JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

HYBRID ANALYSIS OF MALWARE DETECTION IN ANDROID

Mohamed Zehruddin Badusha.M.K[1], Saravanan.M[2], Vignesh.S[3], Rajesh.R[4],Mrs.S.Saraswathi[5].

UG Scholars[1,2,3,4],Assistant Professor[5].

Department of Computer Science and Engineering

Vel Tech High Tech Engineering College, Chennai, Tamil Nadu, India.

ABSTRACT

Android has been the most popular operating system over the past ten years. Despite this quickly rising. Android is a target for malware distribution due to its prevalence. Applications from unofficial marketplaces can be installed on Android devices. The ability to install malicious apps and interact with Android devices is made possible by this reality. Both static analysis and dynamic analysishave so far been used in the development of malware analysis and detection tools. However, the performance of current research is still lacking in terms of correctly and efficiently detecting malware. It frequently makes use of numerous resources from mobile devices with limited resources to identify malware accurately. Therefore, by creating and testing an accurate and efficient machine learning and deep learning model for this issue, this study suggests a solution. The dataset of malware genomes was utilized**.**Test results show that hybrid analysis increases by 5% of malicious software detections.

Keywords— prevalence, malicious, malware gnomes, datasets, hybrid analysis.

Introduction

The android operating system is acknowledged as being the most well-known and widely utilised. By the end of 2023, Android is anticipated to dominate the mobile operating system market, claiming 85% of the total market share. With the growing popularity of the Android OS comes a rise in malware attacks every year. According to TrendMicro, 10.6 million Android adware infections are anticipated by the middle of 2020. This enormous amount of mobile software will keep growing and spreading as more cybercrimes are committed on mobile device

We have received several complaints concerning the existence of malware on the Play Store, despite deploying a variety of security measures including Play Protect. This study demonstrates how difficult it is for even well-known corporations like Google to safeguard their Android environments from infection. The Android operating system has been a top target for mobile adware since it enables users to install programmes bought from third-party marketplaces, in addition to the fact that Android devices are widely used. This makes it possible for an attacker to dupe an Android user into downloading malicious software from her PC.

It is essential to perform malware detection analysis because malware is becoming more sophisticated and prevalent. Numerous researchers work to counteract Android malware cyberattacks in various ways. A malware sample is not executed during the research process in a static-based strategy, and the opposite is true for a dynamic approach.

Anti-malware techniques like static-based detection locate malware by comparing the software's pattern to a database of signatures from well-known threats. Because the byte signature pattern is derived from well-known malware, it is disadvantage of the static analysis approach. Therefore, an attacker utilising code obfuscation approach may easily evade this.

If a piece of spyware has the capacity to change its code as it propagates, it is referred to as metamorphic or adaptable. Additionally, static-based research is unable to recognise this specific type of virus. Signature-based anti-malware systems are unable to identify zero-day malware attacks since they are created using a library of known malware. This collection is getting bigger over time, which will make it harder to find viruses.

On the other hand, a method known as dynamic-based analysis uses a controlled environment to operate a copy of the virus in order to detect it. The characteristics that are extracted using this technique frequently take the form of system operations, memory writes, API requests, or registry adjustments.

Dynamics-based analysis has the disadvantage that it takes time because the environment needs to be built up for evaluating malware samples.

These drawbacks motivate academics to create hybrid-based analytic techniques for better outcomes.Although hybrid-based analyses combine the advantages of static and dynamic analysis to execute with high accuracy, they fall short in ensuring resource economy. The restricted memory, processor, and battery resources on cell phones make this degree of efficiency crucial.desktops. Due to these factors, we suggest creating a hybrid-based malware research approach that not only focuses onboth accurate and resource-friendly in terms of virus identification.

In this study, we merge the data that was extracted from the dynamics-based and signature-based analyses, and we create an analysis model that is based on deep learning and learning machines. Although deep learning and machine learning models are widely employed for malware detection, a reliable deep learning model requires a large amount of training data. The oversampling technique can be used to generate data artificially to finish this step. We balance the quantity and create synthetic data for the malware class in this way. Through the use of dimension reduction, we were employing this model to significantly increase the efficacy of malware identification. The static strategy will address the weaknesses of the dynamic strategy while the dynamic strategy will fill in the gaps left by the static strategy as we develop a hybrid-based approach. Through real-time analysis, our proposed malware detection approach is projected to effectively identify diverse malware types while maintaining effective performance.

Literature survey

Numerous studies have been done on malware analysis for Android devices. Malware identification has also made extensive use of machine learning.

