Journal of Trends and Challenges in Artificial Intelligence

ASPUR PUBLISHING

Vol. 01, No. 03 (2024) 113-122, doi: 10.61552/JAI.2024.03.004 - http://jai.aspur.rs

AN EFFICIENT MALWARE DETECTION MODEL THROUGH REDUCED HYBRID FEATURES AND ENSEMBLE MACHINE LEARNING

Om Prakash Samantray [1]
Siva Sai Geethika Penta
Satya Narayan Tripathy

Received 21.04.2024.
Revised 14.06.2024.
Accepted 29.07.2024.

Keywords:

Malware detection, static analysis, dynamic analysis, Malware API calls, Ensemble Learning, Feature Selection.

A B S T R A C T

Malware attacks are increasing in a higher rate due to extensive use of Internet & handheld devices and misuse of technological advancements. There are many malware detection techniques which use static or dynamic features for classification. This work focuses on creating static feature vector, dynamic feature vector, combining them to form a hybrid feature set and prepare it for better classification. Printable string information and API call sequences of malware and benign samples are used as two feature sets which are passed through feature selection algorithm for selecting best features. The reduced feature sets are combined to form a hybrid feature set. The hybrid feature set is passed through an ensemble model for classification. The ensemble model used in this article includes three supervised classifiers such as SVM, KNN and DT. This proposed model has performed well compared to the individual classifiers as well as individual feature sets. Besides better accuracy, this model has an execution time approximately equal to the individual classifiers which makes this model efficient for malware detection.

1. INTRODUCTION

Malicious software or malware is a program designed to harm security and confidentiality of computing systems. Malware enters into a system or a network to acquire control over the systems, steal sensitive data, overload the server to reduce performance, wasteful use of network bandwidth, and many more illicit intentions. Malware has different forms such as virus, worm, Trojan, adware, bot, rootkit, ransomware, key logger, spyware etc.A brief description of these variants is given in table 1.

1.1 Motivation

As per a study by AV-test organization (AV-test Report), 0.35 millions of malware variants and potentially unwanted Applications (PUA) are being generated every day. Figure 1 shows the progress rate of known and unknown malware (in millions)in the past ten years.

There are so many similar studies available which clearly speak about the exponential growth of malware variants in different sectors. The primary target-sectors are education, entertainment, finance, media, healthcare, public, individual and technology. These sectors are targeted to get sensitive information which creates financial opportunities for the cybercriminals.

Internet plays a vital role in the growth of malware. The easy availability of internet services and extensive use of online services for day-to-day business creates huge amount of information every day. According to a study by Accenture security (Accenture security, 2019), “68%

[1] Corresponding author: Om Prakash Samantray
Email: om.prakash02420@gmail.com

of business leaders feel their cybersecurity risks are increasing".

Table 1. Malware variants

Name of the malware	Description
Virus	Replicates it and connects with other programs. User unknowingly/accidentally executes the program to help the virus to spread.
Worm	Replicates itself and spread over network. Consumes bandwidth and attacks servers.
Trojan	Looks like legitimate file but malicious inside. User thinks it as valid file and installs it which leads to malware attack.
Adware	It comes with advertisements and pop-up windows. Attracts user to install it.
Bot	Self-propagating malicious program which usually attacks and controls a network.
Rootkit	Distantly access a computer secretly.
Ransomware	Attacks a computing system, blocks it and force the user to pay a ransom.
key logger	Records the keys pressed by user in a computing system.
Spyware	Secretly records user actions and key strokes in a computer.

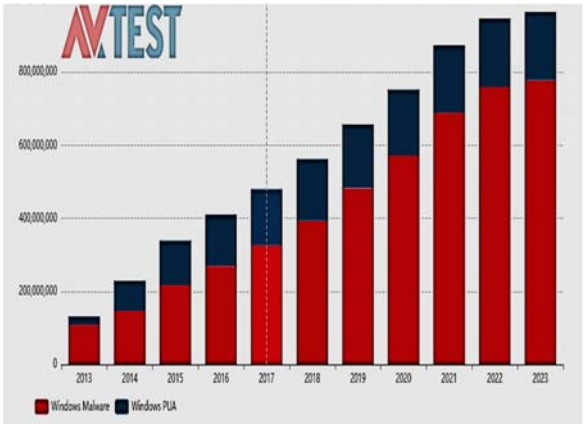


Figure 1. Statistics on Growth of malware by Av-Test (Src: AV-test Report)

There were around 36 billion data breaching incidents recorded in the first half of 2022 (Riskbased Security, 2020). Around 17% of the data breach happened due to malware (Verizon Business, 2020). Therefore, many organizations are focusing on securing their data from attacks. According to (Gartner Research, 2018), global information security market value is predicted to be170.4 billion dollars by the year 2023.

These studies reveal that, the malware attacks are increasing unprecedentedly and many of the organizations are looking for better security approaches to protect their sensitive data from attacks. These predictions motivate the researchers to work in the direction of malware detection systems.

Malware Analysis

Malware analysis is an important step in the process of malware detection. It is a study to know malware details like; origin of malware, type of malware, authorship, behavior & functionalities of malware and impacts of malware on a system or network. These information are later used as features useful in implementing machine learning models for malware detection. Malware analysis is of three types such as; static analysis, dynamic analysis and hybrid analysis.

Static analysis dissects the file under consideration without executing it. In this analysis, low-level information such as byte sequences, strings, opcodes, Control Flow Graphs (CFGs), System calls and Data-Flow Graphs (DFGs) can be extracted. Packed files need to be unpacked, encrypted files need to be decrypted and compressed files need to be decompressed before applying this analysis for better results. Many open source tools, freeware tools and command line utilities are available to perform this analysis. In static analysis, signature of the file is identified by finding out the cryptographic hash value of the file as well as knowing every section of the file. To do so, the file is loaded into a disassembler such as IDApro or OllyDbg which shows the executable code of the file in user understandable format. Then the user is required to analyze the code to understand the actual intention of the file.

