International journal of basic and applied research
www.pragatipublication.com
ISSN 2249-3352 (P) 2278-0505 (E)
Cosmos Impact Factor-5.86

OPTIMAL ENSEMBLE LEARNING FOR AUTOMATED ANDROID MALWARE DETECTION IN CYBERSECURITY

[1]NAINI ARAVIND KUMAR, [2]MOTAM PRAGHNYA SREE
[1]Assistant Professor,[2]Student
Department of CSE
Sree Chaitanya College of Engineering, Karimnagar

ABSTRACT

The development of computer systems in recent years has changed human existence from actual to virtual worlds. Malware is superfluous software that is often used to initiate cyberattacks. With the use of sophisticated packaging and obfuscation techniques, malware versions continue to evolve. Malware detection and categorisation become more difficult using these methods. It is important to use novel approaches that depart from traditional methods in order to successfully counter emerging virus types. All sophisticated and novel malware strains cannot be identified by machine learning (ML) techniques. To identify all malware types, the deep learning (DL) technology may provide a viable answer. This study introduces the Optimal Ensemble Learning Approach for Cybersecurity (AAMD-OELAC) approach for Automated Android Malware Detection. Automated malware detection and categorisation is the primary goal of the AAMD-OELAC approach. The AAMD-OELAC approach does preliminary data preparation to accomplish this. The kernel extreme learning machine (KELM), regularised random vector functional link neural network (RRVFLN), and least square support vector machine (LS-SVM) are the three ML models that the AAMD-OELAC approach uses in its ensemble learning process for Android malware detection. Lastly, the best parameter tuning of the three DL models is achieved by using the hunter-prey optimisation (HPO) technique, which also contributes to better malware detection outcomes. To indicate that the AAMD-OELAC approach is superior, a thorough experimental investigation is carried out. The simulation results demonstrated how superior the AAMD-OELAC technology is compared to other current methods.

1. INTRODUCTION

Network engineers and computer scientists are starting to focus more and more on cyber security, thus it is necessary to find satisfactory answers to a number of issues [1]. As a result of the rapid advancements in technology and its innate incorporation into many facets of life, targets and malware applications have become well-known and researched [2]. The malware kind that has attracted the most attention online is Android malware. Android is one popular operating system that now leads the operating system market [3].

Since few malware apps have more than 50 characteristics, making detection challenging, malware intrusive techniques arise to evade identification [4]. Therefore, developing strategies to address the ongoing proliferation of Android malware is crucial in order to locate, disable, or eradicate it effectively. Because of all these challenges, researchers in the field are encouraged to keep looking for malware and to take appropriate measures to handle it [5]. In order to detect Android malware, researchers have created three mechanisms: dynamic, static, and hybrid analytic approaches. Without requiring the actual deployment of the program, static analysis extracts the attributes that help discover negative performance for applications [6]. However, the research was hampered by code obfuscation techniques that aid virus authors in avoiding static approaches. App malware may be identified during runtime via the use of dynamic analysis [7]. Static analysis often provides the capacity to locate the malware element via source code, while dynamic analysis provides the ability to locate malware inside a runtime context. Malware may expose users and developers of Android devices to needless risks and hazards [8]. This paper discusses malware detection techniques. Android Application Packages (APKs) are included in the ML model for malware detection in order to derive a suitable collection of characteristics. Malicious APKs may be identified using deep learning (DL) and machine learning (ML) techniques [9]. Similar to malware detection, software code vulnerability detection consists of two stages: feature development via code analysis and machine learning training on derived characteristics to identify susceptible code segments [10].

This research introduces the Optimal Ensemble Learning Approach for Cyber Security (AAMD-OELAC) approach for Automated Android Malware Detection. The first step of data preparation is carried out via the AAMDOELAC approach. Three machine learning models—Least Square Support Vector Machine (LS-SVM), Kernel Extreme Learning Machine (KELM), and Regularised Random Vector Functional Link Neural Network (RRVFLN)—are used in the ensemble learning process of the AAMD-OELAC approach for Android malware detection. Lastly, the three DL models' optimum parameter tuning is achieved by using the hunter-prey optimisation (HPO) technique, which also contributes to better malware detection outcomes. A thorough experimental study is performed to demonstrate the superiority of the AAMD-OELAC technique. Briefly said, the following is a list of the major contributions.

