Hindawi
Security and Communication Networks
Volume 2021, Article ID 9099476, 11 pages
https://doi.org/10.1155/2021/9099476

Research Article

PICAndro: Packet InspeCtion-Based Android Malware Detection

Vikas Sihag,[1,2] **Gaurav Choudhary**,[3] **Manu Vardhan**,[2] **Pradeep Singh**,[2] **and Jung Taek Seo**[4]

[1]*Sardar Patel University of Police, Security and Criminal Justice, Jodhpur, India*
[2]*National Institute of Technology, Raipur, India*
[3]*DTU Compute, Technical University of Denmark (DTU), Kongens Lyngby, Denmark*
[4]*Department of Computer Engineering, Gachon University, Seongnam, Republic of Korea*

Correspondence should be addressed to Jung Taek Seo; seojt@gachon.ac.kr

Received 14 September 2021; Revised 9 October 2021; Accepted 21 October 2021; Published 8 November 2021

Academic Editor: Zhe-Li Liu

The post-COVID epidemic world has increased dependence on online businesses for day-to-day life transactions over the Internet, especially using the smartphone or handheld devices. This increased dependence has led to new attack surfaces which need to be evaluated by security researchers. The large market share of Android attracts malware authors to launch more sophisticated malware (12000 per day). The need to detect them is becoming crucial. Therefore, in this paper, we propose PICAndro that can enhance the accuracy and the depth of malware detection and categorization using packet inspection of captured network traffic. The identified network interactions are represented as images, which are fed in the CNN engine. It shows improved performance with the accuracy of 99.12% and 98.91% for malware detection and malware class detection, respectively, with high precision.

1. Introduction

Cell phones have become a vital piece of our routine for accessing valuable services as mobile banking, shopping, food, and governance. The data transferred from these apps are sensitive, and many malicious applications are objectified to get such information using different means [1]. Cybercriminals resort to social engineering tools, the most common of these passing a malicious application off as another popular and desirable one. Recently, a popular and attractive name, "Coronavirus," has been used in different ways for malicious purposes, such as package names concealing spyware and banking Trojans, adwares, and droppers [2]. Of course, this was not limited to naming: the pandemic theme was also used in application user interfaces. Mobile malware and adware in particular often come in the form of a gaming or entertainment app that seems harmless, but what users are unaware of is that their device is doing malicious activities in the background [3]. Therefore, mobile malware is on the rise, with attackers shifting their efforts to smartphones and tablets as global mobile markets come under attack. Staying secure means recognizing your risk and understanding common threats by adopting an effective malware detection mechanism [4]. Figure 1 illustrates the rise in research publications in the domain of malware detection and related terms.

The existing malware detection mechanism relies on two methods, dynamic and static. In addition, having a new literature review with machine learning influenced the research studies and explored some technical details in malware detection using machine learning-based techniques. Numerous past works are identified with Android malware detection, yet the vast majority of the past investigations utilize limited features to distinguish malware [6–8]. Each kind of component can address a couple of properties of the applications. Android malware detectors are vulnerable and can be evaded with a low evasion rate. A robust approach is required to establish a durable defense against these adversarial attacks that are too difficult to bypass. The expanding malware threats risk has constrained the Android

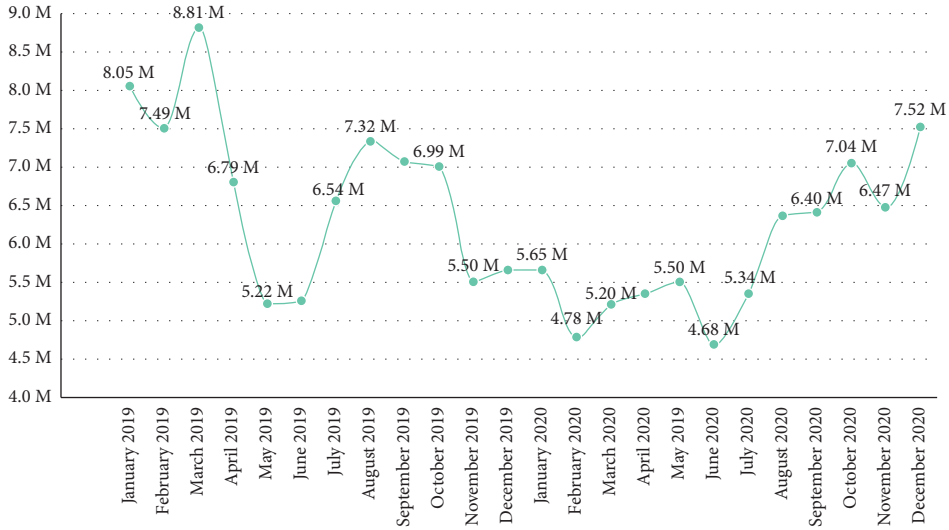

Figure 1: Number of attacks per month on mobile users in 2019 and 2020 [5].

antimalware industry to foster solutions for mitigating malware threats on Android cell phones and other Android gadgets [9, 10].

In this paper, we propose PICAndro (Packet InspeCtion-based Android malware detection) a network interaction-based detection framework. We first generate dynamic analysis network traffic logs for an input APK (an Android executable), followed by conversion of network traffic into network interactions after packet inspection. Furthermore, they are represented as gray-scale images. Images are thus fed into the convolution neural network model for training. Moreover, the model is evaluated against the dataset for malware detection.

Organization of the paper is as follows. Section 2 discusses the related work. In Section 3, we present the proposed framework followed by performance evaluation in Section 4. Furthermore, we discuss issues related with proposed approach and comparison in Section 5, followed with conclusion in Section 6.