In this method, the samples are executed in safe environments to analyze their behavior. This process is also known as behavioral analysis because during execution of the infected file, its behavior, system interaction and effect on the machine are monitored. This study is usually performed in isolated, safe and virtual environments. Many online and automated tools are available to perform this analysis. Prior to execution, the analysis systems are installed & activated with proper monitoring software such as Procmon, Capture BAT, PE-explorer, Hackerreplace, Wireshark, Regshot and so on. This method discloses original behavior of the samples which couldn't be identified in static analysis. Hence, it is considered as a better analysis method as compared to the static method. on the contrary, it increases the analysis overhead in terms of time and resource usage. Because, it needs a safe environmental setup which incurs some cost. The virtual environment used in dynamic analysis may not be same as the real system environment hence, sometimes malcode may act artificially which differs from the original behavior. Online automated tools used in dynamic analysis are CWSandbox, Norman Sandbox, TTAnalyzer, Ether, Anubis & ThreatExpert. The reports of dynamic analysis generated by the tools provide details of malware behavior and actions performed by them. Then the analysis system represents the report outcomes in an organized way which is later used for classification as feature vectors.

Hybrid analysis method is a blend of both static and dynamic methods used to deal with the disadvantages of both the techniques. First it finds out the static signature feature which is then combined with the dynamic behavioral characteristics for better representation of malware and benign samples for classification. The purpose of this analysis is to use the best features of both

static and dynamic analysis to represent a malware or benign sample more accurately and help the detection models to classify known as well as unknown samples more effectively.

1.2 Problem Statement

Most of the malware detection techniques use either static feature or dynamic feature for differentiating malware and benign files. These techniques can detect malware efficiently as long as the malware is a known type. The unknown malware or obfuscated malware may not be detected efficiently by these traditional signature based systems. Therefore, an efficient detection mechanism is required which can consider hybrid features of the samples for ensemble malware classification. The model needs to be optimized so as to increase detection accuracy with minimum execution time.

Contributions of this work are as follows,

- Collecting recent packed and unpacked malware and benign files from online and offline sources.
- Check the validity of these samples through VirusTotal service.
- Unpacking the packed files before performing analysis.
- Performing static analysis to collected printable string information and use static feature vector creation algorithm to represent the frequent and valid PSIs in the form of a binary vector.
- Performing dynamic analysis on the same samples via Cuckoo sandbox environment to get sequence of API calls. Further, use dynamic feature vector creation algorithm for extracting API n-grams of length 3 and 4 which are then represented as a binary vector.
- Performing feature reduction algorithm on individual vectors to reduce the dimension of the dataset.
- Merge the static and dynamic datasets to form a hybrid dataset and apply ensemble model on the hybrid dataset for classification.
- Analyse the detection accuracy and execution time with individual classifiers.

Next section talks about related works in the field of hybrid and ensemble malware detection and analysis systems. Section 3 includes the outline of the proposed system implemented in this work. Experiment results are discussed and analyzed in section 4. Section 5 concludes this article with a future scope of this work.

2. LITERATURE REVIEW

Malware detection techniques use either static features or dynamic features. The static features like strings, operation codes, API calls, Byte N-grams are usually collected using static analysis method. In this method, the samples are not executed but analyzed through disassembling and decompiling tools. The dynamic features like memory usage, register utility, instruction traces, API call traces, network traffic are usually collected using dynamic analysis method. In this method, the samples are executed in a safe and controlled environment to record the dynamic behavior of the samples. Machine learning algorithms play a vital role in solving problems related to classification and prediction. Machine learning algorithms can be implemented easily to achieve more accurate results which made the researchers to use them extensively on a wide range of tasks (Gibert et al., 2020).In this section, we present a few studies related to malware detection using simple machine learning methods and ensemble methods on different feature sets.

(Dai et al., 2019) proposed an ensemble method using features like API call sequences and hardware features such as memory dump images and hardware performance counters. Further they used neural network model to improve each feature for better detection accuracy. Lastly, they combined multiple classification algorithms to perform ensemble learning on the hybrid dataset. The detection accuracy of their model was 97.8%.

(Khasawneh et al., 2020) used ensemble method for improving performance of hybrid malware detection system with less cost. They used logistic Regression and Neural Networks as base classifiers in their experiment. As per their results, the model has reduced FPR value by more than 50% as compared to FPR values of individual models. Similarly, they observed a substantial reduction in detection overhead and detection time during online detection. As per their results, neural network performs better than logistic regression in terms of accuracy, detection overhead and detection time. The neural network based ensemble learning model has better accuracy, less overhead and less detection time as compared to logistic regression based ensemble model.

(Bawazeer et al., 2021) suggested a detection model using hardware feature like processor hardware performance counters collected through dynamic analysis. They used eight ML algorithms and two ensemble models in their experiment to evaluate accuracy, overhead and robustness of their proposal. They observed that, the ensemble model performs better with two HPCs as compared to eight HPCs.

(Ravula et al., 2013) used static and dynamic features extracted through reverse engineering process. They extracted features like API calls, DLLs, registry activity, file system activity and network activity for all the 1103 samples executed in safe environment. They divided the feature set into two datasets one containing 15 features and another containing 141 features. The feature setsare then passed through J48 Decision Tree and Naïve Bayes classifiers in Weka environment. The Naïve Bayes classifier has performed well with 15 selected features whereas decision tree has achieved better results with 141 features.

