Semiconductor Optoelectronics
ISSN: 1001-5868

A MACHINE LEARNING TECHNIQUE TO DETECT BEHAVIOR BASED MALWARE

Jangam Raghunath
Research Scholar, Dept. of Computer Science and Engineering, YSR Engineering College of YVU, Yogi Vemana University, Proddatur, YSR Kadapa, AP, India

S. Kiran
Assistant Professor, Dept. of Computer Science and Engineering, YSR Engineering College of YVU Yogi Vemana University, Proddatur, YSR Kadapa, AP, India

G. Siva Nageswara Rao
Professor, Dept. of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation,Vaddeswaram, Guntur, AP, India

J.R Arun Kumar
Professor, Dept. of Computer Science and Engineering, Model Institute Technology and Research Centre, Alwar, Rajasthan

R. Anasuya
Professor, Dept. of Computer Science and Engineering, Model Institute Technology and Research Centre, Alwar, Rajasthan

C. Siva Kumar
Associate Professor, Dept. of Data Science, Sree Vidyanikethan Engineering College,Tirupati, India

Abstract— While malware has posed a danger to businesses for years, advances in malware detection have lagged. Malware can cause damage to a system by starting unneeded services, which increases the system's workload and prevents it from operating smoothly. Malware detection can be done in one of two ways: the traditional signature-based approach or the more modern behavior-based approach. When malware is activated on a system, it conducts specific actions, such as launching malicious OS services or downloading malicious files from the web, that characterize its behaviour. The described technique detects malicious software based on its actions. The suggested model in this paper combines Support Vector Machine and Principal Component Analysis.

Keywords— Logistic Regression, Naive Bayes, Support Vector Machine.

Introduction

With more and more people relying on computers and the internet, it's become increasingly difficult to keep sensitive information safe. When computers access the internet, they acquire vast amounts of data, which may include malicious software. Malicious software also goes by the titles malicious code, malicious programmes, and malicious executable files. Malware assaults have been on the rise, leaving more and more computers open to intrusion. With the ever-increasing variety of malwares, anti-virus scanners cannot ensure the detection of every type of malware based on its signature, leading to the compromise of countless newly hosted websites and extensive harm to associated data and infrastructure. A computer virus can be thought of as a very short bit of code with the ability to replicate itself. Once a file is obtained or run, it attaches itself and begins its malicious behaviour. Worms, like viruses, can reproduce by copying themselves. A user's click on an infected ad / button, code attached to it accomplishes, and the user's computer is infected with a virus or a bot. The only difference concerning a disease and a parasite is that a worm works on the network and replicates by disseminating copies of itself to the devices linked to that network. Trojans are malicious programmes that trick users into thinking they are legitimate websites, login pages, or communication forms. The term "botnet" is used to describe a group of algorithms that operate together in a network. One line of code represents one machine. which is tasked with making it simple for an intruder to gain access to a user's computer. By breaking into a computer with a Yet, a hacker could install malware, steal sensitive data, or cripple the system.

Related Work

There is a lot of research in this field because malware has plagued users and networks for a long time. In this part, we examine the existing literature and draw conclusions about how adware can be detected most effectively. This part will also provide a brief overview of the benefits and drawbacks of some of the related research:

Fan, Ye, and Chen [1] proposed using the All Nearest Neighbour (ANN) dynamic series mining method to assess malwares in light of their unique techniques. The virus method was built upon the decompression of Portable Executable (PE) files. Both viruses and the ANN method were discovered. Although time and area could be reduced by filtering out the repetitive patterns, this approach was not without its flaws. expensive in terms of time spent, accuracy of classification, and inability to spot newly-released, highly-advanced malware threats. Author used SVM, Naive Bayesian, and Decision Tree to identify malware, which was evaluated by Fan, Hsiao, Chou, and Tseng[2] using the API's (Application Programming Interface) capabilities. The efficiency of classification algorithms was greatly enhanced by the Parameter Selection Method, which employs fewer training characteristics. Reduced training attribute usage via the Attribute Selection Technique led to vastly enhanced categorization algorithm performance. Baldangombo, Jambaljav, and Horng proposed inspecting PE files' raw characteristics as a means of achieving static malware detection. This method operationalized ML/FS/DT concepts like feature selection and data change. By picking out the useful features, the unnecessary ones could be removed. The main drawback of that approach was how memory and processing intensive it was. A novel machine learning approach was proposed by Sanz and coworkers[3] to identify malicious codes in a system based on the characteristics of incoming packages. Primary contributions made by the initial article were: Extraction of Android characteristics was demonstrated. It was illustrative of methods for identifying malicious

