

# Malware Detection Techniques and it's Classification: A Survey

<sup>1</sup>Mahendra Deore, <sup>2</sup>U V Kulkarni

<sup>1</sup>*Dept of Computer Engineering, Cummins College of Engineering for Women, Pune-41152.*

<sup>2</sup>*Dept of Computer Science & Engineering, SGGS Institute of Engineering & Technology, Nanded-431066*

**Abstract**-A recent cyber security industry report says that nearly 20% of enterprise computers are hosting some type of malware. It has become the need of time to mitigate malware. With improvisation in malware design, mutation characteristics like polymorphism and metamorphism are introduced, because of which there is enormous growth in the different malware sample variants. Antivirus software prevents damage from known malware families, having signatures. Different malware classes might have different actions associated to remove or prevent the malware. Hence, family or class of the malware should be correctly identified. However, the new malware variants do not have recognized signatures. Thus, we need to classify it based on some features. Machine learning, which is an ever growing field in computer science, can be effectively used here. Based on the behavioral patterns and previously observed traits in any malicious code, similarities can be identified by means of machine learning algorithms. Malwares can be categorized into classes by different classification algorithms. To classify malware, there are different types of analyzing techniques like static, dynamic and hybrid analysis approaches. Work related to all three approaches is surveyed thoroughly in the paper. In the end, we have our proposed system where we intend to classify malware into their respective classes using feature extraction and classification algorithms like Random forest, Support vector machine, XGBoost.

**Keyword**-*Malware; Classification; Static analysis; Dynamic analysis; Machine learning; Obfuscation; Hybrid analysis.*

## I. INTRODUCTION

Cyber threat increases every year. There was a security incident in the past year reported by some respondents to the 2015 survey of US State of cybercrime [1] by PwC. In 2015, more than 700 million data records were jeopardized as reported by Gemalto [2]. According to the Crime survey of 2011 by PwC [3], cyber crime has risen eventually to become a major threat.

A malware is software that disrupts the computer or different operations. It can gather sensitive data from the computer. Gaining access to private computers or displaying unwanted advertisements is another possible functionality of malwares. Intentions of malware can be stealing or spying on computer users without them knowing. It might also be designed to sabotage payments. Malware can be disguised as a benign file or might be embedded in non-malicious programs.

A recent example of a malware attack experienced worldwide is the *WannaCry crypto worm* ransomware attack in May 2017. WannaCry spread rapidly across a number of computer networks over 150 countries worldwide according to [4]. Microsoft Windows was the operating system that was targeted and data encryption was done followed by demanding ransom payments using Bit coin.

Enormous amounts of malware variants are generated daily because of advancements in malware creation techniques. Techniques like environmental awareness can detect underlying environment and execute only if the environment is some real environment and not a virtual one. So, it might be able to bypass cuckoo sandbox. Some malwares execute only when the system clock timing matches a certain time. The timing is hardcoded in the executable. Some malwares bypass the malware blacklists of common antivirus software's [5]. Other malwares are created by techniques like obfuscation which hides implementation details of the whole malware file by obfuscating. Polymorphism and metamorphism create malwares in bulk.

To eradicate the threat from malware, an analysis is required to understand the behavior of such malicious files and how this behavior differs from the normal or benign files. Malware detection is a technique in which, based on the behavioral differences, files with malicious intent can be identified. Since traditional signature-based malware detection approaches [6] fail to detect packed or obfuscated malware, features of the malware samples collected using behavior analysis are used. Also, because of the similarities in the behavior or attacking patterns of certain malware, they are grouped into families. Classification is a technique by which malware samples can be classified to their respective family for easier analysis. Some of these families are Kelihos [7], Ramnit [8], Tracur [9], Vundo [10], Lollipop [11], Gatak [12], etc. Malware classification can be using various machine learning algorithms.

## II. LITERATURE SURVEY

Malware samples can be analyzed and classified using a *signature-based* approach or *anomaly-based* approach. Either of these approaches can be implemented to analyze malware using Dynamic, Static or Hybrid analysis.