- An intelligent AAMD-OELAC technique comprising data preprocessing, ensemble learning, and HPO-based hyper parameter tuning is presented for Android malware detection. To the best of our knowledge, the AAMD-OELAC technique never existed in the literature.

• Perform ensemble learning-based classification process comprising LS-SVM, KELM, and RRVFLN models for Android malware detection.

• The combination of the HPO algorithm and ensemble learning process improves the detection accuracy of Android malware. By utilizing multiple classifiers and optimization strategies, the model can effectively identify malicious patterns and behaviors in Android applications.

2. LITERATURE SURVEY

‘‘Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses,’’

Today, android smartphones are being used by billions of users and thus have become a lucrative target of malware designers. Therefore being one step ahead in this zero-sum game of malware detection between the anti-malware community and malware developers is more of a necessity than a desire. This work focuses on a proactive adversary-aware framework to develop adversarially superior android malware detection models. We first investigate the adversarial robustness of thirty-six distinct malware detection models constructed using two static features (permission and intent) and eighteen classification algorithms. We designed two Targeted Type-II Evasion Attacks (*TRPO-MalEAttack* and *PPO-MalEAttack*) based on reinforcement learning to exploit vulnerabilities in the above malware detection models. The attacks aim to add minimum perturbations in each malware application and convert it into an adversarial application that can fool the malware detection models. The TRPO-MalEAttack achieves an average fooling rate of 95.75% (with 2.02 mean perturbations), reducing the average accuracy from 86.01% to 49.11% in thirty-six malware detection models. On the other hand, The PPO-MalEAttack achieves a higher average fooling rate of 96.87% (with 2.08 mean perturbations), reducing the average accuracy from 86.01% to 48.65% in the same thirty-six detection models. We also develop a list of the *TEN* most vulnerable android permissions and intents that an adversary can use to generate more adversarial applications. Later, we propose a defense strategy (*MalVPatch*) to counter the adversarial attacks on malware detection models. The MalVPatch defense achieves higher detection accuracy along with a drastic improvement in the adversarial robustness of malware detection models. Finally, we conclude that investigating the adversarial robustness of models is necessary before their real-world deployment and helps achieve adversarial superiority in android malware detection.

‘‘You are what the permissions told me! Android malware detection based on hybrid tactics,’’

Recent years have witnessed a significant increase in the use of Android devices in many aspects of our life. However, users can download Android apps from third-party channels, which provides numerous opportunities for malware. Attackers utilize unsolicited permissions to gain access to the sensitive private intelligence of users. Since signature-based antivirus solutions no longer meet practical needs, efficient and adaptable solutions are desperately needed, especially in new variants. As a remedy, we propose a hybrid Android malware detection approach that combines dynamic and static tactics. We firstly adopt static analysis inferring different permission usage patterns between malware and benign apps based on the machine-learning-based method. To classify the suspicious apps further, we extract the object reference relationships from the memory heap to construct a dynamic feature base. We then present an improved state-based algorithm based on DAMBA. Experimental results on a real-world dataset of 21,708 apps show that our approach outperforms the well-known detector with 97.5% F1-measure. Besides, our system is demonstrated to resist permission abuse behaviors and obfuscation techniques.

‘‘Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification,’

Since the development of information systems during the last decade, cybersecurity has become a critical concern for many groups, organizations, and institutions. Malware applications are among the commonly used tools and tactics for perpetrating a cyberattack on Android devices, and it is becoming a challenging task to develop novel ways of identifying them. There are various malware detection models available to strengthen the Android operating system against such attacks. These malware detectors categorize the target applications based on the patterns that exist in the features present in the Android applications. As the analytics data continue to grow, they negatively affect the Android defense mechanisms. Since large numbers of unwanted features create a performance bottleneck for the detection mechanism, feature selection techniques are found to be beneficial. This work presents a Rock Hyrax Swarm Optimization with deep learning-based Android malware detection (RHSODL-AMD) model. The technique presented includes finding the Application Programming Interfaces (API) calls and the most significant permissions, which results in effective discrimination between the good ware and malware applications. Therefore, an RHSO based feature subset selection (RHSO-FS) technique is derived to improve the classification results. In addition, the Adamax optimizer with attention recurrent autoencoder (ARAE) model is employed for Android malware detection. The experimental validation of the RHSODL-AMD technique on the Andro-AutoPsy dataset exhibits its promising performance, with a maximum accuracy of 99.05%.