2. Related Work

The expanding number of Android malware brings more security issues to mobile users and makes it challenging to identify the malware [11]. Various researchers have been focused on different solutions in Android malware detection.

2.1. Network Traffic-Based Android Malware Detection. To identify and classify Android malware, various solutions have been proposed in the literature. Malik and Kaushal [14] gave CREDROID, a semiautomated Android malware detection approach using network traffic. The authors focused on the DNS server and remote server traffic to identify malware transferring sensitive information. The proposed solution lacks the identification of malware without network interactions. Li et al. [12] proposed a technique to detect malware based on network traffic monitoring and used SVM for feature extractions. The authors focused on the improvement of Android terminal defense ability against malicious attacks. Arora et al. [13] focused on malware detection in Android-based mobiles. The authors used rule-based classifiers in traffic analysis for malware detection.

Zulkifli et al. [16] proposed a dynamic malware detection technique based on decision tree algorithms emphasizing behavioral aspects of the network. The authors used Drebin and Contagio dataset for feature selections. Wang et al. [29] focused on multilayer traffic analysis for malware detection. The authors proposed lightweight malware detection based on the combination of network traffic analysis and machine learning. The proposed approach is applied on the server-side only. Zaman et al. [30] focused on malware detection and proposed a method based on behavioral analysis using syscall tracing. Abuthawabeh and Mahmoud [17] proposed a model for Android malware detection and categorization based on conversation-level network traffic features. However, authors do not include feature extraction at the run time.

2.2. Deep Learning-Based Android Malware Detection. Different methodologies have been proposed in past research works fully intent on identifying Android malware based on deep learning mechanisms. Alzaylaee et al. [21] proposed a deep learning-based malware detection approach for mobile applications based on the dynamic analysis with the help of state entire input generations. The proposed method can be able to detect zero-day Android malware. In this paper, the authors evaluated 31,125 Android applications and 420 static and dynamic features. Wu [31] presented a detailed study on the deep learning-based Android malware detection solutions and classified them as per their techniques. Yuan et al. [19] proposed a deep learning-based malware detector based on rule mining techniques. The authors extracted 192 features from both static and dynamic analysis with the help of the DBN-based deep learning model.

Kim et al. [20] presented a detailed study on multimodal deep learning used for malware detection and proposed a malware detection framework based on static analysis. The

authors provided a flexibility feature that in future more features can be added as per the requirements. Sihag et al. [22] proposed deep learning-based Android malware detection framework using dynamic features. The authors considered dynamic analysis of the logs of Android APK and done processing on features. The proposed approach was tested on 13,533 applications and extract behavioral patterns. Zhang et al. [23] focused on feature selection and processing and proposed an Android malware detection approach based on the text sequence of APPs generated by AndroPyTool. Bayazit et al. [24] proposed a neural network-based Android malware detection mechanism based on IP features selection. The authors used the CICMalDroid2017 dataset for analysis. The IP was converted into integer numbers and subdivided into four numbers.

2.3. Image-Based Android Malware Detection. Darwaish et al. [28] presented an image-based Android malware detection approach that is robust against various adversarial settings. The authors have checked the proposed against two novel attacks. Ding et al. [25] proposed the CNN-deep learning-based static Android malware detection method. The authors used the bytecode file as a binary stream and converted it into the 2D matrix. Mercaldo and Santone [18] focused on the familial classification problem of malwares and evaluated their approach against 50,000 samples. They used mobile applications as a gray-scale image to identify belonging malware facilities.

The static analysis approach can be affected by code obfuscation and code manipulation techniques [32]. Ünver and Bakour [27] proposed a framework for distinguishing between the Android applications as software or malware. Yang and Wen [33] inspected unzipped files from APK files using images patterns with the help of a random forest classifier.

Table 1 provides a comprehensive overview of research work in network, deep learning, and image-based Android malware detection approaches available in the literature.

3. Design of PICAndro

In this section, we discuss the overview and design of the proposed framework.

3.1. Overview. The proposed architectural diagram of the PICAndro framework is illustrated in Figure 2. The objective of the framework is to classify the given Android application executable .APK based on its network behavior. Network behavior of APK is extracted by executing it in an emulated environment. Captured network interactions in the form of packets are inspected to extract network flows and sessions, which are further represented in the form of images. The generated images are fed into convolution neural networks for training the model, which is then evaluated against the test dataset to answer our research questions. The proposed approach consists of below mentioned modules.

3.2. Dynamic Analysis. Two types of approaches, namely, static and dynamic analysis are used to extract application features. Static (code) analysis analyzes an app by scanning its code, whereas dynamic analysis extracts features by executing it. We employ dynamic analysis to record application behavior as it is effective against evasive applications [34]. The first module of the proposed approach involves running sample Android applications on an emulator to log application behavior and capture network traffic [35]. User interactions into the emulator were fed using the Monkey tool. The captured log includes system calls, network traffic, binder calls, and composite behavioral interactions. For our analysis, we focus on network traffic only.

3.3. Image Representation. The captured traffic comprises packets of different sizes and different network interactions. Packet inspection based on different network granularity levels outputs different network interactions. The proposed work uses flow and session as network interactions. It does not consider per packet interactions. A session can be defined as a collection of flows in both directions corresponding to a connection whereas a flow can be defined as packets having the same 5 identifiers, namely, source and destination IP addresses; source and destination port numbers; and protocol. We consider only the first N^2 bytes of a flow/session for representing as an $N \times N$ image for data uniformity. The starting bytes of a flow/session best reflect its characteristics as it contains connection information and few data contents. Each byte of the network interaction is represented as a pixel (e.g., 0 x ff represents a white and 0 x 00 represents a black pixel). Steps involved are defined in Algorithm 1. Figure 3 shows 20×20 image representation of flows in malware samples from different families.