A. Analysis of behaviour and permission allowance

The proliferation of harmful programs on smartphones, particularly those using the Android operating system, is accelerated by the growth of mobile Internet and application stores. In this essay, we provide a system for detecting Android malware that combines permission and behaviour analysis. The MD5 values of APK files that have been identified serve as signatures for detection. The identification of APK files that have not been identified is based on permission and behaviour analysis. Semantic analysis and taint propagation analysis were both a part of behaviour analysis. According to test findings, this system is capable of successfully detecting malware that steals personal information and does destructive deduction.

B. Dynamic analysis in android

Androidisthepreferredtargetformalware.Attacksduetogrowingpopularityamongotherestablishments' systems for smartphones. Its open architecture and large user base provide an open environment for developers and An ice interface to access and launch its code based on malicious activity. This article presents on approach Dynamically analyse and classify Android applications Malicious or non-malicious applications to this end and developed a system to capture system calls. Extract system call traces for all running applications. Runtime interactions with phone platforms after that, all collected system access data is aggregated and analysed Detect and classify Android application behaviour. We used our system analyse the actions of 50 of is malicious Applications derived from the Android malware genome 50 harmless applications from Project and Google Load the game to classify their behaviour Considered application, frequency of system calls Created by each applications its main set of functions. In addition, accurately achieved an acceptable level of accuracy Use J48 to classify applications as malignantorbenign Decision trees and random forest algorithms.

C. Detection by static features and machine learning

Smartphones are vulnerable to cyberattacks. Malware applications can compromise security. Telephone, thereby invading personal or financial privacy information. Machine learning is proven in various fields Areas involving security. This paper proposes a machine Learning Android Malware Detection Focused It’s the use of various static functions in Android apps Packages (APKs). permissions, API calls, Services, opcodes, and activities for training different machines. A learning model that classifies APK files as malware or being. Among machine learning experimental models, Gaussian processes are the most Promising results followed by random forests and decisions wood.

D. Detection on android applications.

Smartphones are now used to establish oneself Multiple tasks over the Internet due to high performance The calculations it has and the big screen it offers So users can take advantage of the easier-to-use side smartphone. Along with various other privacy concerns, Mobile device malware is one of the biggest problems with a smartphone device. Malware is defined as malicious; this pollutes the code and prevents it from working effectively. Malware, for example, can create thousands of threats. Breach of secrecy and privacy, loss of confidentiality, crash Unauthorized use of systems, theft of confidential information, etc. Android malware threats are increasing rapidly, especially this repackaged Android malware. of Understanding Android Malware Based on Dynamic Analysis and static analysis Overview view. these techniques are excellent advantages that allow you to detect accurate information about your data code, but in addition to these, it has certain drawbacks. So, in this work, we proposed a way to prove Smartphone malware, our solution Static and dynamic methods for clearing records with API calls and User permissions in download applications.

E. Deep android malware detection

We suggest a unique deep convolutional neural network-based approach for detecting malware on Android. Static analysis of the raw opcode sequence from a disassembled programme is used to categorise malware. Malware features don't need to be manually engineered because the network automatically learns them from the raw opcode sequence. Because the network is trained end-to-end to jointly learn appropriate features and to perform classification, our proposed system's training pipeline is much simpler than existing n-gram-based malware detection methods. This eliminates the need to explicitly enumerate millions of n-grams during training. Once trained, the network may be operated effectively on a GPU, enabling rapid scanning of a huge number of files.

Proposed Methodology

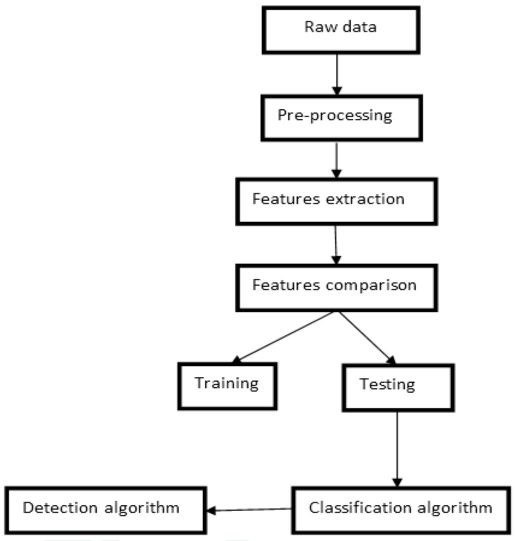
This study proceeded through several stages. The procedure of collecting data is the first step. We conducted experiments on two datasets derived from two publicly accessible collections of malware samples (Android Malgenome project and DREBIN). This collection of 18,835 Android applications includes 215 characteristics and the two classifications of dangerous and benign apps (6,820 malicious and 12,015 benign). We gather the most typical characteristics, such as manifest authorization, To build a powerful and accurate Android malware detection pattern, combine API call signatures, intent filter, command signature, and files from the application under investigation.

