10.48047/jocaaa.2024.32.02.32

A UNIQUE MACHINE LEARNING METHOD FOR ANDRIOD MALWARE DETECTION USING FEATURE CO-EXSISTENCE

Mrs. M. MOUNIKA[1], CHELLUBOINA SHIRISHA[2], MUNAGAPATI SHEEVANI[3], TUPPATHI ANUSHA[4]

[1]Aiistant Professor, Department of Computer Science and Engineering, Sridevi Women's Engineering College, Hyderabad, India

Email: mounikcamanya@gmail.com,

[2,3,4,]B.Tech Student, Department of Computer Science and Engineering, Sridevi Women's Engineering College, Hyderabad, India

Abstract: The paper proposes a new machine learning model for detecting Android malware based on the coexistence of static features. The model assumes that Android malware requests an abnormal set of coexisted permissions and APIs compared to benign applications. To prove this assumption the paper created a new dataset of co-existed permissions and API calls at different levels of combinations and applied them to different APIs frequencies. The frequent pattern growth (FP-growth) algorithm was used to extract the most relevant co-existed features. The new datasets were extracted using Android APK samples from the Drebin, Malgenome and MalDroid2020 datasets. To evaluate the proposed model, several conventional machine learning algorithms such as Random Forest (RF), K- Nearest Neighbours (KNN), Logistic Regression (LR), Decision Tree (J48), and Support Vector Machine (SVM) were used. The maximum accuracy was achieved using the Random Forest algorithm and the co-existence of permissions features at the second combination level. The proposed approach outperformed the state-of-the-art models, and achieving an good accuracy using the Drebin and Malgenome datasets. The results show that using the features co-existence is an effective method to detect Android malware.

Index Terms *- Android malware detection, Machine learning model, Coexistence of static features, Permissions and APIs, Frequent Pattern Growth (FP-growth) algorithm, Dataset creation, Drebin dataset, Malgenome dataset, MalDroid2020 dataset, Random Forest algorithm.*

1. INTRODUCTION

Smartphone market is growing immensely. According to the ICD report. Among the several mobiles operating systems, Android is the dominant operating system with over 2.5 billion active users across over 190 countries. The wide range of capabilities offered by smartphones and the rising number of activities carried out by their users, including social networking, online banking, and gaming, has given rise to very serious concerns about device security and personal privacy. Since Android is an open-source platform, it is easy for malware developer to launch their attacks and develop Android malware apps that pose severe harm. Mobile malware is constantly updated with new features to evade detection by antimalware scanners. Android

10.48047/jocaaa.2024.32.02.32

malware applications usually use three types of breaks through techniques to get access to the user's devices. Repackaging, Update, Downloading. However, these apps are malicious and may harm their devices. So, Machine learning methods are applied to detect Android malware and distinguish them from benign ones without comparing the patterns of the known Android malware. Machine learning methods work smartly by building a model based on sample data, known as ''training data'', to make predictions or decisions without being explicitly programmed. The project focuses on the concept of feature coexistence, an innovative approach that considers the interactions and relationships among various features within Android applications. By analyzing the simultaneous presence or absence of multiple features, the detection model aims to provide a more nuanced understanding of application behaviour, leading to improved accuracy and adaptability. This project holds significant importance in the realm of cybersecurity and mobile device security. As Android malware continues to evolve in complexity, the development of an advanced detection system using feature co-existence is poised to make substantial contributions to user protection, privacy preservation, and the overall resilience of the Android ecosystem. The outcomes of this project aim to not only enhance security but also advance the broader field of machine learning in cybersecurity applications.

