PRO-Net: A Novel Framework for Augmenting Android Security against Botnets and Malware through Advanced Detection Metrics

Iqra Pervaiz[*], Rutaba Irfan[†], Mujeeb-Ur-Rehman[‡], Zoraiz Ali[§]

Abstract

Android is a popular smartphone operating system, which dominates the market with a global share of approximately 70.29%. Over 255 billion applications are available on the official Play Store, with many more available from other sources. Android is the leading platform for smartphone applications, with an increase in the number of applications available. However, the demand for the Android operating systems has also attracted the attention of malicious software developers. A growing number of attackers are targeting mobile devices, converting them into bots for their operations. This enables cybercriminals to gain control of compromised devices, establishing networks known as botnets. These botnets are then utilized to execute harmful activities such as Distributed Denial-of-Service (DDoS) attacks, stealing sensitive data and spamming. Unfortunately, some malicious apps are designed specifically for Android systems to perform different types of offenses, such as worms, exploits, trojans, rootkit viruses etc. These applications are often delivered in various versions to target a larger audience, making them difficult to detect. As the safety of the Android operating system is crucial, Machine Learning (ML) and Deep Learning (DL) algorithms alone are not enough. Therefore, a new PRO-Net system has been devised to protect against data breaches. The proposed framework, PRO-NET, is evaluated using precision, accuracy, and F1 score metrics. The study reveals that the system provides symmetry between apps and malware, which is essential for maintaining the security of the Android operating system.

Keywords: Android Application; Malware Detection; Static Malware Analysis; Machine Learning Algorithms; Deep Learning.

Introduction

Nowadays the smartphone market is experiencing significant sustainable growth, with 4.98 billion people worldwide using mobile devices (Gupta, 2018) (Mat S. R., 2022). Android-based botnets are increasingly employed to attack specific devices.

In Distributed Denial-of-Service (DDoS) attacks, these botnets flood the target system with an excessive volume of requests, effectively

[*]Department of Computer Science, University of Management and Technology, Sialkot 51310, Pakistan, iqrapervez12345@gmail.com

[†]Department of Computer Science, University of Management and Technology, Sialkot 51310, Pakistan, Rutabairfan1@gmail.com

[‡]Corresponding Author: Department of Computer Science, University of Management and Technology, Sialkot 51310, Pakistan, mujeeb.rehman@skt.umt.edu.pk

[§]Department of Computer Science, University of Management and Technology, Sialkot 51310, Pakistan, zoraiz.ali@skt.umt.edu.pk

preventing legitimate traffic from accessing the system and resulting in service disruptions and system failures. (Clarke, 1986). To protect against these attacks, machine learning techniques have demonstrated their effectiveness in identifying and monitoring threats within the Internet of Things (IoT) (Alothman, 2017; Andersen, 2015).

Malicious programs are pieces of code designed to steal user information and damage systems. These programs can be classified into two types: threats that require host programs and threats that are independent of each other (Chen, 2021). Android, being the most extensively used operating system in the smartphone market, is often targeted by scammers. Hackers have developed and distributed a variety of Android malware using modern techniques. The research suggests that by 2024, there will be more than 1.2 billion dangerous apps on the Android platform, with over 11,500 new cases of malware appearing every day. Continuous investigations in this area are crucial. Android malware is malicious apps that can harm Android devices and users in various ways. These include data encryption or destruction, credential theft, data leakage, injecting malicious code into legitimate apps, and changing device settings (Alkahtani, 2022).

Malware has the potential to infiltrate networks, damage critical infrastructures, compromise computers and smart devices, and steal sensitive information (Rathore, 2018). An analysis tool called LimonDroid developed by (Tchakount, 2021), aims to identify malicious characters in Android apps. Malware can penetrate networks, threaten essential infrastructure, compromise both computers and smart devices, and extract sensitive data. The remarkable advancement of technology, along with digitalization, cloud and edge computing (Hartmann, 2022; Jamsa, 2022), quantum computing (Gill, 2022), and the widespread adoption of numerous connected devices (Priyadarshini, 2022), has resulted in unprecedented levels of cybercriminal activities. Studies on malware detection using machine learning are gaining popularity due to their successful strategy, which can achieve a high level of detection accuracy (Mat S. R., 2021). Malicious programs are software applications created to steal user information and disrupt computer systems through various attacks. These programs can be broadly categorized into two types: host programs and independent threats, depending on how they operate. Their behavior further differentiates them, including propagation, remote control, and direct attack methods. Among these categories, there are specific types of malicious software that are commonly found, such as RiskTool applications, mobile banking trojans, and mobile ransomware trojans. Each malicious program poses a unique threat to computer

systems and users, requiring constant vigilance and robust security measures to effectively mitigate their impact (Cinar, 2023).

