2021 7th International Conference on Web Research (ICWR)

Android Malware Detection and Classification Based on Network Traffic Using Deep Learning

Mahshid Gohari, Sattar Hashemi*, Lida Abdi
Department of Computer Science and Engineering, School of Electrical and Computer Engineering
Shiraz University
Shiraz, Iran
mahshid_gohari@yahoo.com, s_hashemi@shirazu.ac.ir , l-abdi@cse.shirazu.ac.ir

2021 7th International Conference on Web Research (ICWR) | 978-1-6654-0426-6/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICWR51868.2021.9443025

Abstract— **Users of smartphones in the world has grown significantly, and attacks against these devices have increased. Many protection techniques for android malware detection have been proposed; however, most of them lack the early detection of malware. Hence, there is an intense need before to expand a mechanism to identify malicious programs before utilizing the data. Moreover, achieving high accuracy in detecting Android malware traffic is another critical problem. This research proposes a deep learning framework using network traffic features to detect Android malware. Commonly, machine learning algorithms need data preprocessing, but these preprocessing phases are time-consuming. Deep learning techniques remove the need for data preprocessing, and they perform well on malware detection problems. We extract local features from network flows by using the one-dimensional CNN and employ LSTM to detect the sequential relationship between the considerable features. We utilize a real-world dataset CICAndMal2017 with network traffic features to identify Android malware. Our model achieves the accuracy of 99.79, 98.90%, and 97.29%, respectively, in binary, category, and family classifications scenarios.**

Keywords— Android, Malware, Network Traffic, Deep learning, CNN, LSTM, Traffic Flow.

I. Introduction

Nowadays, smartphones play a central role in people's lives and communications. The number of smartphone users has reached to 3.8 billion in 2021 in the world, and it is forecasted to increase by several hundred million in upcoming years [1]. Smartphones running Android and iOS accounted for just over 96% of smartphones, and about 54% of global mobile traffic belongs to Android smartphones [2]. Despite the impress of technologies, novel attacks are decreasing daily, having the possibility to infract the security of smartphones. Android is the most popular operating system, and Android smartphones are popular among attackers, and attackers are trying to damage sensitive information stored on smartphones. Attackers are spreading malware in Google Play Store Apps and Android markets.

Malware, which is a collective phrase for malicious software, comprise of any program that can cause damage or destruction to computer systems and enter the system without user permission [3]. Researchers use three principal analyses to detect malware attacks: static, dynamic, and hybrid analysis [4]. Static analysis is a quick approach to find properties or bad code blocks in an application without running the applicants. Researchers monitor dynamic behaviors in dynamic analysis, and mobile applications run in an isolated virtual machine and emulator environment. The researchers should analysis datasets on an isolated environment that is challenge, and not feasible for all of them. So, the existing datasets are good choice for dynamic analysis. To avoid the restrictions of static and dynamic analysis, researchers used hybrid analysis that is a combination of both mechanisms. The hybrid analysis, which is a two-step process, first performs static analysis followed by the dynamic analysis. Nowadays, many datasets are available for Android malware, but all of them have limitations and shortages. As a result, the dynamic analysis is difficult for researchers to perform.

Our contributions in detecting malware are as follows:

1) We propose a network traffic-based malware detection based on network traffic features to identify Android malware and identify the best deep learning algorithm to classify and detect malware categories.

2) We use the "CICAndMal2017"[1] dataset with 86 extract network traffic features and compare our model's performance with state-of-the-art and recently proposed algorithms in the literature. We described three scenarios to analyze the CICAndMal2017 dataset: the first scenario is malware binary detection, the second scenario is malware category classification, and the last one is malware family characterization.

3) Our results indicate that the deep learning model based on Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) can effectively identify malware compared to other deep learning and machine learning algorithms.

The remainder of the paper is organized as follows. Section II provides explanations about the available datasets and related works. Section III explains the detail of the proposed deep learning architecture, and Section IV provides the experimental analyses of the proposed framework. Section V concludes the paper by a conclusion part.

II. Background

There are many security mechanisms for Android malware detection. The most popular mechanisms are static analysis, dynamic analysis, and hybrid analysis. However, many analysis methods focus on static, dynamic, or hybrid detection and seldom consider network traffic. Nowadays, all the attackers interact with users' malicious APPs or get

[1] https://www.unb.ca/cic/datasets/andmal2017.html

978-1-6654-0426-6/21/$31.00 ©2021 IEEE

important information using mobile networks. So, we can analyze Android APPs using network traffic. Some of the studies on Android malware detection utilized the network traffic and several recent proposed methods used static analysis, dynamic analysis or both of them. Also, we reviewed studies that are combinations of deep learning methods to analyze network traffic features and static features. We use deep learning for malware detection and focus on network traffic-based detection using extracted flows from PCAP files. Also, we explain several Android malware datasets containing static and dynamic features.