The intersection of mobile technology and cybersecurity has become a focal point in the contemporary digital landscape, especially with the widespread adoption of Android devices. As the Android ecosystem continues to thrive, so does the potential threat landscape associated with it. This compilation of literature delves into various aspects of Android security, machine learning applications, and malware analysis, providing a comprehensive overview of the challenges and advancements in the field. Elenkov's "Android Security Internals" [1] and Ng's "Machine Learning Yearning" [2] lay the groundwork for understanding the intricacies of Android security and the principles of machine learning, respectively. Building upon this foundation, Dunham and Adams contribute insights into the realm of Android malware and its analysis [3]. Zhou, Jiang, and Lopez explore novel approaches to malware detection through the analysis of Android Manifest files [4], shedding light on the technical nuances of identifying malicious activities. Incorporating a broader perspective, Vijayanand and Arunlal discuss the societal impact of malware in modern society [5], while Abendan's exploration of fake apps affecting Android OS users underscores the real-world implications of cybersecurity threats [6]. The empirical assessments presented by Allix et al. [7] and the performance evaluations by Baldini and Geneiatakis [8] contribute valuable insights into the effectiveness of machine learning-based malware detectors, emphasizing the practical implications of these techniques. This compilation synthesizes knowledge from diverse sources, creating a foundation for understanding the multidimensional challenges posed by Android security threats. By drawing on both theoretical frameworks and empirical assessments, this literature review navigates the complexities of securing Android devices in an era dominated by machine learning and evolving cyber threats.

10.48047/jocaaa.2024.32.02.32

2. LITERATURE REVIEW

The literature survey encompasses a diverse range of sources, addressing topics such as Android security, machine learning, malware analysis, and the impact of malware on modern society. These sources contribute valuable insights to the field and provide a foundation for understanding the challenges and advancements in the areas of Android security and malware detection. Elenkov (2014) offers an in-depth exploration of Android security internals, shedding light on the architecture and mechanisms that safeguard the Android operating system. This foundational work sets the stage for understanding the vulnerabilities and potential exploits that malicious actors may target within the Android ecosystem [1]. Ng's "Machine Learning Yearning" (2018) introduces the reader to the intricacies of machine learning. While not directly related to Android security, the principles and methodologies presented by Ng could potentially be applied to enhance malware detection techniques. This source broadens the perspective by incorporating machine learning concepts into the discussion [2].

Dunham and Adams (2012) focus specifically on Android malware and its analysis. By delving into the characteristics and behavior of Android malware, this source provides insights into the evolving nature of threats targeting the Android platform. The authors offer methodologies for analyzing and understanding the tactics employed by malware to compromise Android devices [3]. Zhou, Jiang, and Lopez (2012) present a research paper on detecting Android malware by analyzing manifest files. This source introduces a technical approach to identifying malicious activity through the examination of Android application manifest files. Manifest file analysis is an essential aspect of understanding the intentions and capabilities of Android applications [4].

Vijayanand and Arunlal (2019) contribute to the literature by discussing the broader societal impact of malware. This source explores the consequences of malware attacks on individuals, organizations, and society at large. Understanding the real-world implications of malware is crucial for motivating research and development efforts in the field of cybersecurity [5]. Abendan (2011) discusses the threat landscape posed by fake apps affecting Android OS users. This source highlights a specific type of malware, emphasizing the need for robust security measures to combat deceptive applications that may compromise the integrity of Android devices [6].

Allix et al. (2016) empirically assess machine learning-based malware detectors for Android. By evaluating the effectiveness of various machine learning approaches, the authors contribute valuable insights into the practical applicability and performance of these detectors in real-world scenarios [7]. Baldini and Geneiatakis (2019) focus on the performance evaluation of distance measures in KNN for mobile malware detection. This source delves into the technical aspects of machine learning algorithms, specifically the K-nearest neighbors (KNN) algorithm, and its application to mobile malware detection [8].

Murphy's "Machine Learning: A Probabilistic Perspective" (2012) serves as a comprehensive resource for understanding the probabilistic

10.48047/jocaaa.2024.32.02.32

foundations of machine learning. This source provides theoretical insights that can enhance the reader's understanding of the underlying principles of machine learning algorithms [9]. Pressman's "Software Engineering: A Practitioner's Approach" (6th edition, 2010) and Booch, Rambaugh, and Jacobson's "The Unified Modeling Language User Guide" (2005) offer foundational knowledge in software engineering and UML, respectively. These texts provide essential background information for understanding software development methodologies and design principles [10] [11]. Lutz's "Programming Python: Powerful Object-Oriented Programming" (4th edition, 2010) contributes to the literature by offering insights into object-oriented programming, which is relevant for understanding the development of secure and well-structured Android applications [12]. In conclusion, the literature survey provides a comprehensive overview of key works in Android security, machine learning, malware analysis, and the societal impact of malware. The combination of technical and theoretical perspectives from these sources contributes to a holistic understanding of the challenges and advancements in the field.