To remove threats, the concept of permission-based detection is explored and determine how it can be utilized alongside artificial intelligence algorithms to detect and prevent malicious attacks on Android systems. The permission-based mechanism works as a background process that detects malicious APKs by performing both dynamic and static examinations. The dynamic analysis examines the behavior of applications during their execution, while the static analysis scrutinizes the source code, bytes, or application binaries to identify any potential security vulnerabilities (Akbar, 2022).

Static analysis is used to flag an application as malicious based on an estimation of its potential runtime behaviors. These estimations are usually derived from methods such as permission analysis, code analysis, and API analysis. Android employs a permission-based security model to secure user data or prevent apps from accessing sensitive user data. Permissions in apps are commonly demoralized as they are regarded as one of the most important security evaluation methods for the Android platform. Therefore, without explicit permission, it is impossible to carry out a management action. This makes authorization perusing a crucial component in the process of detecting malware. Android apps request permissions before they can function and provide their features to users. When combined, multiple permissions can indicate certain negative behaviors. With the static analysis, it is needed to parallel attach with the dynamic analysis (Akbar, 2022).

As it is discussed in Figure 1, the general process of extracting malware from Android apps is using some effective machine and deep learning algorithms. To distinguish between "attack" and "normal" applications by identifying their static and dynamic features separately. Various machine and deep learning models such as support vector machine (SVM), Naive Bayes (NB), neural networks (NN), long short-term memory (LSTM), convolutional neural networks (CNN), and hybrid models are evaluated using these features (Alkahtani, 2022).

Detecting malware is a major security concern for companies, as it can have legal, reputational, and financial implications. One promising approach to improving malware detection systems is through deep learning, but this method presents several challenges. These include selecting features based on correlation, implementing the solid coating model, and utilizing the LSTM model. Each of these approaches is complex and difficult to implement, but they can significantly enhance performance (Alomari, 2023).

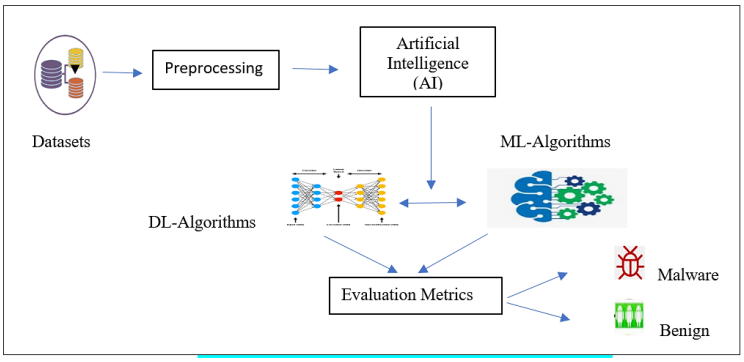

Figure 1: Overview of the proposed framework for malware detection in Android systems

Besides correlation, other than relationship, Quantum innovation has a few conceivable outcomes to upgrade AI capabilities and quicken development by preparing huge sets of information, tackling complex issues speedier, and joining different sets of information. This innovation brings counterfeit insights into a modern period in terms of execution speed and information handling, empowering AI to handle more complex issues. Quantum innovation gives phenomenal energy to fathom issues that require serious calculations, which are becoming progressively challenging as more complex information and connections are included inside the factors. Large-scale quantum computing combined with fake insights is a major insurgency for cybersecurity. Be that as it may, cybercriminals also use powerful quantum computing capabilities to carry out malicious cyber activities with devastating effects (Djenna, 2023).

