2022 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA) | 978-1-6654-6835-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/HORA55278.2022.9800057

A Deep Learning Based Android Malware Detection System with Static Analysis

Esra Calik Bayazit
Fatih Sultan Mehmet Vakif University
Marmara University Institute of Science
Computer Engineering Department
Istanbul, Turkey
ecalik@fsm.edu.tr

Ozgur Koray Sahingoz
Biruni University
Computer Engineering Department
Istanbul,Turkey
osahingoz@biruni.edu.tr

Buket Dogan
Marmara University
Department of Computer Engineering
Faculty of Technology
Istanbul,Turkey
buketb@marmara.edu.tr

***Abstract*—In recent years, smart mobile devices have become indispensable due to the availability of office applications, the Internet, game applications, vehicle guidance or similar most of our daily lives applications in addition to traditional services such as voice calls, SMSs, and multimedia services. Due to Android's open source structure and easy development platforms, the number of applications on Google Play, the official Android app store increased day by day. This also brig some security related issues for the end users. The increased popularity of Android operating system on mobile devices, and the associated financial benefits attracted attackers for developing some malware for these devices, which results a significant increase in the number of Android malware applications. To detect this type of security threats, signature based detection (static detection) in generally preferred due to its easy applicability and fast identification ability. Therefore in this study it is aimed to implement an up-to-date, effective, and reliable malware detection system with the help of some deep learning algorithms. In the proposed system, RNN-based LSTM, BiLSTM and GRU algorithms are evaluated on CICInvesAndMal2019 data set which contains 8115 static features for malware detection. Experimental results show that the BiLSTM model outperforms other proposed RNN-based deep learning methods with an accuracy rate of 98.85%.**

***Index Terms*—malware detection, static analysis, deep learning, RNN, android system**

I. Introduction

In recent years, with the digital transformation that has taken place, the rate of smart mobile device usage is increasing day by day. According to the "Digital 2022" report, the number of smart mobile devices worldwide has reached 5.3 billion [1] and Android operating system is used in 71.63% of these devices [2]. This is an indication that Android mobile devices are an indispensable part of daily life. Android systems have become the target of cyber attackers because of the valuable personal information they contain, which is directly proportional to factors such as the number of users, usage areas, and ease of application access. The attackers may receive access to users' devices through applications containing malware, which they succeeded in uploading to Google Play. In particular, the fact that Android applications can be downloaded from third-party sources as well as from the Google Play Store allows attackers to easily access the systems through an application. The number of mobile apps downloadable, which was 3.14 million in the fourth quarter of 2020, and 3.48 million as of the first quarter of 2021 [3]. However, the total number of Android malware in 2021 is 3.36 million [4]. This sheer when the number of Android users increases, and accordingly, the number of malware is increasing too. Because the amount of valuable information that attackers can be to accessed increases. This clearly requires effective malware detection systems against malware applications that are increasing in complexity.

Various techniques are available for Android malware detection. These are classified as static, dynamic and hybrid analysis. Static analysis enables malware detection without running an application. Thus, the mobile device is not affected by the malicious functionality of the application. The static analysis technique is a safe way to analyze malware. Dynamic analysis requires executing the application in an isolated environment to observe application behavior. In contrast to static analysis, dynamic analysis provides an advantage to reveal the natural behavior of malware due to the presence of code execution processes. On the other hand, the static analysis technique is less costly than the dynamic analysis technique in terms of time and computation, as it does not require an isolated environment. Finally, the hybrid analysis technique consists of a combination of these two techniques. Static and dynamic analysis techniques have advantages and disadvantages over each other. For example, malware can often avoid static analysis techniques with code obfuscation methods, while it can be more easily detected at execution time with dynamic analysis techniques.

In this study, RNN-based deep learning models are proposed on the current and public data set containing static features. The development of detection and classification tools is extremely important, as the popularity of Android systems on mobile platforms leads to an increase in threats in this area. In the literature, the success of malware detection systems is evident in many machine learning-based studies [5]–[7], [23]. However, the rapid growth of Android malware applications and technologies to evade detection systems are rendering defenses based on traditional methods ineffective. For this reason, in this study, we compare RNN, LSTM, BI-LSTM, GRU deep learning approaches on the CICInvesAndMal2019 data set instead of traditional machine learning algorithms.

