Online-Journals.org

iJIM International Journal of Interactive Mobile Technologies

iJIM | eISSN: 1865-7923 | Vol. 18 No. 21 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijim.v18i21.49669

PAPER

An Adaptive Framework for Classification and Detection of Android Malware

Ashraf Al Sharah[1], Yousef Abu Alrub[2], Hamza Abu Owida[3], Esraa Abu Elsoud[4](✉), Nawaf Alshdaifat[5], Hamzah Khtatnaha[2]

[1]Department of Electrical Engineering, College of Engineering Technology, Al-Balqa Applied University, Amman, Jordan

[2]Department of Computer Information Systems, Faculty of Prince Al-Hussein Bin Abdallah II for Information Technology, The Hashemite University, Zarqa, Jordan

[3]Department of Medical Engineering, Faculty of Engineering, Al-Ahliyya Amman University, Amman, Jordan

[4]Department of Computer Science, Faculty of Information Technology, Zarqa University, Zarqa, Jordan

[5]Faculty of Information Technology, Applied Science Private University, Amman, Jordan

eabuelsoud@zu.edu.jo

ABSTRACT

The hardware and software of a computer are controlled by its operating system (OS), which performs essential tasks such as input and output processing, file and memory management, and the management of peripheral devices such as disk drives and printers. Application software refers to programs designed for specific purposes, these applications, often freely available and open source, contribute to the rising number of downloads. In the third quarter of 2022, combined downloads from the Apple App Store and Google Play Reached an estimated 35.3 billion. However, the prevalence of insecurity in these applications and technologies heightens the potential for cybercrimes. Protection against unauthorized intruders is crucial in identifying malicious applications. Machine learning (ML) serves as a promising avenue for detecting malware attacks, offering potential solutions to bolster cybersecurity measures. We propose a novel approach utilizing ML to enhance malware detection accuracy by segmenting datasets into distinct groups. Our research employs supervised ML techniques on the CICMaldroid2020 dataset, which includes comprehensive information such as intent actions, permissions, and sensitive APIs. The dataset was partitioned into four groups, each containing 150 features, and analyzed across four experiments to distinguish between attack and benign classes. Our proposed model demonstrated exceptional performance, with the random forest algorithm achieving an accuracy of 98.6% and a precision of 98.75%. These results highlight the effectiveness of our segmentation approach and its significant contribution to advancing malware detection in Android applications, offering a promising direction for future cybersecurity solutions.

KEYWORDS

Android operating system, machine learning (ML), malware, adware, banking, SMS malware, riskware, CICMaldroid2020, malware detection

1 INTRODUCTION

In recent years, the use of Android phones and tablets has increased, which led to an increase in the volume of downloading applications. Some of these downloaded

Al Sharah, A., Alrub, Y.A., Owida, H.A., Elsoud, E.A., Alshdaifat, N., Khtatnaha, H. (2024). An Adaptive Framework for Classification and Detection of Android Malware. *International Journal of Interactive Mobile Technologies (iJIM)*, 18(21), pp. 59–73. https://doi.org/10.3991/ijim.v18i21.49669

Article submitted 2024-05-13. Revision uploaded 2024-08-16. Final acceptance 2024-08-18.

applications could be malicious and pose risks to users, such as phishing, banking trojans, spyware, etc. The relationship between malware and anti-malware is an inverse relationship that has led to the evolution of malware makers and advocates of systems targeted by malware for decades. The major point of malware detection is how to identify malicious software so that you can take the appropriate action to deal with it. Detection methods are divided into signature-based, change-based, and skew-based [1]. The potential of machine learning (ML) for identifying and classifying Android malware has been acknowledged. Particularly, the handcrafted features play a significant role in the typical ML-based Android malware identification techniques [2]. These characteristics describe what computer security professionals view as the most important inherent traits of Android malware [3]. Most ML-based solutions, however, rely heavily on the experience, level of competence, and breadth of subject knowledge of security specialists to manually define the traits that characterize Android malware [4].

The techniques for detecting malware on Android can be categorized as static, dynamic, or hybrid [5]. Static detection focuses on scrutinizing questionable code without the Android app. This approach achieves extensive code coverage but encounters various challenges, such as dynamic code loading and obfuscation. Conversely, dynamic identification involves executing the code to validate the software application. This method exposes vulnerabilities that are difficult to identify using static analysis but can be detected dynamically with relatively limited computational resources and time costs. Hybrid detection represents a technique for balancing detection performance and effectiveness by amalgamating dynamic and static detection [6].

Our study showed that the random forest technique outperformed other ML models tested, achieving a remarkable accuracy score of 98.6% in detecting Android malware. This algorithm's remarkable accuracy highlights its potential as a trustworthy malware detection tool, which is important given how quickly threats in the Android ecosystem are evolving. By employing ensemble learning techniques, our approach improves detection performance and provides an acceptable answer to the persistent issues caused by malware on Android devices. The overarching goal is to develop a model that can identify malware and reduce the dangers associated with malicious applications. The rest of the paper is divided into sections as follows: The next section presents a literature review related to Android malware detection. Section 3 outlines our suggested way of operation, how the study's findings arrived at, and how the model was constructed. Section 4 discusses and analyzes the results while Section 5 concludes the study with the direction of the contribution provided.

2 LITERATURE REVIEW

Android is considered the most popular operating system. Despite the pandemic, its popularity continues to grow, as it has penetrated most nations, including Turkey, Iran, Indonesia, India, and Brazil [7]. The potential of ML for identifying and classifying Android malware has been acknowledged. The design of various tools and approaches still heavily relies on human intelligence, even though numerous ML techniques have been put out. Particularly, the handcrafted features play a significant role in the typical ML-based Android malware identification techniques [2]. This section will mention some research that uses ML to detect Android malware.