(Niranjan et al., 2018) proposed a Hybrid method comprising k Nearest Neighbor (kNN), Naïve Bayes and ID3 classifiers to achieve better detection accuracy. They

applied weight based feature selection method to select best features in the preprocessing phase. They proposed a feature selection algorithm which selected eight features out of 14 features which was then passed through the proposed classification model using 10-fold cross validation. The proposed method achieved better precision, recall, F1-score and accuracy as compared to existing models.

(Singh & Singh, 2020) extracted malware behavioral features using cuckoo sandbox. They used API calls, printable string information and other behavioral features to form a hybrid dataset. They included 16489 malware and 8422 benign files in their dataset. The accuracy score achieved by Adaboost ensemble model was 99.54%.

(Chen et al., 2020) used image and entropy features to train two different models. Then these models are combined into ensemble model to achieve better accuracy. (Yan et al., 2018) proposed a detection model called MalNet, using opcode sequences as the feature collected through IDApro tool. They used CNN and LSTM networks to train the model using features like grayscale image and opcode sequences. They performed their experiment on a dataset containing 20,650 benign files and 21,736 malware samples of nine malware families. The accuracy, TPR and FPR achieved by their model are 99.88%, 99.14% and 0.1% respectively.

(Feng et al., 2018) used multiple dynamic behavior features for android malware detection. The dynamic behavioral features are extracted through dynamic analysis. They used chi-square method for feature selection. Experiment was done on two datasets using different ML algorithms and stacking method. Stacking method outperformed other classification algorithms for both data sets.

(Li & Li, 2020) used ensemble of deep neural networks for android malware classification. Experimental implementation was done on two datasets containing 26 different types of attacks. They stated that, ensemble methods are more robust if base classifiers are robust. The robustness of deep neural network can be increased with different adversarial training phase.

Another android malware detection strategy is presented by (Martín et al., 2019) which used a benchmark dataset published under a Creative Commons Attribution-Noncommercial-Share Alike 4.0 International License and was constructed using AndroPy Tool which is an automated framework for extracting static and dynamic features from android files. The dataset contains static and dynamic features of 22 thousand malicious and legitimate samples which were extracted through controlled dynamic and static analysis. Finally, they applied ensemble classifiers on the hybrid dataset for effective malware classification.

(Gupta & Rani, 2020) presented two approaches using ensemble method for large scale malware detection. They used weighted voting strategy and stacking methods separately to implement ensemble models. The experiment was done on a huge dataset containing 100200 malware samples and 98150 benign samples using Apache Spark framework. As per the experimental results, the weighted voting method achieved 99.5 percent accuracy which is better than the stacking approach. It is also observed that, stacking the best 3 classifiers achieves same result as that of stacking all the classifiers. Therefore, stacking only 3 best classifiers can provide better accuracy with less computational time.

(Vasan et al., 2020) presented an ensemble learning approach to detect IoT malware using semantic features. Experiment was done using optimized RNN and CNN algorithms on a dataset containing 21137 IoT-malware and benign samples. Verification of the proposed model was done on different system architectures to check the robustness of the cross-architecture property. Two hardware architectures like Raspberry Pi 4 and Core-i5 are considered to examine the architecture overhead. Results shown that, the proposed model has achieved an accuracy of 99.98% with a low detection time of 0.32 seconds.

(Azeez et al., 2021) used an ensemble approach using 1-dimensional and fully connected convolutional neural networks with five algorithms at the end-stage. The five machine learning algorithms are: naïve Bayes, decision tree, random forest, gradient boosting, and AdaBoosting. They performed the experiment on windows PE file features. The ensemble model with 7 neural networks with extra tree as the end-stage classifier has achieved better prediction accuracy close to 100%.

(Zhang et al., 2020) used dynamic analysis method to extract features related to virtual memory, hypervisor and hardware layers. Feature categorization was done based on three factors such as efficiency, system load and security. They applied AdaBoost method with voting strategy in cloud environment which resulted in 99.9% accuracy and less overhead.

(Roseline et al., 2019) presented a hybrid, stacked and multilayered ensemble methodwith vision-based analysis. They used two dimensional grayscale images of malware binaries available in malimg dataset and BIG 2015 dataset for experiment. Experimentations were assessed in two groups, one with serial flow ensemble model and other with stacked multilayered ensemble approach. The suggested ensemble approach performed well with a detection accuracy of 98.91% which was claimed to be better than many modern deep learning based malware detection models.

Based on the above studies, we decided to use an ensemble learning model by combining three top classifiers obtained in our previous studies (Samantray & Tripathy, 2020) & (Samantray & Tripathy, 2021) on malware detection. The PSI static feature and dynamic API call features are selected based on their frequency of use, efficiency and robustness in the literature.

3. METHODOLOGY

Malware and benignware samples collected are analyzed separately using static and dynamic analysis methods. Printable string Information (PSI) of the samples is considered as static feature and Application

Programming Interface (API) calls of the samples are considered as dynamic feature. Frequent PSIs are identified to create a static feature vector. On the other hand, n-grams(n=3& 4) of the collected API calls are determined to create a dynamic feature vector. Both the vectors are binary vectors containing binary 0's and binary 1's to indicate the absence and presence of the features in the sample files.

Feature vectors of all the collected samples altogether constitute static and dynamic datasets for further experiment.

