Artificial Intelligence Applications And Innovations Vol. 1 No. 3 (2024)

Article history:
Received 1 March 2024
Revised 29 April 2024
Accepted 02 June 2024
Published online 01 July 2024

Android Malware Detection by XGBoost Algorithm

Sana. Nazarinezhad[1*], Nafise. Khosrojerdi[2], Ahmad Reza. Shafieesabet[3]

[1] Master in Information Technology Engineering, Department of Information Technology, Faculty of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran
[2] Master in Artificial Intelligence, Faculty of Computer Engineering, Malek Ashtar University of Technology, Tehran, Iran
[3] Master in Information Technology Engineering, Department of Information Technology, Foulad Institute of Technology, Isfahan, Iran

*** Corresponding author email address**: sana.nazari1995@gmail.com

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Nazarinezhad, S., Khosrojerdi, N., & Shafieesabet, A. R. (2024). Android Malware Detection by XGBoost Algorithm. *Artificial Intelligence Applications and Innovations*, *1*(3), 31-37.
https://doi.org/10.61838/jaiai.1.3.4

A B S T R A C T

Today, smartphones are prevalent for personal and corporate use and have become the new personal computer due to their portability, ease of use, and functionality (such as video conferencing, Internet browsing, e-mail, continuous wireless and data connectivity, worldwide map location services, and countless mobile applications such as banking applications). On the other hand, we store many sensitive and private information daily on smart devices. This information is of interest to malicious writers who are developing malware to steal information from mobile devices. Unfortunately, the open source and widespread adoption of the Android operating system has made it the most targeted of the four popular mobile platforms by malware writers. Many researchers have tried to identify malware using program signatures, which have been successful to some extent. However, the signature cannot effectively identify new and unknown malware. For this reason, in this article, we propose a method that designs a machine-learning model for Android malware detection based on the properties of Permissions, Intents APKs. In this study, we evaluated more than 25,000 Android samples belonging to malware and trusted samples. Experimental results show the effectiveness of the proposed method by obtaining 96.27% accuracy.

Keywords: *Malware detection, Artificial Intelligence, Machine Learning, Anti-malware, Android, XGBoost, Ensemble Classifier.*

1. Introduction

The use of smartphones has increased exponentially in recent years. According to Statista statistics, in 2023, 7.33 billion people will use mobile phones, and 72% will use Android smartphones [1, 2]. In the current era, smartphones and the Internet of Things (IoT) are used everywhere as the main media of information communication and

entertainment [3]. Mobile devices are becoming a part of our daily lives and are used even more than conventional computer systems such as personal computers [4].

Despite significant and continuous improvements in cyber security mechanisms, malware remains one of the most severe threats in cyberspace. According to the McAfee report, the total number of malware samples in 2020 gradually increased to about 1.5 billion. Malware everywhere erodes cyberspace and creates sub-branches such as mobile malware, MacOS malware, Internet of Things malware, and Coin Miner malware, which cause substantial financial damage to individuals and industries [5]. According to the latest report published by the computer security company G Data, a new malware is found in online repositories every 8 seconds. In this way, 11,000 malware are produced daily [6].

The Android platform providers have proposed several security measures to prevent the installation of malware, the most important of which is the Android licensing system. Each application must explicitly request the user perform some task on the device during installation, such as sending SMS, etc. [7]. However, many users tend to blindly grant permissions to unknown applications, undermining the licensing system's purpose.

To solve this problem, many research methods have been proposed to analyze and identify Android malware before installation. Traditionally, anti-malware software, whether for Android, Windows, or other operating systems, identifies malware by its signature. Examples of file signatures include file encryption hashes and byte patterns. Then, malware is identified by extracting the program's signature and checking it against a known database [8]. Any slight change in the signature can cause the anti-malware not to detect the malware. These small changes can be introduced by replacing a few lines of code, directives, or even keywords. This limitation prevents the detection of existing malware and malware based on zero-day attacks. Periodic anti-malware updates are insufficient to keep up with this constant stream of new malware. At any given time, anti-malware software will not be aware of thousands, if not more, of malware. As a potential solution to this problem, artificial intelligence (AI) methods such as machine learning (ML) and deep learning (DL) can be used to provide anti-malware products with the ability to identify new malware based on common patterns in existing malware.