The authors in [8] present an innovative combination of convolutional neural networks (CNN) and long short-term memory (LSTM) to detect and predict Android malware. To address the issue of imbalanced datasets, a unique ML strategy is

proposed to improve the performance of the method and alleviate concerns of over- and under-fitting during classification. By extracting both temporal and spatial features from the dataset, the suggested model effectively identifies Android malware. Empirical results show that the hybrid ConvLSTM model achieves exceptional performance, with an accuracy, sensitivity, and AUC of 0.99 each. This model surpasses traditional ML algorithms and provides a promising framework for real-time Android malware detection. However, the evaluation's reliance on the Drebin dataset may limit the generalizability of the findings to other datasets or real-world scenarios. Furthermore, the paper lacks a discussion on potential challenges or limitations in implementing the proposed ML strategy for handling imbalanced datasets, necessitating further research to assess its effectiveness across different contexts.

Unlike previous publications, this one [9] provides a full summary of the data pretreatment approach, the current feature subset selection patterns, the deficiencies, and difficulties in Android malware detection, as well as the problems. The Android OS mechanism and threat categorization techniques are also briefly introduced. a comprehensive examination of current improvements in feature extraction and malware detection methods.

[10] This paper presents a novel machine-learning strategy based on classifiers for the identification of Android malware. Applying 27 criteria from the CICMalDroid 2020 Dataset, the researchers in this system evaluate each Android APK to identify whether it is legitimate software or malicious malware. Each APK is submitted to a blacklisting tool that has been pre-selected. The app will be classified as potentially hazardous if it is discovered there; otherwise, it will be submitted to 4 l ML classification algorithms for classification: RF, NB, J48, and IBK (KNN). The classification metrics of recall, precision, and accuracy utilizing the confusion matrix were used by the researchers to assess each classifier's accuracy in detecting malware. The results demonstrate that the random forest classifier outperforms the rivals in rates of malware identification, with an accuracy rate of 98.6%.

Zhu et al. [11] suggested an ensemble classifier stacking technique for malware identification. The suggested method begins by employing static analysis to retrieve properties. To ensure individual diversity, the training dataset of the model is divided into subgroups, and each subgroup is subsequently submitted to principal component evaluation. The architecture for the stacking ensemble learner has two tiers. Multilayer perceptrons are classifiers that serve as the base learners in the first tier, and SVM is employed in the second layer to create prediction accuracy. Applying two sets of data, the authors evaluated the model's performance. The model's accuracy was 87.89% for the initial sample, which was an actual sample with multi-level characteristics, and 97.04% for the second dataset.

In [12], the CICMALDroid2020 dataset was utilized. For classifying Android malware, the researchers proposed a semi-supervised deep neural network approach. Pseudo-labels that included both unlabeled sample and labeled sample instances were used to train the model. The need to include unlabeled samples in the detection methods was highlighted by this since they are more accessible and less expensive. Their approach has a maximum detection performance of 0.967 after being tested with numerous hidden nodes and hidden neuron counts.

2.1 Background

In this section, we present the Android system and explain how it works, and the architecture of the system Android.

Android architecture. The architecture of Android OS remains largely unchanged, although it is being updated frequently. The Android architecture can be divided into different layers, namely the modified Linux kernel layer, libraries, system runtime library layer, and application framework layer [9]. Figure 1 illustrates these layers.

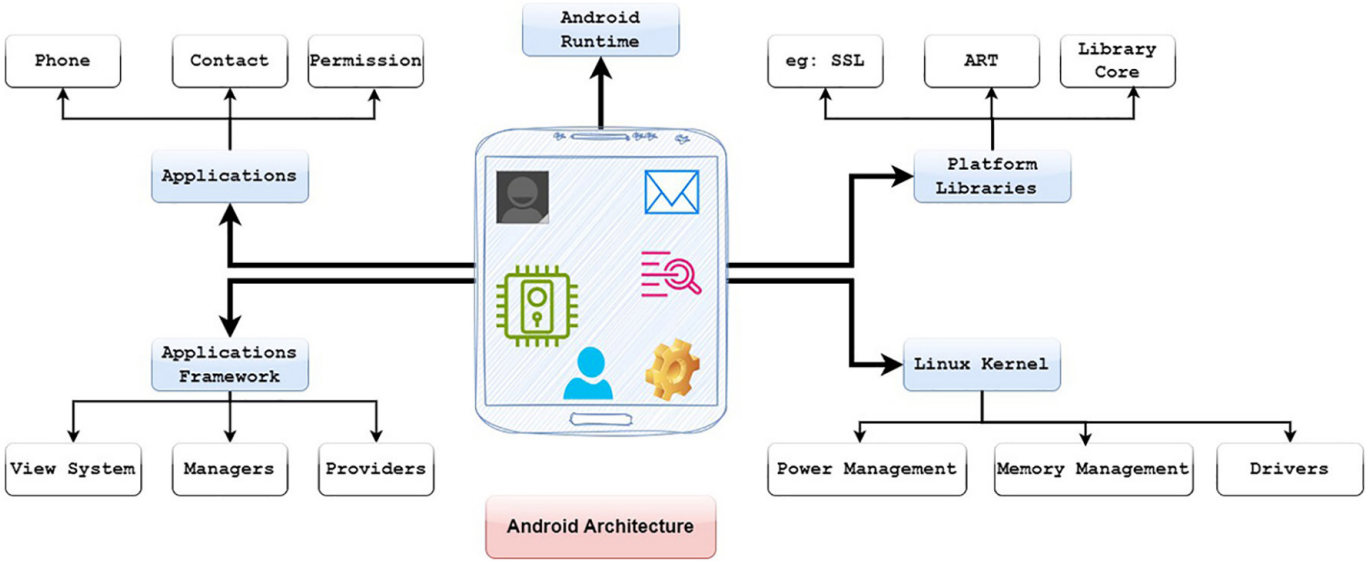

Fig. 1. Architecture of an Android OS

Based on the technique of feature extraction using static features and dynamic features mobile malware detection technology may be split into dynamic analysis, static analysis, and hybrid analysis.