3.4. Convolution Neural Networks. We employ the convolution neural network (CNN)-based deep learning method for image classification. The CNN model is first fed with Network Interaction (NInt) images of size $N \times N$. The first convolution layer (CL_1) performs convolution operation with 32 kernels (of size 3×3). The results of CL_1 ($N \times N \times 32$ output shape) are fed into a 2×2 max-pooling layer MP_1. It is followed by a second convolution layer CL_2 with 64 kernels (of size 3×3) and second 2×2 max-pooling layer MP_2. The last two layers are dense layers (dropout = 0.1). For the output layer, we use sigmoid and softmax functions for binary and multiclass classification, respectively. Rectified linear unit (ReLU) activation function is used for hidden layers as it forwards only the positive part of the argument.

3.5. Classification Model. The representation of network interaction behavior enables us to detect misbehavior by samples effectively. The above discussed CNN model is trained on the dataset, which is then used for classification and detection. The classification results are then used for performance evaluation.

Table 1: A comparison of Android malware detection approaches based on network features, deep learning, and image-based models.

	Author	Year	Detection features	Technique	Dataset
Network	Li et al. [12]	2014	Network features	SVM	Self collected
	Arora et al. [13]	2014	Traffic statistics	Decision tree	Android MalGenome
	Malik and Kaushal [14]	2016	DNS queries	WoT matching	Android MalGenome
	Wang et al. [15]	2017	URL text semantics	SVM	Self collected
	Zulkifli et al. [16]	2018	Traffic statistics	Decision tree	Android MalGenome
	Abuthawabeh and Mahmoud [17]	2019	Conversation level	ExtraTree classifier	CICAndMal2017
	Sanz et al. [18]	2020	TCP/IP header	Random forest	Self collected
Deep learning	Yuan et al. [19]	2016	Advertising, API, intent, network, permission	Deep belief networks	Contagio, MalGenome
	Kim et al. [20]	2018	Opcode, API, library, permission, components	Multimodal deep learning	VirusShare, MalGenome
	Alzaylaee et al. [21]	2020	Permission, events, and application attributes	MLP	McAfee labs
	Sihag et al. [22]	2021	System calls, binder call	Neural network	MalDroid2020
	Zhang et al. [23]	2021	Text sequencing	CNN	Contagio, MalGenome
	Bayazit et al. [24]	2021	IP address	NN	CICAndMal2017
Image	Ding et al. [25]	2020	Byte code	CNN	Drebin
	Mercaldo and Santone [26]	2020	APK raw	Neural network	AMD dataset
	Ünver and Bakour [27]	2020	Binary bitstream	Machine learning	Drebin MalGenome, AMD
	Darwaish et al. [28]	2021	Permissions, intents, components, API	CNN	AndroZoo

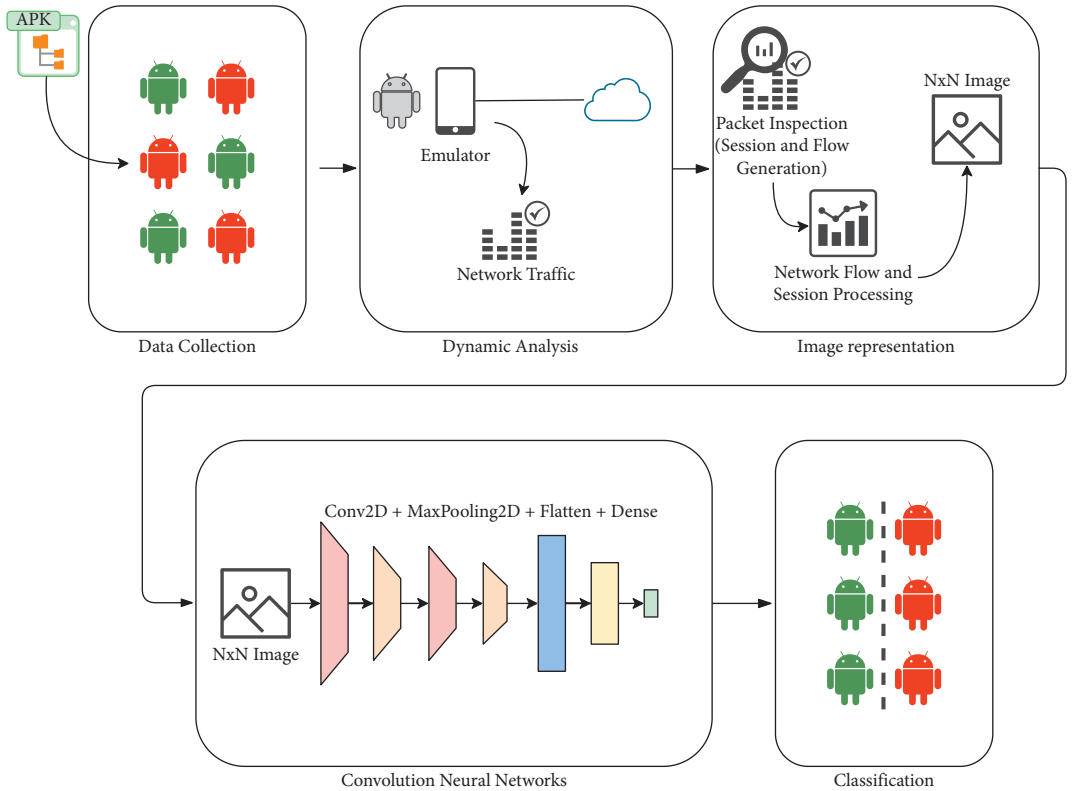

Figure 2: Proposed framework architecture of PICAndro.

