Archives available at journals.mriindia.com

International Journal of Recent Advances in Engineering and Technology

ISSN: 2347-2812
Volume 14 Issue 1s, 2025

Detection of Malware Using Machine Learning Techniques

Yogita Madhukarrao Bhagwat[1], Mr. Kapil D. Dere[2], Dr. Anand A. Khatri[3]
yogitabhagwat4@gmail.com[1], Kapilddere@gmail.com[2], khatrianand@gmail.com[3]
Department of Computer Engineering, Jai Hind College of Engineering, Kuran,India

Peer Review Information

Submission: 20 Jan 2025
Revision: 24 Feb 2025
Acceptance: 27 March 2025

Keywords

Malware
Ransomware
Behavior-Based Methodology
Heuristics
Vulnerabilities
Signature-Based Models

Abstract

The subject of computer security is called malware detection, and it deals with the research and prevention of dangerous software. It's not the exclusive method of defending a company from online threats. To succeed, businesses need to assess their risk and pinpoint their weaknesses. This essay will address the rise in computer malware and how cutting-edge approaches such as behavioral-based models and signature-based models are displacing more conventional detection strategies. Future research directions in this field as well as the many methods for identifying fraudulent websites and computer viruses will be covered. To combat cyber fraud, which has increased recently, particularly in the Asia Pacific area, future instructions call for creating stronger security solutions. Traditional methods of preventing cyber-frauds and other dangerous activities from accessing computers are inadequate because of their many shortcomings. In order to tackle these problems, academics have created new techniques such as heuristic analysis and static and dynamic analysis. With no false positives or negatives, these techniques are able to identify over 90% of malware samples.

INTRODUCTION

Using anti-malware software to safeguard our computers and devices is becoming more and more crucial as the number of malware threats rises. Machine learning is an efficient approach for detecting dangerous software. Millions of samples have been used to train it, enabling it to recognize their traits at scale even in the case of previously undiscovered malware varieties. By using pattern recognition to extract information from the file and compare them to known malware signatures, this artificial intelligence technique may be used to identify malware. It also entails scanning the system as a whole or certain areas, extracting harmful software traits, comparing these traits to known behaviors, and identifying the presence of malware. Malware examiners and designers compete fiercely. Because they don't understand what they imply, many users provide permissions mindlessly, allowing the program to access personal information. The inability to choose which permissions to grant and which to deny is another drawback. Many individuals will consent to install an app even if it requests dubious authorization from a huge number of people who seem trustworthy. Android is now the most widely used operating system for smartphones and tablets, surpassing iOS with an estimated 70–80% of the market. The method advises checking the Play Store Marketplace to see whether an unfamiliar app is dangerous

before introducing it. For each new application or updated version of an existing one, compute a risk score and categorize it according to a preset threshold during execution. A virus detection system was successfully implemented using a real- time Android application environment.

HISTORY & BACKGROUND

Malware detection may be addressed in several ways. However, malware writers have used a number of strategies to avoid detection, prompting the development of new and improved technologies capable of producing more exact findings. Detecting malware is a challenging process. Even after patches, new viruses will emerge, making it an ongoing struggle. Because fresh samples may readily avoid signature- based malware detection, it is not an effective solution for this problem. A recent approach, known as a vision-based approach, uses deep learning techniques in the AI process to detect the features of malicious software. However, this strategy fails if there are no examples or if they are not trainable. To address this, researchers presented a selection and fusion strategy that combines both methods to provide more accurate results.

The authors of these books restrict their research to the most frequently cited assents (or assurances of approvals) [1]. However, depending on the attack, consents such as READ LOGS may also be just as risky as others (such as Web). Each award ought to be carefully evaluated for its potential to be risky when coordinated with other awards.

According to [2], this decision-making procedure results in extremely skewed outcomes. Strategies based on simulated intelligence are thought to have two drawbacks: first, they are seen as having a high rate of fake problems, and second, choosing which characteristics should be mastered during the planning stage is a difficult task. As a result, selecting datasets for preparation is a crucial stage in these structures. The introduction of the classifier improves with time: for a particular month Mi, whose applications were used for the planning datasets, the resultant classifier ends up being less and less prepared for separating all malware in following months.

The majority of these endeavors separate a list of capabilities to address the applications. The information passed on by such traits varies depending upon the gig. Every investigation considers required consents, despite the lack of evidence to demonstrate which characteristics produce the best revelation results. Moonsamy et al. [3] are enthusiastic about including approvals as the sole component to depict programs and perceiving explicit agree guides to perceive immaculate and vindictive applications Using simulated intelligence systems and evaluating the application's isolated assents, Sanz et al. [4] propose a novel method for distinguishing malicious Android applications. The presence of marks uses approval and usages feature in the manifest, as well as how much agrees permitted to each application, are utilized to organize them.

As shown in [5], removing various properties from the Android manifest can be used to create computer-based intelligence classifiers and recognize malware. These components are the specific assents searched for and the reasons feature tag.