''A method for automatic Android malware detection based on static analysis and deep learning,''

The computers nowadays are being replaced by the smartphones for the most of the internet users around the world, and Android is getting the most of the smartphone systems' market. This rise of the usage of smartphones generally, and the Android system specifically, leads to a strong need to effectively secure Android, as the malware developers are targeting it with sophisticated and obfuscated malware applications. Consequently, a lot of studies were performed to propose a robust method to detect and classify android malicious software (malware). Some of them were effective, some were not; with accuracy below 90%, and some of them are being outdated; using datasets that became old containing applications for old versions of Android that are rarely used today. In this paper, a new method is proposed by using static analysis and gathering as most useful features of android applications as possible, along with two new proposed features, and then passing them to a functional API deep learning model we made. This method was implemented on a new and classified android application dataset, using 14079 malware and benign samples in total, with malware samples classified into four malware classes. Two major experiments with this dataset were implemented, one for malware detection with the dataset samples categorized into two classes as just malware and benign, the second one was made for malware detection and classification, using all the five classes of the dataset. As a result, our model overcomes the related works when using just two classes with F1-score of 99.5%. Also, high malware detection and classification performance was obtained by using the five classes, with F1-score of 97%.

3. EXISTING SYSTEM

Shaukat et al. [11] devise a new DL-related method for detecting malware. It delivered superior outcomes to classical methods by merging dynamic and static analysis benefits. Firstly, it visualizes a portable executable (PE) file as coloured images. Secondly, it extracted deep features from colour images utilizing fine-tuned DL method. Thirdly, it finds malware related to the deep features of SVM. Geremias et al. [12] presented a method using image-based DL called novel multi-view Android malware identification, applied threefold. Firstly, as per the many feature sets in multi-view settings, apps were assessed, thereby raising the data presented for the classification. Secondly, the derived feature set is transformed into image formats while preserving the essential elements of data distribution, keeping the data for the classifier task. Thirdly, built images are collectively depicted in one shot, all in a predefined image channel, allowing the implementation of DL structure.

Kim et al. [13] modelled a malware detection system called MAPAS that attains higher precision and adaptable use of computational resources. MAPAS examined the performances of malicious apps based on API call graphs of them through CNN. However, the presented MAPAS technique does not utilize a classifier method produced by CNN, it uses CNN to find typical attributes of the API call graph of malware. Fallah and Bidgoly [14] developed a technique related to LSTM for detecting malware-having the capability of differentiating benign and malware samples and identifying and detecting unseen and new types of malware. In this study, the author has executed many studies to show the abilities of the presented technique, including new malware family detection, malware identification, malware family identification, as well as assessing the minimal time needed to find malware.

Sihag et al. [15] introduced DL-based Android malware identification with the use of DYnamic features (De-LADY), a resilient obfuscation method. It has used behavioural features from dynamic analysis of an application performed in the emulated setting. Wang et al. [16] present a hybrid method related to DAE and CNN. Firstly, to enhance the precision of malware detection, the author reconstructed the high-dimensional feature of apps and used many CNN to find Android malware. Secondly, to diminish the training period, the author used DAE as a pre-training approach for CNN. With the consolidation, DAE and CNN method (DAE-CNN) can study flexible patterns quickly.

Yadav et al. [17] presented a performance comparison of 26 existing pretrained CNN methods in Android malware detection. Depending on the outcomes, to find Android malware, an EfficientNet-B4 CNN-based approach was devised with the use of an image-based malware representation of the Android DEX file. From the malware images, EfficientNet-B4 extracted relevant attributes. Masum and Shahriar [18] devised a DL structure named Droid-NNet, for classifying malware. But this technique Droid-NNet, is a deep learner that surpasses existing cutting-edge ML approaches. Idrees et al. [19] examine PIndroid – a new Permission and Intents-based structure to detect Android malware applications. As we know, PIndroid is the primary solution, which utilizes a group of permissions and purposes supplemented with Ensemble approaches for correct malware detection. In [20], the authors establish that once the concept drift was discussed, permissions create long-lasting and effectual malware detection methods. Taha and Barukab [21] introduce a mechanism for Android malware classification utilizing optimizer ensemble learning depending on GA. The GA was utilized for optimizing the parameter settings from the RF technique for obtaining the maximum Android malware classifier accuracy. Sabanci et al. [22] intended to categorize pepper seeds belonging to distinct cultivars with CNN techniques. Two methods are presented for classification. Initially, the CNN approaches (ResNet50 and ResNet18) are trained for pepper seeds. Secondary, diverse in the first, the features of pre-training CNN approaches are fused, and feature selection has been executed to the fused features. In [23], the authors examine recent algorithms utilized for Android Malware Detection. As a result, an outline of the Android system exposed the underlying processes and the problems facing its security structure.