978-1-6654-6835-0/22/$31.00 ©2022 IEEE

The archive file called Android Package Kit (APK) contains all the necessary data and resource files for Android applications to perform the required functions. The permissions required by an application are kept in the Android Manifest file and are used in malware detection as a feature. Android applications require user permission to use shared memory or access functions. The permissions granted to an application can be used for the desired purpose without asking for consent later. The CICInvesAndMal2019 data set contains permissions and intents, which are static features of applications.

The rest of the paper is organized as follows. Section II briefly covers a summary of the related works on Android malware detection and identification. Section III introduces the data set, Section IV covers the background of Android file structure, static analysis techniques and deep learning methods. Section V presents our proposed approach for the detection of Android malware and comparative experimental results. In Section VI is ends with concluding the work.

II. RELATED WORKS

In essence, attackers develop various methods, aiming to harm the functioning of systems through malware, to steal users' personal data and to use them in line with their goals or to block systems. Researchers suggest many studies in the literature to detect malware. In this part of the study, related studies in the literature are included.

Tang et al., [8], proposed a malware classification model to detect Android malware samples, as well as an algorithmic model, artificial intelligence-based learning solution based on static analysis and feature extraction from source code. The accuracy of the data was found to be approximately 94.4 % after 200 epochs of training, and the e-validation set was used to validate the model.

In 2021, the authors proposed a hybrid DL capable Android malware detection framework is proposed that uses permissions, API calls, and intents to detect malware from Android apps. With the designed approach, CNN and BiLSTM were utilized by using 10-fold cross validation in classifying malware [9]. The study was carried out on Androzoo and AMD data sets. In the proposed study, hybrid DL models and comparative DL-based algorithms were critically evaluated. The proposed hybrid CNN-BiLSTM model has an accuracy rate of 99.05 %. Wenbo et al., [10] proposed a multi-mode deep learning-based Android malware detection system. In the system where API calls, permissions, hardware components, and intent features of applications are used, deep learning algorithms are modeled in three different ways. In the system in which DNN, CNN and CNN-GRU deep learning algorithms are used, it is shown that a 98.74 % accuracy rate is obtained by using 5,560 malware and 16,666 benign samples.

Mu et al., [11] proposed an Android detection system by extracting the API sequence of the malware using text processing in the Cuckoo sandbox. A total of 11000 samples containing 8000 Android malware and 3000 good applications were used in the study. Among them, Android malware is mostly collected from Virus Share, Google Play Store. Benign apps are usually downloaded from the Android app store. They used the Dalvik analysis method-based BiLSTM method, one of the malware static analysis methods, to evaluate the performance of the system. It was stated that the accuracy rate obtained in the study was 96.74%.

A different static alaysis of Android devices is proposed Android malware detection model using static analysis features of benign and malicious apps collected from Google Play and Virus Share [12]. The DBN classification framework is presented in the study, which uses 331 features, including API calls and permissions. The obtained accuracy success rate was reported as 94.64%. The study also models the permissions required by Android systems by looking at features those regular applications require. Chen et al., [13] proposed a new prepossessing technique that is used as a feature in the operation of Dalvik opcode. Their proposed study used a new reprocessing approach combined with the LSTM model to detect malware, solving the long string problem to achieve fast training and high accuracy. It was stated that 95.58 % accuracy rate was obtained with the training model developed in the study using the Drebin data set.

Elayan et al., [14] proposed a malware detection model using the GRU deep learning algorithm. In the study, GRU deep learning algorithm and machine learning algorithms were compared. In the study using the CICAndMal2017 data set, the system proposed with GRU has an accuracy rate of 98.2%.

In the study proposed by Zang et al., [15], in which the deep learning based Deep Classify Droid detection system is presented, consists of three steps. These; feature extraction, feature embedding, and detection. In the first stages, five different feature sets are created using the static analysis method, and in the next stage, CNN-based malware detection is carried out. According to the performance result of the proposed study, an accuracy rate of 97.4% is obtained.

In related studies, data sets that generally do not contain real-world data were used. The data sets used were either collected from various sources or outdated. Malware collected from platforms such as Google Play Store may be removed by the official market and application data may change in the time between two snapshots. This situation imposes a limitation on the data sets created by users. The hottest point that distinguishes this work from the others is the use of deep learning approaches that can best represent the data, make the most of nonlinear functions, and solve problems from start to finish, using the CICInvesAndMal2019 data set, which contains up-to-date and valid application data.