3. METHODOLOGY

Existing system mainly focusses on the Android platform and aim to systematize or characterize existing Android malware. They have managed to collect more than 1,200 malware samples that cover the majority of existing Android malware families. In addition, they systematically characterize them from various aspects, including their installation methods, activation mechanisms as well as the nature of carried malicious app loads payloads. The characterization and a subsequent evolution-based study of very own representative families reveal that they are evolving rapidly to circumvent the detection from existing mobile anti-virus software. Based on the evaluation with four representative mobile security software, our existing system show that the best case detects 79.6% of them while the worst case detects only 20.2% in our dataset. These results clearly call for the need to better develop next-generation anti mobile-malware solutions. One of the researchers presented a system to dynamically identify whether an Android application is malicious or not, based on machine learning and features extracted from Android API calls and system call traces. We evaluated our system with 7,520 apps, 3,780 for training and 3,740 for testing, and obtained a detection rate of 96.66%.

Drawbacks:

1. It works well for known malware but it is easily evaded by obfuscation mechanisms or unknown malware.
2. The results showed that the existing anti-malware apps cannot detect obfuscated or repackaged malware apps.
3. The results showed that none of them can detect malicious applications after obfuscate
4. The developed system are related to shortcomings inherent to dynamic analysis approaches. The analysis system may fail to observe the malicious behaviors of samples in some situations, due to problems when gathering resources, to the lack of the necessary stimulation or to the detection of the analysis environment.

10.48047/jocaaa.2024.32.02.32

In this project we proposed a machine learning model based on the co-existence of static features for Android malware detection. The proposed model assumes that Android malware requests an abnormal set of co-existed permissions and APIs in comparing to those requested by benign applications. To prove this assumption, the paper created a new dataset of co-existed permissions and API calls at different levels of combinations, which are the second level, the third level, the fourth level and the fifth level. The extracted datasets of coexisted features at different levels were applied on permissions only, APIs only, permissions and APIs, and APIs and APIs frequencies. To extract the most relevant co- existed features, the frequent pattern growth (FPgrowth) algorithm, which is an association rule mining technique, was used. The new datasets were extracted using Android APK samples from the Drebin, Malgenome and MalDroid2020 datasets. To evaluate the proposed model, several conventional machine learning algorithms were used.

Benefits:

1. Moreover, the results show that the achieved classification accuracy depends on the classifier and the type of co-existed features.
2. The paper extracted new datasets of co-existed features using the APK samples gathered from three different datasets.
3. The paper applied the Association rule mining technique (FP-growth) to construct a novel coexistence based dataset that depends on the existing of different features together.
4. Different feature selection methods are applied to select the most relevant features that accurately classifies malware apps.

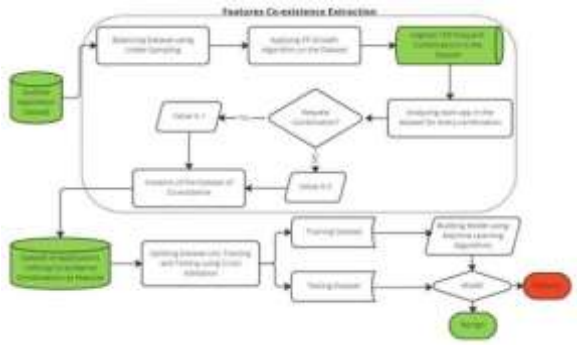

Fig 1 System Architecture

Modules:

The modules are:

- Data exploration: Using this module we will load data into system
- Processing: Using the module we will read data for processing
- Splitting data into train & test: Using this module data will be divided into train & test. Model generation: Building the model – Logistic Regression, Support Vector Machine, KNearest Neighbour, Random Forest, Decision Tree, Stacking Classifier. Algorithms accuracy calculated.
- User signup & login: Using this module will get registration and login
- User input: Using this module will give input for prediction

10.48047/jocaaa.2024.32.02.32

- Prediction: Final predicted displayed

4. IMPLEMENTATION

LR: This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables. Logistic regression is suitable for binary classification problems, making it apt for distinguishing between benign and malicious android apps. In this project, you could use logistic regression to model the probability of an app being malicious based on the coexistence features. This can aid in creating a decision boundary to classify apps effectively. It's simplicity and interpretability make it useful for understanding the significance of coexistence features in the context of malware detection.