Input: RawTraffic.pcap
Output: Gray-ccale images
Extract *NInt* from *RawTraffic.pcap*
foreach *NInt* **do**
  if *NInt is empty* **then**
    continue
  end
  if *NInt already exists* **then**
    continue
  end
  if *NInt size* $\geq N^2$ *bytes* **then**
    consider first N^2 bytes
  end
  if *NInt size* $< N^2$ *bytes* **then**
    pad 0 x 00's till size = N^2 bytes
  end
  Generate $N \times N$ size gray-scale image
end

Algorithm 1: Image Generation.

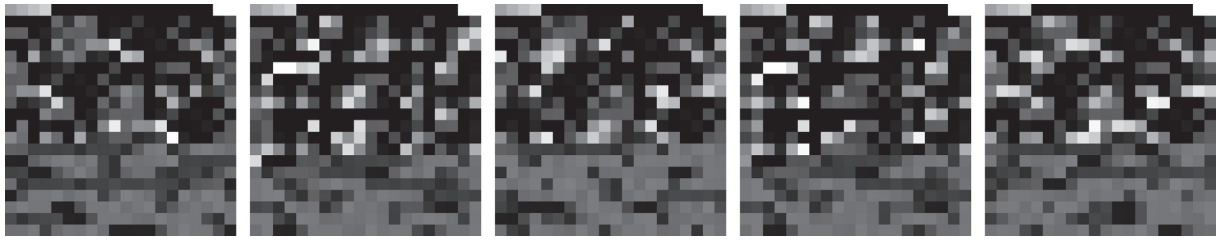

Figure 3: 20×20 gray-scale image representation of malware BankTrojanSVPeng, FakeInstal downloader, SMSreg, RiskwareShedun, and artemis malware samples.

4. Performance Evaluation

In this section, we first introduce datasets and evaluation parameters. It follows with the evaluation of our proposed approach against the following research questions:

RQ1. Can PICAndro detect malware samples with high accuracy?

RQ2. Can PICAndro effectively classify malware samples into their classes?

RQ3. Which network interaction among flow and session is better for network traffic-based detection?

RQ4. Which image size is most effective for representing network interactions?

4.1. Dataset and Evaluation Metrics. To answer the listed RQs, we evaluate the performance and efficiency of PICAndro against a dataset. A dataset of Android application consisting of 13533 samples was collected from different sources (Benign samples = 2621 and malicious samples = 11712) [36–39]. The dataset comprises different Malware types and Benign samples. Samples were categorized for 2-class (Malware and Benign) and 5-class (Adware, Banking, Benign, Riskware, and SMS) scenarios. Samples were analyzed using dynamic analysis, and network traffic was recorded. Of the initially collected APK samples, samples that did not execute during dynamic analysis or generate network traffic were not considered further. The captured traffic was inspected to identify network interactions (flows and sessions). Table 2 describes the successful sample APKs in each category of the dataset, generated flow, and session statistics. It was observed that Riskware category generated most number of interactions (#Sessions/#APKs and #Flows/#APKs), with 34.6 sessions per sample and 61.1 flows per sample. SMS category generated the least network interaction around 2.6 sessions and 5.2 flows per sample. Figure 4 represents the filesize distribution of sample APK files among categories of the dataset. Figure 5 illustrates the number of packets per session and flow for the captured traffic. Each network interaction is then represented as an image of $N \times N$ size. For experimental purpose, we have considered multiple image size 400 (20×20), 625 (25×25), 784 (28×28), and 900 (30×30).

Parameters listed in Table 3 are considered to evaluate the PICAndro framework.

4.1.1. RQ1: Can PICAndro Detect Malware Samples with High Accuracy? The problem of malware detection deals with identifying malicious network interactions from the

Table 2: Description of dataset.

Classification type	Class/category	# APKs	Avg. APK size (bytes)	# Sessions	Avg. # packets per session	# Flows	Avg. # packets per flow
5 class	Adware	1383	196046	23566	16.65	46435	8.47
	Banking	2206	79575	30066	11.22	54000	6.27
	Benign	2155	467907	41145	33.75	81149	17.13
	Riskware	3299	173814	114150	10.46	201570	6.04
	SMS	4013	54275	10559	25.08	20763	12.76
2 class	Benign	2155	467907	41145	33.75	81149	17.13
	Malware	10901	113567	178341	12.28	322768	6.86

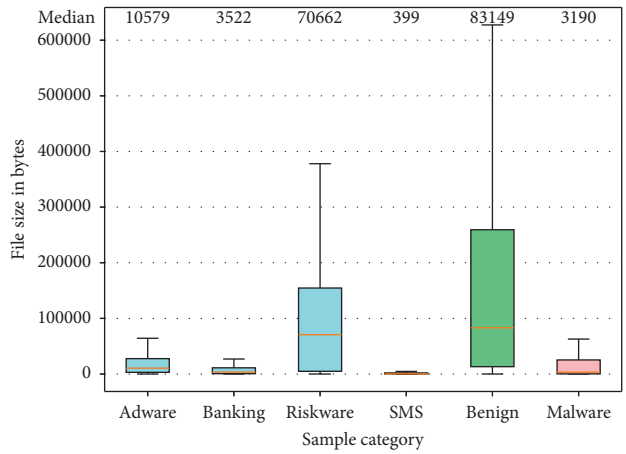

Figure 4: Boxplot representing filesize (bytes) distribution of samples in the dataset among different categories. *Note*. Median value for each category is mentioned on the top.