III. DATASET

In this study, we used CICInvesAndMal2019 data set, which is provided by the University of New Brunswick, Canadian Institute for Cybersecurity. The CICInvesAndMal 2019 dataset is the second part of the CICAndMal 2017 dataset produced and published by Lashkari et al. [16]. CICInvesAndMal2019 data set produced by Taheri et al., [17] which includes static features, includes 426 malware and 5,065 benign labeled samples divided into four categories: Ransomware, Adware,

SMS Malware, Scareware. There are family types belonging to each category. For example, it includes family kinds in other categories such as Charger family, Simplocker family belonging to the Ransomware category. Incorporating these features, CICInvesAndMal2019 is an up-to-date data set consisting of application data from real devices and labeled data based on multiple reliable sources. The used data set consisted of static features, which are: permissions, and intents. The data set used includes 8115 features and 396 malware samples and 1126 benign samples.

IV. BACKGROUND

In recent years, deep learning approach have emerged as a promising research area not only for image based classification algorithms [18], but also in [19] and many computer security applications [12]–[15]. In the use of these approach, The number of features used in detection is so important. Same thing is also valid in malware detection and the preferred deep learning methods are among the factors that directly affect the efficiency of the proposed model. For this detection the used APK file is important and it can contain many features that learning algorithms use to produce optimal solutions. In this part of the study, it is aimed to provide an overview of Android systems, static analysis technique and deep learning algorithms which uses the features for making an analysis in the study.

A. Android File Structures

APK (Android Package Kit) is an archive file format used by the system for distribution and execution of Android applications [20]. To install an application from unofficial sites on an Android device, a direct Apk of that application is needed. Users download Apk, which is likely to contain malicious software, to use applications that have not yet been added to the Google Play Store on their devices. In the applications downloaded from the Google Play Store, the ”Apk” file name is not seen, since all the processes are performed by the system in the background, although the Apk file is automatically installed on the device. The features used in malware detection are extracted from the Apk file, which contains all the data and source files of the application [21]. Android file structure consists of AndroidManifest.xml, Meta-Inf, assets, resources.arsc, lib, classes.dex, res. The structure of these components is shown in Fig. 1.

AndroidManifest.xml— Android Manifest sets the app's package name, app permissions, etc., xml file containing static information.

META-INF — Directory containing APK metadata such as signature and certificate. assets — The directory containing the files where the assets are kept.

resources.arsc — A file structure in which XML files with precompiled resources such as strings, colors, or styles are compiled and put together.

lib— A file that contains libraries (such as armeabi, x86) that the application needs and that are not included in the Android SDK.

Fig. 1. APK File Structure

classes.dex— A file containing application code in dex file format, in which Java codes are compiled into Dalvik Byte coder.

res—the directory containing all uncompiled resources in the ”resources.arsc” file.

Android applications, which are usually written with Java codes, are compiled, and converted to Byte codes. Since Android cannot execute Byte-codes directly, these codes are converted to Dalvik Byte-code that runs in the Dalvik virtual machine. The codes written with this transformation are run by Android. Although the Android operating system is open source, the code developed by users is not open-source code and cannot be accessed directly. These codes can be accessed by Reverse Engineering methods [22]. The structure in which a source file is changed and then converted back to an Apk file is shown in Fig. 2.

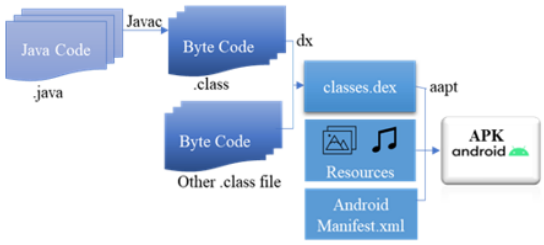

Fig. 2. APK Conversion Steps

B. Static Analysis

Static analysis is a passive approach without running an application. In this approach, the system is not affected by malicious behavior as the detection takes place before the application is executed [23]. The manifest file, which is one of the components of an Apk file, lists the features available for static analysis. In this file, there is notification of the hardware properties of the software to be used in the application, permissions, themes, activity properties. Through the tags in the manifest file, the application's permissions such as internet, camera access, file reading and writing are specified. In the data set used, there are 8115 features extracted from the

AndroidManifest.xml file and converted to vector form for training with learning methods.