KNN: The k-nearest neighbours algorithm, also known as KNN or k-NN, is a a nonparametric, supervised learning classifier, which uses proximity to make classifications or predictions about the grouping of an individual data point. KNN is an instance –based learning algorithm, which means it makes predictions based on the similarity of instances in the feature space. KNN works well when dealing with complex feature spaces. KNN doesn't make assumptions about the underlying data distribution. KNN's ability to adapt to different data patterns can be advantageous in capturing diverse malware behaviours represented by coexistence features. SVM: In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyse data for classification and regression analysis. SVM is designed for binary classification problems, which aligns with the goal of distinguishing between benign and malicious android apps in malware detection. SVM can handle imbalanced datasets, which is common in malware detection where the number of benign apps might outnumber malicious ones. In this project, leveraging SVM involves configuring it with appropriate kernel functions, handling high dimensional coexistence features, and optimizing parameters to achieve a well performing model for android detection.

DT: A decision tree is a decision support hierarchical model that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements. In unique approach for android malware detection using coexistence features, a decision tree is likely employed as a classification algorithm. Decision trees use a tree –like model of decisions to classify instances. The decision tree would be trained on a dataset with labelled examples of both benign and malicious Android apps, learning to make decisions based on these coexistence features. During classification , the decision tree analyses the coexistence features of a given app and assigns it a label, indicating whether it's likely to be malware or not.

RF: Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems. Random

10.48047/jocaaa.2024.32.02.32

Forest algorithm is an ensemble learning method commonly used for both classification and regression tasks. Random Forest can measure the importance of each feature in the classification process. This helps identify which features, including those related to feature co-existence ,contribute significantly to distinguishing between malicious and benign apps. It benefits from its strengths in handling complex ,relationships, providing feature importance insights, and creating a robust and accurate model for Android malware detection based on feature co-existence.

Stacking Classifier: A stacking classifier is an ensemble learning method that combines multiple classification models to create one “super” model. This can often lead to improved performance, since the combined model can learn from the strengths of each individual model. Stacking is an ensemble learning technique where multiple models are trained to predict the same target variable, and a meta-model is the trained to combine their predictions. Extract coexistence features from android applications, capturing interactions and behaviours that may indicate malicious activity. By combining the insights from different base classifiers and allowing the stacking classifier to learn how to best weigh their predictions, the stacking ensemble can potentially provide a more robust and accurate detection system for Android malware using coexistence features.

5. EXPERIMENTAL RESULTS

Dataset Description:

This research introduces a novel dataset designed to enhance the detection of Android malware through the comprehensive analysis of coexisting static features, with a specific emphasis on permissions and API calls. The dataset construction involved a meticulous process, utilizing samples from prominent Android malware datasets such as Drebin, Malgenome, and MalDroid2020. The resulting dataset provides a valuable resource for researchers and practitioners working on Android security, offering insights into the coexistence patterns of static features in malicious Android applications.

Dataset Construction:

Source Datasets:

- *Drebin*
- *Malgenome*
- *MalDroid2020*

Sample Selection: Android APK samples were collected from the aforementioned datasets to ensure diversity and coverage of various malware types.

Static Feature Extraction: Static features were extracted from each APK sample, with a primary focus on permissions and API calls.

Permission information was extracted to understand the access capabilities of the applications.

API call sequences were recorded to capture the behavioral patterns of the applications during execution.

Coexistent Feature Analysis: Different levels of combinations for coexisting features were considered during dataset creation.

The dataset includes various permutations of coexistent permissions and API calls, reflecting

10.48047/jocaaa.2024.32.02.32

realistic scenarios of feature coexistence in Android malware.

Dataset Description: Size: The dataset comprises a substantial number of Android APK samples, providing a diverse set of malicious applications for analysis.