Disadvantages

- The complexity of data: Most of the existing machine learning models must be able to accurately interpret large and complex datasets to detect Android Malware Detection.
- Data availability: Most machine learning models require large amounts of data to create accurate predictions. If data is unavailable in sufficient quantities, then model accuracy may suffer.
- Incorrect labeling: The existing machine learning models are only as accurate as the data trained using the input dataset. If the data has been incorrectly labeled, the model cannot make accurate predictions.

4. PROPOSED SYSTEM

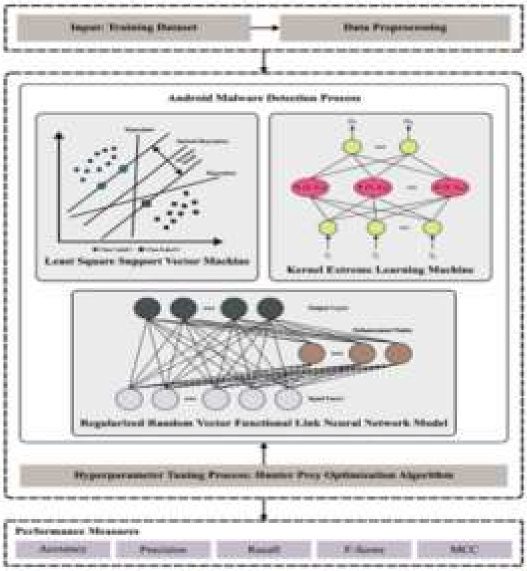
This paper presents an Automated Android Malware Detection using Optimal Ensemble Learning Approach for Cybersecurity (AAMD-OELAC) technique. The AAMDOELAC technique performs data preprocessing at the preliminary stage. For the Android malware detection process, the AAMD-OELAC technique follows an ensemble learning process using three ML models, namely Least Square Support Vector Machine (LS-SVM), kernel extreme learning machine (KELM), and Regularized random vector functional link neural network (RRVFLN). Finally, the hunter-prey optimization (HPO) algorithm is exploited for the optimal parameter tuning of the three DL models, and it helps accomplish improved malware detection results. To indicate the supremacy of the AAMD-OELAC approach, a comprehensive experimental analysis is carried out.

Advantages

- An intelligent AAMD-OELAC technique comprising data preprocessing, ensemble learning, and HPO-based hyperparameter tuning is presented for Android malware detection. To the best of our knowledge, the AAMD-OELAC technique never existed in the literature.

- Perform ensemble learning-based classification process comprising LS-SVM, KELM, and RRVFLN models for Android malware detection.
- The combination of the HPO algorithm and ensemble learning process improves the detection accuracy of Android malware. By utilizing multiple classifiers and optimization strategies, the model can effectively identify malicious patterns and behaviours in Android applications.

5. SYSTEM ARCHITECTURE


Input: Training Dataset
Data Preprocessing
Android Malware Detection Process
Least Square Support Vector Machine
Kernel Extreme Learning Machine
Regularized Random Vector Functional Link Neural Network Model
Hyperparameter Tuning Process: Hunter Prey Optimization Algorithm
Performance Measures
Accuracy
Precision
Recall
F-Score
MCC

6. IMPLEMENTATION

Modules

Service Provider

In this module, the Service Provider has to login by using valid user name and password. After login successful he can do some operations such as Train and Test Data Sets, View Trained and Tested Accuracy in Bar Chart, View Trained and Tested Accuracy Results, View Predicted Android Malware Detection Details, Find Predicted Android Malware Detection Ratio, Download Predicted Datasets, View Android Malware Predicted Ratio Results, View All Remote Users.

View and Authorize Users

In this module, the admin can view the list of users who all registered. In this, the admin can view the user's details such as, user name, email, address and admin authorizes the users.

Remote User

In this module, there are n numbers of users are present. User should register before doing any operations. Once user registers, their details will be stored to the database. After registration successful, he has to login by using authorized user name and password. Once Login is successful user will do some operations like REGISTER AND LOGIN, PREDICT ANDROID MALWARE TYPE, VIEW YOUR PROFILE.