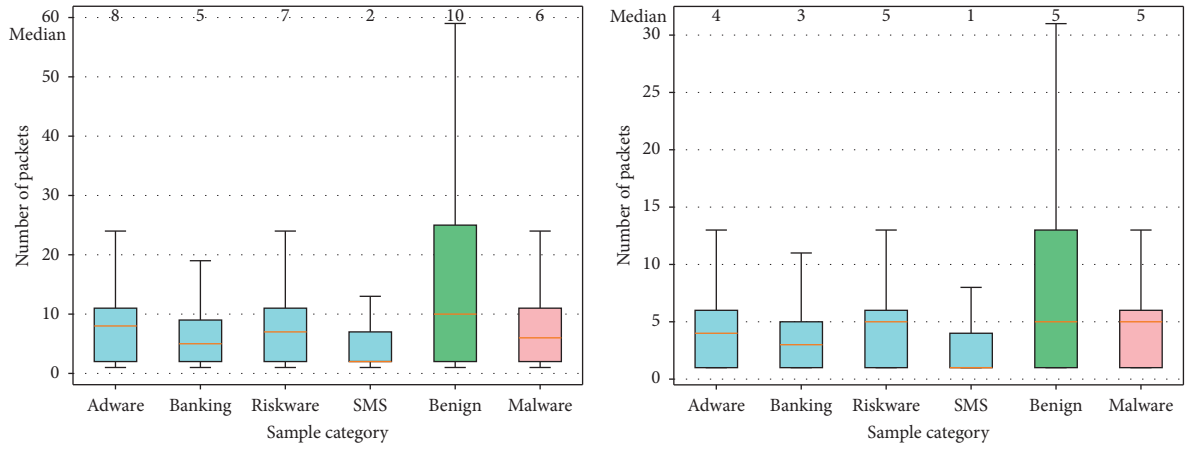

Figure 5: Boxplot of packet count for sessions and flows among sample categories: (a) boxplot for session distribution; (b) boxplot for flow distribution.

Table 3: Malware detection and classification evaluation metrics.

Term	Abbreviation	Definition
True positive	TP	No. of samples correctly classified into class C
True negative	TN	No. of samples correctly not classified into class C
False positive	FP	No. of samples incorrectly classified into class C
False negative	FN	No. of samples incorrectly not classified into class C
Precision	p	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FP})$
Recall	r	$\mathrm{TP}/(\mathrm{TP}+\mathrm{FN})$ also known as sensitivity
F-measure	F_1	$2rp/(r+p)$
Accuracy	Acc.	Percentage of samples correctly classified

dataset. From the dataset, we created a binary classification scenario for both flow- and session-based network interactions. For flow-based binary classification, 81149 Benign and 322768 malicious flows were generated. For session-based binary classification, 41145 Benign and 178341 malicious sessions were generated. Table 4 shows the results of PICAndro against Precision, Recall, F-measure, and Accuracy parameters. Following conclusions are drawn from it:

(i) For binary classification, both flow- and session-based scenarios perform satisfactorily on dataset with accuracy (greater than 99%)

(ii) For 2-class classification, all scenarios with different image sizes perform considerably well with reference to evaluation parameters

RQ1 answer: PICAndro can effectively detect malware samples with high accuracy.

4.1.2. RQ2: Can PICAndro Effectively Classify Malware Samples into Their Classes? The problem of classifying malicious samples into respective malware classes is popularly known as malware type detection/classification. For performance evaluation of PICAndro, we considered a 5-class classification scenario for both flow- and session-based network interactions. Classes considered were Adware, Banking, Benign, Riskware, and SMS type. For session-based classification, 23566 Adware, 30066 Banking, 41145 Benign, 114150 Riskware, and 10559 SMS class unique sessions were generated. For flow-based classification, 46435 Adware, 54000 Banking, 81149 Benign, 201570 Riskware, and 20763 SMS class unique flows were generated. Table 4 shows the results of PICAndro against Precision, Recall, F-measure, and Accuracy parameters for RQ2. Following conclusions are drawn from it:

(i) Proposed work performs satisfactorily on dataset with accuracy greater than 98.5% and *F*-measure greater than 98%

(ii) For 5-class classification, all scenarios with different image sizes perform considerably well with reference to evaluation parameters

RQ2 answer: PICAndro can effectively classify malicious samples into their class/type with high accuracy and *F*-measure.

4.1.3. RQ3: Which Network Interaction among Flow and Session Is Better for Network Traffic-Based Detection? In the proposed work, we study the effectiveness of network interactions for network traffic classification. Multiple works on packet-based classification (malware and intrusion detection) exist in the literature. We try to identify which network interaction amongst flow and session does better network representation. For both binary and 5-class classification, flow-based detection outperforms the session-based approach. For each image representation of size 20×20, 25×25, 28×28, and 30×30, the flow-based approach shows better performance in terms of Precision, Recall, *F*-measure, and Accuracy. Only for a single scenario in 5-class classification with the image size of 25×25, session-based network interaction shows slight improvement over the flow-based one. Accuracy curve for training and test dataset for best results in each scenario is shown in Figure 6. Following conclusions can be drawn from it:

(i) For binary classification, flow-based detection (99.12% accuracy and 97.76% *F*-measure) outperforms session-based (99.09% accuracy and 97.57% *F*-measure) approach

(ii) For 5-class classification, flow-based detection (98.91% accuracy and 98.49% *F*-measure) outperforms session-based (98.56% accuracy and 98.05% *F*-measure) approach

RQ3 answer: flow network interaction is better for network traffic-based detection.

4.1.4. RQ4: Which Image Size Is Most Effective for Representing Network Interactions? In the proposed work, we study the effectiveness of network representation in the form of $N \times N$ images. Multiple works exist on image-based malware detection approaches, where code segments are represented as images. We try to identify which image size does better network representation. Figure 7 illustrates the confusion matrix from 5-class classification of dataset based on flow-based network interactions represented as 20×20 images. Following conclusions can be drawn from it:

(i) For 5-class classification, 20×20 image representation outperforms other image sizes.