A. Related Work

In 2015, Canfora et al. [5] proposed an Android malware detection framework based on sequences of system calls. They collected malicious samples from CICAndMal2017 and used the sequence to distinguished normal families and malicious families based on permissions. They used machine learning methods and obtained an accuracy of 97%.

In 2017, Lashkari et al. [6] proposed a system to detect malicious files and identify them as specific malware on a mobile device. They check the performance of the traffic classifier by using traffic features. Also, they utilized classification techniques for packet-based and flow-based features to identify malware families. They achieved an accuracy of 91.41% and precision of 91.24% for Decision Tree, Random Forest, K-Nearest Neighbor, Regression and Random Tree.

In 2018, Lashkari et al. [7] proposed a principled approach to produce Android malware datasets using real smartphones. They developed a new dataset, CICAndMal2017, a model based on network traffic features to detect Android malware. They extracted more than 80 features to detect and classify malicious families by analyzing network traffic. They utilized Random Forest (RF), K-Nearest Neighbor (KNN), and Decision Tree (DT) classifiers, which indicated recall of 88% and precision of 85% for binary classification (2 classifiers), the precision of 45% and recall of 48% for category classification (4-classifier), and precision of 27% and recall of 25% for family classification (43-classifier).

In 2019, Taheri et al. [8] used the second part of the CICAndMal2017 dataset that uses API calls as dynamic features and intents and permissions as static features. They also used Random Forest (RF), Decision Tree (DT), and K-Nearest Neighbor (KNN) for classifying malware. They achieved a precision of 95.3% in static-based malware binary classification, precision of 83.3% in dynamic based malware category classification, and precision of 59.7% in dynamic based malware family classification.

In 2019, Chen et al. [9] analyzed the Android malware traffic. They used CICAndMal2017 dataset and extracted 15 features, which are time-related network flow features and packets feature. They employed three machine learning methods Random Forest (RF), Decision Tree (DT), and K-Nearest Neighbor (KNN), to detect Android malware traffic. The experiments indicated precision of 95% in binary classification and precision of 84% in category classification.

In 2019, Abuthawabeh et al. [10] used the ensemble learning technique to select useful features. They utilized the CICAndMal2017 dataset, selected 18 features, and used Decision Tree (DT), Extra Trees and Random Forest (RF) classifiers. The experiments showed that the Extra-trees classifier outperforms other classifiers with an accuracy of 87.75% for binary classification and an accuracy of 79.97% for category classification.

In 2020, Feng et al. [11] presented a two-layer method to identify malware. They used CICAndMal2017 dataset and the first layer is based on static malware detection that is intent, permission, and component information. It merges the fully connected neural network with the static features to detect the malware. They suggested a new method CACNN, which is a combination of convolution neural network and AutoEncoder, by using network traffic features of APPs for malware identification. The accuracy in binary classification is 99%, in category classification is 97%, and in family classification is 70%. They showed that the two-layer method could improve the detection rate of Android malware.

In 2020, Lekssays et al. [12] attempted to detect Android malware in static analysis and used Convolutional Neural Networks (CNN). At first, they transform APK format Android applications into gray-scale images; then, they classified Android applications with the machine learning model. They used their self-made and CICAndMal2017 dataset. This architecture indicated the accuracy of 84.9% in malware detection.

In 2021, Imtiaz et al. [13] presented a novel framework called DeepAMD to identify Android malware using Artificial Neural Network (ANN). They concluded that DeepAMD compared to other approaches performs better than detecting malware on static and dynamic layers in the CICAndMal2017 dataset. For binary classification, their framework indicated an accuracy of 93.4%, for category classification 92.5%, and for family classification 90% on static layer. For category classification their framework indicated the accuracy of 80.3% and 59% for malware family on dynamic layer.

In 2021, Wei et al. [14] proposed a traffic behavior modeling method based on 1D CNN with autoencoder and RNN for malware detection. In order to extract local features from network flows they used one-dimensional CNN. They solved consecutive characteristics of anomalies in network traffic. The detection results on the CICAndMal2017 dataset show that they achieved 98% detection accuracy and recall.

B. Datasets

The malware Genome project [15], which is the first large collection of 1260 Android malware samples from 2010 to 2011, characterizes malware based on the breakdown of their exact behavior, including the activation, installation and payloads. They evaluate the effectiveness of existing mobiles' anti-virus software and focus on the static test by assessing the dataset with four representative ones.