In *dynamic analysis*, we analyze the file that contains malware by executing it in a real or virtual environment and identifying the patterns and the way the malware has affected the system. Mamoun Alazab et al

[13], malware is detected by the dynamic analysis of the samples where they have considered the frequency of API calls as their feature. The proposed system unpacks malware and disassembles the binary executable so that it can retrieve the assembly code. The assembly program is further used to pull out the *API calls* from the code and also some relevant machine code is extracted. These features are used in combination. Finally, they have mapped API calls with MSDN library to analyze the malicious code. Based on the features they have used Similarity-based detection methods for identifying unknown malware and classifying them into their respective families. The classification is carried out by utilizing 8 robust classifiers viz. Naive Bayes Algorithm, Sequential Minimal Optimization Algorithm with 4 different kernels: SMO- PolyKernel, SMO - Normalized PolyKernel, SMO – Puk, and SMO- Radial Basis Function, k-Nearest Neighbor Algorithm, Backpropagation Neural Networks Algorithm and J48 decision tree. Their experimental result has attained the accuracy of 98.5%. The approach detects *zero-day malware* attacks [14] successfully.

However, another approach using API calls in [15] uses extraction of API call features from executables and applying pattern recognition to differentiate between malicious and benign files by using an automated tool running in a virtual environment. Their detection is based on detection based on behavioral feature analysis. They have extracted both malware and clean ware binary files to classify into malware and clean ware and further classification of malware into its families. The 4 classification algorithms namely SVM, Random forest, Decision table and IBI were used for classification. The methodology proposed for distinguishing malware from clean ware using a 2-class classification model had an accuracy of 97.3% and their malware family classification model attained the accuracy of 97.4%. This method provides an effective defense against zero-day attacks.

But the major disadvantage of dynamic analysis is that it increases overhead because of execution time for every data sample. Thus, with an increase in the size of the data set there is decrease in efficiency in terms of time complexity. This can be overcome using static analysis. Static analysis is a type of analysis done by examining the executable file without actually having to execute it. Visual inspection is used to scrutinize the code eliminating the need to execute it.

In [16], Deguang Kong et al. use *static analysis* for classification of malware. During their training phase, to obtain the structural information from a malware program they disassembled the code to extract the function call graph as a feature. The more emphasis was given on function call graph because it represents the calling relationship among the functions, gives an overall idea about the structure of malware program. Each vertex of function call graph represents a local function and for each one of them, they have extracted 6 attributes such as *Opcode*, API, memory,

IO, Register, and Flag. After the feature vector was created the automated classification was done. Automated classification of malware requires the computation of the distance between the malware instances. Based on the pair wise malware distance the ensemble of classifiers undergoes the learning process. They are trained on this basis so that they can classify the new malwares correctly. The classification algorithms were k nearest neighbor, Support Vector Machine and their system claims the accuracy to be improvised due to the use of the ensemble learning.

The system proposed by X. Hu et al. [17] is an approach to classify malware based on their static features. Before feature extraction, the data was pre-processed where they have reconstructed the PE headers. The features such as machine instructions and AV label were extracted and an aggregated feature vector was created using these. They have trained and optimized their model and have compared the classification algorithms such as a Weighted nearest neighbor, Logistic regression, and Support Vector Machine and Random forest. The best accuracy for their datasets was obtained by Random forest classifier. They have also used a combination of hashing kernel that helped in reducing the dimensionality of the feature vector which ensured scalability and accuracy. Their experimental results state an accuracy of 99.8% using five-fold cross-validation with a log loss (performance metrics measure) value of 0.0258.

Whereas [18] is a work by Kevadia Kaushal, which uses static analysis approach to detect metamorphic malware. It does not use the signature-based method for malware detection. Their proposed system extracts API calls from the executable and statically analyzes them. The frequency count of API calls is used to generate the feature vector. This method of API call's frequency count is called SAVE. In this method, the signature of malware is determined from the API call sequence. Each sequence is denoted by a vector. Their method includes 4 stages. In the first stage, the binary executables are disassembled and assembly code is retrieved. It is one with the help of *IDA Pro Disassembler* [19]. It also automatically recognizes API calls for various compilers. In the second stage, the features are extracted. In the third stage, the frequency is calculated. In the fourth stage, a similarity measurement is done for unknown binary executable. And with the help of similarity measurement, the malware is detected.

