International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Machine Learning Approach for Malware Detection and Classification Using Malware Analysis Framework

D Anil Kumar [1], Susant Kumar Das[2]

Submitted: 27/10/2022 **Revised:** 18/12/2022 **Accepted:** 05/01/2023

Abstract*:* The world's digitalization is currently being threatened by the daily appearance of new and complicated viruses. As a result, the conventional signature-based approaches for malware detection are practically rendered useless. Modern research studies have demonstrated the effectiveness of machine-learning algorithms in terms of malware identification. In this study, we suggested a system to identify and categorize various files (such as exe, pdf, PHP, etc.), and API calls as benign and harmful utilizing two-level classifiers, namely Macro (for malware detection) and Micro (for classification of malware files as a Trojan, Spyware, Adware, etc.). One of the most used data mining (DM) methods is classification. In this research, we describe a classification technique for DM for malware discovery. On the basis of the characteristics and behaviors of each virus, we suggested many categorization approaches to identify malware. The malware traits have been identified using a dynamic analysis technique. Our solution executes sample files in a virtual environment using Cuckoo Sandbox to generate static and dynamic analysis reports. Additionally, utilizing the data produced by the Cuckoo Sandbox, a unique feature selection, and extraction segment has been produced that operates based on static, behavioral, and network analysis. Machine learning models are created utilizing the Weka Framework and training datasets. The experimental findings utilizing the suggested framework demonstrate high rates of detection and classification using various Machine Learning Algorithms.

Keywords*:* Malware Detection; API-call; Static and dynamic analysis; malware classification; behavior-based analysis.

1. Introduction

One of the biggest hazards on the Internet today is malicious software or malware. Users download many kinds of computer applications on a massive scale from the Internet. Online black markets are used by hackers to create software that violates system security. This gives hackers a significant incentive to alter and make harmful code more sophisticated in an effort to create more uncertainty and reduce their chances of being discovered by anti-virus software. As a result, accessing the Internet is becoming riskier and riskier as a result of growing dangers from malware [1], which is distributed over the Internet in the form of files and software.

Malware is a harmful application that is used to violate a system's data availability, confidentiality, and integrity policies. Malware comes in a variety of forms, including viruses, Trojan horses, spyware, rootkits, trapdoors, etc., depending on how they pose dangers to the system. The overall quantity of malware has increased dramatically since 2008 and reached more than 583 million, in March 2020, according to AV-Test [2]. Finding these files before they violate the system's security perimeter is crucial given the increased incidence of malware. According to the report, the malware detection system comprises the duties of malware analysis [3].

[1 & 2] *Berhampur University, Odisha, India*
ORCID ID : 0000-0003-3998-226X
** Corresponding Author Email: anil.dodala@gmail.com*

1.1. Malware Detection

The use of signature-based and behavioral-based approaches, two well-known detection methods, is made. But signature-based methods are unable to identify Zero-Day attacks. Additionally, it cannot identify sophisticated new malware. On the other hand, it is highly challenging to properly describe the whole variety of appropriate behaviors that a system should show when employing behavior-based methodologies. The majority of malware detection systems typically employ static methods like signature-based and anomaly-based methods of detection. While some systems attempt to discover irregularities in the code structure, others use signature matching to assess whether a program is malicious. Static approaches investigate malware programs without running them in order to understand the code structure [4]. When doing dynamic analysis, malware is run in a virtual environment to track its network interactions and Windows API calls. These observed API call data are utilized by dynamic malicious program detection techniques to identify harmful behavior. The function names, arguments, and return values of an executable are contained in API-call information. Through the use of the occurrence and arrangement of API calls, dynamic approaches attempt to derive distinguishing characteristics to identify malware programs [5].

As per the Malware detection statistics by AV-Test institute, there are more than 1 billion malware programs out there spreading every year. Since 2013 it was spreading

exponentially. Nearly 560,000 new malware pieces are detected on a daily basis worldwide. As per the last statistics, near about 17 million new malware are reported on a monthly basis. As per Sonic Wall data, more than 3.2 billion malware in the 1st half of 2020. As per google statistics, nearly 7% of new websites were affected every year and each week it was reported that nearly 50 websites contained malware. As per Symantec, around 20 million IoT devices were malware affected and out of which 75% were through routers. As per Statista China is the most malware affected globally with 47% followed by Turkey with 42% and Taiat with 39%. As per checkpoint, the viruses are spread through .exe files and the malware is generally delivered through emails.

Different mobile malware statistics have been proposed. A few of them are listed below:

As per Kaspersky total number of mobile malware surpassed 28 million during the 1st half of 2020. And around 14 million are detected in each quarter every year. Adware is one of the most common pieces of mobile malware.

Because malware is becoming more prevalent in technology, understanding how to guard against it is a crucial component of malware detection using machine learning techniques. In general, data mining techniques identified a group of malware applications in the public that included both harmful executable and innocuous software packages [6]. Typically, there are two different types of data mining algorithms based on supervised learning and unsupervised learning techniques. Classification algorithms are the supervised learning techniques that are required for the exercise with the data set [7]. The unsupervised learning techniques, known as clustering algorithms, seek to analyze the organization of data into several clusters [8].