In 2014 [16], the DREBIN dataset was created with 123,453 applications and 5,560 malware instances from 2010 to 2012. DREBIN makes static analysis and collects the features of an application as much as possible. They represent all extracted features as sets of strings, such as permissions, intentions, and API calls. They extract eight sets of strings such as hardware components, App components, requested permissions, intents (Extract from manifest files), restricted API calls, suspicious API calls, network addresses and permissions (extract from disassembled code).

In 2015 [17], authors used AndroTracker dataset and presented an Android malware detection which was based on static analysis. They developed a system that allows fast identification of malware using the certificate serial number information. It checks suspicious behavior, analyzes the suspicious permission requests, and identifies the malicious system commands in the code.

In 2016 [18], SAPIMMDS dataset consists of 906 malware samples composed of 13 malware families and 1776 benign samples presented by the Korea Internet Security Agency (KISA) from March to December 2014. They suggest a doubtful API call patterns analysis approach and a method for function-oriented malware analysis. They focus on extracting samples for malicious functionalities without the need to extract API call patterns. This method compares the predefined suspicious API lists by doubtful API sequences that are extracted from the bytecode.

The Andro-Dumpsys dataset [19] consists of 906 malware instances and 1776 benign instances. Andro-Dumpsys is based on malware information. They classify and detect malware instances into resembling behavior groups utilizing their footprints. They adopt the suspicious API sequence, the serial number of a certificate, permission distribution, intent and the usage of system commands for executing forged files as feature vectors to extract the malware behavior pattern.

The primary goals of the Andro-Profiler dataset [20] are scalability, efficiency and accuracy. Andro-Profiler categorized malware using behavioral profiles which are extracted from system logs. Using a malicious application on the emulator and analyzing the integrated system logs, Andro-profiler generates the integrated system logs and human-readable behavior profiles.

In 2016 [21], the Kharon dataset represented the diversity of malware types as much as possible. They analyzed each malware by reversing their code and extracted their specific behavior by running them on a controlled and monitored real smartphone.

In 2017 [22], authors used smartphones to generate Android network traffic datasets consist of 400 malware samples and 1500 benign samples from 2006 to 2016. To protect cellular infrastructure companies and mobile device users from malicious applications they proposed a system with nine measures of traffic characteristics.

In 2017 [23], the authors generated the AMD dataset, which contains 24,650 samples, 405 malware samples within 71 families from 2010 to 2016. They also developed detailed descriptions of each malware variety's behaviors and included them in their dataset.

In 2018 [24], the authors proposed MalDozer dataset that consists of 33000 malware samples and 38000 benign samples. They proposed a framework for automatically detecting malware and family documents on Android, which relies on sequence classification using deep learning techniques. It can be deployed as a malware detection system everywhere on servers, mobile, and even IoT devices.

This paper uses the Canadian Institute for Cybersecurity (CIC) dataset, called CICAndMal2017, and it is a real-world dataset. CIC collected more than 10,854 instances (6,500 benign and 4,354 malware) from several sources such as VirusTotal service [25], Contagio security blog [25,26], and previous researchers [27, 22, 28]. They have gathered over six thousand benign apps from Google play market published in 2015, 2016, 2017. [7].

This dataset has continuous and discrete features. The continuous features are network traffic, logs, and API calls and discrete features are battery usage, permissions, network and memory dumps [7]. In this paper, we present the network traffic analysis, which can be used to accurately identify and classify Android malware.

For feature extraction, we apply CICFlowMeter-V3[2] to extract 86 features from PCAP files (more than 30 gigabytes), and utilize PCAP files for our work. CICAndMal2017 dataset has three label levels. In the first level, it is categorized into two label classes: benign and malware. In the second level, malware samples are categorized into four categories: Adware, Ransomware, Scareware, and SMS malware.

- **Adware**: Adware causes the user damage by stealing data and sending them to a remote server by displaying ads in the notification area, or adware may steal the device speaker [29]. Developers can conceal malicious code in advertising programs background. They can do this via advertising notification messages or show advertising banners to click on them [30]. Adware is a software package that automatically provides advertisements to users automatically and by guessing from their previous searches. This involves collecting information, in some cases, stealing crucial personal information for ulterior harmful motives. An ad network uses some adware, but intrusive adware may destroy information from the ad network owners [31].
- **Ransomware**: Ransomware is a type of malware which requests an amount of money from the defiled user. Ransomware encrypts and disables the essential data of users and requires payment to retrieve it. Encrypted data cannot be retrieved because the user does not have the decryption key and only the attacker has it. Ransomware can spread simply by untrusted sites, files, or emails. Crypto and lock-screen are two general categories of ransomware on Android. In lock-screen, the stolen resource is blocked by an image covering the screen. The precious data of users are encrypted by Crypto ransomware. Sometimes, the malicious APKs transcribe the common applications' name and icon, or mask it as a valid file in an SMS or email [32].
- **Scareware**: Scareware is a type of malware that scares clients intending to pay for unwanted applications. Sometimes it looks like an anti-virus or another security program. It warns the user that there is malicious code on their system, and the only way is to buy their software to get rid of it. Attackers can rob the certificates of organizations to damage security [33].

