98	98.39
Set-12	36.21	63.78	99.87	99.97	98.48
Set-13	36.23	62.76	99.86	**99.99**	98.39
Set-14	35.99	63.20	99.87	99.98	98.48
Set-15	37.21	63.21	99.87	99.98	99.32
Set-16	42.41	69.01	99.82	99.98	99.29
Set-17	42.43	70.94	99.81	99.98	99.17
Set-18	42.25	71.27	99.82	99.98	99.32

As can be seen from the Table 3, KNN, Decision Trees and SVM classifiers give better training accuracy as compared to SGD and Ridge classifiers. We get the best training accuracy of 99.99% with Decision Trees classifier. We get this accuracy on the set of 13 traffic features. Therefore, we use this set of 13 traffic features to detect anomalies in the given test set. The training results with this optimal feature set is summarized in Table 4.

Table 4: Detection Accuracies with different Classifiers

Classifiers	KNN	Decision Trees	SVM
Testing accuracies (in %)	99.8673	99.9902	97.3216

Hence, we conclude that the proposed model detects Android malware with high accuracy of 99.99% on the best set of 13 traffic features.

5. CONCLUSION AND FUTURE WORK

In this paper, we proposed a dynamic model named *NTDroid* that uses network traffic files to detect Android malware. First, we selected the best subset of features and ranked them using a mutual-information based feature selection algorithm. We then incrementally added the ranked features in our machine learning models, and recorded the training accuracies and saved the models giving the highest accuracy. We then used the saved trained models to predict anomalies on the test set and evaluated their performances based on their accuracies. The experimental results demonstrate that the proposed model has a promising performance in detecting Android malware with a detection accuracy of 99.99% with the best set of 13 traffic features. However, not all samples generate network traffic. Hence, our model can be applied only to a subset of samples that generate network traffic. Therefore, in our future work, we will further try to incorporate other features as well such as static permissions so that we are able to detect all kinds of malicious applications.

6. REFERENCES

1. *MARCH 2021 MOBILE USER STATISTICS (Source: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world)*
2. *Sunita Kumawat, Anjali Kumawat & Anil Kumar Sharma, "Intrusion Detection System and Prevention System in Cloud Computing using Snort",International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR), Vol. 5, Issue 6, pp, 31-40*
3. *Mobile Operating System Market Share Worldwide (Source: https://gs.statcounter.com/os-market-share/mobile/worldwide)*
4. *Development of Android malware worldwide 2016-2020 (Source: https://www.statista.com/statistics/680705/global-android-malware-volume)*
5. *Rajshekhar Tiwari & Manish Sharma, "Comparative Analysis of Trust Based and Intrusion Based Black Hole Prevention in AODV in Manet",International Journal of Computer Networking, Wireless and Mobile Communications (IJCNWMC), Vol. 4, Issue 2,pp , 151- 158*
6. *New Android malware with full range of spying capabilities has been found (Source: https://arstechnica.com/gadgets/2021/03/new-android-malware-with-full-range-of-spying-capabilities-has-been-found/)*

7. *Tens of thousands of malicious Android apps flooding user devices By Joel (Source: https://www.techradar.com/news/tens-of-thousands-of-malicious-android-apps-flooding-google-play-store)*
8. *Tariqahmad Sherasiya, Hardik Upadhyay & Hiren B Patel, "A Survey: Intrusion Detection System for Internet of Things",International Journal of Computer Science and Engineering (IJCSE), Vol. 5, Issue 2, pp, 91-98*
9. *L. E. Menten, A. Chen and D. Stiliadis, "Nobot: Embedded malware detection for endpoint devices," in Bell Labs Technical Journal, vol. 16, no. 1, pp. 155-170, June 2011*
10. *V. Khatri and J. Abendroth, "Mobile Guard Demo: Network Based Malware Detection," 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015*
11. *N. S. Chandolikar & V. D. Nandavadekar, "Investigation of Feature Selection and Ensemble Methods for 133 Performance Improvement of Intrusion Attack Classification",International Journal of Computer Science and Engineering (IJCSE), Vol. 2, Issue 3, pp, 131-136*
12. *P. Yujie, N. Weina, Z. Xiaosong, Z. Jie, H. Wu and C. Ruidong, "End-To-End Android Malware Classification Based on Pure Traffic Images," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020*
13. *I. J. Sanz, M. A. Lopez, E. K. Viegas and V. R. Sanches, "A Lightweight Network-based Android Malware Detection System," 2020 IFIP Networking Conference (Networking), Paris, France, 2020, pp. 695-703.*
14. *M. Zaman, T. Siddiqui, M. R. Amin and M. S. Hossain, "Malware detection in Android by network traffic analysis," 2015 International Conference on Networking Systems and Security (NSysS), Dhaka, Bangladesh, 2015*
15. *A. Arora, S. Garg, and S. K. Peddoju, "Malware detection using network traffic analysis in android based mobile devices", IEEE 8th International Conference on Next Generation Mobile Apps, Services and Technologies, 2014.*
16. *A. Arora, and S. K. Peddoju, "Minimizing Network Traffic Features for Android Mobile Malware Detection", 18th ACM International Conference on Distributed Computing and Networking, 2017.*
17. *K. Ariyapala, G. D. Hoang, N. A. Huynh, K. N. Wee and M. Conti, "A Host and Network Based Intrusion Detection for Android Smartphone's," 2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Crans-Montana, Switzerland, 2016*
18. *J. Zhou, W. Niu, X. Zhang, Y. Peng, H. Wu and T. Hu, "Android Malware Classification Approach Based on Host-Level Encrypted Traffic Shaping," 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2020*
19. *S. Rahmat, Q. Niyaz, A. Mathur, W. Sun and A. Y. Javaid, "Network Traffic-Based Hybrid Malware Detection for Smartphone and Traditional Networked Systems," 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 2019*
20. *Shanshan Wang et al., "TrafficAV: An effective and explainable detection of mobile malware behavior using network traffic," 2016 IEEE/ACM 24th International Symposium on Quality of Service (IWQoS), Beijing, China, 2016*
21. *F. Wu, L. Xiao and J. Zhu, "Bayesian Model Updating Method Based Android Malware Detection for IoT Services," 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019*