1.2. Malware Analysis

This includes static, dynamic, and hybrid analyses. We developed a malware analysis solution utilizing a machine learning approach to distinguish between benign and malicious files in response to the aforementioned limitations of the existing techniques. In order to effectively and efficiently identify and categorize malware, we have suggested an intelligent malware analysis methodology in this study.

Malware programs are often divided into categories including worms, viruses, trojan horses, spyware, back doors, and rootkits [9]. Using signature-based techniques is the cornerstone of conventional and customary approaches to malware identification. Researchers have recently tried to propose more trustworthy methodologies for malware identification with malware behavior after being frustrated by outdated methods' failure to identify malware or polymorphic dangerous files [10]. Static analysis and dynamic analysis have both been used in the process of identifying and locating the malware. Static analysis, which may identify harmful code and place it in one of the available collections depending on various learning techniques, is a technique used in software analyzing approaches. Static analysis uses binary codes to identify harmful files and viruses. The biggest drawback of static analysis is the absence of the program's source codes. It is important to note that extracting binary codes is a difficult and intricate task.

The dynamic analysis, in contrast, detects dangerous scripts based on their runtime behavior [11]. Dynamic analysis, which also refers to behavioral analysis and observation of behavior and system operation, is the term used to describe the examination of runtime code [12]. The infected files need to be run on a virtual system via a dynamic analysis process [13]. To manage the expanding number and variety of malware, dynamic analysis can be employed in conjunction with classification and clustering techniques. The approaches for classifying malware aid in the assignment of unidentified malware to known families. Malware categorization is therefore employed to filter unknown instances, which lowers analysis costs.

The following are some of this paper's contributions:

- Putting out a behavioral analysis detecting system.
- Introducing software that converts an XML file containing a malware behavior executive history into a WEKA input that is appropriate.
- Examining several categorization techniques using a virus case study.
- Comparing the experimental findings from the WEKA tool, such as the proportion of correctly classified instances, and the accuracy optimistic ratio.
- To create a behavioral antivirus, the optimal categorization approach based on critical malware detection criteria is being tested.

The overall arrangement of this article is as follows: Section 2 discusses some historical context and related research in virus detection and data mining approaches. The behavioral analysis of the malware is shown in Section 3. In this part, using a real-world case study, we provide a novel method for deciphering malware behavior and converting dangerous files into data mining files. The classification and prediction methods used with the data mining platform are also described in this section. Then, using the WEKA tool, we apply some of the well-known categorization techniques to our actual case study. Section 4 summarizes the assessment and experimental findings. Section 5 brings the conversation and the work to a close.

2. Related Works

The background information and some associated efforts for malware detection in data mining approaches are covered in this part. First, we quickly go through data mining methodology based on malware and other system classification techniques. Researchers recently revealed

various malware analysis methodologies. A data mining technique was put out by Schultz et al. [14] to identify new dangerous files during runtime execution. Their approach was based on three distinct sorts of DLL calls, including the binary's list of DLLs utilized, the list of DLL functions used, and the number of various systems calls used inside each DLL. Additionally, they use signature techniques to check the byte ordering that was retrieved from an executable file's hex-dump (a hexadecimal schema of computer data). This method's primary structure is based on the Naive-Bayes (NB) algorithm. The experimental findings were compared using conventional signature-based techniques.

Dynamic analysis approaches, which examine program activities while running in a secure environment, have been used in several research. By examining a large number of malware mutation files, Jeffrey, N et al. technique [15] suggests typical patterns of malware programs. Based on the frequency of API calls, Amer et al [16] dynamic malware detection technique is suggested. By tracking API calls Zou et al. [17] examine the malware executables' API-call rates and sequences. In order to identify malware variations, Schofield et al. [18] provide an approach that builds representations of malware behavior by mapping API calls to colors. Function call patterns are used with a Hidden Markov Model to categorize malware. Algorithms for sequence alignment are combined with API call sequences. In order to increase the effectiveness of the detection algorithms, non-essential API functionalities are removed. Text-mining methods are used on the API call sequences to analyze the operation, location, and parameter information of each API call to determine the behavior of malware. Focusing on the examination of dangerous dynamic libraries loaded by portable executable files, Chaganti, R et al. Use both static and dynamic analysis, with a multi-view feature fusion approach suggested and also a hybrid technique suggested along in [19] to extract common properties of malware instances. Using both static and dynamic analysis, Zhu, et al. [20] hybrid approach proposes extracting common aspects of malware instances. They make use of dynamic-link libraries and API-called functions. In order to spot malicious instances, Hasan, et al. [21] analyze the frequency of system actions that portable executables trigger.

It is used to conduct tasks automatically, examine files, and gather thorough analytical data. These discoveries retrieve API call traces, information on registry and file modification, network traffic logs, and particular log data of the malware's flow path within an isolated operating system. Cuckoo Sandbox has been utilized by researchers Sraw, J. S et al. [22] for malware investigation.

