Grenze International Journal of Engineering and Technology, January Issue

Malware Detection for E-Commerce Applications using Machine Learning

Boggula Rajeswari[1], Sodagudi Suhasini[2] and Marri Tanuja[3]
[1-3]Velagapudi Ramakrishna SiddharthaEngineering College / Information Technology, Vijayawada, India
Email: rajeswari9864@gmail.com, s.suhasini@vrsiddhartha.ac.in, marritanuja@gmail.com

***Abstract*—In order to detect malware in e-commerce apps, this research suggests an automated testing approach that makes use of machine learning techniques. The proliferation of Android-based e-commerce platforms has led to a rise in malware threats targeting these platforms. We provide a solution to this issue by applying Artificial Neural Network (ANN) and Support Vector Machine (SVM) techniques to accurately categories APK files as malicious or benign. The framework is based on a vast array of Android applications that were carefully chosen to include samples from various e-commerce platforms. Metrics are taken into consideration when training and evaluating the SVM and ANN models using this dataset. F1-score, recall, accuracy, and precision, for example. Users can engage with the detection findings and sign up or sign in using an intuitive web interface built into the system. Our results confirm the efficacy of the proposed architecture, as both the SVM and ANN models classify APK files with high levels of accuracy. Furthermore, the system provides thorough insights into the classification performance through confusion matrices and classification reports. Overall, this work improves the area of malware identification in e-commerce apps and provides a helpful tool for enhancing cybersecurity in the mobile commerce arena.**

***Index Terms*— APK, Accuracy, Benign, E-Commerce, Machine Learning, Malware.**

I. INTRODUCTION

The necessity of internet security has increased in light of the ever-expanding domain of online applications. Financial and The smartphone industry is growing quickly. According to ICD study [1], it is anticipated that by 2027, annual global sales of mobile phones would exceed 351 million units. Android is the most popular mobile operating system, with over 2.5 billion active users in over 190 countries [2]. There are now a lot more individuals utilizing smartphones for purposes other than social media, internet banking, and gaming, which presents serious concerns regarding device security and personal privacy. Numerous functionalities are available on smartphones. Since Android is an open-source platform, it is simple for malware developers to initiate attacks and produce harmful Android malware applications. It appears that Android virus is having an increasing impact on current culture [3], [4]. The Google Play marketplace, an app marketplace for Android users, had over 3.43 million apps accessible as of January 2021 [5]. Customers may now download programs from a growing variety of non-Google and third-party app stores, such AppBrain and AppChina. The likelihood that the programs provided by third-party markets are malicious apps that evade detection is rather high. Hazardous apps were found in 22% of Google Play applications and 50% of AppChina. applications, according a study by Allix et al. [6]. Numerous security tactics have been used to detect and stop malicious programs. Signature-based malware

Grenze ID: 01.GIJET.11.1.814_1

detection is an early method that matches newly installed programs to those that have already been subjected to patterns. It works well against malware that is already known to be there, but new malware or obfuscation techniques easily get around it. Zhou and Jiang [7] evaluated the effectiveness of existing signature-based anti-malware scanners by comparing four different mobile security applications against over 1200 Android apps. The results showed that the anti-malware software now in use are unable to detect malicious applications that have been repackaged or obfuscated. Similarly, Scott [8] employed an obfuscation technique on ten different malware software from different families. The results showed that none of them can still recognize dangerous applications after they have been obfuscated. Machine learning techniques are utilized to identify and distinguish Android malware from benign ones. This process does not need matching patterns of known Android malware. Through the intelligent use of sample data, or "training data," machine learning techniques create a model that can make predictions or judgements without the need for explicit programming. There are two types of machine learning algorithms-based malware detection techniques: static analysis and dynamic analysis [9]. Analyzing Android apps without actually running them is possible using static analysis. On the other hand, a software is executed within a regulated setting in order to analyze its behavior dynamically. Machine learning is used to identify Android malware and distinguish it from safe versions. It lacks the ability to fit the known Android malware's patterns. Through the intelligent use of sample data, or "training data," machine learning techniques create a model that can make predictions or judgements without the need for explicit programming. There are two types of machine learning algorithms-based malware detection techniques: static analysis and dynamic analysis. Analysing Android apps without actually running them is possible using static analysis. On the other hand, a software is executed within a regulated setting in order to analyse its behaviour dynamically.