- Dynamic analysis involves monitoring the data flow of an application while it is being executed [13].
  - Static analysis, conversely, involves extracting data from Android files such as opcode sequences, API calls, and requested permissions [9].
  - Hybrid analysis combines both dynamic and static analysis to improve the accuracy and efficiency of Android malware detection [9].

Malware. Malware is classified based on how it spreads and its actions once it infects a machine [14]. The classification of malware includes rooted or group malware, which aims to cause as much harm as possible to victim PCs and is relatively easier to guard against. On the other hand, targeted or clever malware, also known as advanced persistent threats (APTs), employs advanced cyber-security techniques and persistently attempts to gain access while causing serious harm to the intended victim [15].

2.2 Android malware detection

The three phases of Android malware detection include data gathering, threat intelligence, and malware detection. For effective virus identification, Android apps with a balance of good and bad instances should be gathered. The data collection should include examples of the different malware families observed in the field. The Android OS supports programs packaged as Android application package (APK). In our work, ML will be used to classify malicious and benign programs with many famous algorithms such as RF, DT, KNN, etc. [16].

3 METHODOLOGY

Due to the increase of malware on devices running the Android OS, this study is intended to help reduce such malware by detecting it and helping to prevent such attacks. Figure 2 presents the methodology followed in this study.

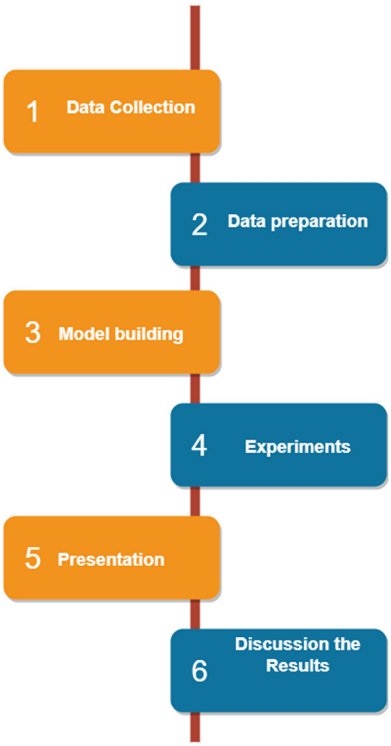

Fig. 2. The proposed methodology

3.1 Dataset

In our research endeavor, we utilized the CICMalDroid 2020 dataset [12] [17], which is a novel, sufficiently extensive, and diverse compilation of data aimed at facilitating the detection of malware. The researchers were able to procure over 17, 341 Android samples from various sources, including the virus total service, Contagion's security blog, AMD, MalDozer, and other datasets. The time frame for data collection spanned from December 2017 to December 2018, resulting in a total of 11,598 instances and 471 features. The dataset is categorized into five distinct groups, namely riskware, adware, SMS malware, banking malware, and Benign [12]. The researchers acquired samples that encompassed comprehensive static and dynamic features. This cybersecurity dataset serves as a ranking mechanism for Android applications in terms of their susceptibility to malware and provides viable strategies for countermeasures and mitigation techniques.

- Adware: Mobile adware refers to advertisements that are clandestinely embedded within genuine applications that have been infected with malware.
- Banking malware: Mobile banking malware is an intricately designed form of malware that mimics authentic banking programs or banking websites to gain unauthorized access to a user's online banking accounts.

- SMS malware: SMS malware intercepts SMS payloads and utilizes the SMS service as a conduit for executing attacks. Before linking the SMS to the malware, attackers upload it to their hosting sites.
- Mobile riskware: Riskware is a term denoting legitimate software that can be exploited by malicious users.
- Benign: Applications that do not fall within any of the categories are deemed benign, indicating their non-malicious nature.

3.2 Dataset pre-processing

The preprocessing steps of the data encompassed an initial screening for noise, which revealed the absence of any noisy data, thereby eliminating the necessity for further noise removal techniques. The dataset was subsequently normalized. However, it was observed that there existed an inherent imbalance between the benign and malicious apps, which prompted the application of preprocessing techniques to enhance accuracy. After the implementation of feature selection, the SMOTE (synthetic minority oversampling technique) was utilized to address the class imbalance. The selection of SMOTE was based on its extensive usage and its straightforward algorithm for generating synthetic data samples. Consequently, the balance of the dataset improved, with the number of cases rising from 11,598 to 17,926 after balancing. These steps played a crucial role in ensuring the effectiveness of the ML models applied to the dataset. By dividing the data into four segments based on the malware category and treating each segment as an independent dataset, we are essentially tailoring the analysis to the distinctive characteristics of each type of malware. This enables a more focused and potentially more effective application of the model to each subset of data, thereby enhancing the overall strength and accuracy of the analysis. It exemplifies a well-thought-out and systematic approach to dealing with different types of malwares within the dataset, which can contribute to the advancement of methodologies for detecting and classifying malware.

3.3 Dataset training and testing

The experimental results were examined using the K-fold cross-validation technique, which separates the examples into several subgroups where the variable K defines the number of folds. In this study, the K parameter was set to 10.

3.4 Evaluation measures

Evaluation measures, alternatively referred to as performance metrics, are employed to evaluate the efficacy and precision of a model or system in resolving a specific task [18]. Accuracy is a commonly used statistic for evaluating model performance. However, it is not always a reliable predictor of performance. It measures the proportion of correct predictions to total predictions. Recall, also known as sensitivity or true positive rate, is the percentage of actual positive instances correctly identified by the model. Precision, also known as positive predictive value. F-measure is the harmonic mean of precision and recall. It is a measure of overall model performance, with a higher F1 score indicating better performance. If either precision or recall is low, the F1 score will decrease significantly. Therefore, a model

with accurate positive predictions and a low number of missed positive cases (recall) performs well in terms of the F1 score.