The next step is the material's pre-processing. The collection's elements without complete values are presently not included. The next step is the material's pre-processing. The collection's elements without complete values are presently not included.

Before separating the samples into training and test bins for machine learning algorithms, we balance the number of samples in each class to account for the malware and harmless sample counts being out of balance. To use every sample equally, we randomly mix the data in each class before balancing the samples. Then, by combining the oversampling strategy with the Synthetic Minority Over Sampling Technique (SMOTE) technique, the number of class samples is equalised.

The next step is the dataset training and confirmation process. The collection data is divided from the training and confirmation data. We employ cross-validation with a 10-fold to ensure that training and validation are effective. The entire data set has been sent. We employ a variety of machine learning and deep learning classification techniques for training, including the Support Vector Machine (SVM), decision trees, random forests, K-Nearest Neighbour (K-NN), multi-layer perceptrons (MLP), and gradient boost (GB). After doing a benchmark, we enhanced the results by changing the hyperparameter of the algorithm that was used to identify the most effective methods for classifying malware. The process makes use of data from static-based and dynamics-based analysis.

The CICMalDroid collection is used to provide the data for the behaviours study. There are 17,341 Android examples in the collection. Using the CopperDroid framework, the information was examined to produce three types of dynamic behaviours: system, binder, and composite behaviour calls. The remaining samples failed owing to problems including time-outs, incorrect APK files, and memory allocation issues, and only about 11,598 samples were successfully evaluated. The other 9803 examples are malware, leaving only 1,795 innocuous samples.



Implementation

After obtaining the optimum categorization value, we combine the characteristics in a static and dynamic collection. For the testing of this combined sample, 311 applications—166 malicious and 145 benign—were employed as test data. Finally, we apply the Principal Component Analysis (PCA) technique to reduce the number of features that best characterise each version. All of these operations are carried out in a Windows 10 development environment with an Intel Core i7 9750 CPU, 32 GB of DDR4 RAM, an Nvidia Quadro T1000 graphics card, and 512 SSD NVMe storage.

Results

We analyse the data using the Scikit-learn library and the Python 3 computer language after it has been gathered. With the SMOTE method, an equal are obtained during the oversampling step. 90% of the data are used for training in the filtered 10-fold cross-validation, and the remaining 10% are used for validation. The techniques utilised for training and assessment include SVM, K-NN, MLP, Random Forest, Decision Tree, Naive Bayes, and GB. Table 1 displays the results for typical training accuracy and delay.

Table 1.

Algorithm	Average score	Train & Val. Time (in secs)
SVM	0.943572	103.566
K-NN	0.808146	1.722
MLP	0.960026	205.45
Random Forest	0.959159	23.33
Decision type	0.929516	4.092
Naïve Bayes	0.808146	2.769
GB	0.963543	139.897

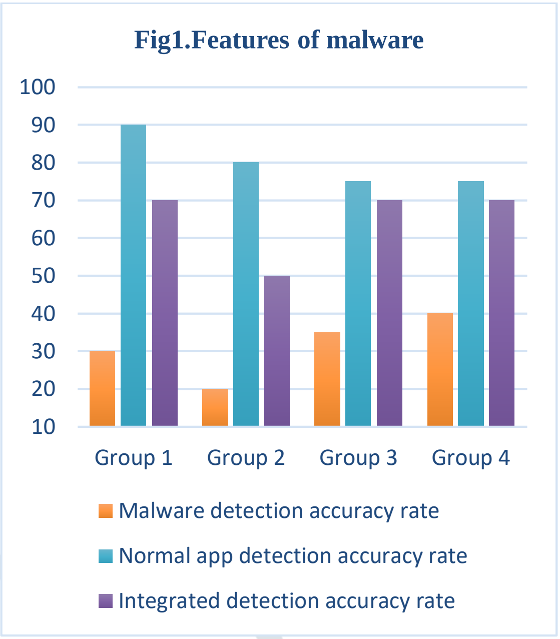
The Naive Bayes method produces the worst result. This outcome is probably a consequence of the algorithm's preference for classifications with three groups or more. The K-NN algorithm only saves the training dataset in memory and uses it later when forecasting, so it has the same subpar accuracy performance as Naive Bayes but the best training time. The forecast time produced by this technique will be much greater than the training time. The SVM method, however, holds a middling position. This is presumably caused by the dataset having a good number of entries but few features.