C. Deep Learning Models

Deep learning is an algorithm system that can detect features using large data sets with the help of multi-layered structures. The premise of these algorithms is that learning takes place directly from large amounts of labeled data. RNN-based deep learning algorithms used in this study are described below.

Recurrent Neural Network (RNN) is a structure whose architecture can handle input size of any length and does not increase in model size despite the increase in input size, remembering past information and making calculations [24].

Long Short-Term Memory (LSTM), is an RNN-based deep learning method. It has a learning architecture that can keep long-term dependencies at random intervals. It is a very successful method, especially in the analysis of data that comes in order according to time or events in a certain relationship [25].

Bidirectional Long Short-Term Memory (Bi-LSTM) consists of two parallel LSTM defined by forward and backward loops that extract data patterns from the past and future to better model time dependency. Compared to LSTM neural networks, it can effectively use contextual information in both directions [26].

Gated Recurrent Unit (GRU) is similar in structure to LSTM because both are designed similarly. Like LSTM, GRU uses gates to control the flow of information. These gates consist of an update gate, a reset gate, and a temporary output. These gates consist of an update gate, a reset gate, and a temporary output [27].

V. METHODOLOGY

In this study, we are aimed to compare the experimental performance results obtained with RNN, LSTM, BiLSTM and GRU deep learning algorithms and present deep learning-based malware binary classification using static analysis techniques.

A. Proposed Method

In the framework of the proposed system, the preprocessed data set is divided into 60% training and 40% test data for binary classification. In the proposed system, malware binary classification was performed using RNN-based deep learning algorithms LSTM, BiLSTM, GRU. The framework of the proposed system is shown in Fig. 3.

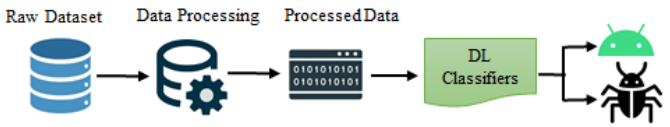

Fig. 3. Proposed System Framework

TABLE I
PROPOSED MODELS ARCHITECTURE.

Algorithm	Layers	Neurons	Epochs
RNN	activation=relu return sequences=true dropout (0.3)	(128,64,16)	100
LSTM	activation=relu return sequences=true dropout (0.2)	(128,64,16)	100
BiLSTM	activation=relu return sequences=true dropout (0.3)	(128,64,16)	100
GRU	activation=relu return sequences=true dropout (0.2)	(128,64,16)	100

Parameters	Values
Output Activation Function	sigmoid
Loss Function	binary crossentropy
Optimizer	adam
Batch Size	256
Learning Rate	0.0001

B. Preprocessing

The public data set containing permissions and intents extracted from Android applications and expressed as static properties is reserved for testing and training in comma-separated values (csv) format. Nan and duplicate values are removed in data set preprocessing. In the study where binary classification was made, family and category features were removed. Samples belonging to the malware category are assigned a value of 1, and benign samples are assigned a value of 0. MinMax Normalization, which is the scaling process in which the real feature values are normalized according to the largest and smallest values in the group, was applied.

C. Proposed Models Architecture

This section details the architecture used in the proposed deep learning algorithms for Android malware detection, which consists of RNN-based algorithms. In all of the proposed models, ”sigmoid” as activation function, ”binary cross-entropy” as loss function, ”adam” as optimizer, 256 as batch size and ”0.0001” as learning rate value are used. The parameter values of the deep learning algorithms proposed in the study are shown in Table 1.

Each proposed deep learning model consists of a three-layer structure. RNN is an algorithm that uses previous outputs as input and has the feature of keep information. However, to overcome the vanishing/exploding gradients problems, which are the disadvantages of RNN, the LSTM algorithm, which has input, forget, and exit gates with decision-making capability, has been proposed LSTM always forwards inputs from all cells in the forward direction before reaching the processing cell, keeping the past information. The fact that there is always forward transmission affects performance in predicting the future, leading to underestimation. To solve this problem, it is aimed to provide full efficiency with the BiLSTM model, which evaluates the past and future information in two directions, forward and backward. The GRU recommended for solving

TABLE II
CLASSIFICATION PERFORMANCE EXAMINATION RESULTS.

Models	Accuracy (%)	F1- Score (%)	Recall (%)	Precision (%)
RNN	98.02	97.36	96.88	97.85
LSTM	98.75	97.35	97.03	97.69
BİLSTM	98.85	98.21	97.52	98.91
GRU	98.10	97.53	97.23	97.85

the problems of standard RNNs, which are very similar to LSTM, are new generation RNNs with fewer parameters since they have two gates.