4 EXPERIMENT AND RESULTS

In this section of our study, we designed our experiments to assess the effectiveness of our approach in detecting malware. The experiments were categorized based on the types of malicious programs, resulting in four distinct sets of experiments that included both malicious and benign programs. Our objective was to compare the performance of our model across different types of malwares by applying the same algorithms in all experiments. Once these experiments are executed and the results are obtained, we will thoroughly analyze and present our findings.

Additionally, we explain the environment utilized for conducting these experiments. Our experimentation relied on the usage of the Waikato environment for knowledge analysis (Weka) version 3.8.6, which spanned a period from 1999 to 2022. The computational setup consisted of a 64-bit OS running on an x64-based processor, specifically Windows 10 Pro version 21H1 with OS build 19043.1889. The hardware configuration included an Intel Core i5-3230M CPU operating at 2.60GHz and 6 GB of random access memory (RAM). These details provide a comprehensive understanding of the computational resources employed in our experimentation and offer context for the interpretation of our results.

4.1 Results of adware experiment

In this experiment, we assessed the performance of various ML algorithms in the classification of Adware malware and benign instances. The outcomes, which are summarized in Table 1 and depicted in Figure 3, reveal noteworthy disparities in accuracy, precision, recall, and F-measure across the algorithms. random forest achieved the highest accuracy, attaining a rate of 98.788%, while the naive bayes (NB) algorithm produced the lowest accuracy at 87.77%. Similarly, random forest also exhibited superior precision (98.8%), with NB falling behind at 88.3%. Concerning recall, random forest outperformed NB, accomplishing a rate of 98.8% compared to NB's 87.8%. Furthermore, random forest excelled in F-measure, registering a percentage of 98.8%, while NB lagged with 87.7%. These results emphasize the efficacy of random forest in precisely classifying Adware malware instances, highlighting its potential for robust malware detection.

Table 1. Adware result

Algorithms	Accuracy	Precision	Recall	F-Measure
RF	98.7884	0.988	0.988	0.988
J48	96.9428	0.969	0.969	0.969
DT	96.7033	0.967	0.967	0.967
KNN	94.9281	0.95	0.949	0.949
SVM	90.0958	0.912	0.901	0.9
NB	87.7712	0.883	0.878	0.877

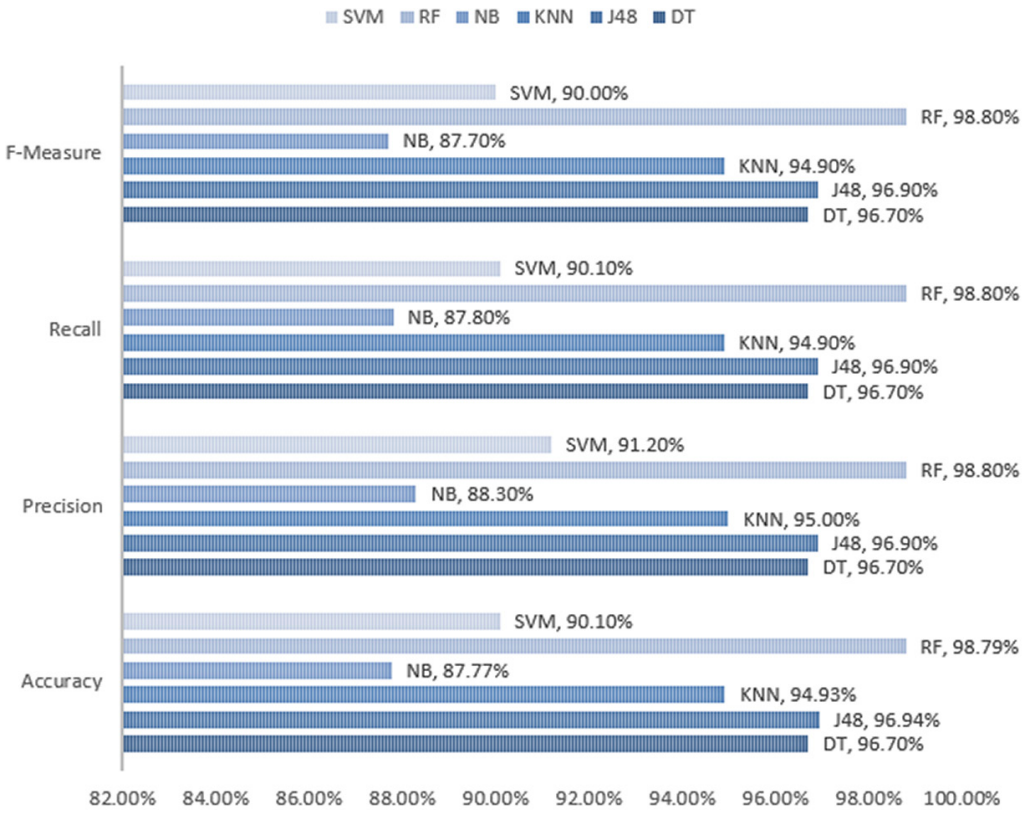

Fig. 3. Adware result

4.2 Results of banking experiment

In this experiment, we investigated the performance of ML algorithms in classifying banking malware and benign instances. The results, depicted in Table 2 and illustrated in Figure 4, demonstrate varying levels of accuracy, precision, recall, and F-measure across the algorithms. Notably, the random forest algorithm achieved the highest accuracy at 98.2878%, while the NB algorithm yielded the lowest accuracy of 87.2374%. Regarding precision, random forest again emerged as the top performer with a precision of 98.3%, contrasting with NB's precision of 88%. Similarly, random forest exhibited superior recall (98.3%) compared to NB's 87.2%. Moreover, random forest excelled in F-measure, registering a percentage of 98.3%, while NB lagged with 87.2%. These findings highlight the efficacy of the random forest algorithm in accurately classifying banking malware instances, suggesting its potential as a robust tool for malware detection.