Android applications. The detection rate for malware was excellent. To detect API call sequence and spot malicious code, Jerlin and Marimuthu [4] suggested employing a powerful Rete-based MDNBS approach. The main motivation behind this methodology was to raise the bar for malware detection precision using the provided dataset. The advantages of these methods are: Faster Computing and Highly Accurate Detection.

Background Study

The purpose of this two-part creation is to detect malicious code using machine learning and other advanced methods.

Study of malware

The first step in the detection procedure is network forensics. In this step, we compile data on viruses that are already known to exist. A virus's traits can be used to inform the development of an algorithm that can detect newly arriving infections

Malware Detection

Once the analysis is complete, a suitable algorithm can be generated to identify malware with a high degree of accuracy. The created algorithm is applied to inbound packets, which are then analyzed to determine whether or not they contain malicious code

Anti Malware Methods

Assault prediction tools include data mining[5, 6], deep learning[6, 7], and idea exploration[8, 9]. However, learning has become increasingly common as a means of virus detection. Malware identification can be broken down into two categories (see figure 1). Malware is initially identified using a traditional handwriting technique. The other, more modern technique of malware detection is behavior-based, and it recognises spyware by the activities it plans to carry out on the target system. The behave method is more advanced because it can detect newly created malware by analysing the actions it takes on a computer.

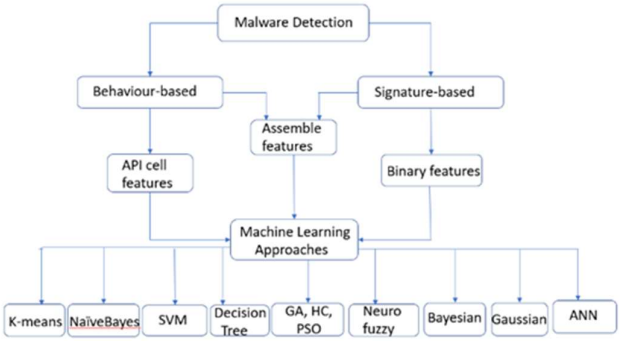

Fig. 1. Architecture of Malware Detection Approaches

Behavior Based Approach

When a piece of code is run on a computer, its behaviour is the action it takes. Payload Persistence, Stealth Methods, Mapping the Environment[9], etc. are all examples of such actions. Behaviour patterns malware detection[10] analyses incoming images in light of the tasks and actions they were programmed to perform before deciding whether or not to proceed with an operation. Examining a file's functions and possibly its danger level is being investigation. Dynamic analysis refers to the process of investigating a system in real time for malevolent behaviour. Behavior-based approaches are used in technology to rapidly spot novel

and unanticipated risks, despite the fact that no computer can ever be 100% accurate.

Methodology

Support Vector Machine

As a popular Supervised Learning technique, Support Vector Machine (SVM) can be applied to both regression and classification issues. However, its primary application is in computer vision, where it is used to address issues of categorization and regression. The goal of SVM software is to find the best decision boundary or vector for classifying an n-dimensional space, making it simple to add new data points to the appropriate group in the future. If you must pick a boundary, make it vertical. SVM finds the extreme values and locations that contribute to the growth of the edges. Support vector machines are used in this technique, which is also known as a machine learning machine. Take a quick look at the diagram below, which shows how a higher-dimensional region of interest can be used to successfully separate points into two separate groups. They are an established SVM consensus.