Hassan Takabi et al. [20] Proposed a *heuristic* method for malware detection. They used a unique structure in PE files i.e. *Dynamic-link library* dependency tree. Firstly dependency trees are extracted from benign as well as malicious PE files which are then converted into string encoding formats. After that with the help of closed frequent trees, feature vectors are constructed. And the classifier used is Random forest classifier.

Static analysis is a more thorough approach and is cost-efficient. However, a subtle flaw or vulnerability can get too complicated for static analysis alone to reveal which is why

it is often used in conjunction with dynamic analysis. This approach is called hybrid analysis.

Elhadi et al.[21] Proposed the *hybrid analysis* is used for the classification of malware. Firstly they've undergone dynamic analysis and then the static features are also examined. The file which is suspected of having malware is executed in the safe and controlled environment and then API calls are extracted using kernel hooking. If the file is packed then it is unpacked first. The static part includes construction of *call graphs* with the help of the API calls extracted and the resources of the OS used by these API calls. The nodes in the graph represent: 1. API calls 2. Operating system resource. This graph is decreased by removing some nodes. Only those API calls which are used by the majority of malware and others are removed which results in decreasing the sizes of the constructed graphs. Databases are created using nodes on the graph. And then with the help of similarity algorithms, two graphs are compared and malware is classified.

One more method of hybrid analysis for malware classification was given by Taegyu Kim et al. [22]. This system is divided into three parts i.e. Analyzer, Converter and Resource Manager. Converter does the task of unpacking, decompiling and structuring and converts the input binaries into SCFSs. These modified SCFSs are passed on to the analyzer. It is the analyzer which detects whether the code is malicious by measuring set similarities with the already existing samples of malware in databases. Similarity measurement is done with the help of SCFSs obtained from converter. The analyzer includes three components: malware databases, C2C matcher, and I-filter. All three of them work together in the detection of malware. VMs are used by converters and analyzers for performing their functions. Each VM does the conversion and analyzing. However, their workloads change situation-wise. It is the job of the resource manager to allocate resources.

### III. MALWARE CREATION TECHNIQUES

For generating malware, the attackers use different ways varying from writing a small piece of code to complex algorithms that adversely affect the machines to create various kinds of malware such as polymorphic, metamorphic malware, obfuscated malware. The malware that are created by using these basic techniques can be recognized easily by extracting basic features and characteristics. The major techniques that attackers use to create malware are polymorphism [23], metamorphism [24] and obfuscation [25].

In *polymorphic malware*, there is a malware code whose syntax mutates itself with each iteration but the semantic remains the same. Various common methods to create polymorphic malware are using encryption, data appending/ pretending but the limitation that the decrypted code essentially remains the same makes the polymorphic malware easy to detect using memory-based signature detection. Nur Syuhada Selamat et al. [26] Has given a

method to detect polymorphic malwares using based on files dropped by malware. They have used dynamic tool for behavior analysis.

Metamorphic malware is rewritten with each iteration. Each succeeding code version is different from the preceding one. This malware automatically codes them each time they are distributed or propagated. Methods to create metamorphic malware are by adding the varying length of NOP instructions, permuting the registers that are already in use, adding additional unnecessary instructions in the code, adding irrelevant loops that acts as dead piece of code and does nothing, reordering the functions that are written in the code, static data structure modification. The metamorphic malware was detected by automated code identification and analysis of memory snapshots. Duaa Ekhtoom et al. [27] Uses a compression based approach to identify the metamorphic malware on the basis of its similarity measurement with respect to the other malware variants taken from 13 different families.

Obfuscated malware includes the combination of both polymorphic and metamorphic malware technique. It is used to generate multiple variants of code, with indistinguishable functions. The obfuscated code has identical functions but different morphs (smallest constituent). Identifying obfuscated code is a very difficult task. Some of the techniques to generate obfuscations are appending garbage code (adding futile instructions), register renaming (replacing the register in use with the register which is not in use), subroutine reordering (change the order of subroutines), dead code insertion (adding some code which accomplish nothing), substitution of equivalent instruction (replacing xor eax, ebx with sub eax, ebx), code transportation (adding the unconditional jump statements in the program). Remote execution of these different types of malware is done by the attacker to achieve their intentions. Malware obfuscation has made it very difficult for the signature-based antivirus software's. In Scott Treadwell et al. [28] has proposed a heuristic based approach to detect the obfuscated malware. They have analyzed certain features such as section names, entry point location, import functions count, DLL characteristics, PE characteristics and various other features as well. They have generated the risk score by performing static analysis on Windows PE file. Based on the threshold value which has been decided and the risk score obtained the actions to be performed are justified.