  SMS Malware: SMS malware sends messages without the validation of user [34]. SMS malware

[2] https://www.unb.ca/cic/research/applications.html#CICFlowMeter

involves distributing malware by cyber attackers designed to aim at a victim's mobile device. These attacks plan to make unauthorized calls or send unwanted texts without the user's permission [35].

In the third level, each malware sample categorized into 43 family types which are presented in Table I.

III. PROPOSED METHOD

This paper proposes an Android malware detection model based on deep learning, and it is a combination of Convolution Neural Network (CNN) and Long-Short Term Memory (LSTM) architecture. Our CNN-LSTM model can also classify malware by binary, category, and family. CNN-LSTM architecture is designed to predict sequences in spatial input problems such as images or videos. Regarding the sequential nature of our dataset, CNN-LSTM architecture is the best model for detecting malware. Generally, our model consists of feature extraction, flow, and training deep learning classifier. Fig.1 indicates the complete framework of our proposed Android malware detection model.

A. *Feature Extraction*

Feature extraction involves extracting 86 network-flow features from each flow of the dataset by using CICFlowMeter. A flow is defined with five attributes that are Source IP, Destination IP, Source Port, Destination Port and Protocol. These features are labelled and then sent to the supervised deep learning algorithm in order to identify benign and malware traffic samples.

B. *Training and Testing*

The CICAndMal2017 dataset is divided into two disjoint training (80%) and testing (20%) sets. We evaluate our models via two standard deep learning classifiers, namely CNN and the combination of CNN and LSTM based on the following scenarios:

- Scenario A: malware binary that we identify benign applications from malware applications.
- Scenario B: malware category that we classify malware category.
- Scenario C: malware family that we distinguish malware families.

C. *Base Classifier*

In this section, we explain the neural networks architecture that used in the experiments. The suggested architecture has a one-dimensional convolutional part and then a recurrent part, i.e., LSTM. Fig.2 illustrates the overall picture of the proposed architecture. We provided concise description for each layer below:

- Conv1d: To process one-dimensional vector data used a 1D convolutional layer. 1D convolutions slide the kernel on input vector and move the kernel in one dimension.
- LSTM: This solves the exploding gradient problem and is a type of the Recurrent Neural Network (RNN) layer. It crosses the entire chain by using a state cell. Therefore, the information associated with them remains unchanged. The LSTM uses input, output, and forget gates and also can add or delete information to the state cell.

Because of the improved network structure, CNNs are strong models; they can reduce computational complexity, which leads to better efficiency. Also, convolutional filters can extract high-level features. An LSTM layer is added after CNN to support sequence prediction. LSTM does not have any problem like exploding or vanishing gradient compared to other RNN models that can work with long sequences. The benefit of LSTM is to have a memory that reserves data of the previous state [36].

The convolutional part of the architecture consists of four one-dimensional convolutional layers. Each convolution uses a Batch Normalization (BN) layer and it then uses a ReLU activation layer. The Batch Normalization layer is exclusively crucial as it decreases the internal covariance alternation. This speed up training and creates order effect between batches. One recurrent layer is employed with an activation function that is ReLU, after the convolutions. Next

TABLE I. MALWARE CATEGORY AND FAMILY KIND OF CICANDMAL2017 DATASET.

Category	*Family Type*		
Adware	Dowgin	Ewind	Feiwo
	Gooligan	Kemoge	koodous
	Mobidash	Selfmite	Shuanet
	Youmi		
Ransomware	Charger	Jisut	Koler
	LockerPin	Simplocker	Pletor
	PornDroid	RansomBO	Svpeng
	WannaLocker		
Scareware	AndroidDefender	AndroidSpy	AV for Android
	AVpass	FakeApp	FakeApp.AL
	FakeAV	FakeJobOffer	FakeTaoBao
	Penetho	VirusShield	
SMS Malware	BeanBot	Biige	FakeInst
	FakeMart	FakeNotify	Jifake
	Mazarbot	Nandrobox	Plankton
	SMSsniffer	Zsone	