Now it is required to explore and evaluate the effectiveness of different machine (ML) and deep learning (DL) algorithms in detecting mobile malware outbreaks. This research aims to offer the best model for monitoring Android applications against malicious attacks. It is essential to achieve the following goals in study:

A. It is needed to create an intrusion detection system for the Android platform utilizing a range of machine learning and deep learning algorithms

B. A comparison of the tested algorithms is presented with different art models.

Literature Review

It is needed to discuss the permission-based approach, which acts as a basic part of Android's security model, overseeing the entrance privileges of applications to different gadget assets and functionalities. Customary permission-based approaches include static investigation of an

application's proclaimed authorizations to evaluate its potential security gambles. It proposes a permissions-based malware detection system called PerDRaML, designed to address the issue of malware applications targeting Android devices (Akbar, 2022).The difference between useful and unrelated permissions for benign apps and how the Android open-source policy and integration for unofficial app stores make them vulnerable to malicious intrusions. The proposed system uses a multi-level approach that involves extracting features from a dataset of 10,000 applications and utilizing ML models to classify apps as either malicious or benign.

The new method is inspired by (Zhu, 2018), and employs a permission-based detection strategy to identify malicious APKs. The goal of this new approach is to enhance the detection of malicious APKs while minimizing the number of permissions needed for classification. Unlike the (Zhu, 2018), which only utilizes Support Vector Machine (SVM) and Rotation Forest classifiers, the proposed strategy incorporates Support Vector Machine (SVM), Naïve Bayes (NB), and Random Forest (RF) classifiers for classification. The selection of permissions is based on their significant impact on virus detection effectiveness. The following are the main components of the planned research:

Gathering Both Benign and Malicious APKs; Developing/Determining the Features Set; Refinement, Completion, and Acquisition of Permissions (Features) Dataset; Classifying Android Malware Using Supervised ML Algorithms (Akbar, 2022).

Extensive experimentation has demonstrated PerDRaML's ability to detect malware with high accuracies, optimize the feature set, and improve evaluation metrics compared to existing techniques. The system employs various ML models, such as SVM, RF, and NB, which achieve accuracies of 89.7%, 86.25%, and 89.52%, respectively. The proposed system also optimizes the feature set by up to 77% compared to recent methods, while improving evaluation metrics such as precision, sensitivity, and accuracy (Akbar, 2022). It is required to compare multiple machine-learning and deep-learning algorithms using two datasets, to answer specific inquiry questions.

Which ML and DL algorithms are suitable for detecting Android malware?

What are the suggested ML and DL models validation accuracy, robustness, and efficiency in identifying Android malware? (Alkahtani, 2022).

Experimentations are conducted on two commonly used datasets: CICAndMal2017 and Drebin. The Drebin dataset comprises over 100,000 Android apps, containing both benign and malware samples, which are

widely used to guess the act of Android malware detection procedures. On the other hand, CICAndMal2017 consists of Android adware and ransomware samples from 2017, which provides a strong foundation for evaluating detection algorithms (Alkahtani, 2022). The combination of CICAndMal2017 and Drebin allows for a comprehensive evaluation of the PRO-Net framework. CICAndMal2017 offers insights into detecting specific categories like ransomware and adware, while Drebin tests the framework's ability to handle diverse malware families. This dual-dataset approach ensures that PRO-Net is assessed under varying conditions, validating its effectiveness and robustness.

This study analyzes the effectiveness of different models in identifying malicious Android packets using standard Android malware datasets. Three models, SVM, Linear Discriminant Analysis (LDA), and K-Nearest Neighbor (KNN) are used to achieve the objective. The network has a complex structure, and to achieve high accuracy, nonlinear models are proposed. The SVM algorithm shows the highest precision, achieving 100% results in overall performance measurements (Alkahtani, 2022).

However, the linear models such as LDA and KNN does not perform well in detecting Android malware. The accuracy of LDA is only 45.37% in the CICAndMal2017 dataset and enhances to 81.39% when using the Drebin dataset. The KNN model achieves just 82% accuracy with the Drebin dataset, indicating that both the LDA and KNN models are unsuitable for detecting Android malware (Alkahtani, 2022). It also examines the results of deep learning models using the AE mode in detecting mobile attacks. However, the results are not satisfactory. The AE achieves only 75.99% and 56.78% accuracy for the CICAndMal2017 and Drebin datasets, respectively (Alkahtani, 2022).