Table 2. Banking result

Algorithms	Accuracy	Precision	Recall	F-Measure
RF	98.2878	0.983	0.983	0.983
J48	96.5612	0.966	0.966	0.966
DT	95.9137	0.959	0.959	0.959
KNN	95.9281	0.959	0.959	0.959
SVM	88.3022	0.894	0.883	0.883
NB	87.2374	0.88	0.872	0.872

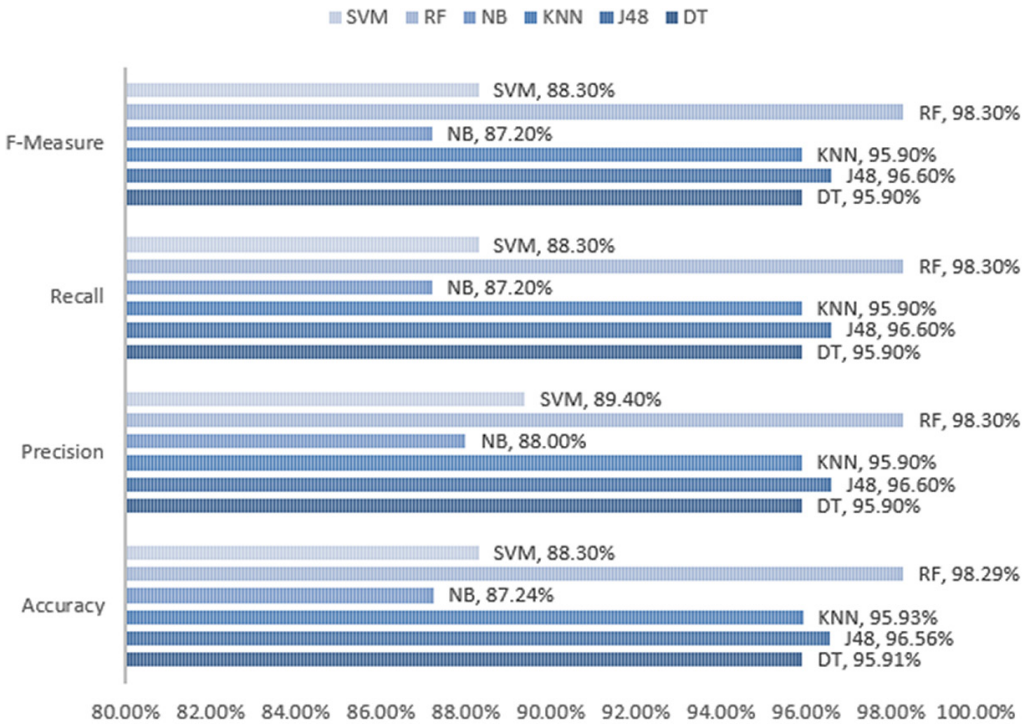

Fig. 4. Banking result

4.3 Results of riskware experiment

In this experiment, we investigated the performance of ML algorithms in the classification of riskware malware and benign instances. The outcomes, presented in Table 3 and illustrated in Figure 5, demonstrate diverse levels of accuracy, precision, recall, and F-measure across the algorithms. Notably, the random forest algorithm achieved the highest level of accuracy at 97.9312%, while the NB algorithm produced the lowest level of accuracy, amounting to 82.8488%. Concerning precision, random forest once again emerged as the top performer, exhibiting a precision of 97.7%, in contrast to NB's precision of 83.3%. Likewise, random forest demonstrated superior recall (97.7%) compared to NB's recall of 82.8%. Furthermore, random forest excelled in F-measure, registering a percentage of 97.7%, while NB lagged at 82.8%. These findings highlight the effectiveness of the random forest algorithm in accurately classifying instances of riskware malware, suggesting its potential as a robust tool for the detection of malware in this category.

Table 3. Riskware result

Algorithms	Accuracy	Precision	Recall	F-Measure
RF	97.9312	0.979	0.979	0.979
J48	96.2259	0.962	0.962	0.962
DT	95.3313	0.953	0.953	0.953
KNN	94.7582	0.948	0.948	0.948
SVM	87.4616	0.875	0.875	0.875
NB	82.8488	0.833	0.828	0.828