The system makes recommendations for malware detection techniques. We used the following techniques: (ii) similarity-based permission feature selection; (iii) association rule mining; (iv) permission ranking-based feature selection; (v) updated random forest classifier parameters; and (iii) similarity-based permission feature selection. A frequency rating is given to the characteristics by both the permission ranking-based feature selection approach and the permission feature selection technique based on similarity.

Architecture

According to [6], present a plan to help an artificial intelligence-based malware identification system for Android recognize malware applications and improve mobile phone customers' security and insurance. This structure assembles different assent based characteristics and events from Android applications and assessments them using artificial intelligence classifiers to conclude whether the application is innocuous or vindictive.

Approvals key and referenced are joined with an additional six characteristics from the manifest and the destroyed code in DREBIN [7]. Estimates based on artificial intelligence are used to distinguish between dangerous and harmless tasks.

Algorithms

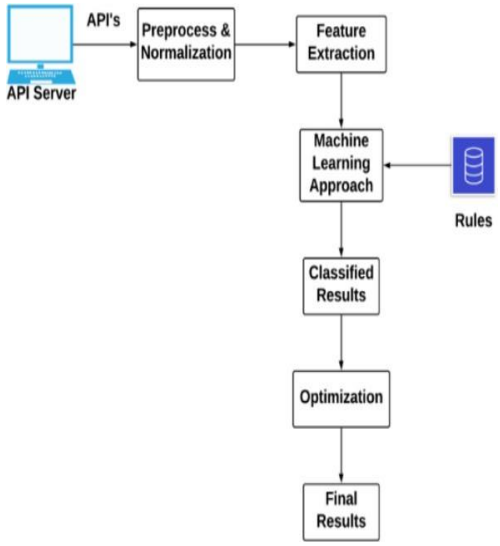

Figure1. System Architecture

The Assist Vector With Machining only sends the learned model to the phone for detecting potentially dangerous applications when disengaged on a dedicated machine.

Using assents and 20 components from application packs, Huang et al. [8] investigate the probability of identifying bogus Android applications. According to their disclosures, a singular classifier can perceive around 81% of phony undertakings. It very well may be a fast channel to distinguish more suspect applications, according to them, by integrating revelations from a couple of classifiers.

Liu and Liu [9] use referenced and indispensable assents by an application in like manner. Artificial intelligence estimations and assents are used in this system to arrange an application as innocuous or unsafe.

Shabtai et al. [10] divide Android applications into utilities and games categories. Successful parcel among games and gadgets, as they might want to think, should offer a good indication of such structures' capacity to learn and exhibit Android innocuous undertakings and possibly recognize malware reports using man-made intelligence (ML) methods on static credits accumulated from Android program records.

DESIGN ISSUES

Algorithms 1: RNN

Input: Test Dataset which contains various test instances TestDB-Lits [], Train dataset which is built by training phase TrainDB-Lits [] , Threshold Th.

Output: HashMap ¡ class label, Similarity Weight ¿ all instances which weight violates the threshold score.

Step 1: For each testing records as given below equation

$$testFeature(m) = \sum_{m=1}^{n} (.featureSet[A[i].....A[n] \leftarrow TestDBLits)$$

Step 2 : extract each feature as a hot vector or input neuron from testFeature(m) using below equation.
ExtractedFeatureSetx[t] contains the feature vector of respective domain

$$Extracted_FeatureSetx[t.....n] = \sum_{x=1}^{n} (t) \leftarrow testFeature(m)$$

Step 3: For each read each train instances using below equation

$$trainFeature(m) = \sum_{m=1}^{n} (.featureSet[A[i]........A[n] \leftarrow TrainDBList)$$

Step 4 : Extract each feature as a hot vector or input neuron from testFeature(m) using below equation.
Extracted Feature Setx[t] contains the feature vector of respective domain.

$$Extracted_FeatureSetx[t.....n] = \sum_{x=1}^{n} (t) \leftarrow testFeature(m)$$

Step 5 : Now map each test feature set to all respective training feature set

$$weight = calcSim(FeatureSetx \parallel \sum_{i=1}^{n} FeatureSety[y])$$

Step 6: Return Weight

Algorithms 2: Naive Bayes (NB)

Input: Traininput TrF[], Testfeatures TsF[], Threshold T, Feedback count_n

Output: Refine weight for each object.

Step 1: Read Train feature TrF Step 2: Read Train feature TsF Step 3: for each (tsf into TsF)

Step 4: for each (trf into TrF) If (feed-back count != n) Step 5: send feed layer to tsf again tsF ¡- feed-Layer [] execute for all neurons early stodolgy

Step 6: optimized feed-Layer weight

Step 7:weight = FeedLayer[0]

Step 8: return cweight

Algorithms 3: Support Vector Machine

Input: Train features TF [], Test features Ts[], threshold T

Output: Classification of weight

Step 1: vector is given as an input

Step 2: Each values in the given vector is extracted
Step 3: The extracted values is searched in the dataset Step 4: for each (x[] into TF[] when!=null)
Step 5: Get all features x1[] =Ts []
Step 6: w=Cal Distance(x[], x1[])
Step 7: evaluate w with T Step 8: Classify weight

RESULT AND ANALYSIS

The Java open-source environment was used to finish the implementation process. The gadget, which has a distributed INTEL 3.0 GHz i5 CPU and 4 GB RAM, runs on the Java 3- tier analytics platform. The APK dataset has been used to detect if an email is spam or not. To confirm the results, we conducted experiment analysis on the ensemble machine implementation.