### IV. MALWARE DETECTION AND CLASSIFICATION TECHNIQUES

The steps to be followed for malware detection are:

I. Analyze the executable to detect if it has malicious code present or not (feature extraction) II. If it has a malicious code, then assign most appropriate malware family it belongs to (classification mechanism)

#### 4.1 Malware detection

Malware are detected using two major techniques:

## 4.1.1 Signature-based detection

Signature-based detection is an old technology which used to be popular in the 1990s, and is extremely effective in identifying previously known threats. Signature of the malware is a code string or a pattern of actions corresponding to a known attack. There is a database maintained of such signatures against which network traffic and files (also known as blacklists) are checked to see if any of the known threats are present in them. An alert is issued if there are any threats present and the mitigation process is triggered. The problem with this detection which is based on knowledge entirely is that it cannot detect the malicious codes that do not have any signature like new malware variants created in bulk by advance techniques and the scale of this threat makes it harder to keep count and list of such signatures up to date. A signature-based detector uses distinct signatures for every malware variant, thus leading to an exponential increase in the size of the database of signatures.

Abhay Kumar Sahoo et al. [29] Proposes a system that detects malware present in unstructured data in the Hadoop Distributed file system. The system uses map reduces to detect malware in data stored in HDFS. Mila Dalla Preda et al. [30] Explains how signature based malware detection approach is syntactical and thus is not feasible for obfuscated malware. Thus, making semantic behavior based approach an alternative.

## 4.1.2 Anomaly-based detection

This detection uses two phases.

1) Training phase: During this phase, the detector learns the normal or usual program behavior. The detector might learn behavior of the PUI or the host or a combination of both.

2) Testing phase: Based on the learned behavior, the programs that violate the normal execution flow are considered to be anomalous and are identified as malicious.

It's advantageous to use anomaly-based approach because of its ability of detecting zero-day attacks i.e attacks that are not known by the malware detector previously. However, there can be a possibility, where a program can exhibit unseen behavior but, does not have any malicious intent, thus raising false positives. This technique was used by R. Sekar et al. [31] in which a Finite State Automata was created to detect anomaly. Program counter in the PUI is represented by nodes that are created in FSA. Transitions in the automata are given by system calls. Another approach by K. Wang et al. [32], a payload based (PAYL) anomaly detection approach is presented. The model computes byte distribution for normal traffic. The new packet's payload is checked for similarity with its corresponding model. Large distance from the normal model will mark the packet as an anomaly.

The above techniques can apply any one of the following three approaches.

Static analysis, In static analysis the actual execution of program is not required. Various information obtained using static methods are Opcode sequences (extracted from disassembling the binary file), byte code sequence n-grams, control flow graphs, syntactic library calls. Such feature sets are used in combination or individually for detecting malware.

Table.1: A summary of systems that employed static malware analysis

Sr no.	Author	Publishing year	Features extracted	Type	
				Detection	Classification
1	Deguang Kong et al. [16]	2013	Function call graph		✓
2	Nir Nissim et al. [33]	2014	Dll files, n-gram	✓	
3	Masoud Narouei et al. [34]	2015	Dll (dynamic link library) dependency tree extracted from PE files.	✓	
4	X. Hu et al.[17]	2016	Instruction sequences, strings, section information.		✓
5	Mehadi Hassen et al.[35]	2017	Opcode n-gram, control statement shingling		✓

*Dynamic analysis*, In the dynamic analysis, the behavior of the program is monitored based on analyzing the API calls to know how the program interacts with the operating system. In dynamic analysis, one is supposed to

execute the program most probably in a virtual environment. Information like registry changes, file system, memory writes, API calls, system calls, system change

Table.2: A summary of systems that employed dynamic malware analysis

Sr no.	Author	Publishing year	Features extracted	Type	
				Detection	Classification
1	M Shankarapani et al. [36]	2010	API calls		✓
2	Mamoun Alazab et al. [13]	2011	API calls		✓
3	Mohammad Imran et al.[37]	2015	System calls		✓
4	Andrii Shalaginov et al.[38]	2016	API calls, PE32, disk activity, memory footprints		✓
5	Oscar Somarriba et al.[39]	2017	DNS queries, URLs	✓	

Hybrid analysis, It is the combination of both static and dynamic analysis. It uses features from both the

methods and tries to identify the malicious code present in the system.