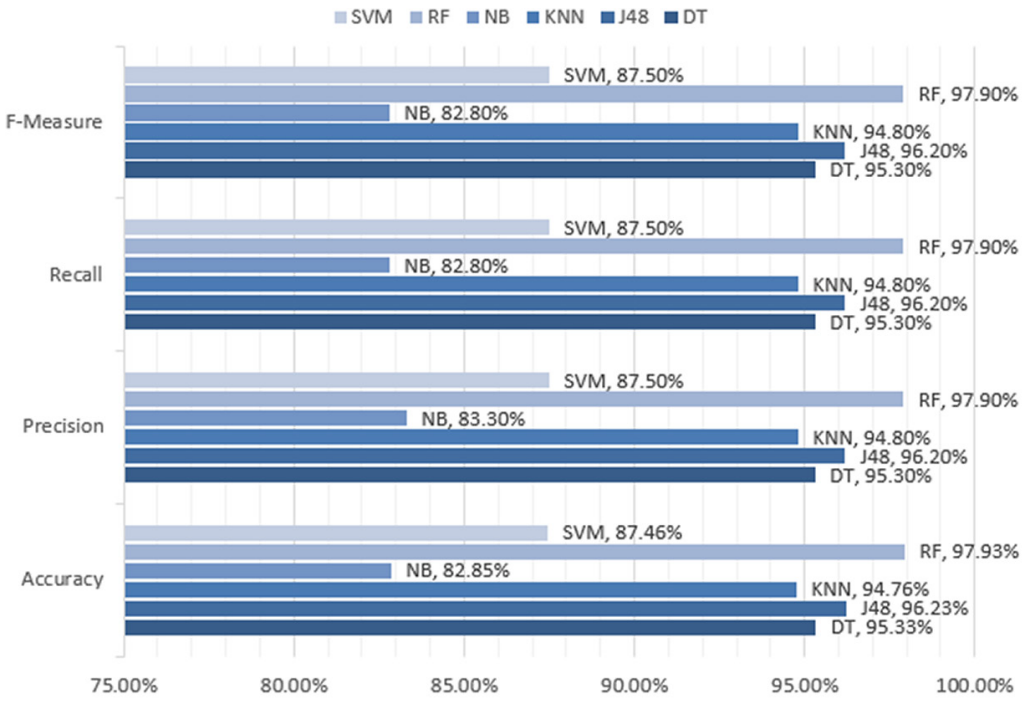

Fig. 5. Riskware result

4.4 Results of SMS malware experiment

In this experiment, an investigation was conducted on the performance of ML algorithms in the classification of SMS malware and benign instances. The outcomes, which are outlined in Table 4 and depicted in Figure 6, highlight significant disparities in accuracy, precision, recall, and F-measure among the algorithms. Remarkably, the random forest algorithm attained the highest level of accuracy, achieving a remarkable 99.3251%, while the NB algorithm exhibited the lowest accuracy at 93.1%. It is worth noting that random forest also demonstrated superior precision with a precision of 99.3% in comparison to NB's precision of 93.6%. Furthermore, random forest displayed a higher recall of 99.3% as opposed to NB's 93.1%. In addition, random forest excelled in F-measure, registering a percentage of 99.3%, while NB lagged behind at 93.1%. These observations emphasize the effectiveness of the random forest algorithm in accurately classifying SMS malware instances, thus suggesting its potential as a robust tool for SMS malware detection.

Table 4. SMS malware result

Algorithms	Accuracy	Precision	Recall	F-Measure
RF	99.3251	0.993	0.993	0.993
J48	98.9685	0.99	0.99	0.99
DT	98.5993	0.986	0.986	0.986
KNN	98.7648	0.988	0.988	0.988
SVM	93.6457	0.941	0.936	0.936
NB	93.1109	0.936	0.931	0.931

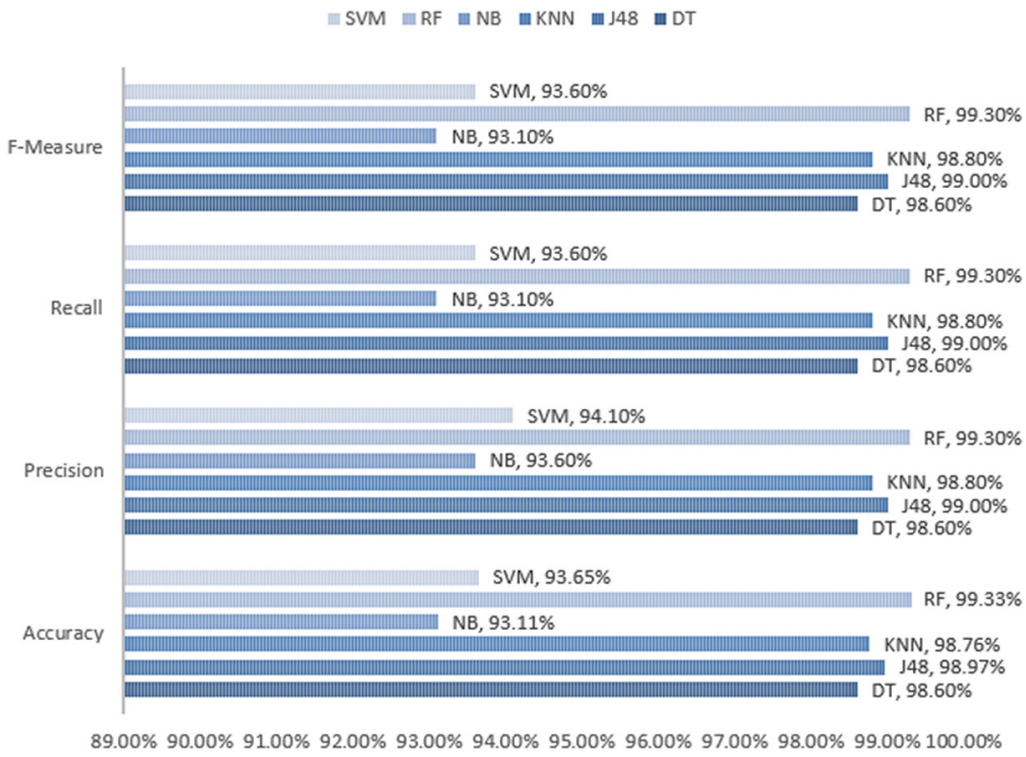

Fig. 6. SMS malware result

4.5 Results of averaged for all experiment

In this extensive analysis, we have consolidated the outcomes from all experiments involving adware, banking, riskware, SMS malware, and benign instances. The findings, which are presented in Table 5 and illustrated in Figure 7, provide valuable insights into the overall performance of ML algorithms across a range of malware categories.