II. LITERATURE STUDY

Most people carry about one or more personal computers with internet access these days [4]. The internet is starting to affect our day-to-day lives. Along with computers and mobile devices, traditional independent items like gadgets are now connected to the internet to become intelligent. The vital infrastructure of cities, hospitals, and other enterprises has been linked to the internet for enhanced intelligence (SCADA). The expansion of the internet enhances human welfare.

[1]We use an extensive dataset of over 50,000 Android applications that we collected from various sources utilizing state-of-the-art techniques for data curation. We show that our approach outperforms existing machine learning-based approaches in the laboratory. However, this outstanding performance is not the same as great performance in the wild. The increased usage of smartphones has greatly expedited the spread of mobile malware, especially on well-known platforms like Android [10] [11]. Given their increasing proliferation, it is vital to provide workable solutions.

But our ability to combat these newly emerging mobile viruses is severely limited by our lack of timely access to pertinent samples and our incomplete knowledge of them. [7] We will be focusing on the Android platform in this study, which includes the majority of the known Android malware families, in an effort to systematize or describe the present Android malware [7] [9]. We also systematically explain their installation methods, modes of operation, and the kinds of malicious payloads they contain. The characterization and the ensuing evolution-based analysis demonstrate how rapidly these representative families are evolving to avoid being picked up by the mobile anti-virus software that is available today.

Based on the evaluation utilizing four typical mobile security software, our investigations show that the best scenario finds 79.6% of them, while the worst case only detects 20.2% of them in our dataset. These results clearly show that next-generation anti-mobile malware development has to be enhanced. The number of people using smartphones has increased by billions each year due to consumers' increasing storage of sensitive and private data on these devices. This gives hackers a great chance to get private user information [10]. Our method can detect malware on Android devices by using permissions and APIs. We have generated two different types of feature vectors: combined and common. Making use of logistics regression, For common characteristics, we were able to attain 97.25% accuracy, and for composite features, 96.56% accuracy. In order to achieve 95.87% accuracy, we further refined the feature to 131 by removing low variance features. This made the training and testing periods for categorization shorter. Malicious apps put the security of the Android platform at risk. Traditional defenses are practically ineffective due to the growing amount and variety of these applications, making Android phones susceptible to fresh infections. [9] In this study, we present DREBIN, a low-effort method for detecting Android malware that lets users identify potentially dangerous apps directly from their smartphone. Apps cannot be monitored while they are operating due to resource limitations, thus DREBIN does a comprehensive static analysis to gather as much information as possible.

These features are combined into a single vector space, which makes it possible to automatically recognize patterns that are indicative of malware and, thus, explains the decisions our system made. DREBIN outperforms several similar methods, detecting 94% of malware with few false positives in an experiment with 5,560 malware samples and 123,453 programmes. Relevant features of the malware that was found are highlighted in the explanations provided for each detection [7][12]. The method may be used to assess downloaded applications directly on the smartphone because it analyses data on five popular smartphones in an average of ten seconds.

III. METHODOLOGY

A. Proposed Work

Co-existing permissions and APIs are essential for distinguishing between malware and benign apps, as the proposed system—a specialized machine learning model for Android malware detection—demonstrates. On datasets like TUANDROMD, it outperforms existing models in terms of accuracy [9]. Its primary objectives are to increase Android security and protect user privacy. This study introduced a Stacking Classifier, which leverages the best features of Artificial Neural Network (ANN) and Support Vector Machine (SVM) models for enhanced feature extraction. Using this ensemble learning approach results in much higher prediction accuracy. Moreover, we developed a user-friendly Flask framework that is integrated with SQLite for safe signup, signing, and user testing. Because of its streamlined methodology, which allows for input provision and prediction retrieval.