Table.3: A summary of systems that employed hybrid malware analysis

Sr no.	Author	Publishing year	Features extracted	Type	
				Detection	Classification
1	Mohamad Fadli et al. [40]	2011	Behaviour analysis, runtime analysis, resource monitoring.		✓
2	Taegyu Kim et al. [22]	2015	Binaries, I-filter, dynamic resource allocation.		✓
3	R. J. Mangialardo et al. [41]	2015	Number of sections, file size, Suspicious strings, API calls		✓
4	Ammar Ahmed E. Elhadi et al. [21]	2016	API call graph	✓	
5	Sean Kilgallon et al.[42]	2017	Byte features, PE import, string features, metadata, API system calls		✓

#### 4.2 Malware classification techniques

Classification is done using various machine learning algorithms [43] which takes the training data and based on that data it builds a classifier. Once the classifier is built the testing data is given to it for classification. Based on how correctly the testing data is classified the accuracy of the model is determined. Some of the major classification algorithms are given below

##### 4.2.1 Logistic Regression

Logistic regression [44] is a method based on statistics to analyze the data set. Outcome of this model is determined by one or more independent variables. Only two possible outcomes can be there i.e. is it a binary classifier. It is also called as log it regression or log it model. The output of logit model can take only two values, 0 and 1, which represents pass or fail respectively. The aim is finding a model which describes relationship between characteristics of interest and independent variables in the best way. The formula to predict logit transformation where,

$p$  - probability of presence of the required characteristic is given by:

$$\text{logit}(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + \dots + b_kX_k$$

The logged odds are given by:

$$\text{odds} = \frac{p}{1-p} = \frac{\text{probability of presence of characteristic}}{\text{probability of absence of characteristic}}$$

Thus,  $\text{logit}(p)$  is given by:

$$\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$$

#### 4.2.2 Naive Bayes

The Naive Bayes classification algorithm [45] is based on the Bayesian theorem. Bayes theorem states the relationship as below

$$P(y | x_1, \dots, x_n) = \frac{P(y)P(x_1, \dots, x_n | y)}{P(x_1, \dots, x_n)}$$

$$P(x_i | y, x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_n) = P(x_i | y),$$

This relationship can be simplified as

$$P(y | x_1, \dots, x_n) = \frac{P(y) \prod_{i=1}^n P(x_i | y)}{P(x_1, \dots, x_n)}$$

Thus, Naive Bayes classifier predicts probability which tells if the given record or data point belongs to some class for each class i.e. the probability that given record or data point belongs to a particular class. The class with maximum probability is the most likely class. Naive Bayes is called a naive approach because it assumes that all the features being used are unrelated to each other.

#### 4.2.3 Random Forest

Random forest employs ensemble learning for classification which uses decision trees. The decision trees are constructed during training phase. The modal class of the classes or regression of individual trees is returned as result. Random forest algorithm is like the bootstrapping algorithm with Decision tree model. The Random Forests approach uses the construction of multiple decision trees [46]. The only difference is that instead of using Gini index[47] and information gain[48] parameters, the root node selection is done randomly forming a forest of multiple decision trees. The more the number of trees in the forest, the more accurate are the results obtained.

#### 4.2.4 Support Vector Machine (SVM)

Support vector machine [49] is a type of supervised learning approach. In SVM, every data item is plotted in an n-dimensional space as a point. Every axis is labeled with a feature. Classification Fourth step, applying similarity measurement for unknown binary executable: is performed by the hyper plane that differentiates both the classes very well, the dimensions of this hyperplane being n-1 for n-dimensional space. There can be many such hyperplanes possible. The basic rule to select the best hyperplane is the one that has the maximum margin and best differentiates between the elements belonging to two different classes. For

a linear classifier identified by the set of pairs  $(w,b)$ , identifying  $w,b$  which satisfies the following equation will give the optimum hyperplane.

$$\begin{cases} \mathbf{w} \cdot \mathbf{x}_i + b \geq +1 & \text{if } y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b \leq -1 & \text{if } y_i = -1 \end{cases}$$

#### 4.2.5 k-Nearest Neighbor (kNN):

A lot of training data is fed to this algorithm in the learning phase where the kNN algorithm [50] plots the data samples according to their classes in an n-dimensional space. Whenever test data samples are given to the algorithm, the algorithm plots the samples in the same n-dimensional space and searches for its nearest k neighbors from the training samples based on distance measures like Euclidean distance given as

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^n (q_i - p_i)^2}.$$

In this algorithm the selection of k plays a very important role.