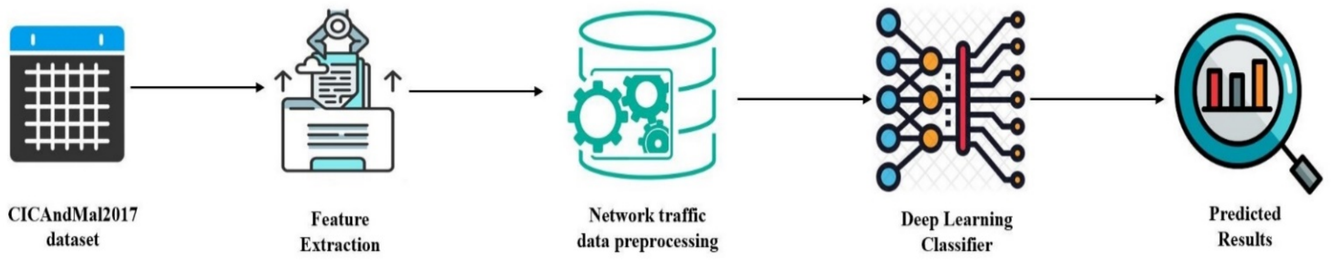

Fig. 1. The overall architecture of the proposed model.

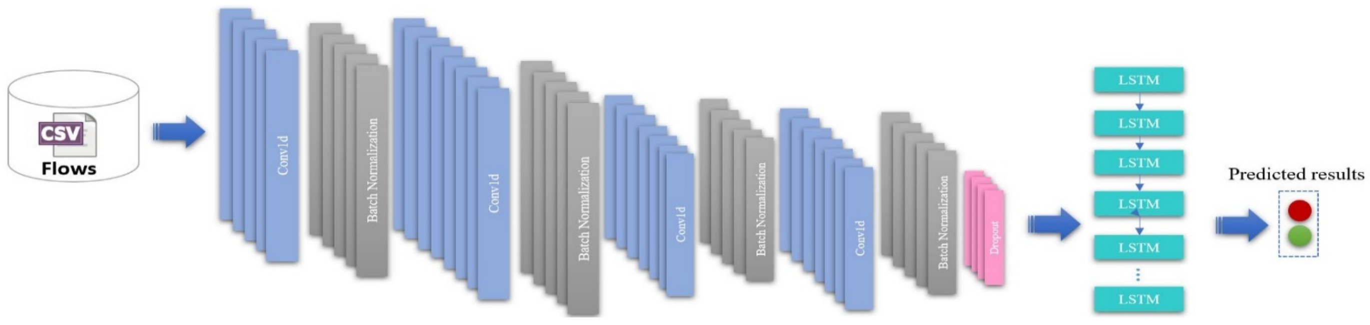

Fig. 2. The architecture of CNN-LSTM.

we utilize three dropout layers to arrange the activations to prevent overfitting, followed by a flatten layer. Finally, for generating the output we exert a dense layer. A sigmoid activation function is used for this layer because the output is a value between 0 and 1. This value indicates the probability of belongingness to class 0 or 1. This value is then rounded to obtain a binary label. This architecture is trained in supervised mode using back-propagation. We used binary cross-entropy and category cross-entropy for loss functions. The combination of CNN and LSTM optimize the efficiency of the model and led to explore data locally. Table II summarizes the parameters used to train the architectures.

IV. Experiments

This section analyzes the results of the experiments. The performance of our model is evaluated over three mentioned scenarios with precision and recall evaluation metrics.

A. *Evaluation Metrics*

We use the precision (P) and recall (R) to assess the performance of the deep learning algorithm. These confusion metrics are contained true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). True positives (TP) and true negatives (TN) are samples that are predicted true as malware or benign. Accordingly, false positives (FP) and false negatives (FN) are the samples that are predicted wrong as malware or benign.

These criteria are as follows:

- True Positives (TP): Predicted case that give the class 1 and in fact is in it.
- False Positives (FP): Predicted case that give the class 1 but in fact is not in it.
- True Negative (TN): Predicted case that give the class 0 and in fact is not in it.
- False Negative (FN): Predicted case that give the class 0 but in fact is in it.

Precision indicates the percentage of all instances predicted as malware traffic that is correctly malware. The formula is calculated in (1).

$$Precision = \frac{TP}{(TP + FP)} \quad (1)$$

Recall shows the percentage of all malware traffic instances that are predicted to be correctly malware. The formula is calculated in (2).

TABLE II. Parameter specification for the architecture employed in the experiments.

Parameter	*Value*
Conv.filters	20
Recurrent unit	128
Dropouts	0.45, 0.35, 0.25
Activation function for all layers	Relu
Activation function at the output layer	Sigmoid
Epochs	10
Batch size	50
Learning optimizer	Adam
Error function	Binary & Category Cross-Entropy

$$Recall = \frac{TP}{(TP + FN)} \quad (2)$$

B. *Experimental Results*

Our results indicate that the proposed architecture performs well in all three scenarios. In Figs 3, 4, and 5, we provide the results of our experiments. The combination of CNN and LSTM has better performance than CNN alone.