According to the mentioned cases, this article aims to present a machine learning model using a static analysis method to identify Android malware with the ability to identify zero-day malware. The rest of this paper is organized as follows: Related works are introduced in Section 2. An overview of the architecture is presented in Section 3. The implementation and evaluation of our proposed method are discussed in detail in Section 4 and Section 5, respectively. Section 6 concludes the paper.

2. Literature Review

Due to the number of Android devices and the security risks associated with Android malware, the field of Android malware detection using machine learning has grown significantly in recent years. Researchers have proposed supervised, unsupervised, and deep-learning strategies to detect Android malware [9].

Support vector machines (SVM) and decision trees, two examples of supervised learning techniques, have been widely used in Android malware detection [10]. In order to build a model capable of distinguishing between legitimate and malicious Android apps, these methods rely on labeled training data.

Android malware detection also uses unsupervised learning techniques like clustering and dimensionality reduction. These techniques can detect patterns in data that may indicate malware and do not require labeled training data [11].

However, using machine learning techniques to detect Android malware is the subject of this survey. In the following, we examine some of the research done in this field.

The authors in [12], PUMA (Permission to Use for Malware Detection on Android), presented a new strategy for detecting malware on Android devices. The authors claim that malware's excessive use of permissions can serve as an identification signature for malicious programs. PUMA uses an ML-based algorithm that trains a classifier from a dataset (more than 4000 APKs containing malicious and malware) of malware and benign applications. The permissions requested for the application and their usage patterns are the characteristics used for classification. The authors stated that PUMA detects malware with over 90% accuracy and a low false positive rate.

The malware detection approach in [13] uses a feature-based learning framework where permissions and API calls are used as features. This work uses machine learning methods such as SVM, decision trees, and bundling

approaches and achieves an accuracy of 96.88%. The work in [13] used a static analysis technique.

Jung et al. [14] used a static analysis technique and presented an approach to selecting a set of API calls to detect malicious Android apps. This approach selects two sets of API calls: a benign API list and a malicious API list, which contain the most commonly called APIs in benign and malicious applications, respectively. This is a very simple method to determine whether a new program is benign or malicious, but it gives a high accuracy of nearly 90%.

[15] proposed an SVM-based malware detection scheme using suspicious API calls and dangerous authorization as features. This feature data trains the SVM classifier, which is then used to identify unknown malware in the future. The test results showed that the proposed plan's accuracy is 81% without taking into account the risky licenses and 86% with the licenses taken into account.

In [16], a static detection method based on 151 trained Android system permissions was used for knowledge analysis. This model is based on training a set of 10,000 programs, including 5,000 malicious programs and 5,000 malware. The malware is from the Drebin [17] dataset. Benign apps are among the top 500 apps in each category on the Google Play Store. In this regard, the declared accuracy in the test set is 94.62%.

MAMA technique detects executable Android malicious apps. The manifest file of Android applications is analyzed to extract application properties and permissions using the "Android Asset Packaging" tool. The extracted features are used to build a supervised machine-learning classifier to identify malicious programs [17].

In another study [18], DT, Naive Bayes, and Random Forest Machine Learning algorithms were used to detect Android attacks. An information acquisition method was used to select important features. The random forest algorithm achieved 94.6% accuracy.

In [19], they presented an ensemble hybrid model called SEDMDroid to detect sophisticated Android malware. The proposed framework used bootstrap and principal component analysis (PCA) to ensure diversity and generate random feature subspaces. Multilayer Perception (MLP) and Support Vector Machine (SVM) are trained to learn additional information that provides the final prediction. The test is performed on official Android apps, and the malicious apps are collected from the VirusShare repository. SEDMDroid achieved a recognition accuracy of 94.92%.

3. Methodology

This section describes the proposed methodology, data set, and preparation steps for the data set and the proposed model of this study.

3.1. *Method: XGBoost*

Extreme gradient boosting (XGB, also known as XGBoost) is another gradient boosting-based boosting method that uses more accurate approximations to generate the best prediction model (among gradient boosting models) [20]. The most apparent advantage of XGBoost, among other gradient-boosting methods, is its speed. The essence of this model is the construction of several CART trees. The model predicts each tree separately and finally combines the prediction results of each tree to achieve the final prediction value. Several weak learners are constructed by taking the decision tree as the base learner, and then the model is continuously trained in the gradient descent direction. Its structure is shown in Figure 1.