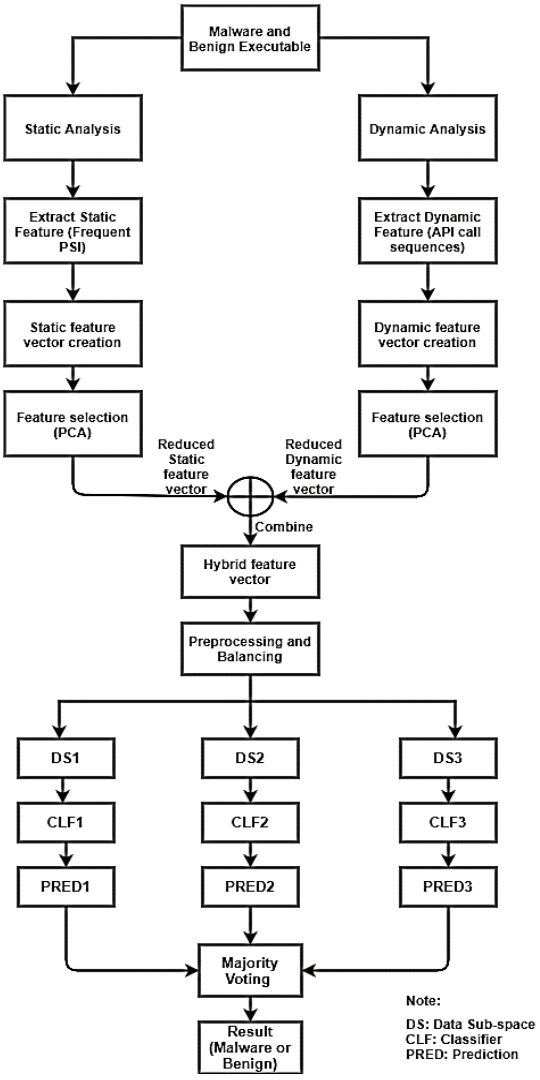


Figure 2. Ensemble model for Malware detection

Both the datasets are passed individually through Principle Component Analysis (PCA) method for dimensionality reduction (Wold et al., 1987). The number of components in PCA is decided by using a variance threshold of 90%. The reduced static dataset and dynamic dataset are combined together to create a hybrid dataset which is further sent for classification. In this model we have suggested the use of bagging ensemble model with three different classifiers such as; Support vector machine (SVM), K-nearest neighbor(KNN) and Decision Tree(DT). The dataset is divided into three subspaces using random sampling method. The three subspaces are given to the above three algorithms individually. Final prediction is done using majority voting methodology. The proposed system architecture is shown in figure 2.

4. EXPERIMENT & RESULTS DISCUSSION

Linux OS is used in this experiment to perform malware analysis. The "strings" utility of Linux is used to fetch string information from different file samples. Each input sample of both malware and benign types are analyzed using this Linux utility and the analysis results are stored in files. The procedure to use the strings utility in Linux is demonstrated in the figure 3. As depicted in this figure, the samples named malwaresample1.exe and malwaresample2.exe files are supplied to the string utility and the outputs are stored in malwaresample1.doc and malwaresample2.doc respectively.

```
~/OmPrakash-System$ strings malwaresample1.exe >> malwaresample1.doc
~/OmPrakash-System$ strings malwaresample2.exe >> malwaresample2.doc
~/OmPrakash-System$ strings --data malwaresample2.exe
SVWh@P@
WWWWh8P@
tOWh
jcPV
h P@
VWhhP@
hTP@
[htP@
F;t$
A;L$
VUj@
; uG
^]_[Y
VWjA3
Sleep
OutputDebugStringA
InterlockedExchange
GlobalFree
GlobalAlloc
GetSystemDirectoryA
GetWindowsDirectoryA
FreeLibrary
GetProcAddress
LoadLibraryA
GetSystemDefaultUILanguage
GlobalMemoryStatusEx
GetVersionExA
lstrcpyA
CloseHandle
```

Figure 3. String utility to collect strings of a sample and store them in a document.

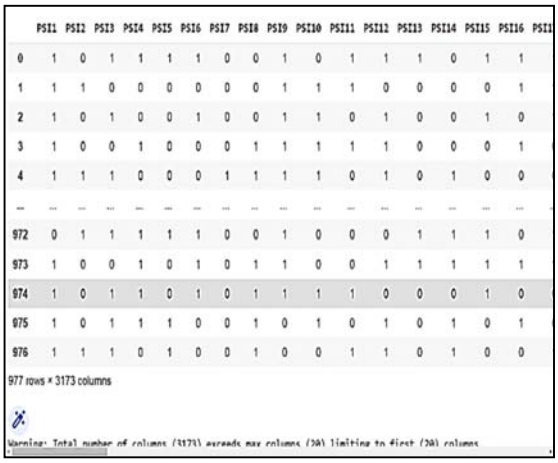
Total number of malware samples collected for this test is 562 and number of benignware samples is 415. These samples are collected from online sources like virus sign, virushare, malshare and other offline sources. The files are unpacked before sending for analysis. Static analysis is performed first on these samples to recognize PSI of size greater than 8 bytes for every sample and the same are documented. The valid strings are identified by

eliminating unnecessary strings which were placed in the data section to obfuscate the malware. Further, then the static feature vector creation (SFVC) algorithm implemented in (Shijo & Salim, 2015) is used to maintain a list of features whose frequency exceeds a particular threshold (40% of the count of samples). The SFVC algorithm is shown in algorithm 1. The files containing PSI information of the samples are given as input to the algorithm. The algorithm checks every file and identifies the frequent PSI based on the threshold value and the identified PSI are added to a list. The list is then compared against PSI of each file to create the binary vector. The vector contains binary 1's and 0's to indicate the existence and non-existence of features in each file.
The static dataset contains 3172 static features assigned with binary values. A portion of the static PSI dataset is shown in figure 4. The columns contain features PSI1 to PSI3172 to represent 3172 features. The first column denotes the sequence number of all sample files. The last column is the class label with binary values to represent malware and benign classes.