Table 5. Average of all results

Algorithms	Accuracy	Precision	Recall	F-Measure
RF	98.58	98.75	98.75	98.75
J48	97.17	97.17	97.17	97.17
DT	96.63	96.6	96.6	96.6
KNN	96.1	96.1	96.1	96.1
SVM	89.87	90.55	89.87	89.85
NB	87.74	88.3	87.7	87.7

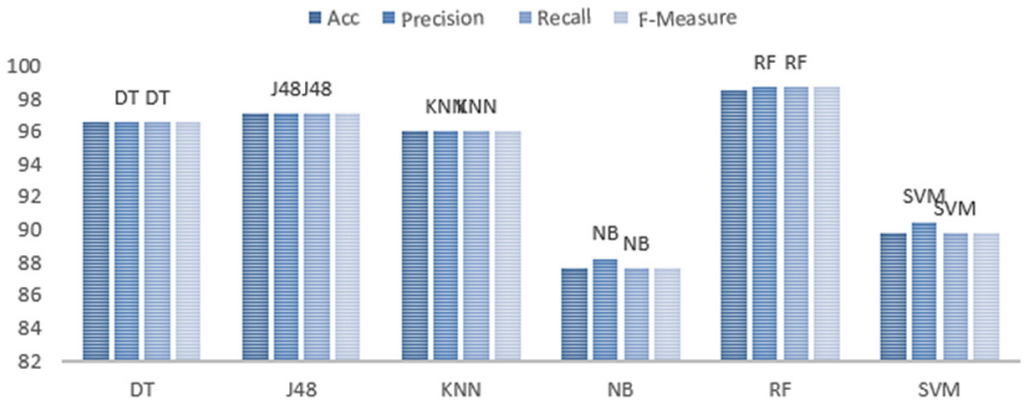

Fig. 7. Averaged all result

The random forest algorithm emerged as the leading performer in terms of overall accuracy, achieving an impressive accuracy rate of 99.3251%. Conversely, the NB algorithm displayed the lowest overall accuracy at 93.1109%. This significant difference highlights the superior capability of random forest in accurately categorizing malware instances across multiple categories.

Furthermore, when examining precision, random forest once again demonstrated superiority with an overall precision rate of 99.3%, while NB lagged behind with a precision rate of 93.6%. This indicates that random forest is more effective in minimizing false positives, thereby enhancing the reliability of malware classification.

Similarly, random forest exhibited a higher overall recall rate (99.3%) compared to NB's 93.1%, indicating its ability to capture a greater proportion of true positives across various malware types. This implies that random forest is more skilled at identifying malicious instances, making it a more suitable choice for robust malware detection.

Moreover, when evaluating the overall F-measure, random forest outperformed NB with a score of 99.3% compared to NB's 93.1%. The F-measure provides a balanced assessment of both precision and recall, highlighting random forest's superiority in achieving a harmonious balance between these metrics across different malware categories.

To summarize, the consolidated results emphasize the effectiveness of the random forest algorithm in accurately classifying malware instances across diverse categories, showcasing its potential as a powerful tool for comprehensive malware detection and classification.

4.6 Comparing our approach with previous works

This section presents the results of our comparison analysis with previous studies on Android OS malware detection. Similarities and differences in dataset utilization and performance across different methodologies were assessed in the comparison, which is outlined in Table 6. Our approach, utilizing the random forest algorithm, achieved an accuracy of 98.6%, aligning closely with the findings of [10]. However, our approach surpassed [10] in precision, achieving a higher precision of 98.75%. Notably, our approach exhibited the best accuracy among all compared works, highlighting its efficacy in accurately detecting malware in Android systems.

Table 6. Comparing with previous works

Reference	RF				J48			
	AC	PR	Recall	F-M	AC	PR	Recall	F-M
[10]	98.6051	98.6	98.6	X	96.9207	96.9	96.9	X
[19]	97.5	97.5	97	97	X	X	X	X
[20]	X	X	X	X	X	X	X	X
Our Approach	98.6	98.75	98.75	98.75	97.17	97.17	97.17	97.17
Reference	**DT**				**KNN**			
	AC	PR	Recall	F-M	AC	PR	Recall	F-M
[10]	X	X	X	X	92.6097	92.7	92.6	X
[19]	X	X	X	X	84	81	80	80
[20]	88	80	92.4	75.2	85	82.2	82.8	82.6
Our approach	96.63	96.6	96.6	96.6	96.1	96.1	96.1	96.1
Reference	**SVM**				**NB**			
	AC	PR	Recall	F-M	AC	PR	Recall	F-M
[10]	X	X	X	X	50.658	58.9	50.7	X
[19]	45	62	34	31	53	44	57	45
[20]	86.6	91.6	88.2	X	X	X	X	
Our approach	89.87	90.55	89.87	89.85	87.74	88.3	87.7	87.7

In contrast, when considering the support vector machine (SVM) algorithm, the accuracy achieved by [19] was notably lower at 45% compared to our approach, which achieved an accuracy of 89.87%. This substantial difference underscores the superiority of our approach in leveraging the SVM algorithm for malware detection.

Similarly, when evaluating the NB algorithm, our approach outperformed the findings of [10], achieving an accuracy of 87.74% compared to their accuracy of 50.658%. This significant difference further highlights the effectiveness of our approach in utilizing the NB algorithm for malware detection.

Overall, our approach demonstrates superior performance across various algorithms compared to previous works, showcasing its potential as an advanced and effective solution for malware detection in Android OS.

5 CONCLUSION AND FUTURE WORK

In conclusion, this study offers a thorough analysis of the use of ML techniques for malware detection in Android OS. We have shown via thorough experimentation and analysis that our approach is effective in correctly classifying both benign and malicious instances of a variety of malware, including Adware, banking malware, riskware, and SMS malware.

Our results show that the random forest method is the best at obtaining high recall, accuracy, precision, and F-measure for all types of malwares. This demonstrates random forest's ability to reliably identify and classify malware cases, which makes it the recommended option for malware detection tasks. Moreover, our comparison

analysis with earlier works demonstrates the advancements achieved with our approach, with superior outcomes seen in many algorithms when compared to past research. This confirms that our approach is effective in handling an evolving variety of malware threats on Android devices.