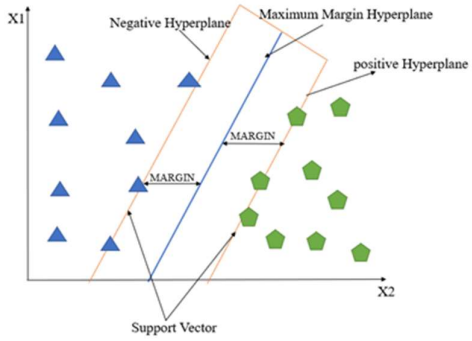

Fig. 2. Support Vector machines

Linear SVM

Data that can be classified into two groups using a single linear model is said to have unit approximations in their input, and so this data is classified using the linear SVM technique.

Non-linear SVM

Non-linear SVM is being used, which indicates that non-linear Classification models have been implemented, for variables that have been non-linearly divided. when a sample defies linear classification.

Algorithms

Support vector classifiers algorithm

The SVM algorithm, also known as the algorithm for Support Vector Machines, is a simple yet effective technique for machine learning with guidance that can be used to construct regression and classification models simultaneously. The SVM method works very well with both continuously and non-linearly separable data. No matter how much data you have, the supported vector machine method usually does quite well.

- Use Pandas to import the information and the Numpy arrays library
- Identify the subject and the attributes.
- Before creating the SVM analysis point of view, use Vary depending on the source to divide the dataset into to the training and test categories.

- Before generating the SVM database in order, use Descriptive term to partition the information into to the trained and various quality parameters.
- Use a modelling of the SVM classifier to estimate the values.
- Assess the output of the supported vector machine.

PCA(Principle Component Analysis)

A methodological approach called principal component analysis (PCA) is utilized to lessen the dimensionality. transforms a collection of data of potentially dependent variables via an orthogonal transformation into a set of principal component values, which are values of linearly uncorrelated variables. It is frequently employed as a method of dimension reduction. Standardize the Dataset: Assuming we have the dataset below, which consists of 4 characteristics and 5 training instances altogether.

z1	z2	z3	z4
2	1	1	3
4	5	3	1
3	6	2	4

Create an overall dataset covariance matrix. Covariance matrix calculation formula: The dataset must first be standardized, and to do that, the means and standard deviations of each characteristic must be determined.

Before standardization, the mean and standard deviation.

	z1	z2	z3	z4
μ	3	4	2	2.7
σ	1.41421	2.64575	1.41421	1.52807

Standardization formula:

$$x_{new} = \frac{x - \mu}{\sigma}$$

When the formula is used, the following changes are made to each feature in the dataset:

z1	z2	z3	z4
-0.70710	-1.33389	-0.70710	0.19632
-0.70710	0.37796	0.70710	-1.11251
0	0.75592	0	0.85074

Perform full-dataset covariance matrix computations.

Method for determining the correlation matrix:

For Population	$\text{cov}(x, y) = \frac{\sum(x_i - \bar{x})*(y_i - \bar{y})}{N}$
For Sample	$\text{cov}(x, y) = \frac{\sum(x_i - \bar{x})*(y_i - \bar{y})}{(N - 1)}$

The following formula will be used to determine the covariance matrix for the specified information.

	z1	z2	z3	z4
z1	var(z1)	cov(z1, z2)	cov(z1, z3)	cov(z1, z4)
z2	cov(z2, z1)	var(z2)	cov(z2, z3)	cov(z2, z4)
z3	cov(z3, z1)	cov(z3, z2)	var(z3)	cov(z3, z4)
z4	cov(z4, z1)	cov(z4, z2)	cov(z4, z3)	var(z4)

We have normalized the data, so now the average value of each feature is zero and the standard variation is one.

var(z1)=((-0.70710-0)²+(0..70710-0)²+(0-0)²)/3

var (z1) = 0.33

cov(z1,z2)=((-0.70710–0)*(-1.33389-0)+(-0.70710–0)*(0.37796-0)+(0–0)*(0.75592 -0)))/3

cov(z1, z2) = 0.22531

The other co values can be calculated in a similar fashion, yielding the covariance matrix shown below.

	z1	z2	z3	z4
z1	0.16	**0.22531**	0	0.64784
z2	0.22531	0.83120	0.40348	-0.01308
z3	0	0.40348	0.33332	-0.30849
z4	0.64784	-0.01308	-0.30849	0.66666

Determine eigenvalues and eigenvectors; an eigenvector is a nonzero vector whose magnitude varies by no more than a scalar factor as a result of applying that linear transformation. The eigenvalue is the multiplier used to transform the eigenvector. An eigenvalue linked with an eigenvector ν of A square matrix (here, the covariance matrix) is denoted by the scalar if and only if $A\nu=\lambda\nu$.