Alomari (2023) discusses how malware traffic is asymmetrical in nature, unlike benign traffic, which is symmetrical. Nonetheless, there are several artificial intelligence techniques that can be employed to detect malware and differentiate it from normal activities. But handling voluminous and high-dimensional data is still a challenge. The paper presents a high-performance malware detection system that uses deep learning and feature selection methodologies. To check its effectiveness, the authors employs two distinct sets of malware data to train and test deep models. The initial dataset comprises a large number of entries but had a limited set of attributes. In contrast, the second dataset features fewer entries but includes a vast number of attributes, resulting in a complex and high-dimensional structure.

Different researchers deal differently with malware as malware scanners and traditional antivirus solutions are no longer sufficient to protect against modern malware threats. To effectively predict and prevent

damage caused by malware, it is important to conduct a thorough examination of malware to create new and effective solutions. A systematic method is proposed that integrates dynamic deep learning-based techniques with heuristic approaches to classify and identify five families of malware—rootkits, adware, SMS malware, and ransomware—using the Android dataset (CICAndMal2017). It evaluates the enactment of future detection approaches using various estimate measures. The experimental results suggest that combining behavior-based DNN with a heuristic-based approach leads to better performance compared to using ML and DL methods alone (Djenna, 2023).

Various methodologies utilized in malware detection has been compared while highlighting their evaluation metrics, research gaps and their datasets. By using Android malware databases, Akbar (2022) has concentrated on selected permission sets to maximize malware detection rates and has ultimately achieved an accuracy of 89.70% and 89.96% accuracy with SVM and random forests, respectively. A mixed-method approach using CICAndMal2017, and Drebin datasets is employed by Alkahtani (2022) which yields a remarkable accuracy of 100% for SVM and 99.40% for LSTM, but this approach lacks of real time detection and feature engineering. Cinar (2023) has solved issues of user awareness and bias mitigation that evolved during emerging threats but has not defined a dataset. A combination of heuristic and behavior-based approaches, suggested by Djenna (2023), has detected five malware families using advancedCICAndMal2017, highlighting the need for larger datasets and advanced learning models.

This research aims to develop an efficient malware detection model that is both robust and requires low computational resources. To achieve this, a feature selection approach is employed that reduces the number of features and minimizes computational time. It is needed utilize both static and dynamic analysis techniques to enhance performance and detect advanced, complex malware in these applications. By combining deep learning, high performance, and feature selection, it is expected to introduce a novel malware detection model that surpasses previous studies. Feature selection and data preprocessing as recommended by Alomari (2023), is used to enhance malware detection with Android malware datasets, exploring the impact of feature selection on model performance.

Proposed Framework

While ensuring security, a technique called PRO-NET has been developed to detect Android malware among benign applications, which further incudes sub-modules discussed below.

Feature Extraction

Using signature-based and behavior-based classification methods, this method comprises extracting features from both benign and malicious data. In the signature-based approach, API features are extracted, while behavior-based data is transformed into binary data for feature extraction. It is needed to use dualistic grouping to determine whether an Android application is nonviolent or risky based on static features. It is required to monitor virus actions and perform dynamic analysis by running malware in a simulated sandbox environment for a few minutes.

The proposed PRO-Net framework has been fabricated with scalability and efficiency in mind to make sure its applicability across a wide range of Android devices, from high-end smartphones to low-resource devices. The following deliberations highlight its impact on real-world applications.

Low-Resource Device Compatibility

The framework utilizes lightweight static analysis techniques, such as permission and API call evaluations, which need minimal computational resources. Dynamic analysis is performed selectively, leveraging sandbox environments to ensure efficient runtime behavior monitoring without overburdening device hardware. These optimizations make PRO-Net viable for mid- and low-range devices.

High-End Smartphone Advantages: On high-end devices, PRO-Net utilizes its full potential by integrating more advanced dynamic analysis techniques, such as real-time monitoring of network traffic and in-depth behavioral assessments. This enhances detection accuracy and real-time response capabilities. The reason for this approach is to instruct a learning specialist to choose features for classifying progressively. The framework consecutively chooses features beneath the greedy technique until it comes to an end state. In Figure 2 (above), it is explained that the Android System uses the APK (Android Package Kit) file format to install and distribute apps on devices. APK files contain all the necessary components for a mobile application to install and run properly. The APK TOOL is used to decompile both malicious and helpful files to extract the required information. There are two groups of analysis in this record.