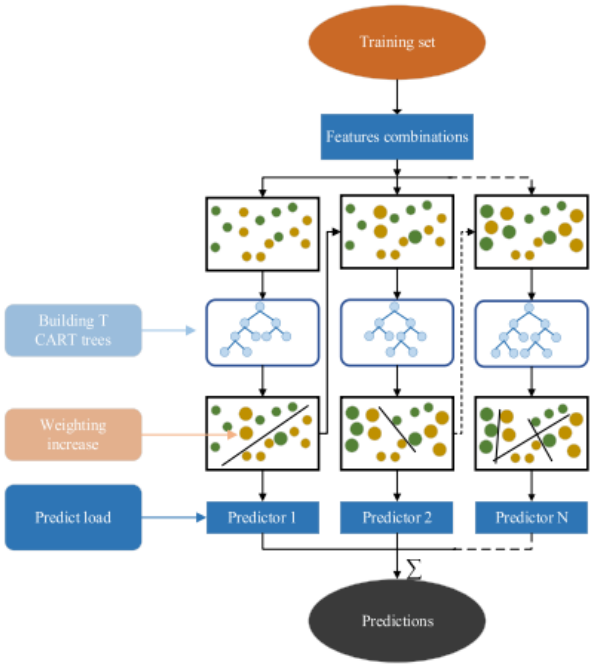

Figure 1. Schematic image of the XGBoost model

3.2. *Proposed Method*

This section explains the dataset in the first part, and then the proposed model for detecting Android malware is presented in the second part.

3.2.1. *Data Collection*

In this section, the data set used in this research is reviewed. The desired data set is collected from 3 different data repositories. These 3 data repositories are:

1. BazarFarsi: This dataset contains 294 samples downloaded from Cafe Bazar.
2. LibreAV: This dataset contains 8315 samples.
3. OmniDroid: This dataset contains 21992 instances of AndroPyTool [21].

3.2.1.1 *The Initial Raw Dataset*

30601 primary raw data were considered to build the proposed model. Table 1 shows the names of the data warehouses used in the model-building process. The initial number of data related to each data warehouse is given in the Number column. Our data is APK.

Table 1. Initial raw dataset

Repository name	Type	Number	Total of each Type
BazarFarsi	Benign	294	4598
LibreAV	Benign	4304	
OmniDroid	Benign/Malware	21992	21992
LibreAV	Malware	4011	4011

3.2.2. *Data Preprocessing*

This section deals with data preprocessing and preparing them for training the artificial intelligence model. The preprocessing step performs the following steps for both Benign and Malware categories.

1. First, the VirusTotal site checks the entire data set, which includes 30,601 samples, to ensure the accuracy of each sample's labeling.
2. If noisy data is detected in the previous step, it is removed from the data set in this step.
3. In this step, features are extracted by AndroPyTool Framework, and if a sample is broken and cannot be extracted, it is removed from the dataset.
4. Finally, the data set required for training the artificial intelligence model is obtained from the difference of the total data with the noisy and corrupted data.

3.2.2.1 *The Final Dataset*

Finally, in Table 2, you can see the dataset used to build the artificial intelligence model of this research. The number of each category is written according to the name of the corresponding data warehouse.

Table 2. The final data set of this study

Repository name	Type	Number	Total of each Type
BazarFarsi	Benign	280	13784
LibreAV	Benign	4017	
OmniDroid	Benign	9487	
LibreAV	Malware	2750	11770
OmniDroid	Malware	9020	

As seen in Table 2, our data set includes a total of 25554 samples.

3.2.2.2 *Feature Extraction*

In this article, AndroPyTool is used to extract static features. There are many tools for extracting features from Android applications. However, a program that can group individually obtained results to build a complete dataset is rare, which is the motivation behind using AndroPyTool.

AndroPyTool is an integration framework developed in Python that aims to extract diverse features from various Android applications. It embeds the most widely used Android malware analysis tools, inspects the source code, and retrieves behavioral information when the sample is run in a controlled environment. The tool provides a detailed report for each application analyzed, including a large set of features. In this study, 1000 features were considered for training the artificial intelligence model.

3.2.3. *Modeling*

Machine learning methods are generally divided into descriptive and predictive categories [22]. In this article, 7 different methods are used to predict malware. In the discussion section, the accuracy of these algorithms is compared with each other. Also, this section will explain the XGBoost model designed to achieve the highest accuracy.