Altering the above formula,

$$A\nu-\lambda\nu = 0;\ (A-\lambda I)\nu = 0$$

The only way this equation can equal zero is if $\det(A-\lambda I) = 0$, since we know v is not a zero-length vector.

Solving the above equation = 0

λ =0.1474, 0.3983, 0.6848, 1.1619

	z1	z2	z3	z4
z1	0.16-λ	**0.22531**	0	0.64784
z2	0.22531	0.83120-λ	0.40348	-0.01308
z3	0	0.40348	0.3333-λ	-0.30849
z4	0.64784	-0.01308	-0.30849	0.66666-λ

Calculating (A-I)v = 0 for a variety of velocities v yields the eigenvectors:

$$\begin{pmatrix} 0.16000-\lambda & 0.22531 & 0.00000 & -0.64784 \\ 0.22531 & 0.83120-\lambda & 0.40348 & -0.01308 \\ 0.00000 & 0.40348 & 0.3333-\lambda & -0.30849 \\ 0.64784 & -0.01308 & -0.30849 & 0.66666-\lambda \end{pmatrix} * \begin{pmatrix} v1 \\ v2 \\ v3 \\ v4 \end{pmatrix} = 0$$

After plugging the above equation into Cramer's formula, we get the following values for the v vector when

λ=1.1619.

V1	=	*0.8502*
V2	=	*-1.0106*
V3	=	*-1.2599*
V4	=	*-0.5569*

Following the same method, we can determine the eigen vectors for the remaining eigen values. From a matrix, we can derive the corresponding eigenvectors.

eig1	eig2	eig3	eig4
-1.0019	-0.2253	0.0000	-0.6478
0.2253	-0.3308	0.4035	-0.0131
0.0000	0.4035	-0.1286	-0.3085
-0.6478	-0.0131	-0.3085	0.4959

eigenvectors (4 * 4 matrix)

Arrange eigenvalues and eigenvectors in a meaningful way. There is no need to perform a second classification on the eigenvalues here because they are already in order. To create an eigenvector matrix, select k eigenvalues. This is what the matrix looks like if we pick the first two eigenvectors:

eig1	eig2
-1.0019	-0.2253
0.2253	-0.3308
0.0000	0.4035
-0.6478	-0.0131

Top 2 eigenvectors (4*2 matrix)

Transform the original matrix.

Proposed Method

In this section, a comprehensive description of the planned effort for identifying malware will be provided. In this method we use SVM supported by PCA. PCA helps in dimensionality reduction by calculating variance. The characteristic extractor was given a collection of malicious and benign programmes that had previously been identified by a variety of sources. These programmes had previously been compiled and submitted. After the feature extraction process was finished, a total of 77 features had been developed, all of which are going to be utilized during the training of the model. Advantages: The level of accuracy is at its highest. It can be predicted accurately.

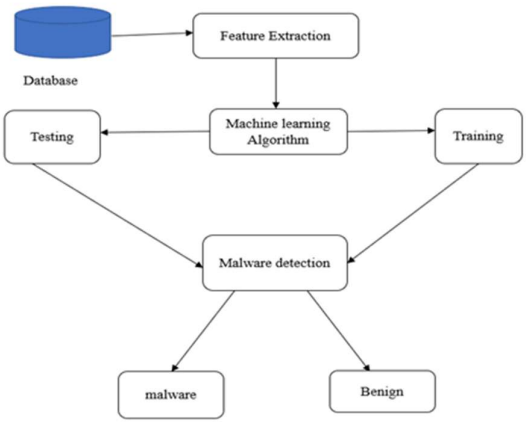

Fig. 3. Architecture of Proposed System

Result Comparision

In this study, the Support Vector Machine (SVM) and Principal Component Analysis (PCA) were utilized as classifications. A total of 93.0% of the SVM classifiers were found to perform best overall, with an average accuracy of 96.8%, according to a research that analyzed all of the tests and experimental findings.