7. ALGORITHIMS

Naïve Bayes

The naive bayes approach is a supervised learning method which is based on a simplistic hypothesis: it assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature .

Yet, despite this, it appears robust and efficient. Its performance is comparable to other supervised learning techniques. Various reasons have been advanced in the literature. In this tutorial, we highlight an explanation based on the representation bias. The naive bayes classifier is a linear classifier, as well as linear discriminant analysis, logistic regression or linear SVM (support vector machine). The difference lies on the method of estimating the parameters of the classifier (the learning bias).

While the Naive Bayes classifier is widely used in the research world, it is not widespread among practitioners which want to obtain usable results. On the one hand, the researchers found especially it is very easy to program and implement it, its parameters are easy to estimate, learning is very fast even on very large databases, its accuracy is reasonably good in comparison to the other approaches. On the other hand, the final users do not obtain a model easy to interpret and deploy, they does not understand the interest of such a technique.

Thus, we introduce in a new presentation of the results of the learning process. The classifier is easier to understand, and its deployment is also made easier. In the first part of this tutorial, we present some theoretical aspects of the naive bayes classifier. Then, we implement the approach on a dataset with Tanagra. We compare the obtained results (the parameters of the model) to those obtained with other linear approaches such as the logistic regression, the linear discriminant analysis and the linear SVM. We note that the results are highly consistent. This largely explains the good performance of the method in comparison to others. In the second part, we use various tools on the same dataset (**Weka 3.6.0**, **R 2.9.2**, **Knime 2.1.1**, **Orange 2.0b** and **RapidMiner 4.6.0)**. We try above all to understand the obtained results.

Logistic regression Classifiers

Logistic regression analysis studies the association between a categorical dependent variable and a set of independent (explanatory) variables. The name logistic regression is used when the dependent variable has only two values, such as 0 and 1 or Yes and No. The name multinomial logistic regression is usually reserved for the case when the dependent variable has three or more unique values, such as Married, Single, Divorced, or Widowed. Although the type of data used for the dependent variable is different from that of multiple regression, the practical use of the procedure is similar.

Logistic regression competes with discriminant analysis as a method for analyzing categorical-response variables. Many statisticians feel that logistic regression is more versatile and better suited for modeling most situations than is discriminant analysis. This is because logistic regression does not assume that the independent variables are normally distributed, as discriminant analysis does.

This program computes binary logistic regression and multinomial logistic regression on both numeric and categorical independent variables. It reports on the regression equation as well as the goodness of fit, odds ratios, confidence limits, likelihood, and deviance. It performs a comprehensive residual analysis including diagnostic residual reports and plots. It can perform an independent variable subset selection search, looking for the best regression model with the fewest independent variables. It provides confidence intervals on predicted values and provides ROC curves to help determine the best cutoff point for classification. It allows you to validate your results by automatically classifying rows that are not used during the analysis.

Decision tree classifiers

Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing descriptive decision making knowledge from the supplied data. Decision tree can be generated from training sets. The procedure for such generation based on the set of objects (S), each belonging to one of the classes C1, C2, …, Ck is as follows:

Step 1. If all the objects in S belong to the same class, for example Ci, the decision tree for S consists of a leaf labeled with this class

Step 2. Otherwise, let T be some test with possible outcomes O1, O2,…, On. Each object in S has one outcome for T so the test partitions S into subsets S1, S2,… Sn where each object in Si has outcome Oi for T. T becomes the root of the decision tree and for each outcome Oi we build a subsidiary decision tree by invoking the same procedure recursively on the set Si.

8. CONCLUSION AND FUTURE ENHANCEMENT

We have developed the AAMD-OELAC approach in this work to provide an automated and precise method for detecting Android malware. The goal of the AAMD-OELAC method was to automatically identify and categorise Android malware. The AAMD-OELAC approach uses ensemble classification, HPO-based parameter adjustment, and data preparation to accomplish this. The AAMD-OELAC approach uses three machine learning models (ML): LS-SVM, KELM, and RRVFLN, in an ensemble learning process for Android malware detection. Ultimately, the HPO technique is used to get the best possible parameter tuning for the three DL models, which contributes to better malware detection outcomes. A comprehensive experimental investigation is carried out to demonstrate the superiority of the AAMD-OELAC approach. The simulation results demonstrated the AAMDOELAC technique's superiority over other current methods.