ALGORITHM-1 : Static feature vector creation (SFVC) algorithm

Input: sample Dataset (D)
Output: Static feature vector
Start
repeat for each fi in D
Collect and scan all strings (S) from each fi to identifyuseful strings (S)
repeat for each (Sj) for all j (j=no. of S in fi)
Calculate F (Sj). // F(S)= frequency/count of strings
if (F(Sj)>= N * 0.4) then// N=Number of samples
Include Sjin the SF-List. // SF-List = List of selected Static features
end of loop
end of loop
Build a matrix (M) with every String feature of the SF-List as columns;
repeat for each fi in D
repeat for each Sj in SF-List
if (Sj∈ fi) then
Set, M[i][j]:=1. //To represent presence of the string in the sample
else
Set, M[i][j]:=0. // To represent absence of the string in the sample
end of loop
end of loop
stop

On the other side, the dynamic analysis is performed on the same collection of samples to get API calls. This is done via Cuckoo sandbox environment in a virtual system to avoid the host system from being attacked accidentally during dynamic analysis. The result of the analysis was stored in the log file in which API calls of the samples were available. In order to get sequence of API calls, the dynamic feature vector creation (DFVC) algorithm implemented in (Shijo & Salim, 2015) is applied to extract n-grams (n = 3 and 4) of the API calls.



	PSI1	PSI2	PSI3	PSI4	PSI5	PSI6	PSI7	PSI8	PSI9	PSI10	PSI11	PSI12	PSI13	PSI14	PSI15	PSI16	PSI1
0	1	0	1	1	1	1	0	0	1	0	1	1	1	0	1	1	
1	1	1	0	0	0	0	0	0	1	1	1	0	0	0	0	1	
2	1	0	1	0	0	1	0	0	1	1	0	1	0	0	1	0	
3	1	0	0	1	0	0	0	1	1	1	1	1	0	0	0	1	
4	1	1	1	0	0	0	1	1	1	1	0	1	0	1	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
972	0	1	1	1	1	1	0	0	1	0	0	0	1	1	1	0	
973	1	0	0	1	0	1	0	1	1	0	0	1	1	1	1	1	
974	1	0	1	1	0	1	0	1	1	1	1	0	0	0	1	0	
975	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	
976	1	1	1	0	1	0	0	1	0	0	1	1	0	1	0	0	

977 rows × 3173 columns

Warning: Total number of columns (3173) exceeds max_columns (20) limiting to first (20) columns

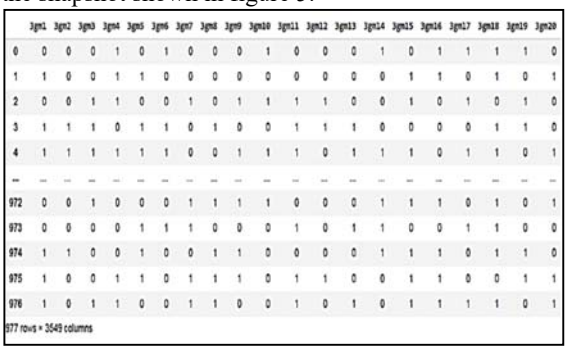
Figure 4. Static binary vector

Only 3-gram and 4-gram features are considered because these features work well in malware detection as per the previous works. The DFVC algorithm is given in Algorithm 2. This algorithm extracts the 3-grams and 4-grams of API calls from every sample. The frequent 3-grams and 4-grams are identified and added to the 3-gram feature list and 4-gram feature lists respectively. These feature lists are then compared with the list of 3-gram and 4-gram API calls extracted initially. A dynamic binary feature vector is created based on the presence (represented as 1) and absence (represented as 0) of these API call grams in all the samples.

ALGORITHM-2 : Dynamic feature vector creation (DFVC) algorithm

Input: Sample Dataset (D)
Output: Dynamic feature vector
Start
repeat for each fi in D
Generate log file and collect API3g and API4g call sequences.
// API3g= API call sequence of length 3 , API4g= API call sequence of length 4
repeat for each API3g and API4g
calculate F(API3g) and F(API4g) // F(API3g and API4g)= Frequency or count
if (F(API3g >=threshold) then
IncludeAPI3g to the DF-List. // DF-List = List of selected Dynamic features
if (F(API4g > = Threshold) then
Include API4g to the DF-List.
end of loop
end of loop
Create a matrix (M) with selected API3g and API4g as attributes;
repeat for each fi in D
repeat for each API3gand API4gin DF-List
if (selected API3g or API4gis available in the fi) then,
Set '1' as the value of the API3g or API4g in M.
Else
Set '0'as the value of the API3g or API4g in M.
end of loop
end of loop
stop

The result covered around 1642 number of 4-gram and 1906 number of 3-grams which are then represented as a feature vector. The frequent n-grams (which exceeds a specific threshold value) are selected from all the files and included in the dataset. The dynamic dataset containing 3-gram and 4-gram features of all the samples are represented as a binary dataset which looks similar to the snapshot shown in figure 5.