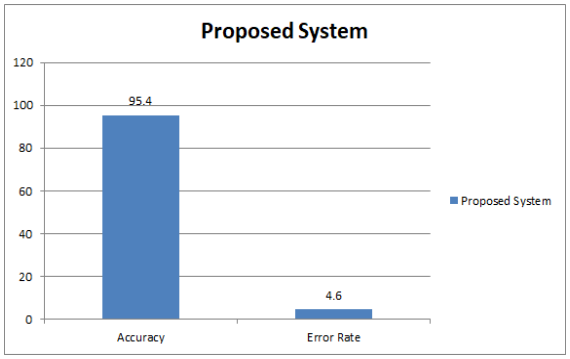

Figure 2: Accuracy of system analysis

The classification accuracy of the proposed system is displayed in Fig. 2 along with a comparison to a number of cutting-edge systems.The accuracy of identifying dangerous or non-malicious APKs using various machine learning and deep learning classifiers is depicted in the following image.With a high accuracy rate of up to 95.40%, the proposed classifier has been utilized to determine if an APK is malicious or not.

Our model was trained with a large number of classifiers, which were then checked and compared to ensure higher accuracy. Each classification result will be displayed in graphs and tables for ease of comprehension.

Table I. Comparison Table

Classifiers	**Accuracy**	**Precision**	**Recall**	**F-score**
SVM	80.50	100	89.50	85.40
NB	81.40	100	87.40	82.40
RNN	95.40	96.50	94.10	97.15

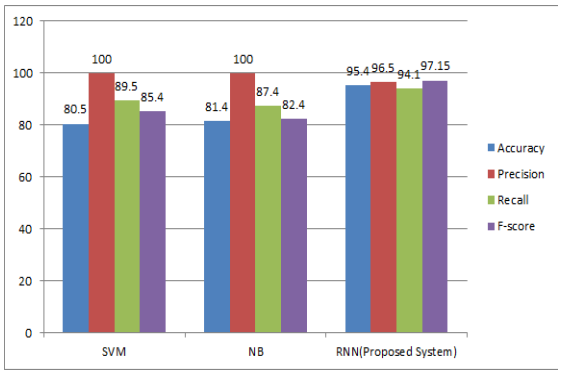

Figure 3: Comparison of Existing and Proposed System algorithms

CONCLUSION

Finding a machine learning solution to the malware issue is the aim of this study. We need automated techniques to detect tainted data since malware is getting increasingly complex. Using both clean and infected executables, we created the data set for the first part of the study. A Python script was then used to extract the data we needed to generate the data set. To train machine learning algorithms, the data collection has to be prepared after it is created. For the proposed malicious activity module, we examined many classification techniques. Support Vector Machine, Navie Bayes, and RNN were the algorithms that were employed. RNN classification has a 95% accuracy rate on numerous cross validations, whereas SVM and NB have the lowest accuracy.

References

Narayanrao, Purude Vaishali, and P. Lalitha Surya Kumari. "Analysis of machine learning algorithms for predicting depression." 2020 international conference on computer science, engineering and applications (iccsea). IEEE, 2020.

Laijawala, Vidit, et al." Classification algorithms based mental health prediction using data mining." 2020 5th International Conference on Communication and Electronics Systems (ICCES). IEEE, 2020.

AlSagri, Hatoon S., and Mourad Ykhlef. "Machine learning-based approach for depression detection in twitter using content and activity features." IEICE Transactions on Information and Systems 103.8 (2020): 1825-1832

William, David, and Derwin Suhartono. "Text-based depression detection on social media posts: A systematic literature review." Procedia Computer Science 179 (2021): 582-589.

Tadesse, Michael M., et al. "Detection of depression- related posts in reddit social media forum." IEEE Access 7 (2019): 44883-44893.

Shah, Faisal Muhammad, et al. "Early depression detection from social network using deep learning techniques." 2020 IEEE Region 10 Symposium (TENSYMP). IEEE, 2020.

Burdisso, Sergio G., Marcelo Errecalde, and Manuel Montes-y-G˜A lmez. "A text classification framework for simple and effective early depression detection over social media streams." Expert Systems with Applications 133 (2019): 182-197

Stankevich, Maxim, et al. "Depression detection from social media profiles." International Conference on Data Analytics and Management in Data Intensive Domains. Springer, Cham, 2019.

Al Asad, Nafiz, et al. "Depression detection by analyzing social media posts of user." 2019 IEEE International Conference on Signal Processing, Information, Communication Systems (SPICSCON). IEEE, 2019.

Chiong, Raymond, Gregorious Satia Budhi, and Sandeep Dhakal. "Combining sentiment lexicons and content-based features for depression detection." IEEE Intelligent Systems 36.6 (2021): 99-105.