To extract elements like registry activity, API calls, and imported libraries, another effort concentrated on the memory pictures. Additionally, it evaluated the effectiveness of several machine learning methods and discovered that SVM (support vector machine) outperformed the others. While leveraging supplied arguments, Thakur et al. [23] built API calls with in-depth analysis. They also attempted to categorize and evaluate a big quantity of malware. They employed a mix of characteristics in different studies to reach a high categorization rate[24]. We go into further depth about our suggested approach to malware analysis in the next section.

3. Malware Behavior Analysis

As seen in Table 1, we have considered malware datasets for malware behavioral analysis techniques. It consists of two malware datasets. This approach will use a suggested program to transform an XML file containing the executive history of malware action into a non-sparse matrix [25]. This app was created using the VB.Net programming language. A screenshot of our proposed application's XML converter to a non-sparse matrix. The amount of library file calls targeted by malicious program and their volume are two components of turning each XML file into an appropriate WEKA input. For instance, in Box 1, the malware has called the XML library file ntdll.dll 16 times, ranging (0, 2). We next convert this matrix into the WEKA input data set [26]. Some classification algorithms will come before the training techniques. The new data set virus will choose the classifications with the greatest performance for the test platform. Finally, a behavioral antiviral may be developed using this process [27]. We employ 10540 rows of files for our experiment. For each malware, the dataset has 57 attributes. Then, using our recommended application, we transform this XML file into a non-sparse matrix. Non-sparse matrix has two integers, the first of which indicates the number of qualities and the second of which indicates the significance of those properties. This matrix's first row is displayed as follows.

(SystemSettings.DeviceEncryptionHandlers.dll|f226d169
22369a8ea24e8156db40a373|34404|240|8226|14|12|1024
00|62464|0|98176|4096|0|6442450944|4096|512|10|0|10|0|
10|0|180224|1024|202454|3|16736|262144|4096|1048576|
4096|0|16|6|4.60129473839|2.71413309005|6.381756727
95|27136.0|1536|102400|27197.3333333|1176|102000|38|
174|6|4|1|3.44737833601|3.44737833601|3.44737833601|
1076.0|1076|1076|256|16|1)

Where the '|' indicate the separation of parameters of one row of the file. The 1st part contains the name of the file, the second is the Hash code, etc.

The decision-making history of the malicious program in the WEKA platform is examined last. To execute malware safely in computer systems and stop it from spreading, some programs, including the SandBox tool and virtual machine, may create a malware executive history [28]. The XML file contains useful information, including calls to system library files, file creation, search and change operations, registry operations, information about primary

processes, creation of mutexes (which allow multiple programs thread for sharing a single resource), alterations to virtual memory, email transmissions, registry operations, and switch communications. The proposed software reads and stores all the data in a non-sparse matrix [29].

4. Malware Analysis Framework

We provide our suggested process for identifying and categorizing a sample of a file in this section. The proposed method-operational logic flow is shown in the flow graph in Figure 1. Behavior monitoring, feature extraction, Data collection, analysis, report handling, and detection and classification are the first four steps of this process. The following subsections offer a full discussion of these phases.

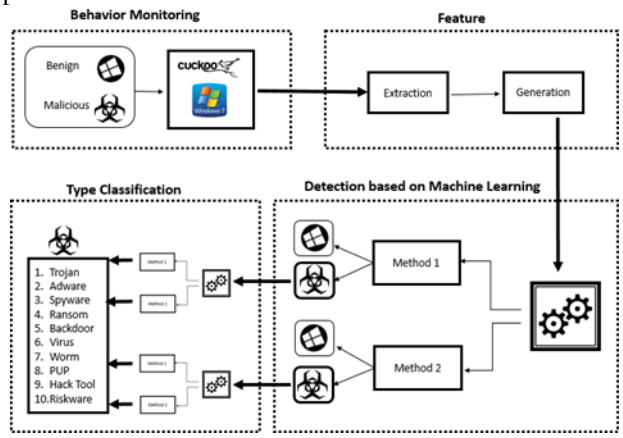

Fig. 1. Flow of operation for malware classification

The Guests are the isolated environments where the malware samples are really securely performed and studied during the whole analysis process. Two subphases comprise the analysis phase: i) Sandbox configuration and (ii) Sandbox configuration. The following is a detailed discussion of these subphases:

4.1. Sandbox Configuration

Configuring Cuckoo Sandbox [30] is crucial if you want to obtain malware behavior reports and make sure malware samples operate correctly, including all of their capabilities. In the real world, many malware samples take use of various flaws that may exist in certain software products. As a result, it's crucial to include a variety of services in the virtual machines that the sandbox creates. VirtualBox serves as the hypervisor for the virtual machines utilized by Cuckoo. One Intel Core i5 2.13 GHz CPU and 8 GB of RAM, and an internet connection make up a virtual machine's specifications. Adobe PDF reader, Python, and Windows 10 (64-bit) are the installed programs on the virtual system.