3.2.3.1 *Proposed Model*

This section presents the proposed model of this study to predict Android malware. Figure 2 shows the proposed model. The XGBoost algorithm was used to implement the proposed model in an attempt to detect malware on Android devices using a machine learning approach. The proposed model is tuned to perform effectively on mobile devices with limited computing resources. Experiments

show that the proposed model works efficiently and effectively on low-end mobile devices. With the proposed model, it is possible to scan all the programs installed on the device quickly.

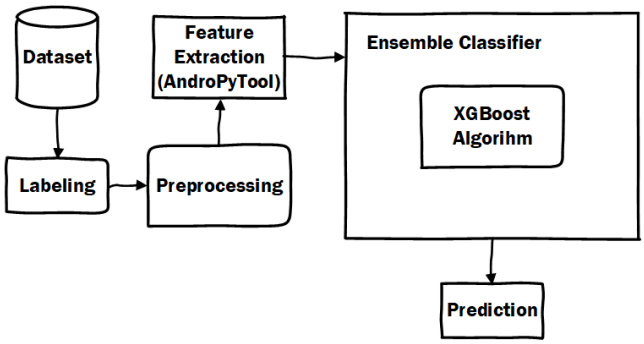

Figure 2. The proposed model of this study

4. Discussion

The dataset used in this study includes 25554 different samples. 1000 essential features have been selected from each sample's Permissions and Intents features. AndroPyTool was used to extract each sample's characteristics. The Pycharm tool and Python programming language were also used to conduct experiments.

In this study, we considered 80% of the data for model training and the other 20% for testing. Also, in this research, eleven different algorithms were used to predict the label of the dataset. Then, four criteria- accuracy, correctness, recall, and F1 score- were used to compare and evaluate the performance of each algorithm. Algorithms used in this study are Linear Regression, Decision Tree, KNN, Gradient Boosting, XGBoost, Support Vector Machine, and Logistic Regression.

In Table 3 and Figure 3, you can see the results of comparing the used algorithms.

Table 3. Check the performance of the algorithms used using the data set used

Algorithms	Accuracy	Precision	Recall	F1-Score
Linear Regression	91.6416115453998%	94.8855989232840%	87.4689826302729%	91.0264686894770%
Decision Tree	91.4010823812387%	90.9765142150803%	91.3151364764268%	91.1455108359133%
KNN	93.2651834034876%	94.6015424164524%	91.3151364764268%	92.9292929292929%
Gradient Boosting	92.6638604930847%	94.4155844155844%	90.1985111662531%	92.2588832487309%
XGBoost	94.9869031870114%	95.4468710089399%	92.4317617866005%	93.7696664568911%
SVM	92.3631990378833%	95.2063914780292%	88.7096774193548%	91.8432883750802%
Logistic Regression	93.0246542393265%	95.3947368421052%	89.9503722084367%	92.5925925925926%