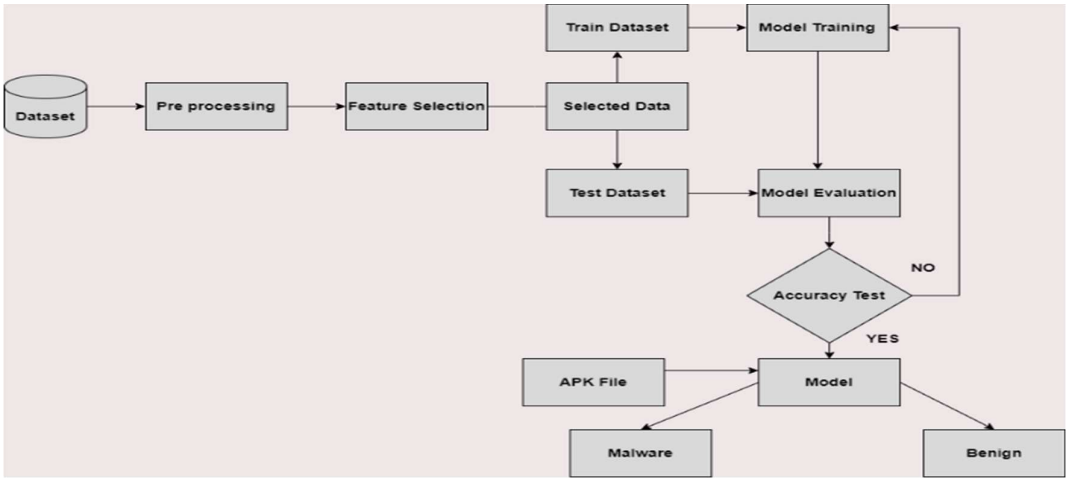

Fig 1: Architecture Diagram

B. Algorithms

Support Vector Machine (SVM)

A supervised machine learning approach called Support Vector Machine (SVM) is utilized for regression as well as classification. Even yet, classification issues are the most appropriate use for regression problems. The SVM algorithm's primary goal is to locate the best hyperplane in an N-dimensional space that may be used to divide data points into various feature space classes. The hyperplane attempts to maintain the largest feasible buffer between the nearest points of various classes. The number of features determines the hyperplane's dimension. The hyperplane is essentially a line if there are just two input characteristics. The hyperplane transforms into a 2-D plane if there are three input characteristics. If there are more than three characteristics, it gets hard to envision.

Think of a binary classification task where there are two classes, denoted by the labels +1 and -1. The input feature vectors (X) and the matching class labels (Y) comprise our training dataset. The linear hyperplane equation is expressed as follows:

$w^Tx + b = 0$

The direction perpendicular to the hyperplane, or the plane of the normal vector, is represented by the vector W. The offset, or distance, of the hyperplane from the origin along the normal vector w is represented by the

parameter b in the equation. One can compute the following to get the distance between a data point x_i and the decision boundary:

$$d_i = \frac{w^T x_i + b}{||w||}$$

Where the weight vector w's Euclidean norm is denoted by ||w||. Euclidean norm of vector W's normal.

Genetic Algorithm

It is mainly used for feature selection problems to find a subset of features that optimize a given objective function.

a. Create initial random population of organisms.
b. Evaluate fitness for each organism.
c. Select individuals from the current population to form the next generation.
i. Pi = Fi/∑ Fj
ii. Fi = Fitness for string
iii. Pi = Probability of string being selected
iv. N = no of individuals in population
d. Perform crossover on selected pair of Individuals. Use after crossing over on an individual basis for every child. In randomly picked text, bits are altered at random positions from zero to one or from one to zero.
e. Replace the old population with the newly created ones that is potentially better than previous generation.
f. Repeat the evaluation, selection, cross over, mutation and replacement steps for a specified number of generations.
g. Extract the best individual (subset of features) from the final population.