22. *M. Conti, L. V. Mancini, R. Spolaor and N. V. Verde, "Analyzing Android Encrypted Network Traffic to Identify User Actions," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 1, pp. 114-125, Jan. 2016*
23. *D. Su, J. Liu, X. Wang and W. Wang, "Detecting Android Locker-Ransomware on Chinese Social Networks," in IEEE Access, vol. 7, pp. 20381-20393, 2019*
24. *S. Wei, Gaoxiang Wu, Ziyang Zhou and L. Yang, "Mining network traffic for application category recognition on Android platform," 2015 IEEE International Conference on Progress in Informatics and Computing (PIC), Nanjing, China, 2015*
25. *Y. Xue et al., "Auditing Anti-Malware Tools by Evolving Android Malware and Dynamic Loading Technique," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 7, pp. 1529-1544, July 2017, doi: 10.1109/TIFS.2017.2661723.*
26. *M. K. A. Abuthawabeh and K. W. Mahmoud, "Android Malware Detection and Categorization Based on Conversation-level Network Traffic Features," 2019 International Arab Conference on Information Technology, Al Ain, United Arab Emirates, 2019*
27. *J. Feng, L. Shen, Z. Chen, Y. Wang and H. Li, "A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic," IEEE Access, vol. 8, pp. 125786-125796, 2020.*
28. *S. Wang et al., "Deep and Broad Learning Based Detection of Android Malware via Network Traffic," 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS), Banff, AB, Canada, 2018*
29. *S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao and M. Conti, "Detecting Android Malware Leveraging Text Semantics of Network Flows," in IEEE Transactions on Information Forensics and Security, vol. 13, no. 5, pp. 1096-1109, May 2018*
30. *Z. Chen et al., "A First Look at Android Malware Traffic in First Few Minutes," 2015 IEEE Trustcom/BigDataSE/ISPA, Helsinki, Finland, 2015*
31. *C. Manzano, C. Meneses and P. Leger, "An Empirical Comparison of Supervised Algorithms for Ransomware Identification on Network Traffic," 2020 39th International Conference of the Chilean Computer Science Society (SCCC), Chile, 2020*
32. *Y. Pang et al., "Finding Android Malware Trace from Highly Imbalanced Network Traffic," 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 2017*
33. *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," 2017 15th Annual Conference on Privacy, Security and Trust (PST), Calgary, AB, Canada, 2017*
34. *A. Feizollah, N. B. Anuar, R. Salleh and F. Amalina, "Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis," 2014 International Symposium on Biometrics and Security Technologies (ISBAST), Kuala Lumpur, Malaysia, 2014*
35. *J. Ribeiro, F. B. Saghezchi, G. Mantas, J. Rodriguez and R. A. Abd-Alhameed, "HIDROID: Prototyping a Behavioral Host-Based Intrusion Detection and Prevention System for Android," in IEEE Access, vol. 8, pp. 23154-23168, 2020*
36. *A. Saracino, D. Sgandurra, G. Dini and F. Martinelli, "MADAM: Effective and Efficient Behavior-based Android Malware Detection and Prevention," in IEEE Transactions on Dependable and Secure Computing, vol. 15, no. 1, pp. 83-97, 1 Jan.-Feb. 2018*

37. *S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song and H. Yu, "SAMADroid: A Novel 3-Level Hybrid Malware Detection Model for Android Operating System," in IEEE Access, vol. 6, pp. 4321-4339, 2018*

38. *H. Cai, N. Meng, B. Ryder and D. Yao, "DroidCat: Effective Android Malware Detection and Categorization via App-Level Profiling," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 6, pp. 1455-1470, June 2019, doi: 10.1109/TIFS.2018.2879302.*

39. *H. Zhang, S. Luo, Y. Zhang and L. Pan, "An Efficient Android Malware Detection System Based on Method-Level Behavioral Semantic Analysis," IEEE Access, vol. 7, pp. 69246-69256, 2019.*

40. *V.M. Afonso et al., "Identifying Android malware using dynamically obtained features", Journal of Computer Virology and Hacking Techniques, vol. 11, pp.9-17,2015.*

41. *Kraskov, Alexander & Stögbauer, Harald & Grassberger, Peter. (2004). Estimating Mutual Information. Physical review. E, Statistical, nonlinear, and soft matter physics. 69. 066138. 10.1103/PhysRevE.69.066138.*

42. *M. A. Ambusaidi, X. He, P. Nanda and Z. Tan, "Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm," in IEEE Transactions on Computers, vol. 65, no. 10, pp. 2986-2998, 1 Oct. 2016, doi: 10.1109/TC.2016.2519914.*