Comparing with different Classifiers

Classifier	Accuracy	precision
KNN	90.45	89.1
Naïve bayes	86.3	62.8
Support Vector Machine	93.0	96.8

Accuracy: Accuracy can be defined as the proportion of times that a model's predictions match up with the actual data being tested. It is a straightforward calculation that can be performed by dividing the number of accurate predictions by the overall number of predictions.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Precision: One indicator of a machine learning model's performance is its level of precision, which refers to the quality of a favourable prediction that the model has made. The term "precision" alludes to the ratio of the number of actual positive results to the overall number of successful predictions (i.e., the number of true positives plus the number of false positives).

$$\text{Precision} = \frac{TP}{TP+FP}$$

Here TP = True Positive

TF = True Negative

FP = False Positive

FN = False Negative

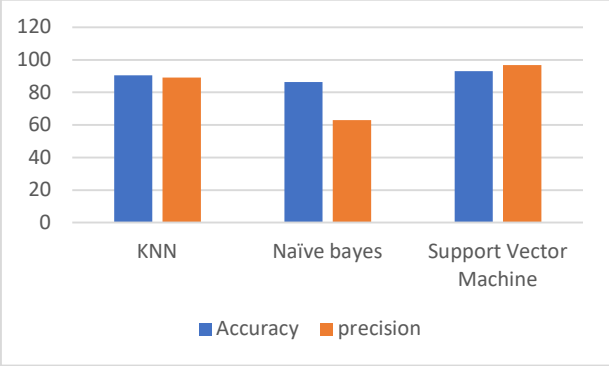

Fig. 4. precision and Accuracy of Different Classifiers

Figure 4 explains the accuracy and precisions of different classifiers. The Accuracy and Precision of KNN is 90.45 and 89.1. The Accuracy and Precision of Naïve Bayes is 86.3 and 62.8. The Proposed System Accuracy and Precision is 93.0 and 96.8. Here Support Vector Machine has highest Accuracy and Precision comparing to KNN and Naive Bayes.

Conclusion

In conclusion, it is possible to assert that a proof-of-concept for an alternative malware detection method has been developed as a result of this research. Feature selection was demonstrated in this research using K-Nearest Neighbor. By conducting PCA (Principle Component Analysis), the features were reduced drastically in a dimensional fashion. As a result, the amount of time necessary to train and construct the model requires less time, although this comes at the expense of a modest reduction in its overall performance. There are some circumstances in which the performance can even marginally improve. The performance comparison of 2 different classifiers was also demonstrated. The overall best performance was obtained by SVM with 91.3 accuracy. The findings of the tests and the experiments, when analyzed, showed that this proof-of-concept is quite effective and efficient in identifying malicious software.

References

[1] J. Landage and M. Wankhade, “Malware and Malware Detection Techniques: A Survey,” Int. J. Eng. Res. Technol., vol.
2, no. 12, pp. 61–68, 2013
[2] Kaspersky, “Machine learning for Cybersecurity.”
[3] P. Kaur and S. Sharma, “Literature Analysis on Malware Detection,” Int. J. Electron. Electr. Eng., vol. 7, no. 7, pp. 717–
722, 2014.
[4] I. A. Saeed, A. Selamat, and A. M. A. Abuagoub, “2013-A Survey on Malware and Malware Detection Systems.pdf,” vol.
67, no. 16, pp. 25–31, 2013.
[5] U. Baldangombo, N. Jambaljav, and S.-J. Horng, “A Static Malware Detection System Using Data Mining Methods,” 2013.
[6] Y. Saint Yen and H. M. Sun, “An Android mutation malware detection based on deep