4.2. Malware analysis lab set up

A malware analysis environment was developed. In the guest machine's starting menu, the required Cuckoo agent is installed. The host computer has the Cuckoo host installed. Cuckoo's host configuration is set up in accordance with the virtual machine that will be utilized to run the sample. While the NAT adapter is used to connect the Cuckoo guest (XP virtual machine) to the internet, Virtual Box only Adapter (Vboxnet0) is used to connect the virtual machine and Cuckoo host. A snapshot is kept of the virtual machine's initial state, which is a malware-free and unharmed condition. The Python script cuckoo.py (cuckoo host) is run with root privileges to begin analysis on any file. Once the Cuckoo host is started, we may send files to a virtual computer for examination in accordance with Cuckoo parameters. Cuckoo Sandbox executes the virtual machine's files in a clean state when a sample is given, monitors every activity taking place in the virtual environment, and creates a report for each sample [31-32]. The web interface and API Calls may both be used to get the Analyzed Report "AR". To create Macro and Micro datasets, the Report Handler is used to retrieve the AR, as mentioned in the next part.

5. Experimental Results and Discussion

We used the WEKA tool in this part to put our strategy into practice. For the categorization techniques, we utilize a PC with an Intel Core i5 2.13 GHz CPU and 8 GB of RAM. Several classification techniques, including K-Neighbor, XGB, Random Forest, and Light GBM approach, were used for this investigation. We compared how well various categorization techniques performed in two malware data sets.

For the suggested classification techniques, Table 1 details the analysis of statistical Data Sets 1 and 2. The elements that make up the classification techniques include Correctly Classified Instances and Incorrectly Classified Instances. Through this comparison, we are able to demonstrate that the regression classification algorithm detects malware the best. As an illustration, the 5281 malicious programs and 5259 benign programs.

In our datasets, a total of 10540 samples out of which there are 5281 harmful samples and 5259 benign samples. Using a daily downloading routine, the Mal share website is used to download the infected samples [27]. Then, using VirusTotal [30], each sample is verified and stored according to its date. To be included in our dataset, the sample must have the support of five antivirus engines. As previously mentioned, malware samples of the same sort have comparable characteristics and actions. It is difficult to determine the malware type's ground-truth label, as different anti-virus providers may assign several detection labels (types) to the same scanned sample.

TABLE 1: Dataset description

Sample	Type	No. of samples	%
Malicious	Adware	135	1.28083491
	Backdoor	132	1.25237192
	Hack Tool	13	0.12333966
	PUP	21	0.19924099
	Ransom	221	2.09677419
	Riskware	7	0.06641366
	Spyware	241	2.28652751
	Trojan	4302	40.8159393
	Virus	74	0.70208729
	Worm	135	1.28083491
Benign	APIMDS	142	1.34724858
	CNET	153	1.4516129
	CYGWIN	2864	27.1726755
	DLL files	568	5.38899431
	File Hippo	27	0.25616698
	Portable applications	263	2.49525617
	WINDOWS 10	996	9.44971537
	Windows executable	246	2.33396584
Total		10540	-

Researchers thus start investigating other malware sample tagging methods. For instance, in [23], Thakur, D et al. use the open-source, automated program AVClass to identify the type of malware from a sample, in addition to a confidence level that was determined by using the level of anti-virus application from VirusTotal and the engine-level agreement. The ground-truth labeling is outside the purview of this research, but for our malicious dataset, each sample's malware type is identified according to data provided by the Malware- bytes engine in VirusTotal. Based on VirusTotal data, the age of our harmful samples is in the range of April 2020 and June 2021. Table 2 lists the different malware categories and the number of samples for each category.

The benign samples come from a total of eight sources. We downloaded the APIMDS dataset after installing a new copy of Windows 10 and extracting from c:windowssystem32 directory's, the Windows executables and DLL files. (1) In order to test legal downloading, we used free websites. (2) From the file Hippo website, we downloaded the top 43 programs and the top 300 portable Windows apps(3) We extracted the two folders, CYGWIN and WINDOWS 10 benign samples, from the benign dataset from downloaded files. Windows executable files are included in both directories and were copied from the required author sources. Using VirusTotal, each and every benign sample from the eight sources has been confirmed. Table 1 lists the no of trials from each safe source (1) Among the four algorithms, the MLP and MLR outperform the SVR and SLR in slope stability prediction. The MLP has an accuracy parameter, Kappa value, and AUC of 90.89%, 0.799, and 0.908, which are considered to be excellent predictions result.

Researchers thus start investigating other malware sample tagging methods. For instance, in [23], Thakur, D et al. use the open-source, automated program AVClass to identify the type of malware from a sample, in addition to a confidence level that was determined by using the level of anti-virus application from VirusTotal and the engine-level agreement. The ground-truth labeling is outside the purview of this research, but for our malicious dataset, each sample's malware type is identified according to data provided by the Malware- bytes engine in VirusTotal. Based on VirusTotal data, the age of our harmful samples is in the range of April 2020 and June 2021. Table 2 lists the different malware categories and the number of samples for each category.

The benign samples come from a total of eight sources. We downloaded the APIMDS dataset after installing a new copy of Windows 10 and extracting from c:windowssystem32 directory's, the Windows executables and DLL files. (1) In order to test legal downloading, we used free websites. (2) From the file Hippo website, we downloaded the top 43 programs and the top 300 portable Windows apps(3) We extracted the two folders, CYGWIN and WINDOWS 10 benign samples, from the benign dataset from downloaded files. Windows executable files are included in both directories and were copied from the required author sources. Using VirusTotal, each and every benign sample from the eight sources has been confirmed. Table 1 lists the no of trials from each safe source.