	3gm1	3gm2	3gm3	3gm4	3gm5	3gm6	3gm7	3gm8	3gm9	3gm10	3gm11	3gm12	3gm13	3gm14	3gm15	3gm16	3gm17	3gm18	3gm19	3gm20
0	0	0	0	1	0	1	0	0	0	1	0	0	0	1	0	1	1	1	1	0
1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1
2	0	0	1	1	0	0	1	0	1	1	1	1	0	0	1	0	1	0	1	0
3	1	1	1	0	1	1	0	1	0	0	1	1	1	0	0	0	0	1	1	0
4	1	1	1	1	1	1	0	0	1	1	1	0	1	1	1	0	1	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
972	0	0	1	0	0	0	1	1	1	1	0	0	0	1	1	1	0	1	0	1
973	0	0	0	0	1	1	1	0	0	0	1	0	1	1	0	0	1	1	0	0
974	1	1	0	0	1	0	0	1	1	0	0	0	0	1	1	1	0	1	1	0
975	1	0	0	1	1	0	1	1	1	0	1	1	0	0	1	1	0	0	1	1
976	1	0	1	1	0	0	1	1	0	0	1	0	1	0	1	1	1	1	0	1

977 rows × 3548 columns

Figure 5. Dynamic feature binary vector

The static dataset contains around 3172 features whereas the dynamic dataset contains a total of 3548 features of type 3-grams and 4-grams. Merging these two datasets resulted in a hybrid dataset but, the number of features are more which led to an over fitted and time-consuming model. For this reason, we applied Principle Component Analysis technique on the individual datasets shrink the dimension. The variance threshold in PCA is set to 90% to select the number of features for reduction. After applying PCA the feature sets are reduced by approximately 45%. The reduced static feature set contains 1957 number of features after applying PCA. The dynamic feature set contains 1335 number of 3-grams and 916 numbers of 4-grams after feature reduction using PCA. The PCA is implemented using Python inbuilt libraries. The reduced data sets are joined to generate a hybrid dataset which is then passed through the Synthetic Minority Oversampling Technique (SMOTE) for balancing the number of samples of both the classes. This technique identifies the instances which are closer in the feature space and draws a line between the instances in the feature space. Then pulls a new instance at a point on the line. Usually, an arbitrary instance from the smaller class is selected. Then k of the closest instances for that instance are identified (usually k=5). A random close-instance is chosen and an artificial example is formed at a randomly selected point between the 2 instances in feature space.

The selected ensemble model is applied individually on static feature set, dynamic feature set and then combination of these two sets. Linear SVM with regularization parameter value 0.1 is used as the first classifier. The parameter value and kernel selection is done based on the study done in the combined analytical model. The second classifier of this ensemble model is DT with Gini as the splitting criteria. The other parameters like max_depth, min_samples_split, min_samples_leaf are used with their default values. The max_features parameter value is set to "sqrt" which considers less number of features (approximately equal to square root of total number of features) during splitting. The combination of these parameter values gave us the best result as compared to other parameter values.

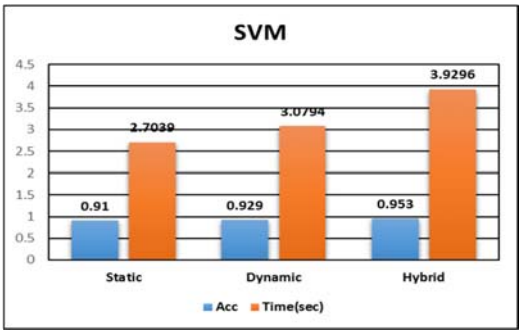


Figure 7(a). Accuracy and execution time of SVM for all three datasets

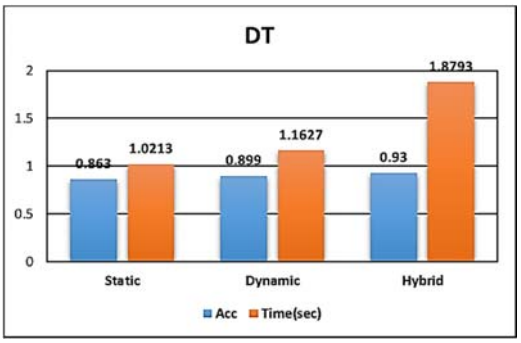


Figure 7(b). Accuracy and execution time of DT for all three datasets

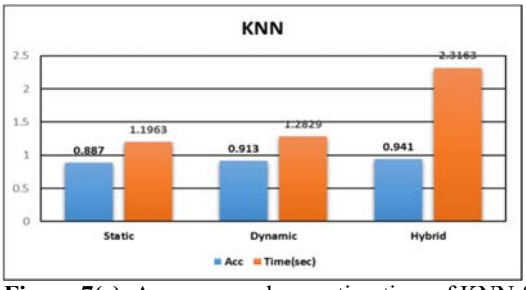


Figure 7(c). Accuracy and execution time of KNN for all three datasets

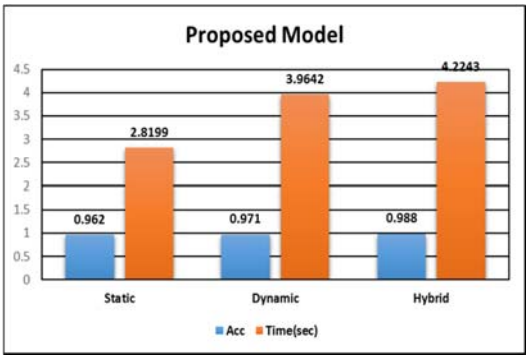


Figure 7(d). Accuracy and execution time of Proposed model for all three datasets

The third classifier is KNN which is applied with the default distance metric "Minkowski" and n_neighbors value 10. The algorithm parameter was set to "Auto" which chooses the best-fit algorithm to find the nearby neighbors. The parameter tuning of these three algorithms are done by rigorous experiment with different parameter values. The metrics used to evaluate this model are: FPR, Precision, Recall, Accuracy, Error rate and F-Score. Results of this model before applying PCA has achieved approximately same as that of combined analytical model but takes more training time. After applying PCA the performance is increased slightly with approximately equal training time. The values of the metrics for the three types of datasets are given in table 2.