C. Artificial Neural Network

They handle each record separately and learn by comparing their categorization which is mostly subjective with the actual classification of the record as it is understood. The neural network equation is formed by linearly combining the independent variables with the weights and bias factors (or intercept) corresponding to each neuron.

Step1: Initialize the Weights.
Step 2: Propagating the inputs forward.
Step 3: Back Propagating the Error.
Step 4: Terminating Condition.

IV. Results And Observations

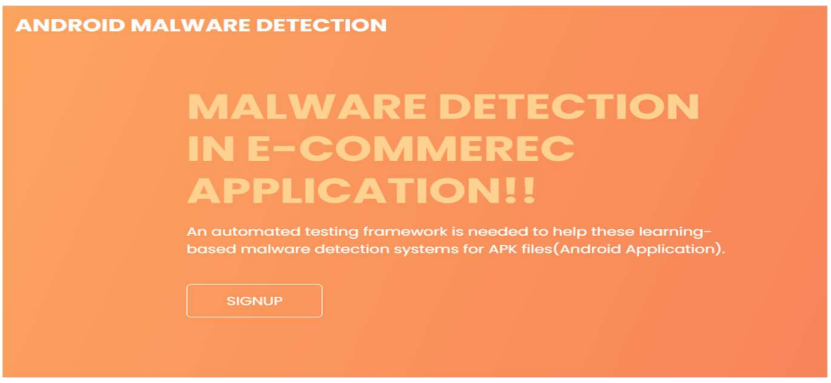

Fig 2: Home Page

Fig 4: It describes that the ANN Algorithm detect the apk file named Rareism as malware with 90% accuracy. Fig 5: It describes that the ANN Algorithm detect the apk file named FirstCry as Benign with 90% accuracy. Fig 6: It describes that the SVN Algorithm detect the apk file named Dmart Ready as malware with 88% accuracy. Fig 7: It describes that the SVN Algorithm detect the apk file named AJIO as Benign with 88% accuracy.