In binary classification, our model achieves 99.79%, and CNN achieve 98.25% (Fig.3). Our model indicates 98.90% in category classification, and CNN achieves 88.33% (Fig.4). In family classification, our model achieves 77.85%, and CNN 98.19% (Fig.5).

In the second part of the experiments, we compare the performance of the proposed method with other state-of-the-art methods in literature. It is worth mentioning that these methods, also, used CICAndMal2017 dataset. These results are provided in Tables III, IV, and V.

The studies in [7, 8, 9, 10] used a random forest to compute the precision and recall of the dataset using network traffic features. Some of them used static features and feature selection to achieve better results. They achieved precision of 85.8%, 95.3%, 95%, and 86.65%

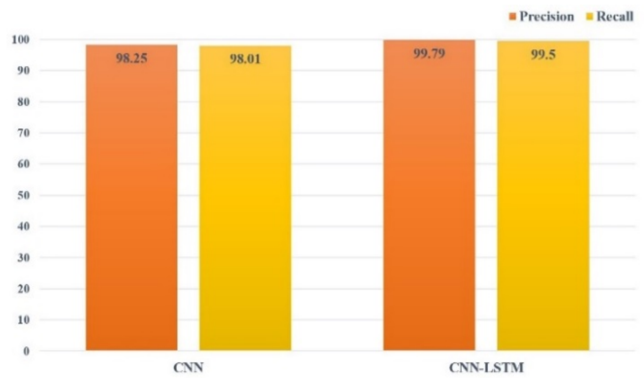

Fig. 3. Precision and recall of the proposed architectures in binary classification scenario.

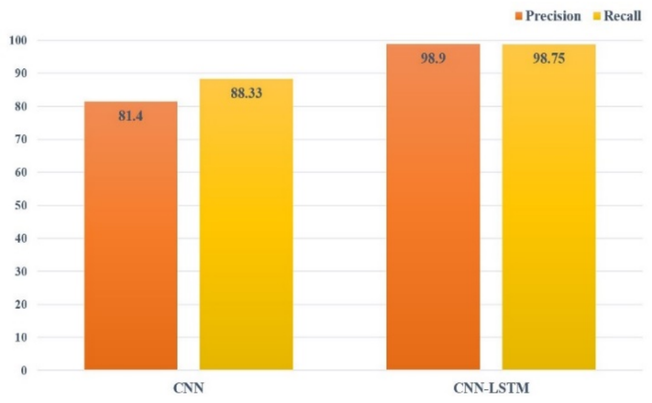

Fig. 4. Precision and recall of the proposed architectures in category classification scenario

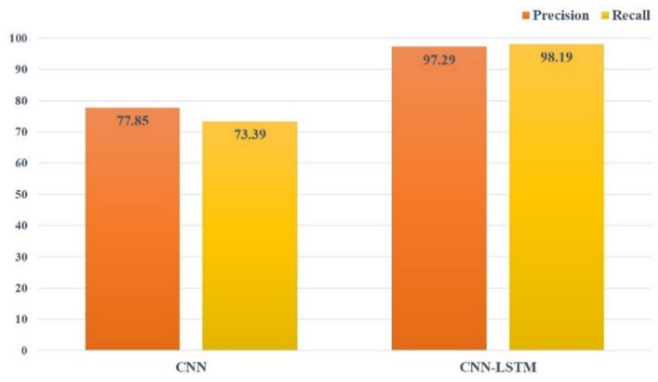

Fig. 5. Precision and recall of the proposed architectures in family classification scenario.

TABLE III. PRECISION AND RECALL OF COMPARATIVE TECHNIQUES IN BINARY CLASSIFICATION SCENARIO.

Research	*Algorithm*	*Precision*	*Recall*
Lashkari [7]	RF	85.8%	88.3%
Taheri [8]	RF	95.3%	95.3%
Chen [9]	RF	95%	95%
Abuthawabeh [10]	RF	86.65%	89%
Feng [11]	CACNN	99.2%	98.2%
Imtiaz [13]	DeepAMD	93.5%	93.4%
Our work	**CNN-LSTM**	**99.79%**	**99.5%**

TABLE IV. PRECISION AND RECALL OF COMPARATIVE TECHNIQUES IN CATEGORY CLASSIFICATION SCENARIO.