5.1. Method Evaluation

In this part, we assess the effectiveness of our suggested approaches for distinguishing malware from harmless samples and then categorizing them into the appropriate classifications. For training and testing, the benign dataset is divided into 4207 (80 percent) and 1052 (20 percent) samples, respectively. In addition to this, the training and testing datasets are separated for each malware-type dataset where each sample set contains 80% and 20% of the samples, respectively, to ensure a fair evaluation. As a result, we have a total of 1056 malware samples for testing and 4225 malware samples for training purposes.

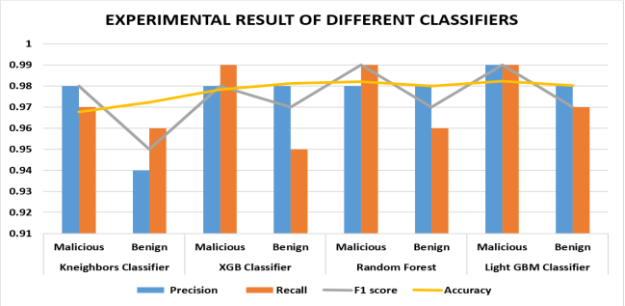

Fig. 2. Experimental Result of different Classifiers

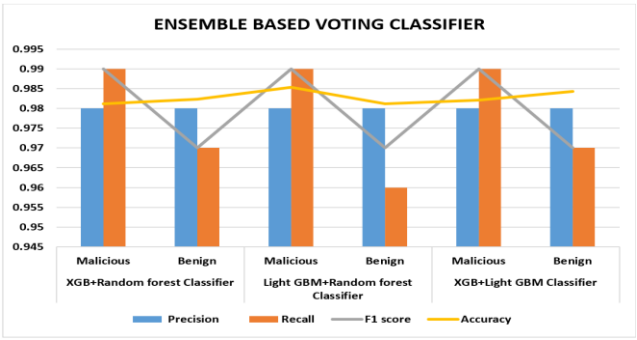

Fig. 3. Ensemble-Based Voting Classifier

5.2. Detection of Malicious Behavior

In this series of studies, we test how well Methods 1 and 2 distinguish harmful samples from benign ones. To achieve that, in method 1, On the malicious and benign training datasets (4225 and 4207, respectively), we have trained our models using four distinct machine-learning approaches as mentioned above in Table 2. Using the aforementioned machine-learning techniques, we perform 10-fold cross-validation on the datasets to avoid overfitting where the total dataset was divided into 10 parts and the same model was run ten times for the same dataset with a different set of test sets. The 1056 and 1052 harmful and benign testing datasets, respectively, are tested using the models. However, in method 2 we have used three different ensemble techniques on the above-mentioned datasets. The different ensemble techniques used were (XGB+ Random Forest Classifier), Light GBM+ Random Forest Classifier and XGB+ Light GBM Classifier. In each of the cases, the accuracy and other statistical criteria are analyzed. The effectiveness of the suggested techniques is assessed using the conventional machine learning performance criteria listed below:

- TP (True Positive): It is the % of datasets that are actually positive and also predicted as positive.
- FP (False Positives): These are the % of data samples that are actually negative but wrongly predicted as a positive sample.
- TN (True Negatives): % of samples that were expected to be negative and turned out to be negative
- FN (False Negatives): % of samples that were expected to be positive but turned out to be negative.
- Recall: The proportion of positive results that were really expected to be positive, or the TP rate (also known as sensitivity)
- Precision: The percentage of favorable predictions that actually materialize
- Accuracy: The ratio of samples accurately predicted (TP+NP) to all samples collected for testing (TP + TN + FP + FN)
- F-Measure: It is the measure of harmonic mean Recall and precision.

Here we have considered two sets of malware, one as malicious and the other as benign. Hence positive refers to actual malicious or benign malware type and negative refers to not a malware type sample.

TABLE 2: EXPERIMENTAL RESULT OF DIFFERENT CLASSIFIERS

Classifiers	Dataset	Precision	Recall	F1 score	Accuracy
Kneighbors Classifier	Malicious	0.98	0.97	0.98	0.967742
	Benign	0.94	0.96	0.95	0.972231
XGB Classifier	Malicious	0.98	0.99	0.98	0.978178
	Benign	0.98	0.95	0.97	**0.981105**
Random Forest	Malicious	0.98	0.99	0.99	0.981973
	Benign	0.98	0.96	0.97	0.979932
Light GBM Classifier	Malicious	0.99	0.99	0.99	**0.982314**
	Benign	0.98	0.97	0.97	0.980115

Performance results for Methods 1 and 2 are shown in Table 2 and Table 3 respectively. According to Table 2, utilizing Method 1, XGBoost outperforms the other three machine learning algorithms in terms of accuracy for benign datasets with scoring 98.1105 and Light GBM outperforms the other three machine learning algorithms in terms of accuracy for Malicious datasets scoring 98.2314. The detail of method 1 is shown in figure 2. According to Table 3, utilizing Method 2, XGB+ Light GBM Classifier outperforms the other three machine learning algorithms in terms of accuracy for benign datasets scoring 98.4325, and Light GBM+ Random Forest Classifier outperforms the other three machine learning algorithms in terms of accuracy for Malicious datasets with scoring 98.5312. Method 2 scoring 98.4325. As a result, we carry out more Light GBM+ Random forest trials employing 10-fold cross-validation. The evaluation details are shown in figure 3. The performance outcomes of Methods 1 and 2 are then assessed in terms of various parameters.