The confusion matrix is an evaluation model that provides information about the classification performance of the learning algorithm with a holistic approach. We evaluated accuracy, precision, recall, and F1 score for performance evaluation with the complexity matrix for the performance of the classifiers. The formulas of the performance evaluation metrics used are given below [28].

Where, the True Positive (TP) is the number of positive predicted values that are actually positive. The True Negative (TN) is the number of negative predicted values that are actually negative. The False Positive (FP) is the number of values predicted as positive but actually negative. The False Negative (FN) is the number of values predicted as negative but actually positive.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

D. Experimental Results

This study aims to compare deep learning models in terms of performance using static features in malware detection. The performance results of the experimental studies performed are shown in Table 2.

VI. CONCLUSIONS

The fact that smart devices in the Android market are becoming more accessible day by day has increased their usability in all areas of our daily life. This situation increases the need for Android applications as the dominant operating system in this area. Malicious applications are great threat not only in desktop computers but also in this rapidly growing mobile operating system market and it requires a robust and effective detection system. In this study, a deep learning-based model with static analysis technique is proposed to detect malwares in Android operating systems. The proposed detection system was presented by testing it on the publicly available CICInvesAndMal2019 dataset using some up-to-date deep learning algorithms based on RNN. The proposed models have competent performance values in terms of detection and accuracy. The BiLSTM model achieved an accuracy rate of 98.85%, providing better performance than other proposed algorithms.

As an ongoing work, we are planning to propose a hybrid model using supervised and semi-supervised deep learning algorithms in the future against cyber threats that continue to increase and change methods. In addition, the size of dataset is increasing day by day. This is also important for data science concept. Currently in the used dataset, there exist 8115 features in it. To chose the better features set is a trivial problem for the researchers. It is planned to use of different feature selection algorithms in our future work for decreasing the noise and making an efficient analysis. Additionally to decrease the train time and increase the effectiveness of the system GPU power can be taken into consideration as mentioned in [29].

ACKNOWLEDGMENT

This work has been supported by Marmara University Scientific Research Projects Coordination Unit under grant number FDK-2020-10066.

REFERENCES

[1] S. Kemp, Digital 2022: Global Overview Report, DataReportal – Global Digital Insights, Jan. 26, 2022. https://datareportal.com/reports/digital-2022-global-overview-report. (accessed Apr. 19, 2022).

[2] Mobile Operating System Market Share Worldwide, StatCounter Global Stats. https://gs.statcounter.com/os-market-share/mobile/worldwide/monthly-202103-202203-bar (accessed Apr. 19, 2022).

[3] Google Play: growth of available apps 2021,- Statista. https://www.statista.com/statistics/185729/google-play-quarterly-growth-of-available-app. (accessed Apr. 12, 2022).

[4] "Malware Statistics and Trends Report — AV-TEST," Av-test.org. https://www.av-test.org/en/statistics/malware/ (accessed Apr. 19, 2022).

[5] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, "Neural Network Based Android Malware Detection with Different IP Coding Methods," IEEE Xplore, Jun. 01, 2021. https://ieeexplore.ieee.org/document/9461302. (accessed Apr. 19, 2022).

[6] A. Kapratwar, F. Di Troia, and M. Stamp, "Static and Dynamic Analysis of Android Malware," Proceedings of the 3rd International Conference on Information Systems Security and Privacy, 2017, doi: 10.5220/0006256706530662.

[7] 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, Mar. 2019, doi: 10.1109/TIFS.2018.2866319.

[8] B. H. Tang et al., "Android Malware Detection Based on Deep Learning Techniques," 2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), Aug. 2021, doi: 10.1109/prai53619.2021.9551073.

[9] I. U. Haq, T. A. Khan, and A. Akhunzada, "A Dynamic Robust DL-Based Model for Android Malware Detection," IEEE Access, vol. 9, pp. 74510–74521, 2021, doi: 10.1109/access.2021.3079370.

[10] F. Wenbo, Z. Linlin, W. Chenyue, H. Yingjie, Y. Yuaner and Z. Kai,"AMC-MDL: A Novel Approach of Android Malware Classification using Multimodel Deep Learning," 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), 2020, pp. 251-256, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00052.