The development of more complex methods for capturing and analysing fine-grained behaviours may be the main focus of future research in order to improve the detection of malicious software. Furthermore, future research might investigate privacy-preserving techniques like federated learning or safe multi-party computing, which allow for cooperative malware detection without jeopardising user privacy.

REFERENCES

[1] H. Rathore, A. Nandanwar, S. K. Sahay, and M. Sewak, ‘‘Adversarial superiority in Android malware detection: Lessons from reinforcement learning based evasion attacks and defenses,’’ *Forensic Sci. Int., Digit. Invest.*, vol. 44, Mar. 2023, Art. no. 301511.

[2] H. Wang, W. Zhang, and H. He, ‘‘You are that the permissions told me! Android malware detection based on hybrid tactics,’’ *J. Inf. Secur. Appl.*, vol. 66, May 2022, Art. no. 103159.

[3] A. Albakri, F. Alhayan, N. Alturki, S. Ahamed, and S. Shamsudheen, ‘‘Metaheuristics with deep learning model for cybersecurity and Android malware detection and classification,’’ *Appl. Sci.*, vol. 13, no. 4, p. 2172, Feb. 2023.

[4] M. Ibrahim, B. Issa, and M. B. Jasser, ‘‘A method for automatic Android malware detection based on static analysis and deep learning,’’ *IEEE Access*, vol. 10, pp. 117334–117352, 2022.

[5] L. Hammood, İ. A. Doğru, and K. Kılıç, ‘‘Machine learning-based adaptive genetic algorithm for Android malware detection in auto-driving vehicles,’’ *Appl. Sci.*, vol. 13, no. 9, p. 5403, Apr. 2023.

[6] P. Bhat and K. Dutta, ‘‘A multi-tiered feature selection model for Android malware detection based on feature discrimination and information gain,’’ *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9464–9477, Nov. 2022.

[7] D.Wang, T. Chen, Z. Zhang, and N. Zhang, ‘‘A survey of Android malware detection based on deep learning,’’ in *Proc. Int. Conf. Mach. Learn. Cyber Secur.* Cham, Switzerland: Springer, 2023, pp. 228–242.

[8] Y. Zhao, L. Li, H. Wang, H. Cai, T. F. Bissyandé, J. Klein, and J. Grundy,
‘‘On the impact of sample duplication in machine-learning-based Android
malware detection,’’ *ACM Trans. Softw. Eng. Methodol.*, vol. 30, no. 3, pp. 1–38, Jul. 2021.

[9] E. C. Bayazit, O. K. Sahingoz, and B. Dogan, ‘‘Deep learning based malware detection for Android systems: A comparative analysis,’’ *Tehnički vjesnik*, vol. 30, no. 3, pp. 787–796, 2023.

[10] H.-J. Zhu, W. Gu, L.-M. Wang, Z.-C. Xu, and V. S. Sheng, ‘‘Android malware detection based on multi-head squeeze-and-excitation residual network,’’ *Expert Syst. Appl.*, vol. 212, Feb. 2023, Art. no. 118705.

[11] K. Shaukat, S. Luo, and V. Varadharajan, ‘‘A novel deep learning-based approach for malware detection,’’

Eng. Appl. Artif. Intell., vol. 122, Jun. 2023, Art. no. 106030.

[12] J. Geremias, E. K. Viegas, A. O. Santin, A. Britto, and P. Horchulhack, ‘‘Towards multi-view Android malware detection through image-based deep learning,’’ in *Proc. Int.Wireless Commun. Mobile Comput. (IWCMC)*, May 2022, pp. 572–577. 72516 VOLUME 11, 2023 IEEE Transaction on MachineLearning,Volume:11,Issue Date:11.July.2023

[13] J. Kim, Y. Ban, E. Ko, H. Cho, and J. H. Yi, ‘‘MAPAS: A practical deep learning-based Android malware detection system,’’ *Int. J. Inf. Secur.*, vol. 21, no. 4, pp. 725–738, Aug. 2022.

[14] S. Fallah and A. J. Bidgoly, ‘‘Android malware detection using network traffic based on sequential deep learning models,’’ *Softw., Pract. Exper.*, vol. 52, no. 9, pp. 1987–2004, Sep. 2022.

[15] V. Sihag, M. Vardhan, P. Singh, G. Choudhary, and S. Son, ‘‘De-LADY: Deep learning-based Android malware detection using dynamic features,’’ *J. Internet Serv. Inf. Secur.*, vol. 11, no. 2, p. 34, 2021.