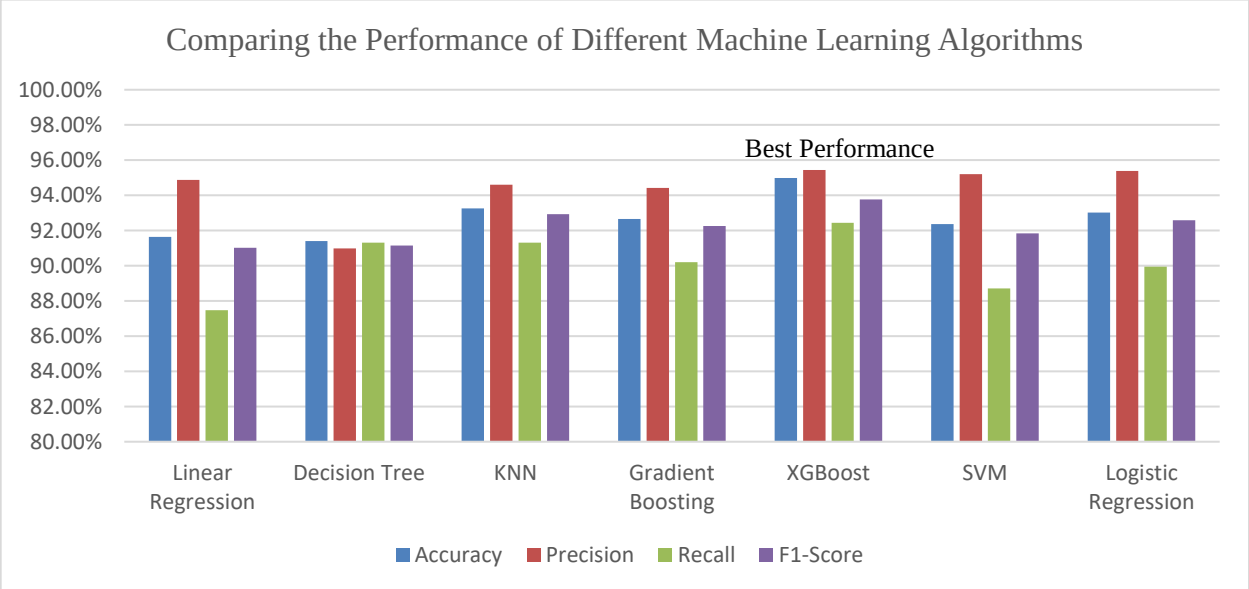

Figure 3. Performance review of the used algorithms using the data set of this study.

According to the results obtained above, the XGBoost model (proposed model), KNN, and Logistic Regression have the best performance compared to other algorithms. These three models perform well in all four evaluation criteria.

Because the XGBoost model had the best performance, this model is considered the final model. In the following, the proposed model of this research was examined with the proposed model of study [19]. A static malware detection framework called SEDMDroid was proposed in a 2020 study by Zhu et al. Their study used 2 different data sets to evaluate the model. Their first dataset consisted of 2000 real programs; the second dataset was MUDFLOW [23] with 17000 samples. Also, this framework is a two-layer architecture, including group learning. They considered MLP to be the base learners and SVM to be the output model. Finally, their proposed model achieved 89.07% accuracy for the first data set and 94.92% accuracy for the second data set.

After comparing this study's proposed model with the model proposed by Zhu et al., it can be seen that our proposed model uses a larger data set, performs better, and achieves an accuracy of 94.98%.

5. Conclusions

Smartphones are becoming increasingly popular and are a lucrative target for hackers due to their vulnerability to security breaches. Android is an open gateway for attackers who exploit it with malicious apps and take advantage of the system's security flaws. ML-based solutions have been proposed and implemented to address this critical issue. Another reason that makes using ML algorithms important in malware detection is their ability to detect zero-day attacks. ML-based approaches can at least detect previously unseen malware and, therefore, have the potential to prevent zero-day attacks.

In this work, we focused on the static analysis of Android applications. We intensively tested 25,554 Android apps and 1,000 features to detect Android app malware using 7 machine learning algorithms, among which XGBoost had the best performance. This showed the competitive results of the proposed model compared to existing approaches. Our approach provides an overall accuracy of 94.98%, precision of 95.44%, F1-score of 93.76%, and recall rate of 92.43%. In future work, we will focus on analyzing more features of Android apps, and larger datasets will be considered. Also, as a future work, this research can pay attention to other artificial intelligence algorithms, such as neural networks, which are widely used today.

Authors' Contributions

HM contributed to design, AAK and MB native RL algorithm, HM, AAK, MB bio-statistical analysis; MB participated in most of the study steps. AAK used RL in exploring drug combinations. All authors have read and approved the content of the manuscript.

Declaration

In order to correct and improve the academic writing of our paper, we have used the language model ChatGPT.

Transparency Statement

Data are available for research purposes upon reasonable request to the corresponding author.

Acknowledgments

We would like to express our gratitude to all individuals helped us to do the project.

Declaration of Interest

The authors declare that they have no conflict of interest. The authors also declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

According to the authors, this article has no financial support.

Ethical Considerations

The study placed a high emphasis on ethical considerations. Informed consent obtained from all participants, ensuring they are fully aware of the nature of the study and their role in it. Confidentiality strictly maintained, with data anonymized to protect individual privacy. The study adhered to the ethical guidelines for research with human subjects as outlined in the Declaration of Helsinki.

References

[1] A. Muzaffar, H. R. Hassen, H. Zantout, and M. A. Lones, "A Comprehensive Investigation of Feature and Model Importance in Android Malware Detection," 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2301.12778.

[2] S. O'Dea, "Forecast number of mobile users worldwide from 2020 to 2025," 2020. [Online]. Available: https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/.

[3] I. U. Haq, T. A. Khan, A. Akhunzada, and X. Liu, "MalDroid: Secure DL-enabled intelligent malware detection framework," *IET Communications*, vol. 16, no. 10, pp. 1160-1171, 2022, doi: 10.1049/cmu2.12265.

[4] A. Fournier, F. El Khoury, and S. Pierre, "Classification method for malware detection on android devices," in *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3*: Springer International Publishing, 2021, pp. 810-829.