#### 4.2.6 XGBoost

XGBoost [51], i.e. Extreme Gradient Boosting is an algorithm which is used for supervised learning problems. It is based on tree ensemble learning using CART [52] (Classification and Regression Trees). Multiple trees are used here because a single tree might not always yield the best results. Every leaf node of this tree is one of the distinct classes, that the data is supposed to be classified into and has a prediction score (probability that the given data sample belongs to this class). Since, there are multiple such trees, sum of the prediction score of every tree is calculated and final score is assigned to every leaf node. This is same as random forest algorithm. However, the difference is that boosted trees are trained differently.

Gradient Boosting starts with a not very deep tree and will model the original target. After the first round of predictions, errors are found out and passed as a target to the second tree. The second tree will model errors from the first tree, find out new errors and pass those as a target to the third tree and the process will continue.

The tree ensemble model can be mathematically written as:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), f_k \in \mathcal{F}$$

Where,

$f$  - Functional space  $\mathcal{F}$  contains function  $f$

$\mathcal{F}$  - It is a set of all the possible classification and regression trees(CARTs)

$K$  - Number of trees.

## V. PROPOSED SYSTEM

Based on the study of the different classification methods and features, our proposed system uses static analysis technique to classify the malware samples. The dataset consisting of malware samples considered for classification is from the Kaggle BIG 2015 challenge [53].

In this approach, the emphasis is given to the effective selection of features so as to decrease the additional computational overhead. This makes it beneficial for handling huge amounts of data. Only the most relevant

features are considered. There are two stages. Firstly, the essential features like metadata, symbols, Opcode, registers, sections, entropy, data define and control flow graph are extracted and a feature vector is created..This feature vector is given to the classifier which uses xgboost, random forest and other machine learning algorithms to classify the samples in their 9 major malware families shown in fig1. We are currently working on building this system.

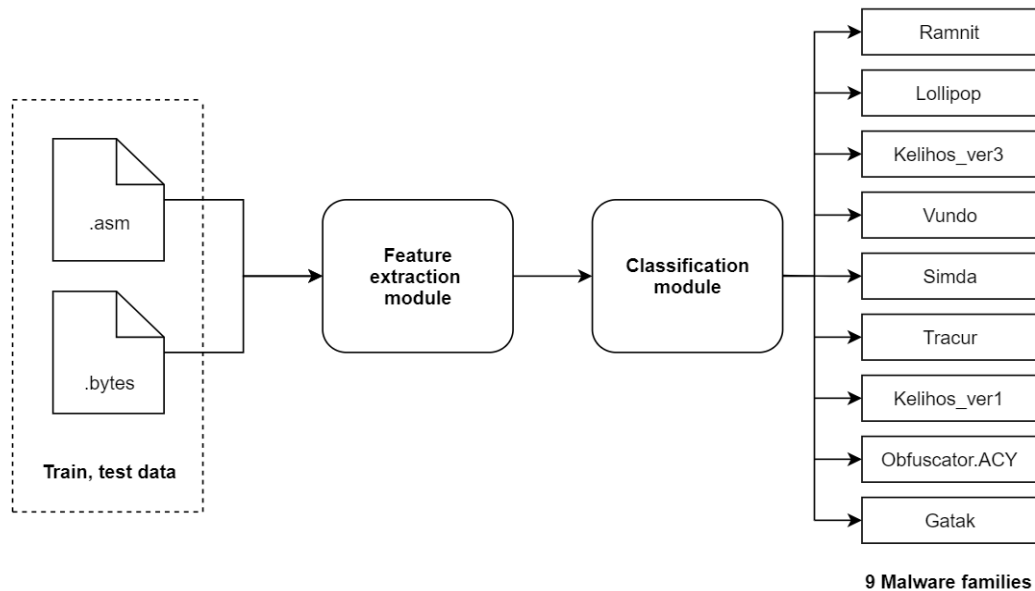


Fig.1: The system architecture of proposed system.