Table 2. Result of Ensemble Model for Malware Detection

Data set Type / Metrics	FPR	Acc	Error Rate	Precision	Recall	F-Score
Static	0.053	0.962	0.037	0.961	0.973	0.966
Dynamic	0.041	0.971	0.028	0.97	0.981	0.975
Hybrid	0.02	0.988	0.011	0.985	0.994	0.989

The F-score value for static, dynamic and hybrid dataset are 0.966, 0.975 and 0.989 respectively. The accuracy score of this model for the hybrid dataset is 98.8% which is more than the accuracy scores of the other two types of datasets used in this experiment. The FPR of this model for hybrid dataset is 0.011 which is lower than the FPR of other two data sets. The comparison of Accuracy, F-score and Error rate for the three datasets is given in figure 6.

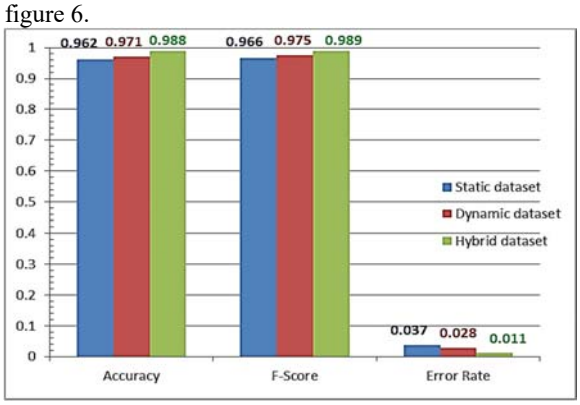


Figure 2. Result of Ensemble model for the three datasets

The accuracy scores (ACC) and execution times of individual algorithms and the proposed ensemble model are given in table 3.

Table 3: Results of individual classifiers and proposed Model for all the three datasets.

	SVM		DT		KNN		Proposed model	
Dataset	Acc	Time (sec)	Acc	Time (sec)	Acc	Time (sec)	Acc	Time (sec)
Static	0.91	2.70	0.86	1.02	0.88	1.19	0.96	2.81
Dynamic	0.92	3.07	0.89	1.16	0.91	1.28	0.97	3.96
Hybrid	0.95	3.92	0.93	1.87	0.94	2.31	0.98	4.22

Accuracy and execution time comparison of SVM, DT, KNN and the proposed model for all the three types of datasets is given in figures 7(a), 7(b), 7(c) and 7(d) respectively.

The execution time of Decision tree is less than the other algorithms used in this work. SVM has good detection accuracy than DT and KNN, but it is slower than these two methods.

Comparison of all the algorithms with the proposed model is shown in figure 8. The proposed ensemble model takes a little more execution time as compared to SVM algorithm. This extra execution time is very less which can be ignored looking at the benefit of the ensemble model in terms of higher detection accuracy.

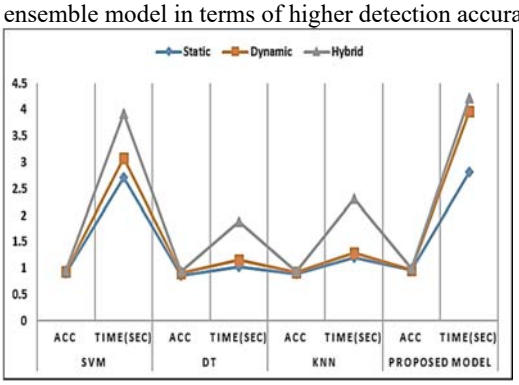


Figure 8. Comparison of all the algorithms with the proposed model

The combined use of multi-feature hybrid dataset, PCA and the ensemble model makes this detection model unique and efficient than the other detection models discussed in the literature.

5. CONCLUSION

In this work, static binary vector is prepared using PSI features from malware and benign samples. Dynamic binary vector is created using 3-gram and 4-gram API call sequences collected from the same set of samples used in static analysis. These two vectors are given to PCA model to reduce their dimensions. The reduced datasets are then merged to combine a hybrid feature vector that best describes the samples. The prepared hybrid dataset is passed through an ensemble model which uses SVM, DT and KNN algorithms. The ensemble model with reduced hybrid feature set has performed exceptionally well with an accuracy score of 98.8%. The dynamic dataset has better performance than static dataset in all cases in this experiment. The execution time of SVM is more than DT and KNN whereas the execution time of proposed model is slightly more than SVM algorithm. This little difference in execution time can be ignored looking at the higher detection accuracy provided by the proposed model. The proposed model also achieved higher detection accuracy when compared with most of the previous works discussed in this article.

Acknowledgement: The outstanding assistance of my co-authors was indispensable to the completion of this study and the research that supported it. I owe my co-authors a debt of gratitude for providing the utmost assistance in conducting this study. Many thanks to The Management of REC, Visakhapatnam for providing the infrastructure needed to do this study.

References:

Accenture security. (2019). Accenture security. Accenture Security. Retrieved April 26, 2023, from https://www.accenture.com/_acnmedia/PDF-96/Accenture-2019-Cost-of-Cybercrime-Study-Final.pdf

Azeez, N. A., Odufuwa, O. E., Misra, S., Oluranti, J., & Damaševičius, R. (2021). Windows PE Malware Detection Using Ensemble Learning. Informatics, 8(1), 10. DOI: 10.3390/informatics8010010

Bawazeer, O., Helmy, T., & Al-Hadhrami, S. (2021). Malware Detection Using Machine Learning Algorithms Based on Hardware Performance Counters: Analysis and Simulation. Journal of Physics Conference Series, 1962(1), 012010. DOI: 10.1088/1742-6596/1962/1/012010

Chen, L., Lv, H., Fan, K., Yang, H., Kuang, X., Xu, A., & Suo, S. (2020). An Ensemble Learning Approach to Detect Malwares Based on Static Information. In Lecture notes in computer science (pp. 676–686). DOI: 10.1007/978-3-030-60248-2_47