The MLP algorithm shows the second-best recognition results when compared to other algorithms. However, this approach requires a lot more time for training and validation compared to other algorithms.Deep learning builds levelled algorithms to create an artificial neural network that can learn and make decisions on its own. Deep learning enables us to analyse problems that would be far more challenging to directly programme because of its hidden layer design.The drawback of the neural network's extensive training period is its many nodes, though.

When compared to other methods, gradient boosting (GB), one form of machine learning boosting, achieves the highest accuracy levels. It is predicated on the understanding that using the best subsequent model in conjunction with earlier models will lower total prediction error. The most precise outcomeappears to be the result of setting the goal results for this next model to reduce the variance.Gradient boosting typically requires a lengthy training period because, unlike random forests, the decision trees used to build the final forecast model are linked in GB.

The findings of the steady and dynamic analyses are then combined. There are 261 characteristics total, which combine 215 static analysis features and 46 dynamic analysis features.To determine which of the top 10 characteristics best distinguishes malware groups, we performed a PCA analysis. Fig.1 represents the analysis's finding.

The most recent trial measured the model for mixed static and dynamic analysis. We reverse engineer sample feature extraction apps for this reason. While harmful apps are downloaded from Virus Share, good apps can be downloaded from the Play Store.We gathered 311 example apps, and using the vector features that were present in each dataset, we extracted both static and dynamic features. Additionally, the gathered datasets are used to evaluate the training and validation models.After the model's hyperparameter has been improved for the highest precision, this test is run. Table 2 displays the benchmark's best 3 outcomes.



.Table2

Algorithm	Average score	Prediction time (in secs)
GB	0.99	0.66
Random forest	0.97	1.22
MLP	0.95	2.45

The GB algorithm has the best accuracy judgements and the quickest latency prediction time, as shown in Table 2.These findings are consistent with trials using earlier static analysis data.

The results of the GB algorithm closely resemble those of Random Forest. Similar to GB, the boosting method builds a model using a random forest technique. GB constructs one tree at a time, whereas random forests create each tree separately. When there is a lot of disturbance in the data, random forest algorithms outperform GB, which overfits the data.

Further testing revealed that the extreme gradient boosting (XGB) modification of the assembly model, in particular, was successful in cutting the forecast time by 15% while maintaining the same precision.

Conclusion

We put a model to the test using static analysis data from the Drebin project dataset and the malware genome project. Based on these databases, we sought to develop the finest machine learning and deep learning malware detection algorithm. The findings indicate that the gradient boosting method has a detection rate of about 99%. The characteristics of the dynamic analysis findings from the CICMalDroid dataset of apps were then also examined. The two traits of the static and dynamic analysis results are then blended to build a hybrid-based malware analysis model. Through extensive testing, we also found that the gradient boosting approach surpassed all other deep learning and machine learning algorithms in terms of accuracy effectiveness and latency time.

Acknowledgements

We would like to express our gratitude towards department(CSE) for providing us a great opportunity to complete a project on “Hybrid analysis of malware detection in android” .

Our sincere thanks to Mrs. Saraswathi for coordinating with us.

References

[1]. Khan (2012, August). A machine learning approach to android malware detection. (pp. 141-147). IEEE.

[2]. Kaushal (2017, June). Malware detection in android based on dynamic analysis. (pp. 1-6). IEEE.

[3]. Zaman, & Hossain, M. S. (2015, January). Malware detection in Android by network traffic analysis. (pp. 1-5). IEEE.

[4]. Tong (2017). A hybrid approach of mobile malware detection in Android., *103*, 22-31.

[5]. Yang & Li (2021). Deep learning feature exploration for android malware detection. *Applied Soft Computing*, *102*, 107069.

[6]. Ali, Aung & Lukman (2017, December). Malware detection in android mobile platform using machine learning algorithms. (pp. 763-768). IEEE.

[7]. Kambourakis, G. A comprehensive survey on machine learning techniques for android malware detection. *Information*.

[8]. Baskaran (2016). A study of android malware detection techniques and machine learning.

[9]. Ye Wang (2007, August). IMDS: Intelligent malware detection system. (pp. 1043-1047).

[10]. McLaughlin, Martinez , Kang & Joon (2017, March). Deep android malware detection (pp. 301-308).