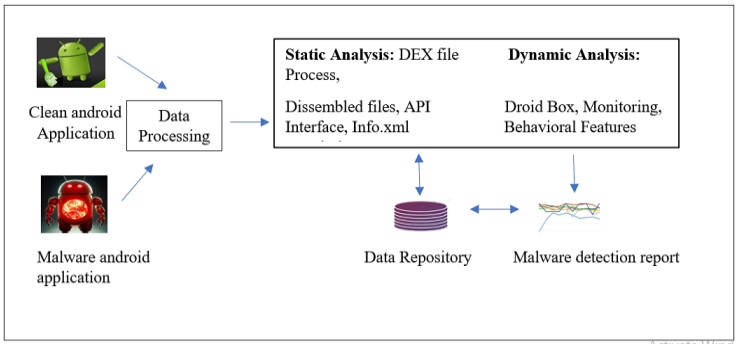

Figure 2: Feature extraction process for Android malware detection, highlighting binary and API Features.

Static Analysis

(a) Permissions: Applications need approval to access facts and features like cameras, storage, and calls. Always review these requests carefully before accepting them to ensure your privacy.

(b) Suspicious API calls: Unauthorized access to private data and resources leads to harmful behavior. It is important to ensure that such access is prevented and that proper security measures are in place to protect sensitive information.

(c) Without its execution and testing, static analysis inspects the application's code and associated metadata. This technique is computationally fast and efficient and identifies potential vulnerabilities or malicious behaviors in the application's structure.

Dynamic Analysis

This is very demanding type of analysis that concentrates on the characteristics that can be obtained by implementing the application. It is needed to indicate that it is possible to retrieve numerous dynamic features from Android based devices and applications. Dynamic analysis monitors the application's behavior during execution, often in a controlled environment like a sandbox. This method is effective in detecting runtime threats that static analysis misses most often.

Figure 3 portrays the progression of feature extraction and procedure utilized in the PRO-Net framework to classify Android applications as benign or malicious. This step is pivotal in ensuring that meaningful and relevant data is fed into the machine learning and deep learning models for accurate malware detection. The goal of the classification model is to predict a class label from a set of available options. There are two main types of class problems: multiclass classifications and binary classifications. When it comes to detecting

malware on Android, it is considered a binary classification problem. It is essential to use binary ordering to determine if an app is harmless or risky based on static features. Multiple techniques are then applied to verify accuracy. Different algorithms yield varying accuracy based on the feature extraction list. To achieve its objective through the use of binary arrangement. SVM excels in analyzing static features, providing high accuracy for detecting malware based on permissions, file properties, and API calls. LSTM complements SVM by focusing on dynamic features, such as runtime behavior and sequential API interactions, which are critical for detecting advanced and evasive malware.

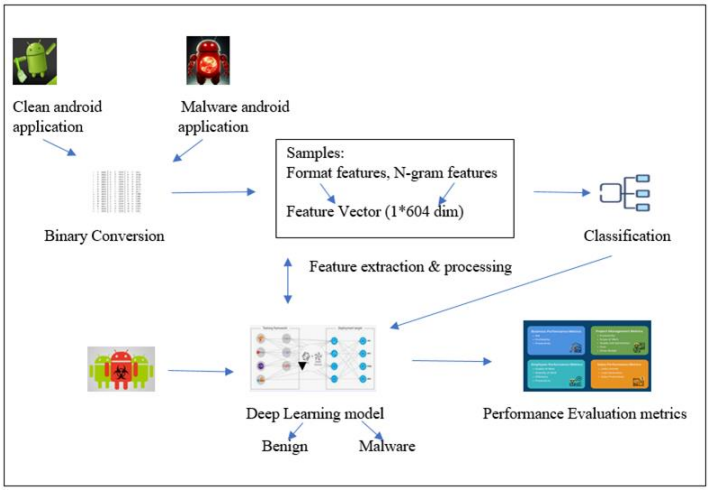

Figure 3: Classification model for distinguishing between benign and malicious Android applications.