TABLE 3: RESULT OF ENSEMBLE-BASED VOTING CLASSIFIER

Classifiers	Dataset	Precision	Recall	F1 score	Accuracy
XGB+ Random forest Classifier	Malicious	0.98	0.99	0.99	0.981205
	Benign	0.98	0.97	0.97	0.982305
Light GBM+ Random forest Classifier	Malicious	0.98	0.99	0.99	**0.985312**
	Benign	0.98	0.96	0.97	0.981201
XGB+ Light GBM Classifier	Malicious	0.98	0.99	0.99	0.982114
	Benign	0.98	0.97	0.97	**0.984325**

5.3. Performance of the Methods and Misclassifications

In this part, we computed the performance of Methods 1 and 2 and explain when Method 2 might perform better than Method 1. As previously said, all approaches accomplish the same goal; the tokenization methods are

what really set them apart from one another. While Method 2 regards each AP call's argument as a unique feature, Method 1 interprets the complete collection of arguments for each API call as a single token. If the sample has called a few API calls but many arguments have been provided for each API call, Method 2 performs better than Method 1 since the many arguments for each call can make up for the few total API calls. Table 3 demonstrates that this finding is correct. It is evident that Method 2 outperforms Method 1 marginally in terms of malware detection. The causes of the misclassifications in our suggested techniques are then discussed.

5.4. Classification of Malware Types

The purpose of this collection of experiments is to assess how well Methods 1 and 2 perform in categorizing malware samples into their appropriate classes. The malicious samples are divided into their categories using the same features that were used to divide the samples into harmful and benign classifications. Our harmful samples fit into one of ten malware categories, as was already explained (Table 1). Each malware-type dataset is divided into training and testing datasets, which each include 80% and 20% of the samples, respectively, in order to provide a thorough validation and ensure that the machine learning modules are taught using a suitable number of samples from each sort. As a result, we have 2108 samples for testing and 8432 pieces of malware and benign code for training. The same five machine-learning techniques are employed to train models using the training dataset. Tables 2 and 3 respectively present the performance results for Methods 1 and 2. Utilizing Method 1, XGBoost outperforms the other three machine learning algorithms in terms of accuracy for benign datasets with a score of 98.1105, and Light GBM outperforms the other three machine learning algorithms in terms of accuracy for malicious datasets with a score of 98.2314, as shown in Table 2. Table 3 shows that using Method 2, the XGB+ Light GBM Classifier outperforms the other three machine learning algorithms in terms of accuracy for benign datasets with a score of 98.4325, and the Light GBM+ Random Forest Classifier outperforms the other three machine learning algorithms in terms of accuracy for malicious datasets with a score of 98.5312. The Score of 98.4325 for method 2. We do more Light GBM+ Random forest experiments using 10-fold cross-validation as a result. Next, numerous factors are responsible for giving different kind of accuracy of the models that were used in Methods 1 and 2.

(2) All of the study's parameters are vulnerable to slope failure, therefore determining slope stability using a single metric is useless. The variable δ is perhaps the most profound aspect to MLR model and MLP models, while slope geometry attributes are also critical. It should also be highlighted that neither of the supervised learning techniques is suitable for all kinds of slope scenarios, and none was sufficient to address the existing problem.

6. State-of-the-art of the different models

Here in this section, we contrast our strategies with those of previous works that take API parameters into account. Our comparison takes into account (i) Detection accuracy (ii) Necessary API data, including determining and the limitations include the frequency counter for a specific API request, recognizing API sequence trends, and more. The API parameters have been utilized in the research listed below to create malware detection and/or type categorization models. Both [10] and [11] employ pattern recognition algorithms to identify a shared sequence of API calls and parameters, as was mentioned in Section 2. However, by removing and/or introducing certain API calls. A pattern may be changed. In contrast, [1, 5, 6] employ malware detection methods based on the frequency of API calls. In [24], the distinction between benign and malicious samples was made using the frequency metric of calling particular API calls and their parameters. For malware identification, Yong et al. employed frequent item sets of API calls and their parameters in [14]. Statistics pertaining about and their parameters include the frequency, mean, and size of parameter arguments that were proposed by Hasan, H et al. [21] utilized to identify harmful software activity. By the removal of and/or adding API calls and by changing the frequency counter values, malware developers can easily get around the aforementioned frequency-based techniques.

Compared to the previous research, our methodologies are distinct. i) As a result of the fact that we don't rely on the sequence or pattern of the API calls, nor do we consider their individual methods are resistant to malware mutation and obfuscation tactics (such as changing the order of API calls or repeatedly using certain API calls and/or arguments). Instead, our approaches solely take into account the frequency of API calls and the values of such requests. (ii) Our method does not consider statistical traits like mean, frequency, or the size of the API parameters. (iii) Because our approaches employ unique feature generation functions to improve the retrieved API-based characteristics for improved processing, domain knowledge of the complicated arguments is not necessary. (iv) None of the current methods have investigated the potential for using each API call's parameter element independently, as demonstrated in Method 2.