[11] T. Mu, H. Chen, J. Du, and A. Xu, "An Android Malware Detection Method Using Deep Learning Based on API Calls," 2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Oct. 2019, doi: 10.1109/imcec46724.2019.8983860.

[12] S. HR, "Static Analysis of Android Malware Detection using Deep Learning," 2019 International Conference on Intelligent Computing and Control Systems (ICCS), 2019, pp. 841-845, doi: 10.1109/ICCS45141.2019.9065765.
[13] Y. M. Chen, C. H. Hsu, and K. C. Kuo Chung, "A Novel Preprocessing Method for Solving Long Sequence Problem in Android Malware Detection," 2019 Twelfth International Conference on Ubi-Media Computing (Ubi-Media), Aug. 2019, doi: 10.1109/ubi-media.2019.00012.
[14] O. N. Elayan and A. M. Mustafa, "Android Malware Detection Using Deep Learning," Procedia Computer Science, vol. 184, pp. 847–852, 2021, doi: 10.1016/j.procs.2021.03.106.
[15] Y. Zhang, Y. Yang, and X. Wang, "A Novel Android Malware Detection Approach Based on Convolutional Neural Network," Proceedings of the 2nd International Conference on Cryptography, Security and Privacy, Mar. 2018, doi: 10.1145/3199478.3199492.
[16] 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," 2018 International Carnahan Conference on Security Technology (ICCST), 2018, pp. 1-7, doi: 10.1109/CCST.2018.8585560.
[17] L. Taheri, A. F. A. Kadir and A. H. Lashkari, "Extensible Android Malware Detection and Family Classification Using Network-Flows and API-Calls," 2019 International Carnahan Conference on Security Technology (ICCST), 2019, pp. 1-8, doi: 10.1109/CCST.2019.8888430.
[18] S. C. Kalkan and O. K. Sahingoz, "Deep Learning Based Classification of Malaria from Slide Images," 2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT), 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741702.
[19] G. Sismanoglu, M. A. Onde, F. Kocer and O. K. Sahingoz, "Deep Learning Based Forecasting in Stock Market with Big Data Analytics," 2019 Scientific Meeting on Electrical-Electronics Biomedical Engineering and Computer Science (EBBT), 2019, pp. 1-4, doi: 10.1109/EBBT.2019.8741818.
[20] "Android Security Features," Android Open Source Project. https://source.android.com/security/features (accessed Apr. 20, 2022).
[21] H. Kim, T. Cho, G.-J. Ahn, and J. Yi, "Risk assessment of mobile applications based on machine learned malware dataset," Multimedia Tools and Applications, 2017, doi: 10.1007/s11042-017-4756-0.
[22] M. G. Rekoff, "On reverse engineering," IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-15, no. 2, pp. 244–252, Mar. 1985, doi: 10.1109/tsmc.1985.6313354.
[23] E. C. Bayazit, O. Koray Sahingoz and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey," 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1-8, doi: 10.1109/HORA49412.2020.9152840.
[24] S. Jha, D. Prashar, H. V. Long, and D. Taniar, "Recurrent Neural Network for Detecting Malware," Computers and Security, p. 102037, Sep. 2020, doi:10.1016/j.cose.2020.102037.
[25] F. A. Gers, "Learning to forget: continual prediction with LSTM," 9th International Conference on Artificial Neural Networks: ICANN '99, 1999, doi: 10.1049/cp:19991218.
[26] T. Zhang and R. Xu, "Performance Comparisons of Bi-LSTM and Bi-GRU Networks in Chinese Word Segmentation," 2021 5th International Conference on Deep Learning Technologies (ICDLT), Jul. 2021, doi: 10.1145/3480001.3480011.
[27] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models and a character-level CNN," IEEE Xplore, Mar. 01, 2017. https://ieeexplore.ieee.org/abstract/document/7952603 (accessed Apr. 20, 2022).
[28] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," arXiv:2010.16061 [cs, stat], Oct. 2020, [Online]. Available: https://arxiv.org/abs/2010.16061. (accessed Apr. 19, 2022).
[29] S. I. Baykal, D. Bulut and O. K. Sahingoz, "Comparing deep learning performance on BigData by using CPUs and GPUs," 2018 Electric Electronics, Computer Science, Biomedical Engineerings' Meeting (EBBT), 2018, pp. 1-6, doi: 10.1109/EBBT.2018.8391429.