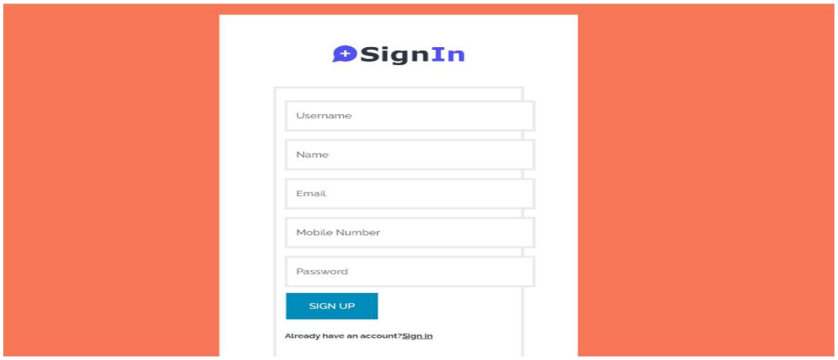

Fig 3: Signup Page

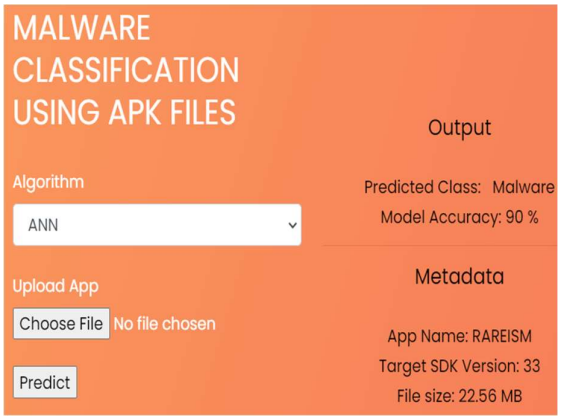

Fig 4: Classify as Malware using ANN Algorithm

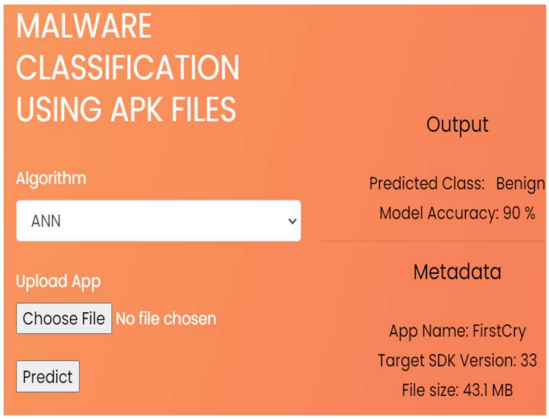

Fig 5: Classify as Benign using ANN Algorithm

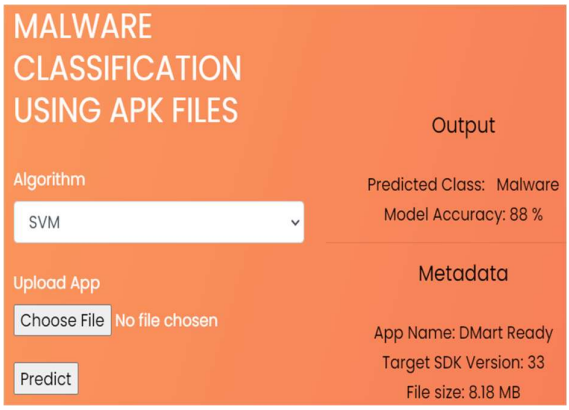

Fig 6: Classify as Malware using SVM Algorithm

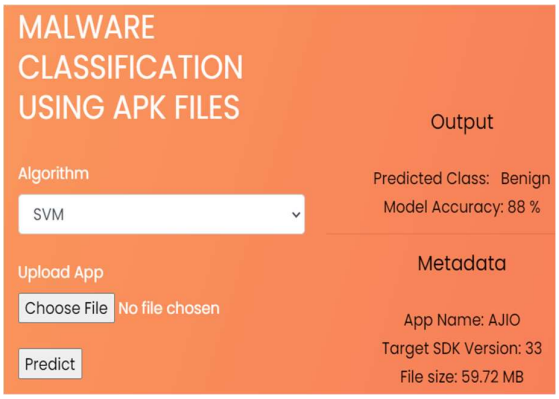

Fig 7: Classify as Benign using SVM Algorithm

Fig 8: The classification report shows the metrics of ANN Algorithm. The precision, recall, f1-score of class label 0 are 91, 94 and 93 respectively and precision, recall, f1-score of class label 1 are 84, 78 and 81 respectively and the accuracy of the model is 90%.

Fig 9: The classification report shows the metrics of SVM Algorithm. The precision, recall, f1-score of class label 0 are 88, 95 and 92 respectively and precision, recall, f1-score of class label 1 are 88, 73 and 80 respectively and the accuracy of the model is 88%.

From the above we can notice that the performance of ANN is better than the SVM i.e ANN has the more accuracy when compared to SVM.

```
[ ]  # Print classification report
     print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.91	0.94	0.93	240
1	0.84	0.78	0.81	96
accuracy			0.90	336
macro avg	0.88	0.86	0.87	336
weighted avg	0.89	0.90	0.89	336

Fig 8: Classification report of ANN

```
[ ]  # Print classification report
     print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.95	0.92	228
1	0.88	0.73	0.80	108
accuracy			0.88	336
macro avg	0.88	0.84	0.86	336
weighted avg	0.88	0.88	0.88	336