(ii) For 2-class classification, 20×20 image representation outperforms other image sizes. The 28×28 image representation during flow-based scenario also performs equally well on one instance.

RQ4 answer: image size 20×20 is most effective for representing network interactions.

5. Discussion

In this section, we compare our proposed system against state-of-the-art Android malware detection systems using network traffic. The efficiency and performance of the proposed solution are compared with those of previous studies in Table 5. It lists features employed solving Android malware detection problems using network traffic, furthermore the dataset used, techniques, and performance. The previous evaluation demonstrates the efficacy of our method in detecting recent malware using their network traffic.

Our proposed approach suffers from few limitations. Dynamic analysis is being used to execute the sample APK in the emulator. As dynamic analysis suffers from code coverage issues, random events in the emulator were generated using the Monkey tool to explore components of each

Table 4: Comparative analysis of 2-class and 5-class classification by PICAndro using flow-based and session-based network granularity.

Classification type	Network granularity	Image size	Precision	Recall	*F*-measure	Accuracy
2 class	Flow	20 × 20	**98.97**	96.59	**97.76**	**99.12**
		25 × 25	96.49	94.95	95.71	98.29
		28 × 28	98.54	**97.00**	**97.76**	99.10
		30 × 30	98.33	96.85	97.58	99.04
	Session	20 × 20	**98.41**	**96.75**	**97.57**	**99.09**
		25 × 25	95.80	94.20	94.99	98.10
		28 × 28	94.49	93.62	94.05	97.79
		30 × 30	95.54	95.64	95.59	98.33
5 class	Flow	20 × 20	**98.51**	**98.46**	**98.49**	98.91
		25 × 25	97.07	96.74	96.88	97.74
		28 × 28	97.61	97.58	97.59	98.28
		30 × 30	97.99	97.86	97.92	98.43
	Session	20 × 20	**98.17**	**97.94**	**98.05**	**98.56**
		25 × 25	97.14	97.07	97.10	98.01
		28 × 28	97.36	97.19	97.27	98.09
		30 × 30	97.45	97.21	97.33	98.15

Bold value highlights best performance results.

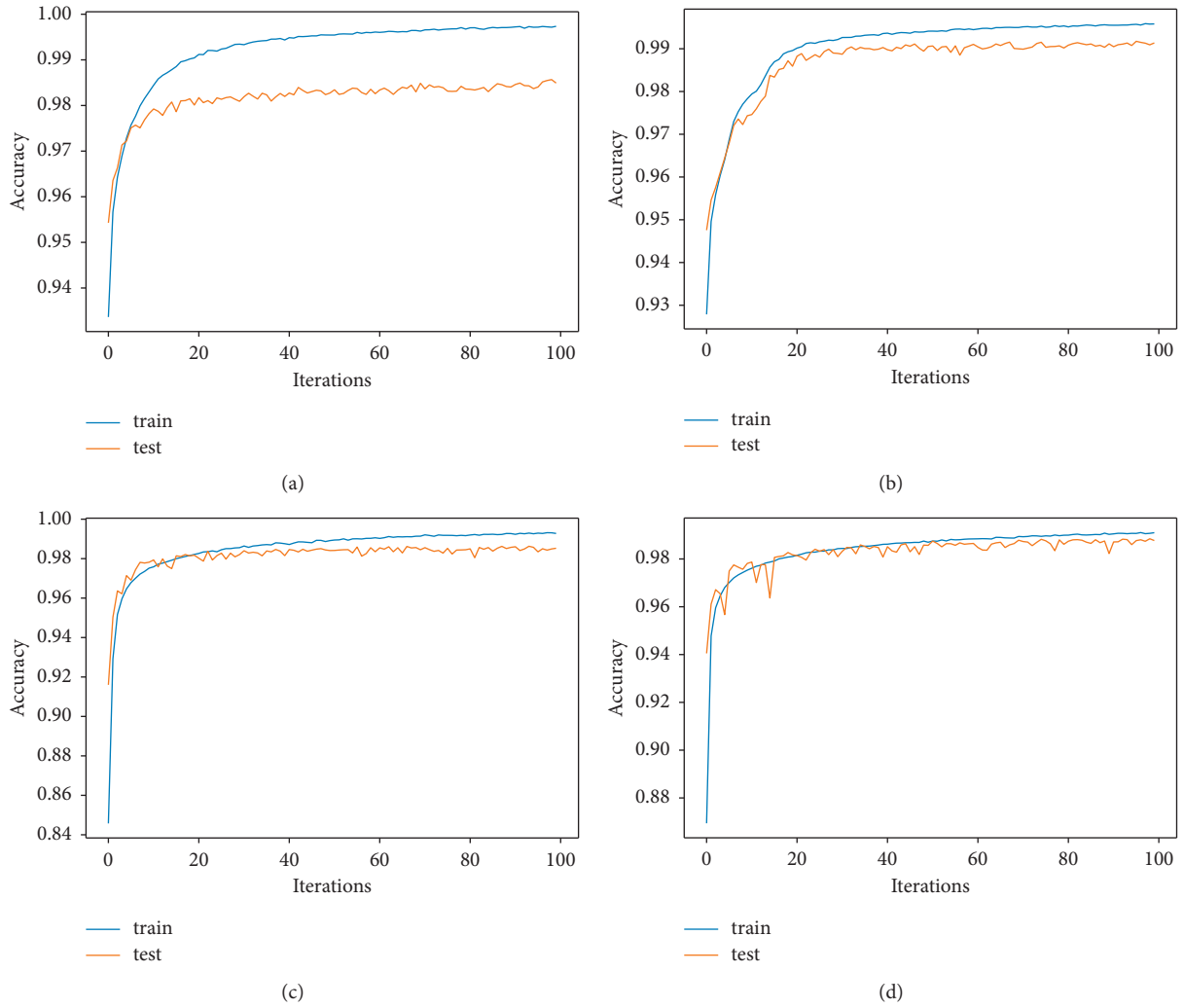

Figure 6: Accuracy graph of PICAndro over 100 iterations for training and test samples: (a) session-based 2-class classification; (b) flow-based 2-class classification; (c) session-based 5-class classification; (d) flow-based 5-class classification.