After the feature selection step, the reliability of the detection model is based on the following parameters.

$$\text{Accuracy}=\frac{TP+TN}{TP+FP+TN+FN}\text{x}100\% \quad (1)$$

$$\text{Sensitivity}=\frac{TP}{TP+FN}\text{ x }100\% \quad (2)$$

$$\text{Precision}=\frac{TP}{TP+FP}\text{x }100\% \quad (3)$$

$$\text{F1-Score}=\frac{2xprecisionxsensitivity}{precision+sensitivity}\ x\ 100\% \quad (4)$$

The following terms are used to describe the accuracy of a testing sample. True positive (TP) denotes the number of positive samples correctly identified as positive. False positive (FP) refers to the number of negative samples incorrectly classified as positive. True negative (TN) represents the number of negative samples accurately identified as negative. False negative (FN) is the number of positive samples incorrectly

classified as negative. To attain high performance, it is essential to evaluate various machine learning and deep learning algorithms.

To evaluate our proposed framework, 'MalDroid' dataset is utilized which contains eighty-five instances whereas target column contains three different malware classes. We used ML (Random Forest) and DL (Artificial Neural Network) algorithms for classification. The selected dataset is preprocessed, unnecessary columns are removed and missing values are handled. Further, categorical data is converted into numeric form, finally, relevant features and target labels are separated to enhance the efficiency of algorithms.

Results and Discussion

The first step is the feature extraction and selection. It is an essential process that aims to choose the attributes with the highest accuracy while reducing complexity and avoiding overfitting. Historically, researchers have used various techniques to categorize features to identify malware in apps as shown in Table 1. For this attempt, the feature-overgrown strategy is specifically chosen which selects the necessary features to build malware detection models. The most highly ranked features are considered and defined.

Table 1: Sample of efficient feature selection)

Features	Analysis type	Feature type	Details
.DEX file	Dynamic	Behavior based	Process extraction
Task Intents	Static	Signature based	Internal words
Process ID	Static	Behavior based	Process tracking
SMS	Static	Signature based	Opcode verification
Power Usage	Static	Behavior based	Hashing tricks
Log files	Dynamic	Behavior based	Process monitor

The performance of the Random Forest and Artificial Neural Network (ANN) models is evaluated on the chosen dataset using four key metrics: Accuracy, Precision, Recall, and F1 Score. Below is an analysis of the results presented in Table 2.

Table 2: Performance Metrics Table.

Metric	Random Forest	ANN
Accuracy	88.89%	90.12%
Precision	89.18%	90.28%
Recall	88.89%	90.12%
F1 Score	88.77%	89.96%

Figure 4 compares visually these metrics for both models. Each group of bars represents a specific metric (Accuracy, Precision, Recall, and F1 Score), with one bar for Random Forest and other for ANN.

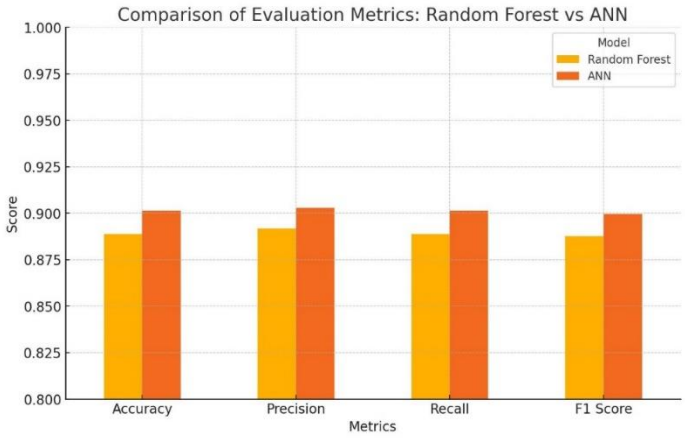

Figure 4: Comparative analysis using Machine learning and deep learning Models.

The analysis reveals that while both Random Forest and ANN are effective in classifying the dataset, ANN outperforms Random Forest in all evaluated metrics. This makes ANN a preferable choice for this classification task, as it captures complex patterns in the data slightly better.

By joining static and dynamic analysis, the PRO-Net framework compromises a significant improvement in Android malware detection that ensures compatibility with both high-end and low-resource devices. Contrasting existing methods like Zhu's static analysis or Akbar's permissions-based detection, which lack runtime behavior monitoring, PRO-Net integrates sandbox-based dynamic analysis to detect evasive malware more efficiently and effectively. Its feature extraction approach integrates lightweight and advanced techniques, addressing the weaknesses of models like Alkahtani's, which emphasize dataset diversity but overlook real-time detection. However, PRO-Net's reliance on binary classification may limit its ability to identify unknown malware families compared to heuristic or clustering methods, as suggested by Djenna. While its precision and scalability make it versatile and robust, further optimization for potential threats and dataset expansion could augment its generalizability and reliability.