Research	*Algorithm*	*Precision*	*Reall*
Lashkari [7]	RF	49.9%	48.5%
Taheri [8]	RF	83.3%	81%
Chen [9]	RF	86%	85%
Abuthawabeh [10]	RF	80.2%	79.64%
Feng [11]	CACNN	98.4%	96.4%
Imtiaz [13]	DeepAMD	82.2%	80.3%
Our work	**CNN-LSTM**	**98.9%**	**98.75%**

TABLE V. PRECISION AND RECALL OF COMPARATIVE TECHNIQUES IN FAMILY CLASSIFICATION SCENARIO.

Research	*Algorithm*	*Precision*	*Recall*
Lashkari [7]	RF	27.5%	25.5%
Taheri [8]	RF	59.7%	61.2%
Feng [11]	CACNN	73.5%	74.2%
Imtiaz [13]	DeepAMD	65%	59%
Our work	**CNN-LSTM**	**97.29%**	**98.19%**

respectively. The researches in [11,13], used deep learning methods to improve previous works that employed machine learning methods. They achieved the precision of 99.2% and 93.5%, respectively. We improve our results in network traffic-based detection using all network traffic features by the deep learning method. We achieve the highest precision of 99.79% and recall of 99.5% as shown in Table III for malware binary classification. This deep learning model with network traffic features indicates far better performance than the random forest or the other DL methods with improvements. In Table IV, CNN-LSTM achieves precision of 98.9% for malware category classification. Other studies in [7, 8, 9, 10, 11, 13] achieved the highest precision of 49.9%, 83.3%, 86%, 80.2%, 98.4%, and 82.2% respectively. In Table V, CNN-LSTM achieves the highest precision of 97.29% for malware family classification. Other studies in [7,8,11,13] achieved the highest precision of 27.5%, 59.7%, 73.5%, and 65%, respectively.

According to the results, our model achieves the highest precision and recall in all the considered scenarios. CNN-LSTM outperforms other deep and machine learning algorithms, significantly and network traffic features are enough for malware detection.

V. CONCLUSION

In this research, we proposed a deep learning model to detect Android malware. We utilized a real-world dataset called CICAndMal2017. For feature extraction, we employed the CICFlowMeter tool and used network traffic features to identify malware Android. Moreover, we evaluated the dataset with two deep learning models: CNN, CNN-LSTM. Finally, we compared our results with other algorithms in the literature. The experimental results indicate that CNN-LSTM has the highest precision on malware binary classification. On the other hand, we compared our results with the previous works that used feature selections to achieve better results and some of them used static features.

We utilized network traffic-based detection and all the network flow features. Our model achieved the precision of 98.9% on malware category classification, and finally, it achieved the precision of 97.29% in malware family classification. Overall, the CNN-LSTM model significantly improves the malware detection rate compared to other deep and machine learning algorithms

REFERENCES

[1] https://www.statista.com/statistics/330695/number-of-smartphone users-worldwide/

[2] https://marketingland.com/report-apple-Android-now-96-percent-smartphones-globally-119487

[3] Vinod P, Jaipur R, Laxmi V, Gaur M. Survey on malware detection methods. InProceedings of the 3rd Hackers' Workshop on computer and internet security (IITKHACK'09) 2009 Mar 17 (pp. 74-79).

[4] Bayazit EC, Sahingoz OK, Dogan B. Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey. In2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) 2020 Jun 26 (pp. 1-8). IEEE

[5] G. Canfora, E. Medvet, F. Mercaldo, C.A. Visaggio, Detecting androidmalware using sequences of system calls, in: Proceedings of the 3rdInternational Workshop on Software Development Lifecycle for Mobile,2015, pp. 13–20.

[6] A.H. Lashkari, A.F.A. Kadir, H. Gonzalez, K.F. Mbah, A.A. Ghorbani, Towardsa network-based framework for android malware detection and characterization,in: 2017 15th Annual Conference on Privacy, Security and Trust(PST), IEEE, 2017, pp. 233–23309.

[7] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark android malware datasets and classification. In 2018 International Carnahan Conference on Security Technology (ICCST), pages 1–7. IEEE, 2018.

[8] L. Taheri, A. F. A. Kadir, and A. H. Lashkari, Extensible Android malware detection and family classi_cation using network-flows and APIcalls," in Proc. Int. Carnahan Conf. Secur. Technol., Oct. 2019, pp. 1_8.

[9] R. Chen, Y. Li, W. Fang, Android malware identification based on traffic analysis, in: International Conference on Artificial Intelligence and Security, Springer, 2019, pp. 293–303.

[10] M.K.A. Abuthawabeh, K.W. Mahmoud, Android malware detection and categorization based on conversation-level network traffic features, in: 2019 International Arab Conference on Information Technology (ACIT), IEEE, 2019, pp. 42–47.