Dai, Y., Li, H., Qian, Y., Yang, R., & Zheng, M. (2019). SMASH: A Malware Detection Method Based on Multi-Feature Ensemble Learning. IEEE Access, 7, 112588–112597. DOI: 10.1109/access.2019.2934012

Feng, P., Ma, J., Sun, C., Xu, X., & Ma, Y. (2018). A Novel Dynamic Android Malware Detection System with Ensemble Learning. IEEE Access, 6, 30996–31011. DOI: 10.1109/access.2018.2844349

Gartner Research. (2018, September 14). Forecast Analysis: Information Security, Worldwide, 2Q18 Update. Gartner. Retrieved July 26, 2024, from https://www.gartner.com/en/documents/3889055

Gibert, D., Mateu, C., & Planes, J. (2020). The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, 153, 102526. DOI: 10.1016/j.jnca.2019.102526

Gupta, D., & Rani, R. (2020). Improving malware detection using big data and ensemble learning. Computers & Electrical Engineering, 86, 106729. DOI: 10.1016/j.compeleceng.2020.106729

Khasawneh, K. N., Ozsoy, M., Donovick, C., Abu-Ghazaleh, N., & Ponomarev, D. (2020). EnsembleHMD: Accurate Hardware Malware Detectors with Specialized Ensemble Classifiers. IEEE Transactions on Dependable and Secure Computing/IEEE Transactions on Dependable and Secure Computing, 17(3), 620–633. DOI: 10.1109/tdsc.2018.2801858

Li, D., & Li, Q. (2020). Adversarial Deep Ensemble: Evasion Attacks and Defenses for Malware Detection. IEEE Transactions on Information Forensics and Security, 15, 3886–3900. DOI: 10.1109/tifs.2020.3003571

Martín, A., Ldeeara-Cabrera, R., & Camacho, D. (2019). Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the Omni Droid dataset. Information Fusion, 52, 128–142. DOI: 10.1016/j.inffus.2018.12.006

Niranjan, A., Akshobhya, K. M., Shenoy, P. D., & Venugopal, K. R. (2018). EKNIS: Ensemble of KNN, Naïve Bayes Kernel and ID3 for Efficient Botnet Classification Using Stacking. DOI: 10.1109/icdse.2018.8527791

Ravula, R. R., Liszka, K. J., & Chan, C. C. (2013). Learning Attack Features from Static and Dynamic Analysis of Malware. In Communications in computer and information science (pp. 109–125). DOI: 10.1007/978-3-642-37186-8_7

Risk based Security. (2020). Risk Based Security. Q3 report: Data breach quick view. Retrieved April 26, 2023, from https://pages.riskbasedsecurity.com/hubfs/Reports/2020/2020%20Q3%20Data%20Breach%20QuickView%20Report.pdf

Roseline, S. A., Sasisri, A. D., Geetha, S., & Balasubramanian, C. (2019). Towards Efficient Malware Detection and Classification using Multilayered Random Forest Ensemble Technique. DOI: 10.1109/ccst.2019.8888406

Samantray, O. P., & Tripathy, S. N. (2020). A Knowledge-Domain Analyser for Malware Classification. 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA). DOI: 10.1109/iccsea49143.2020.9132916

Samantray, O. P., & Tripathy, S. N. (2021). An Opcode-Based Malware Detection Model Using Supervised Learning Algorithms. International Journal of Information Security and Privacy, 15(4), 18–30. DOI: 10.4018/ijisp.2021100102

Shijo, P., & Salim, A. (2015). Integrated Static and Dynamic Analysis for Malware Detection. Procedia Computer Science, 46, 804–811. DOI: 10.1016/j.procs.2015.02.149

Singh, J., & Singh, J. (2020). Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms. Information and Software Technology, 121, 106273. DOI: 10.1016/j.infsof.2020.106273

Vasan, D., Alazab, M., Venkatraman, S., Akram, J., & Qin, Z. (2020). MTHAEL: Cross-Architecture IoT Malware Detection Based on Neural Network Advanced Ensemble Learning. I.E.E.E. Transactions on Computers/IEEE Transactions on Computers, 69(11), 1654–1667. DOI: 10.1109/tc.2020.3015584

Verizon Business. (2020). 2020 Data Breach Investigations Report - Executive Summary. Retrieved July 26, 2024, from https://www.verizon.com/business/resources/T9a2/executivebriefs/2020-dbir-executive-brief.pdf

Wold, S., Esbensen, K., & Geladi, P. (1987). Principal component analysis. Chemometrics and Intelligent Laboratory Systems, 2(1–3), 37–52. DOI: 10.1016/0169-7439(87)80084-9

Yan, J., Qi, Y., & Rao, Q. (2018). Detecting Malware with an Ensemble Method Based on Deep Neural Network. Security and Communication Networks, 2018, 1–16. DOI: 10.1155/2018/7247095

Zhang, J., Gao, C., Gong, L., Gu, Z., Man, D., Yang, W., & Li, W. (2020). Malware Detection Based on Multi-level and Dynamic Multi-feature Using Ensemble Learning at Hypervisor. *Journal on Special Topics in Mobile Networks and Applications/Mobile Networks and Applications*, *26*(4), 1668–1685. DOI: 10.1007/s11036-019-01503-4

Om Prakash Samantray
Raghu Engineering College,
Visakhapatnam,
India.
om.prakash02420@gmail.com
ORCID: 0000-0003-1422-7014

Siva Sai Geethika Penta
MVGR College of Engineering,
Vizianagaram, India
Geethikasatya1216@gmail.com
ORCID: 0009-0001-7194-5971

Satya Narayan Tripathy
Berhampur University
Odisha, India.
snt.cs@gmail.com
ORCID: 0000-0002-6005-687X