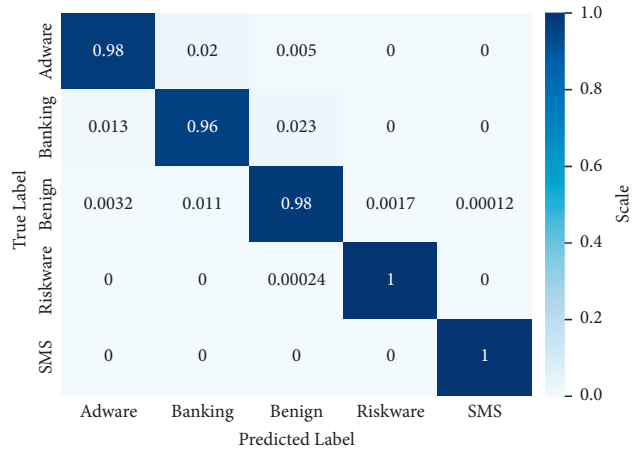

Figure 7: Confusion matrix obtained from 5-class classification of dataset based on network flow features represented by 20×20 images.

Table 5: Comparison of the proposed work with state-of-the-art network traffic-based malware detection solutions in Android.

Author	Year	Network features	Dataset	Technique	Accuracy
Arora et al. [13]	2014	Traffic statistics	Android MalGenome	Decision tree	93.75
Malik and Kaushal [14]	2016	DNS queries	Android MalGenome	Web of trust matching	—
Wang et al. [15]	2017	URL text semantics	Self collected	SVM	99.15
Zulkifli et al. [16]	2018	Traffic statistics	Drebin, Contagio	Decision tree	98.4
Abuthawabeh and Mahmoud [17]	2019	Conversation level	CICAndMal2017	ExtraTree classifier	87.75
Sanz et al. [18]	2020	TCP/IP header	Self collected	Random forest	90
Proposed Work	2021	Flow and session	CICMalDroid2020	CNN	99.12

activity. This increases the probability of triggering malicious behavior. However, it is possible that some of the malicious code segments were not triggered. A stateful input generator for emulation can be explored in future to gain advanced code coverage and real-world traffic.

6. Conclusion

Malware is an increasing threat to smartphone users. Antivirus scanners are evaded by ever-evolving malware with hardening methods. We introduce Android malware detection methods using network interactions (flows and sessions generated by packet inspection) represented as images. Evaluation of the proposed approach demonstrates its potential as it outperforms existing approaches and identifies malicious interactions with few false alarms. It shows improved performance with the accuracy of 99.12% and 98.91% for malware detection and malware class detection, respectively. In the future, the PICAndro framework can be extended to include other static and dynamic features (for example, system call, API, permissions, and network statistical information) than network features alone. Visual image analysis for malware families can also be explored for identification.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1A2C101218712) and was supported by the Nuclear Safety Research Program through the Korea Foundation of Nuclear Safety (KoFONS) using the financial resource granted by the Nuclear Safety and Security Commission (NSSC) of the Republic of Korea (No. 2101058).

References

[1] I. Kholod, A. Shorov, and S. Gorlatch, "Efficient distribution and processing of data for parallelizing data mining in mobile clouds," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 1, pp. 2–17, 2020.

[2] B. Schacht and P. Kieseberg, "An analysis of 5 million openpgp keys," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 3, pp. 107–140, 2020.

[3] D. Caputo, L. Verderame, A. Ranieri, A. Merlo, and L. Caviglione, "Fine-hearing google home: why silence will not protect your privacy," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 1, pp. 35–53, 2020.

[4] N. C. Thang and M. Park, "Detecting malicious middleboxes in service function chaining," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 2, pp. 82–90, 2020.

[5] V. Chebyshev, "Mobile malware evolution 2020," 2021, https://securelist.com/mobile-malware-evolution-2020/101029/.

[6] R. Feng, S. Chen, X. Xie, G. Meng, S. W. Lin, and Y. Liu, "A performance-sensitive malware detection system using deep learning on mobile devices," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 1563–1578, 2020.

[7] V. Sihag, A. Mitharwal, M. Vardhan, and P. Singh, "Opcode n-gram based malware classification in android," in *Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, pp. 645–650, IEEE, London, UK, July 2020.

[8] S. Talegaon and R. Krishnan, "Administrative models for role based access control in android," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 3, pp. 31–46, 2020.

[9] A. L. Marra, F. Martinelli, F. Mercaldo, A. Saracino, and M. Sheikhalishahi, "D-bridemaid: a distributed framework for collaborative and dynamic analysis of android malware," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 3, pp. 1–28, 2020.

[10] M. S. M. Pozi and M. H. Omar, "A kernel density estimation method to generate synthetic shifted datasets in privacy-preserving task," *Journal of Internet Services and Information Security (JISIS)*, vol. 10, no. 4, pp. 70–89, 2020.