Conclusion

The proposed PRO-NET framework employs a triple-feature extraction technique that helps to identify potentially malicious behavior in the Android environment. It doesn't rely on label datasets but also focuses on feature engineering techniques to extract relevant features such as file properties, system calls, API calls, network traffic patterns and permissions requested by applications. By cautiously choosing and designing these important features, the framework trains a model to detect suspicious patterns. Dynamic behavioral analysis techniques are used to observe the runtime behavior of applications, after analyzing static behavior. This technique can detect malicious activities based on actions performed by the application, such as unauthorized access to personal data or suspicious network communications. The proposed PRO-Net framework has shown promising results in detecting Android based malware. However, there are areas for further exploration: Incorporate additional datasets representing emerging malware threats to enhance model generalization, Optimize the framework for low-end Android devices, ensuring scalability and efficiency across varying hardware configurations. It is the need of hour to explore clustering techniques to identify unknown malware families without relying on labeled data. To improve detection accuracies, the theme is evaluated using both supervised and unsupervised deep learning algorithms and advanced technologies, with and without labeled data, to detect new and highly dangerous malware.

References

Akbar, F. a.-H. (2022). Permissions-based detection of android malware using machine learning. *Symmetry* ,14(4), 718.

Alkahtani, H. a. (2022). Artificial intelligence algorithms for malware detection in android-operated mobile devices. *Sensors*.

Alomari, E. S. (2023). Malware detection using deep learning and correlation-based feature selection. . *Symmetry*.

Alothman, B. a. (2017). Android botnet detection: An integrated source code mining approach. *12th International Conference for Internet Technology and Secured Transactions (ICITST)*, (pp. 111--115.).

Andersen, J. R. (2015). CAAL: concurrency workbench, aalborg edition. In Theoretical Aspects of Computing. *12th International Colloquium, Cali,* . Colombia.

Chen, L. a. (2021). Detection, traceability, and propagation of mobile malware threats. *IEEE Access*, ,9, 14576-14598.

Cinar, A. C. (2023). The current state and future of mobile security in the light of the recent mobile security threat reports. . *Multimedia Tools and Applications.*

Clarke, E. M. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. . *ACM Transactions on Programming Languages and Systems (TOPLAS).*

Djenna, A. a. (2023). Artificial intelligence-based malware detection, analysis, and mitigation. *Symmetry.*

Gill, S. S. (2022). Quantum computing: A taxonomy, systematic review and future directions. *Software: Practice and Experience.*, 52(1), 66-114.

Gupta, B. B. (2018). Advances in security and privacy of multimedia big data in mobile and cloud computing. *Multimedia Tools and Applications*, 9203-9208.

Hartmann, M. a. (2022). Edge computing in smart health care systems: Review, challenges, and research directions. *Transactions on Emerging Telecommunications Technologies.*

Jamsa, K. (2022). Cloud computing. *Jones \& Bartlett Learning*.

Mat, S. R. (2021). Towards a systematic description of the field using bibliometric analysis: malware evolution. *Scientometrics.*, 126: 2013-2055.

Mat, S. R. (2022). A Bayesian probability model for Android malware detection. *ICT Express*, 8, no. 3 (2022): 424-431.

Priyadarshini, S. B. (2022). *In The Role of the Internet of Things (Iot) in Biomedical Engineering: Present Scenario and Challenges.* Apple Academic Press.

Rathore, H. a. (2018). Malware detection using machine learning and deep learning. . *6th International Conference, BDA 2018, Warangal.* India.

Tchakount, F. a. (2021). LimonDroid: a system coupling three signature-based schemes for profiling Android malware. *Iran Journal of Computer Science.*

Poornima, S., & Mahalakshmi, R. (2024). Automated malware detection using machine learning and deep learning approaches for android applications. Measurement: Sensors, 32, 100955.

Zhu, H. J. (2018). DroidDet: effective and robust detection of android malware using static analysis along with rotation forest model. *Neurocomputing.*, 272, 638-646.