Fig 9: Classification report of SVM

	Accuracy	Precision	recall	F1-score
ANN	90	84	78	81
SVM	88	88	73	80

Fig 10: Metrics Comparison of ANN and SVM

V. Conclusion

Our project's objective was to develop a novel method for identifying Android malware by combining sophisticated algorithms with state-of-the-art web technologies. We successfully achieved our goals and developed a comprehensive solution that addresses the dynamic cybersecurity risks during the process. Our work began with a comprehensive analysis of the state-of-the-art research and methods for detecting Android malware. We have also found the shortcomings of traditional signature-based techniques, even though machine learning techniques, particularly Support Vector Machines (SVM), Artificial Neural Networks (ANN), and Genetic algorithms (GA), have the potential to improve detection accuracy and adaptability. One of the project's most significant achievements is the development of the TUANDROMD dataset, an extensive collection of Android applications that includes instances of both malicious and benign software. TUANDROMD, which contains 4464 instances and 241 characteristics collected from various sources inside the applications, provides a strong foundation for training and testing our detection algorithms. We have been able to further advance Android malware detection and contribute significantly to the cybersecurity research community by curating this dataset. Throughout the project, we encountered a number of challenges and obstacles, including complex data preparation, frontend design issues, and algorithm optimization. But with perseverance, cooperation, and innovative problem-solving, we were able to get over these challenges and emerge with a fully functional system that surpasses our initial expectations.

Future Scope

Future research may focus on improving the suggested system's real-time detection capabilities by continuously monitoring and analyzing dynamic data. By doing this, the system would be able to respond to more sophisticated attacks from Android infections. By looking at ways to choose the most relevant dynamic features for malware detection, models may be made more accurate and effective. Techniques for feature selection such as mutual information and recursive feature removal may be studied. To enhance security, it might be possible to increase the system's capacity to identify anomalous activity in Android applications. Part of this involves identifying distinctions between normal conduct and potential malware. It’s possible that the system will be designed to upgrade and modify its models and detection methods as new varieties of Android malware surface. The system has to be updated often with fresh threat intelligence in order to be effective over the long run. Expanding the system's ability to enable cross-platform malware detection—which covers iOS and other mobile operating systems—is one method to handle mobile security more thoroughly.

REFERENCES

[1] H. Menear. (2021). IDC Predicts Used Smartphone Market Will Grow 11.2% by 2024. Accessed: Oct. 30, 2022. [Online]. Available: https://mobile-magazine.com/mobile-operators/idc-predicts-usedsmartphone-market-will-grow-112-2024?page=1

[2] D.Curry.(2022).Android Statistics .Accessed: Oct.30,2022. [Online]. Available: https://www.businessofapps.com/data/android-statistics/

[3] O. Abendan. (2011). Fake Apps Affect Android Os Users. Accessed: Oct. 30, 2022. [Online]. Available: https://www.trendmicro.com/ vinfo/us/threat-encyclopedia/web-attack/72/fake-apps-affect-android-osusers

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

[5] M. Iqbal. (2022). App Download Data. Accessed: Oct. 30, 2022. [Online]. Available: https://www.businessofapps.com/data/app-statistics/

[6] 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.

[7] Y. Zhou and X. Jiang, ''Dissecting Android malware: Characterization and evolution,'' in Proc. IEEE Symp. Secur. Privacy, May 2012, pp. 95–109.

[8] J. Scott. (2017). Signature Based Malware Detection is Dead. Accessed: Oct. 30, 2022. [Online]. Available: https://icitech.org/wpcontent/uploads/2017/02/ICIT-Analysis-Signature-Based-MalwareDetection-is-Dead.pdf

[9] Q. M. Y. E. Odat. Accessed: Dec. 27, 2022. [Online]. Available: https://github.com/esraa-cell28/a-novel-machine-learning-approach-forandroid-malware-detection-based-on-the-co-existence

[10] S. R. Tiwari and R. U. Shukla, ''An Android malware detection technique based on optimized permissions and API,'' in Proc. Int. Conf. Inventive Res. Comput. Appl. (ICIRCA), Jul. 2018, pp. 258–263. [15] H. Cheng, X. Yan, J. Han, and C.-W. Hsu, ''Discriminative frequent pattern analysis for effective classification,'' in Proc. IEEE 23rd Int. Conf. Data Eng., Apr. 2007, pp. 716–725.

[11] C.-F. Tsai, Y.-C. Lin, and C.-P. Chen, ''A new fast algorithms for mining association rules in large databases,'' in Proc. IEEE Int. Conf. Syst., Man Cybern. San Francisco, CA, USA: Morgan Kaufmann, Oct. 1994, pp. 487–499.