These benefits enable our method to overcome the scaling challenge posed by the high memory consumption and computational complexity associated with the use of high dimensional feature space. Table 3 provides a comparison of our strategy with the comparable research stated previously. As seen in Tables 2 and 3, our suggested approaches have outperformed the most recent methods.

7. Conclusion and Future Work

This research presented a novel classification-based data mining method for identifying malware behavior. First, our proposed application is used to transform a malware behavior executive history XML file into a non-sparse matrix. The WEKA input data set was then translated from this matrix. We used the WEKA tool to apply the suggested procedures to an actual case study data set to demonstrate performance effectiveness. We have performed two operations in method 1 and method 2 on the same data sets. Classification techniques including K- neighbor, Random Forest, and Light GBM algorithms in method 1 and also a few ensemble techniques were used in method 2 for the same datasets. For classifying malware detection, the regression classification approach performed best. Additionally, we used the ensemble classification approach to examine the same data set. The evaluation's findings showed how useful the suggested data mining and ensemble method were more effective in finding malware. With reference to figure 2 and figure 3 and by paying attention to the experimental findings, classifying behavioral characteristics of malware can be an easy way to create behavioral antivirus. A genuine behavioral antiviral platform based on categorization via an ensemble algorithm will be developed and examined in the next work.

All authors have equally contributed to this research work.

Conflicts of interest

The authors declare no conflicts of interest.

References

[1] Kumar, R., Alenezi, M., Ansari, M. T. J., Gupta, B. K., Agrawal, A., & Khan, R. A. , "Evaluating the impact of malware analysis techniques for securing web applications through a decision-making framework under fuzzy environment". Int. J. Intell. Eng. Syst, 13(6), 94-109, 2020

[2] Balaji, K. M., & Subbulakshmi, T., "Malware Analysis Using Classification and Clustering Algorithms", International Journal of e-Collaboration (IJeC),18(1), 1-26,2022

[3] Akhtar, M. S., & Feng, T., "Malware Analysis and Detection Using Machine Learning Algorithms", Symmetry, 14(11), 2304, 2022.

[4] Hadiprakoso, R. B., Kabetta, H., & Buana, I. K. S, "Hybrid-based malware analysis for effective and efficiency android malware detection". In 2020 International Conference on Informatics, Multimedia, Cyber and Information System (ICIMCIS), (pp. 8-12). IEEE,2020.

[5] Hwang, C., Hwang, J., Kwak, J., & Lee, T., "Platform-independent malware analysis applicable to windows and Linux environments", Electronics, 9(5), 793,2020.

[6] Bermejo Higuera, J., Abad Aramburu, C., Bermejo Higuera, J. R., Sicilia Urban, M. A., & Sicilia Montalvo, J. A., " Systematic approach to malware analysis (SAMA)", Applied Sciences, 10(4), 1360,2020.

[7] Mehtab, A., Shahid, W. B., Yaqoob, T., Amjad, M. F., Abbas, H., Afzal, H., & Saqib, M. N., "AdDroid: rule-based machine learning framework for android malware analysis. Mobile Networks and Applications", 25(1), 180-192,2020.

[8] Akhtar, M. S., & Feng, T., "Malware Analysis and Detection Using Machine Learning Algorithms", Symmetry, 14(11), 2304,2022.

[9] S Aboaoja, F. A., Zainal, A., Ghaleb, F. A., Al-rimy, B. A. S., Eisa, T.A. E., & Elnour, A. A. H., "Malware Detection Issues, Challenges, and Future Directions: A Survey", Applied Sciences, 12(17), 8482,2022.

[10] Smith, M. R., Johnson, N. T., Ingram, J. B., Carbajal, A. J., Haus, B. I., Domschot, E., .& Kegelmeyer, W. P, "Mind the gap: On bridging the semantic gap between machine learning and malware analysis", In Proceedings of the 13th ACM Workshop on Artificial Intelligence and Security, (pp. 49-60),2020.

[11] de Vicente Mohino, J. J., Bermejo-Higuera, J., Bermejo Higuera, J. R., Sicilia, J. A., Sánchez Rubio, M., & Martínez Herraiz, J. J. "MMALE a methodology for malware analysis in linux environments",2021.

[12] Pereberina, A., Kostyushko, A., & Tormasov, A., "An approach to dynamic malware analysis based on system and application code split", Journal of Computer Virology and Hacking Techniques,1-11,2022.

[13] Almomani, I., Ahmed, M., & El-Shafai, W., "Android malware analysis in a nutshell", PloS one,17(7), e0270647,2022.