[11] V. Sihag, M. Vardhan, and P. Singh, "Blade: robust malware detection against obfuscation in android," *Forensic Science International: Digital Investigation*, vol. 38, Article ID 301176, 2021.

[12] J. Li, L. Zhai, X. Zhang, and D. Quan, "Research of android malware detection based on network traffic monitoring," in *Proceedings of the 2014 9th IEEE Conference on Industrial Electronics and Applications*, pp. 1739–1744, IEEE, Hangzhou, China, June 2014.

[13] A. Arora, S. Garg, and S. K. Peddoju, "Malware detection using network traffic analysis in android based mobile devices," in *Proceedings of the 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies*, pp. 66–71, IEEE, NW Washington, DC, USA, September 2014.

[14] J. Malik and R. Kaushal, "Credroid: android malware detection by network traffic analysis," in *Proceedings of the 1st acm workshop on privacy-aware mobile computing*, pp. 28–36, New York, NY, USA, July 2016.

[15] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting android malware leveraging text semantics of network flows," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 5, pp. 1096–1109, 2017.

[16] A. Zulkifli, I. R. A. Hamid, W. M. Shah, and Z. Abdullah, "Android malware detection based on network traffic using decision tree algorithm," *Advances in Intelligent Systems and Computing*, Springer, in *Proceedings of the International Conference on Soft Computing and Data Mining*, pp. 485–494, January 2018.

[17] M. K. A. Abuthawabeh and K. W. Mahmoud, "Android malware detection and categorization based on conversation-level network traffic features," in *Proceedings of the 2019 International Arab Conference on Information Technology (ACIT)*, pp. 42–47, IEEE, Al Ain, UAE, December 2019.

[18] I. J. Sanz, M. A. Lopez, E. K. Viegas, and V. R. Sanches, "A lightweight network-based android malware detection system," in *Proceedings of the 2020 IFIP Networking Conference (Networking)*, pp. 695–703, IEEE, USA, June 2020.

[19] Z. Yuan, Y. Lu, and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 114–123, 2016.

[20] T. Kim, B. Kang, M. Rho, S. Sezer, and E. G. Im, "A multimodal deep learning method for android malware detection using various features," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 3, pp. 773–788, 2018.

[21] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Dl-droid: deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, Article ID 101663, 2020.

[22] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, "De-lady: deep learning based android malware detection using dynamic features," *Journal of Internet Services and Information Security (JISIS)*, vol. 11, no. 2, pp. 34–45, 2021.

[23] N. Zhang, Y.-a. Tan, C. Yang, and Y. Li, "Deep learning feature exploration for android malware detection," *Applied Soft Computing*, vol. 102, Article ID 107069, 2021.

[24] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "Neural network based android malware detection with different ip coding methods," in *Proceedings of the 2021 3rd International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–6, IEEE, Ankara, Turkey, June 2021.

[25] Y. Ding, X. Zhang, J. Hu, and W. Xu, "Android malware detection method based on bytecode image," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–10, 2020.

[26] F. Mercaldo and A. Santone, "Deep learning for image-based mobile malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 16, pp. 1–15, 2020.

[27] H. M. Ünver and K. Bakour, "Android malware detection based on image-based features and machine learning techniques," *SN Applied Sciences*, vol. 2, no. 7, pp. 1–15, 2020.

[28] A. Darwaish, F. Naït-Abdesselam, C. Titouna, and S. Sattar, "Robustness of image-based android malware detection under adversarial attacks," in *Proceedings of the ICC 2021-IEEE International Conference on Communications*, pp. 1–6, IEEE, Xiamen, China, July 2021.

[29] S. Wang, Z. Chen, Q. Yan, B. Yang, L. Peng, and Z. Jia, "A mobile malware detection method using behavior features in network traffic," *Journal of Network and Computer Applications*, vol. 133, pp. 15–25, 2019.

[30] M. Zaman, T. Siddiqui, M. R. Amin, and M. S. Hossain, "Malware detection in android by network traffic analysis," in *Proceedings of the 2015 international conference on networking systems and security (NSysS)*, pp. 1–5, IEEE, Dhaka, Bangladesh, January 2015.

[31] H. Wu, "A systematical study for deep learning based android malware detection," in *Proceedings of the 2020 9th international conference on software and computer applications*, pp. 177–182, 2020.

[32] C. Johnson, B. Khadka, R. B. Basnet, and T. Doleck, "Towards detecting and classifying malicious urls using deep learning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 11, no. 4, pp. 31–48, 2020.

[33] M. Yang and Q. Wen, "Detecting android malware by applying classification techniques on images patterns," in *Proceedings of the 2017 IEEE 2nd International Conference on*

Cloud Computing and Big Data Analysis (ICCCBDA),, pp. 344–347, IEEE, New York, NY, USA, February 2017.

[34] V. Sihag, M. Vardhan, and P. Singh, "A survey of android application and malware hardening," *Computer Science Review*, vol. 39, Article ID 100365, 2021.

[35] K. Tam, S. J. Khan, A. Fattori, and L. Cavallaro, "Copperdroid: automatic reconstruction of android malware behaviors," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, California, USA, March 2015.

[36] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "An empirical analysis of android banking malware," *Protecting Mobile Networks and Devices: Challenges and Solutions*, Vol. 209, CRC Press, , Boca Raton, Florida, 2016.

[37] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *Proceedings of the 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pp. 515–522, IEEE, Calgary, AB, Canada, August 2020.

[38] C. Mobile, "Mobile malware mini dump. EB/OL].[2016-6-12]," 2013.

[39] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou, "Deep ground truth analysis of current android malware," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 252–276, Springer, Berlin, Heidelberg, July 2017.