[5] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Explainable AI for Android Malware Detection: Towards Understanding Why the Models Perform So Well?," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, 2022, pp. 169-180, doi: 10.1109/ISSRE55969.2022.00026.

[6] K. Beckert-Plewka, H. Gierow, V. Haake, and S. Karpenstein, "G DATA Mobile Malware Report: Harmful Android Apps Every Eight Seconds," 2020.

[7] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "Drebin: Effective and explainable detection of android malware in your pocket," in *NDSS*, 2014, vol. 14, pp. 23-26, doi: 10.14722/ndss.2014.23247.

[8] B. Yu, Y. Fang, Q. Yang, Y. Tang, and L. Liu, "A survey of malware behavior description and analysis," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, pp. 583-603, 2018, doi: 10.1631/FITEE.1601745.

[9] Z. Liu, R. Wang, N. Japkowicz, D. Tang, W. Zhang, and J. Zhao, "Research on unsupervised feature learning for android malware detection based on restricted Boltzmann machines," *Future Generation Computer Systems*, vol. 120, pp. 91-108, 2021, doi: 10.1016/j.future.2021.02.015.

[10] M. Yang, X. Chen, Y. Luo, and H. Zhang, "An Android Malware Detection Model Based on DT-SVM," *Security and Communication Networks*, vol. 2020, no. 1, p. 8841233, 2020, doi: 10.1155/2020/8841233.

[11] M. N. AlJarrah, Q. M. Yaseen, and A. M. Mustafa, "A context-aware android malware detection approach using machine learning," *Information*, vol. 13, no. 12, p. 563, 2022, doi: 10.3390/info13120563.

[12] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P. G. Bringas, and G. Álvarez, "Puma: Permission usage to detect malware in android," in *International Joint Conference CISIS'12-ICEUTE 12-SOCO 12 Special Sessions*: Springer Berlin Heidelberg, 2013, pp. 289-298.

[13] N. Peiravian and X. Zhu, "Machine learning for android malware detection using permission and API calls," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence*, 2013, pp. 300-305, doi: 10.1109/ICTAI.2013.53.

[14] J. Jung, K. Lim, B. Kim, S. J. Cho, S. Han, and K. Suh, "Detecting malicious android apps using the popularity and relations of APIs," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2019, pp. 309-312, doi: 10.1109/AIKE.2019.00062.

[15] W. Li, J. Ge, and G. Dai, "Detecting malware for android platform: An SVM-based approach," in *2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing*, 2015, pp. 464-469, doi: 10.1109/CSCloud.2015.50.

[16] A. Fournier, F. El Khoury, and S. Pierre, "A client/server malware detection model based on machine learning for android devices," *IoT*, vol. 2, no. 3, pp. 355-374, 2021, doi: 10.3390/iot2030019.

[17] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A survey on android malware detection techniques using machine learning algorithms," in *2019 Sixth International Conference on Software Defined Systems (SDS)*, 2019, pp. 110-117, doi: 10.1109/SDS.2019.8768729.

[18] J. Toldinas, A. Venčkauskas, R. Damaševičius, Š. Grigaliūnas, N. Morkevičius, and E. Baranauskas, "A novel approach for network intrusion detection using multistage deep learning image recognition," *Electronics*, vol. 10, no. 15, p. 1854, 2021, doi: 10.3390/electronics10151854.

[19] H. Zhu, Y. Li, R. Li, J. Li, Z. You, and H. Song, "SEDMDroid: An enhanced stacking ensemble framework for Android malware detection," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 984-994, 2020, doi: 10.1109/TNSE.2020.2996379.

[20] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp. 1189-1232, 2001, doi: 10.1214/aos/1013203451.

[21] A. Martín, R. Lara-Cabrera, and D. Camacho, "Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset," *Information Fusion*, vol. 52, pp. 128-142, 2019, doi: 10.1016/j.inffus.2018.12.006.

[22] S. Nazari Nezhad, M. H. Zahedi, and E. Farahani, "Detecting diseases in medical prescriptions using data mining methods," *BioData Mining*, vol. 15, no. 1, p. 29, 2022, doi: 10.1186/s13040-022-00314-w.

[23] V. Avdiienko *et al.*, "Mining apps for abnormal usage of sensitive data," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 2015, vol. 1, pp. 426-436, doi: 10.1109/ICSE.2015.61.