Features:

Permissions: Enumerates the permissions requested by each application.

API Calls: Describes the sequences of API calls made by the applications.

Labeling: Each sample is labeled based on its classification in the source datasets (Drebin, Malgenome, MalDroid2020).

Metadata: Supplementary information about the source, malware family, and characteristics of each sample is included for comprehensive analysis.

Fig 2 Path for Link Generation

Fig 3 Anaconda prompt

Fig 4 URL

Fig 5 Dashboard

Fig 6 Signup Page

10.48047/jocaaa.2024.32.02.32

Fig 7 Validation Form page

Fig 8 Sign In Page

Fig 9 Input Parameters

Fig 10 Result detection

Fig 11 Entering Another parameters

Fig 12 Result detection

6. CONCLUSION

The project proposed a novel approach for detecting Android malware using the FP growth algorithm, which is an association rule mining technique that is used to extract the frequent patterns of features at different feature co-existence levels. This project has created three datasets of co-existed features, which are co-existed permissions features only, coexisted API features only, and co-existed permissions and API features. Furthermore, the project has created the features co-existence at four levels, which are level two, level three, level four, and level five. To test the proposed approach, several machine learning algorithms were used, which are Random Forest, Decision Trees, Logistic Regression, SVM, and KNN. The experiments have shown that Random Forest achieved the best accuracy of about 98% at

10.48047/jocaaa.2024.32.02.32

level 2 of co-existence using permissionAPI co-existence in the CIC_MALDROID2020 dataset. The proposed approach outperformed the state-of-the-art models, achieving an high accuracy using the Drebin and Malgenome datasets. The results show that using the features coexistence is an effective method to detect Android malware.

FUTURE ENHANCEMENTS

In the future development of an Android malware detection system using feature coexistence, a holistic approach to enhancement is essential. Consider delving into advanced deep learning architectures, such as recurrent neural networks (RNNs) or long short-term memory networks (LSTMs), to capture intricate temporal dependencies in app behavior. The integration of pre-trained models, leveraging transfer learning, can potentially elevate the system's performance. Behavioral analysis, encompassing the app's actions throughout its lifecycle, adds a layer of sophistication to the detection system. Collaborative and federated learning approaches can facilitate information sharing across devices while preserving user privacy. Implementing a continuous monitoring system with a feedback loop allows for realworld performance evaluation and adaptation to evolving malware patterns. Integration with external threat intelligence feeds enriches the feature set, enhancing the model's capability to recognize known malicious patterns.

REFERENCES

[1] Elenkov, N. (2014). Android Security Internals. No Starch Press.

[2] Ng, A. (2018). Machine Learning Yearning. deeplearning.ai.

[3] Dunham, K., & Adams, K. (2012). Android Malware and Analysis. Elsevier.

[4] Zhou, Y., Jiang, X., & Lopez, J. (2012). Detecting Android Malware by Analyzing Manifest Files.

[5] C. D. Vijayanand and K. S. Arunlal, "Impact of malware in modern society", J. Sci. Res. Develop., vol. 2, pp. 593-600, Jun. 2019

[6] O. Abendan, Fake Apps Affect Android Os Users, Oct. 2011, [online] Available: https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/72/fake-apps- affect-android-os-users.

[7] K. Allix, T. Bissyand, Q. Jarome, J. Klein, R. State and Y. L. Traon, "Empirical assessment of machine learning-based malware detectors for android", Empirical Softw. Eng., vol. 21, pp. 183-211, Jun. 2016.

[8] G. Baldini and D. Geneiatakis, "A performance evaluation on distance measures in KNN for mobile malware detection", Proc. 6th Int. Conf. Control Decis. Inf. Technol. (CoDIT), pp. 193-198, Apr. 2019.

[9] Kevin P.murphy .(2012). Machine Learning: A Probabilistic Perspective [online].

[10] Software engineering , A practitioner’s Approach-Roger S.Pressman, 6th edition, McGraw Hill International Edition

10.48047/jocaaa.2024.32.02.32

[11] The unified modelling language user guide Grady Booch James Rambaugh Ivar Jacobson, Pearson Education

[12] Programming Python: Powerful object-oriented Programming, 4th Edition Mark Lutz.