learning using visualization of
importance from codes," Microelectron. Reliab., vol. 93, no. October 2018, pp. 109–114, 2019.
[7] S. Sohrabi, O. Udrea, and A. V. Riabov, "Hypothesis Exploration for Malware Detection Using Planning," Twenty-Seventh
AAAI Conf. Artif. Intell., pp. 883– 889, 2013.
[8] J. C. Rosales, "Rehumanización y metáfora religiosa en Luis Rosales," Insula, vol. 767, pp. 32–34, 2010.
[9] D. Nieuwenhuizen, "A behavioural-based approach to ransomware detection," Whitepaper. MWR Labs Whitepaper, 2017.
[10] H. S. Galal, Y. Bassyouni, and M. A. Atiea, "Behavior-based features model for malware detection," J. Comput. Virol.
Hacking Tech., no. April, 2018.
[11] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," Expert Syst. Appl.,
vol. 52, pp. 16–25, 2016.
[12] C. I. Fan, H. W. Hsiao, C. H. Chou, and Y. F. Tseng, "Malware detection systems based on API log data mining," Proc. - Int. Comput. Softw. Appl. Conf., vol. 3, pp. 255–260, 2015.

[13] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, P.G. Bringas, and G. Álvarez, "PUMA: Permission usage to detect
malware in android," Adv. Intell. Syst. Comput., vol. 189 AISC, pp. 289–298, 2013.

[14] M. A. Jerlin and K. Marimuthu, "A New Malware Detection System Using Machine Learning Techniques for API Call
Sequences," J. Appl. Secur. Res., vol. 13, no. 1, pp. 45–62, 2018.

[15] S. Niranjana, I. Chandra, G. Charulatha, S. Leopauline, C. Prathima and T. Geetha, "A Novel Orientation Approach in
Artificial Intelligence for Mounting Robots Utilizing a Three-Dimensional Framework of the Broadcast Tower," 2023
International Conference on Artificial Intelligence and Knowledge Discovery in Concurrent Engineering
(ICECONF), Chennai, India, 2023, pp. 1-7, doi: 10.1109/ICECONF57129.2023.10084216.

[16] Infectious diseases of Rice plants classified using a deep learning-powered Least Squares Support Vector Machine
Model,Goluguri, N.V.R., Suganya Devi, K., Prathima, C.H.Indian Journal of Computer Science and Engineering this link
is disabled, 2022, 13(5), pp. 1640–1659.
[17] A Novel Deep Learning Method for the Identification and Categorization of Footpath Defects based on Thermography,
Vanitha, L., Kavitha, R., Panneer selvam, M., Prathima, C.H., Valantina, G.M.3rd

International Conference
on Smart Electronics and Communication, ICOSEC 2022 - Proceedings, 2022, pp. 1401–1408.

[18] Auto Encoders and Decoders Techniques of Convolutional Neural Network Approach for Image Denoising In Deep
Learning, P Chilukuri, JRA Kumar, R Anusuya, MR Prabhu, Journal of Pharmaceutical Negative Results 13 (4), 1036-
1040,2022.

[19] Prathima, C., Reddy, L.S.S. (2019). A Survey on Efficient Data Deduplication in Data Analytics. In: Soft Computing
and Medical Bioinformatics. Springer Briefs in Applied Sciences and Technology. Springer, Singapore.
https://doi.org/10.1007/978-981-13-0059-2_12.

[20] Chilukuri, P (Chilukuri, Prathima) ; Anusuya, R (Anusuya, R.) ; Prabhu, MR (Prabhu, M. Ramkumar), Comprehensive Design Analysis Of Digital Marketing In Agriculture Sector, International Journal Of Early Childhood Special Education ,vol 14,Issue :5 page:814-821, DOI10.9756/INTJECSE/V14I5.81.

[21] B. Dinesh, P. Chilukuri, G. P. Sree, K. Venkatesh, M. Delli and K. R. Nandish, "Chat and Voice Bot Implementation for
Cardio and ENT Queries Using NLP," 2023 International Conference on Innovative Data Communication Technologies
and Application (ICIDCA), Uttarakhand, India, 2023, pp. 124-130, doi: 10.1109/ICIDCA56705.2023.10099942.

[22] V. Rakesh, P. Chilukuri, P. Vaishnavi, P. Sreekaran, P. Sujala and D. R. Krishna Yadav, "Real Time Object Recognition
Using OpenCV and Numpy in Python," 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), Uttarakhand, India, 2023, pp. 421-426, doi: 0.1109/ICIDCA56705.2023.10099584.