[14] McDole, A., Gupta, M., Abdelsalam, M., Mittal, S., Alazab, M., "Deep Learning Techniques for Behavioral Malware Analysis in Cloud IaaS", In: Stamp, M., Alazab, M., Shalaginov, A. (eds) Malware Analysis Using Artificial Intelligence and Deep Learning. Springer, Cham, (pp. 269-285), 2021

[15] Jeffrey, N., Tan, Q., & Villar, J. R., "Anomaly Detection of Security Threats to Cyber-Physical Systems: A Study", In International Workshop on Soft Computing Models in Industrial and Environmental Applications,(pp. 3-12). Springer, Cham,2023

[16] Amer, E., Zelinka, I., & El-Sappagh, S., "A multi-perspective malware detection approach through behavioral fusion of API call sequence", Computers & Security,110, 102449,2021

[17] Zou, D., Wu, Y., Yang, S., Chauhan, A., Yang, W., Zhong, J., ... & Jin, H., "IntDroid: Android malware detection based on API intimacy analysis", ACM Transactions on Software Engineering and Methodology (TOSEM), 30(3), 1-32,2021

[18] Schofield, M., Alicioglu, G., Binaco, R., Turner, P., Thatcher, C., Lam, A., & Sun, B, "Convolutional neural network for malware classification based on API call sequence", In Proceedings of the 8th International Conference on Artificial Intelligence and Applications, (AIAP 2021),2021

[19] Chaganti, R., Ravi, V., & Pham, T. D, "A multi-view feature fusion approach for effective malware classification using Deep Learning", Journal of Information Security and Applications, 72, 103402,2023

[20] Zhu, H. J., Gu, W., Wang, L. M., Xu, Z. C., & Sheng, V. S., "Android malware detection based on multi-head squeeze-and-excitation residual network", Expert Systems with Applications, 212, 118705,2023

[21] Hasan, H., Ladani, B. T., & Zamani, B., "MEGDroid: A model-driven event generation framework for dynamic android malware analysis", Information and Software Technology, 135, 106569,2021

[22] Sraw, J. S., & Kumar, K., "Using static and dynamic malware features to perform malware ascription", ECS Transactions, 107(1), 3187,2022.

[23] Thakur, D., Singh, J., Dhiman, G., Shabaz, M., & Gera, T., "Identifying major research areas and minor research themes of android malware analysis and detection field using LSA", Complexity,2021

[24] Al-Dwairi, M., Shatnawi, A. S., Al-Khaleel, O., & Al-Duwairi, B., "Ransomware-Resilient Self-Healing XML Documents. Future Internet", 14(4), 115,2022.

[25] Rafiq, H., Aslam, N., Ahmed, U., & Lin, J. C. W., "Mitigating Malicious Adversaries Evasion Attacks in Industrial Internet of Things", IEEE Transactions on Industrial Informatics, 2022

[26] Lebbie, M., Prabhu, S. R., & Agrawal, A. K., "Comparative Analysis of Dynamic Malware Analysis Tools. In Proceedings of the International Conference on Paradigms of Communication", Computing and Data Sciences, (pp. 359-368). Springer, Singapore,2022

[27] Kartel, A., Novikova, E., & Volosiuk, A., "Analysis of visualization techniques for malware detection", In 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus) (pp. 337-340), 2020

[28] Liu, S., Feng, P., Wang, S., Sun, K., & Cao, J., "Enhancing malware analysis sandboxes with emulated user behavior", 2022, Computers & Security, 115, 102613,2020

[29] Yadav, C. S., Singh, J., Yadav, A., Pattanayak, H. S., Kumar, R., Khan, A. A., ... & Alharby, S., "Malware Analysis in IoT & Android Systems with Defensive Mechanism", Electronics, 11(15), 2354,2022.

[30] Lebbie, M., Prabhu, S. R., & Agrawal, A. K., "Comparative Analysis of Dynamic Malware Analysis Tools", In Proceedings of the International Conference on Paradigms of Communication, Computing and Data Sciences, (pp. 359-368), Springer, Singapore,2022.

[31] Palša, J., Ádám, N., Hurtuk, J., Chovancová, E., Madoš, B., Chovanec, M., & Kocan, S., "MLMD—A Malware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm, Applied Sciences, 12(13), 6672,2022

[32] Louk, M. H. L., & Tama, B. A., "Tree-Based Classifier Ensembles for PE Malware Analysis: A Performance Revisit", Algorithms, 15(9), 332,2022

Biography

Mr. D Anil Kumar completed his M.Tech degree in Computer Science from Berhampur University Odisha in the year 2009 Currenly pursuing Ph.D from Berhampur University, Odisha. He has 18+ teaching experience worked as a different organizations as an Assistant Professor He has published no. of international journal and Conference papers. His research area is Cyber Security, Data Ware Housing and Data Mining Computer Organization and Architecture.

Dr. Susanta Kumar Das joined the Dept. of Computer Science in 1993. He has teaching experience of 23 years in the department. He has attended no. of national & international conferences. To his credit, he has served as H.O.D for 2 years in the department. At present he is the coordinator of M.Tech(S.F) course & as coordinator of spoken tutorial project conducted by IIT Bombay & funded by MHRD, Govt of India. Fourteen no. of scholars are awarded Ph.D under his guidance. One D.Sc degree is awarded in Computer Science under his guidance. He has been felicitated award of honour by Dept. of Mathematics, Maharshi Dayanand University Rohtak, Haryana in the international conference on History & Development of Mathematical Science & Symposium on Nonlinear Analysis. His research are in Software Engineering & Network Security. .