6 REFERENCES

[1] T. Alsmadi and N. Alqudah, "A survey on malware detection techniques," in *2021 International Conference on Information Technology (ICIT)*, Amman, Jordan, 2021, pp. 371–376. https://doi.org/10.1109/ICIT52682.2021.9491765

[2] J. Liu, J. Zeng, F. Pierazzi, L. Cavallaro, and Z. Liang, "Unraveling the key of machine learning solutions for android malware detection," *arXiv preprint arXiv:2402.02953*, 2024. https://doi.org/10.48550/arXiv.2402.02953

[3] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers & Security*, vol. 89, p. 101663, 2019. https://doi.org/10.1016/j.cose.2019.101663

[4] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android malware detection with deep neural models," *ACM Computing Surveys (CSUR)*, vol. 53, no. 6, pp. 1–36, 2020. https://doi.org/10.1145/3417978

[5] F. Taher, O. AlFandi, M. Al-kfairy, H. Al Hamadi, and S. Alrabaee, "Droiddetectmw: A hybrid intelligent model for android malware detection," *Applied Sciences*, vol. 13, no. 13, p. 7720, 2023. https://doi.org/10.3390/app13137720

[6] A. S. de Oliveira and R. J. Sassi, "Chimera: An android malware detection method based on multi-modal deep learning and hybrid analysis," *Authorea Preprints*, 2023. https://doi.org/10.36227/techrxiv.13359767.v1

[7] C. Sahin, "Do popular apps have issues regarding energy efficiency?" *PeerJ Computer Science*, vol. 10, p. e1891, 2024. https://doi.org/10.7717/peerj-cs.1891

[8] A. Wajahat *et al.*, "An adaptive semi-supervised deep learning-based framework for the detection of android malware," *Journal of Intelligent & Fuzzy Systems*, vol. 45, no. 3, pp. 5141–5157, 2023. https://doi.org/10.3233/JIFS-231969

[9] L. Meijin *et al.*, "A systematic overview of android malware detection," *Applied Artificial Intelligence*, vol. 36, no. 1, 2022. https://doi.org/10.1080/08839514.2021.2007327

[10] A. Droos, A. Al-Mahadeen, T. Al-Harasis, R. Al-Attar, and M. Ababneh, "Android malware detection using machine learning," in *2022 13th International Conference on Information and Communication Systems (ICICS)*, 2022, pp. 36–41. https://doi.org/10.1109/ICICS55353.2022.9811130

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

[12] S. Mahdavifar, A. F. A. Kadir, R. Fatemi, D. Alhadidi, and A. A. Ghorbani, "Dynamic android malware category classification using semi-supervised deep learning," in *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. 515–522. https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00094

[13] Z. Salah and E. Abu Elsoud, "Enhancing network security: A machine learning-based approach for detecting and mitigating krack and kr00k attacks in IEEE 802.11," *Future Internet*, vol. 15, no. 8, p. 269, 2023. https://doi.org/10.3390/fi15080269

[14] M. S. Akhtar and T. Feng, "Evaluation of machine learning algorithms for malware detection," *Sensors*, vol. 23, no. 2, p. 946, 2023. https://doi.org/10.3390/s23020946

[15] A. Y. Alhusenat, H. A. Owida, H. A. Rababah, J. I. Al-Nabulsi, and S. Abuowaida, "A secured multi-stages authentication protocol for IoT devices," *Mathematical Modelling of Engineering Problems*, vol. 10, no. 4, pp. 1352–1358, 2023. https://doi.org/10.18280/mmep.100429

[16] S. Selvaganapathy, S. Sadasivam, and V. Ravi, "A review on android malware: Attacks, counter-measures and challenges ahead," *Journal of Cyber Security and Mobility*, vol. 10, no. 1, pp. 177–230, 2021. https://doi.org/10.13052/jcsm2245-1439.1017

[17] S. Mahdavifar *et al.*, "Effective and efficient hybrid android malware classification using pseudo-label stacked auto-encoder," *Journal of Network and Systems Management*, vol. 30, 2022. https://doi.org/10.1007/s10922-021-09634-4

[18] R. Mohan, *Measurement, Evaluation and Assessment in Education*. Delhi, India: PHI Learning Pvt. Ltd., 2023.

[19] M. F. Ahmed *et al.*, "Shieldroid: A hybrid approach integrating machine and deep learning for android malware detection," in *2022 International Conference on Decision Aid Sciences and Applications (DASA)*. 2022, pp. 911–916. https://doi.org/10.1109/DASA54658.2022.9764984

[20] S. E. Mohamed, M. Ashaf, A. Ehab, O. Shereef, H. Metwaie, and E. Amer, "Detecting malicious android applications based on API calls and permissions using machine learning algorithms," in *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, 2021, pp. 1–6. https://doi.org/10.1109/MIUCC52538.2021.9447594

7 AUTHORS

Ashraf Al Sharah is with the Department of Electrical Engineering, College of Engineering Technology, Al-Balqa Applied University, Amman, Jordan.

Yousef Abu Alrub is with the Department of Computer Information Systems, Faculty of Prince Al-Hussein Bin Abdallah II for Information Technology, The Hashemite University, Zarqa, Jordan.

Hamza Abu Owida is with the Department of Medical Engineering, Faculty of Engineering, Al-Ahliyya Amman University, Amman, Jordan.

Esraa Abu Elsoud is with the Department of Computer Science, Faculty of Information Technology, Zarqa University, Zarqa, Jordan (E-mail: eabuelsoud@zu.edu.jo).

Nawaf Alshdaifat is with the Faculty of Information Technology, Applied Science Private University, Amman, Jordan.

Hamzah Khtatnaha is with the Department of Computer Information Systems, Faculty of Prince Al-Hussein Bin Abdallah II for Information Technology, The Hashemite University, Zarqa, Jordan.