[11] Feng, Jiayin, Limin Shen, Zhen Chen, Yuying Wang, and Hui Li. "A Two-Layer Deep Learning Method for Android Malware Detection Using Network Traffic." IEEE Access 8 (2020): 125786-125796.

[12] Lekssays, Ahmed, Bouchaib Falah, and Sameer Abufardeh. "A Novel Approach for Android Malware Detection and Classification using Convolutional Neural Networks"2020.

[13] Imtiaz, Syed Ibrahim, Saif ur Rehman, Abdul Rehman Javed, Zunera Jalil, Xuan Liu, and Waleed S. Alnumay. "DeepAMD: Detection and identification of Android malware using high-efficient Deep Artificial Neural Network." Future Generation Computer Systems 115: 844-856.

[14] Wei, S., Zhang, Z., Li, S., & Jiang, P. (2021). Calibrating Network Traffic with One-Dimensional Convolutional Neural Network with Autoencoder and Independent Recurrent Neural Network for Mobile Malware Detection. Security and Communication Networks, 2021

[15] Y. Zhou and X. Jiang. Dissecting android malware: Characterization and evolution. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 95–109. IEEE, 2012.

[16] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in Ndss, vol. 14, pp. 23–26, 2014.

[17] H. Kang, J.-w. Jang, A. Mohaisen, and H. K. Kim, "Detecting and classifying android malware using static analysis along with creator information," vol. 11, p. 479174, SAGE Publications Sage UK: London, England, 2015.

[18] J.-w. Jang and H. K. Kim, "Function-oriented mobile malware analysis as first aid," vol. 2016, Hindawi, 2016.

[19] J. wook Jang, H. Kang, J. Woo, A. Mohaisen, and H. K. Kim, "Androdumpsys: Anti-malware system based on the similarity of malware creator and malware centric information," vol. 58, pp. 125 – 138, 2016.

[20] J.-w. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim, "Detecting and classifying method based on similarity matching of android malware behavior with profile," SpringerPlus, vol. 5, no. 1, p. 273, 2016.

[21] E. CIDRE, "Kharon dataset: Android malware under a microscope,"Learning from Authoritative Security Experiment Results, p. 1, 2016.

[22] . 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," in Proceeding of the 15th international conference on privacy, security and trust, 2017.

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

[24] E. B. Karbab, M. Debbabi, A. Derhab, and D. Mouheb. Maldozer: Automatic framework for android malware detection using deep learning. Digital Investigation, 24:S48–S59, 2018.

[25] V. Total, "Virustotal-free online virus, malware and url scanner," Online: https://www.virustotal. com/en, 2012.

[26] V. Total, "Contagio mobile malware mini dump," Online: http://contagiominidump.blogspot.ca/, 2016.

[27] A. F. A. Kadir, N. Stakhanova, and A. A. Ghorbani, "Android botnets:What urls are telling us," in International Conference on Network andSystem Security, pp. 78–91, Springer, 2015.

[28] H. Gonzalez, N. Stakhanova, and A. A. Ghorbani, "Droidkin:Lightweight detection of android apps similarity," in International Conference on Security and Privacy in Communication Systems, pp. 436–453, Springer, 2014.

[29] I. Ideses, A. Neuberger, Adware detection and privacy control in mobile devices, in: 2014 IEEE 28th Convention of Electrical & Electronics Engineers in Israel (IEEEI), IEEE, 2014, pp. 1–5.

[30] M.T. Ahvanooey, Q. Li, M. Rabbani, A.R. Rajput, A survey on smartphones security: Software vulnerabilities, malware, and attacks, 2020, arXiv preprint arXiv:2001.09406.

[31] E. Erturk, A case study in open-source software security and privacy: Android adware, in: World Congress on Internet Security (WorldCIS-2012), IEEE, 2012, pp. 189–191.

[32] Alsoghyer, S., and Almomani, I. (2019). Ransomware detection system for Android applications. Electronics, 8(8), 868.

[33] M. Sikorski, A. Honig, Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, no starch press, 2012.

[34] K. Hamandi, A. Chehab, I.H. Elhajj, A. Kayssi, Android sms malware: Vulnerability and mitigation, in: 2013 27th International Conference on Advanced Information Networking and Applications Workshops, IEEE, 2013, pp. 1004–1009.

[35] SMS attacks and mobile malware threats, 2020, https://www.kaspersky.com/resource-center/threats/sms-attacks (Accessed: 2020-03-12).

[36] Moti, Zahra, Sattar Hashemi, and Amir Namavar Jahromi. "A Deep Learning-based Malware Hunting Technique to Handle Imbalanced Data." In 2020 17th International ISC Conference on Information Security and Cryptology (ISCISC), pp. 48-53. IEEE, 2020.