## VI. CONCLUSION

Advances in malware creation techniques like polymorphism, metamorphism, and obfuscation pose a threat to networked organizations and individuals. The signature-based approach for malware classification can prove to be inefficient to identify new malware variants created by the above techniques because of having unrecognized signatures. Hence, research is being done in behavior-based analysis of the executable file samples. The analysis can be carried out either statically or dynamically, or using the hybrid approach which uses both the approaches that is static analysis and the dynamic analysis in conjunction. The performances of each of all these methods are compared and evaluated in this paper. Research works related to all of the above approaches are surveyed in the paper. Based on the research, we also propose a system to classify the malware samples using static analysis with optimum features to increase accuracy of classification for Big Data.

Future scope of the proposed model can be classifying the misclassified malware samples using dynamic approach: Misclassified files if any can be fed to a dynamic classifier and can be classified based on the behavioral analysis. Also, the Distributed programming can be used to enhance the

speed of operation: If the computation is distributed across a network, the high-performance system will be achieved for even huge datasets (Big Data).



## REFERENCES

- [1] <https://www.pwc.com/us/en/increasing-it-effectiveness/publications/assets/2015-us-cybercrime-survey.pdf>
- [2] <https://www.gemalto.com/press/pages/gemalto-releases-findings-of-2015-breach-level-index.aspx>
- [3] <https://www.pwc.com/gx/en/economic-crime-survey/pdf/GlobalEconomicCrimeSurvey2016.pdf>
- [4] <https://www.tripwire.com/state-of-security/security-data-protection/the-four-most-common-evasive-techniques-used-by-malware/>
- [5] <http://malware.wikia.com/wiki/WannaCry>
- [6] <https://pdfs.semanticscholar.org/646c/8b08dd5c3c70785550eab01e766798be80b5.pdf>
- [7] <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Backdoor:Win32/Kelihos.F>
- [8] [https://www.symantec.com/security\\_response/writeup.jsp?docid=2010-011922-2056-99](https://www.symantec.com/security_response/writeup.jsp?docid=2010-011922-2056-99)
- [9] [https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win\\_32%2FTracur](https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Win_32%2FTracur)

- [10] [https://www.symantec.com/security\\_response/writeup.jsp?docid=2004-112111-3912-9](https://www.symantec.com/security_response/writeup.jsp?docid=2004-112111-3912-9)
- [11] <https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Adware:Win32/Lollipop>
- [12] [https://www.symantec.com/security\\_response/writeup.jsp?docid=2012-012813-0854-99](https://www.symantec.com/security_response/writeup.jsp?docid=2012-012813-0854-99)
- [13] Mamoun Alazab et al., “Zero-day Malware Detection based on Supervised Learning Algorithms of API call Signatures”, 2011, Australian Computer Society, vol. 121, 2011.
- [14] [https://en.wikipedia.org/wiki/Zero-day\\_\(computing\)](https://en.wikipedia.org/wiki/Zero-day_(computing))
- [15] Ronghua Tian et al. “Differentiating Malware from Cleanware Using Behavioural Analysis”, Malicious and Unwanted Software (MALWARE), 5th International Conference, 2010.
- [16] Deguang Kong et al. ,”Discriminant Malware Distance Learning on Structural Information for Automated Malware Classification”, 19th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 1357-1365, 2013.
- [17] X. Hu et al., ”Scalable malware classification with multifaceted content features and threat intelligence”, IBM Journal of Research and Development, vol. 60, issue 4, 2016.
- [18] Kevadia Kaushal et al., ”Metamorphic Malware Detection Using Statistical Analysis”, International Journal of Soft Computing and Engineering ISSN: 2231-2307 (Online), vol. 2, issue 3, 2012.
- [19] <https://www.hex-rays.com/products/ida/>
- [20] Hassan Takabi et al., “DLLMiner: structural mining for malware detection”, Security and Communication Networks vol. 8,issue 8,pp 3311-3322, 2015.
- [21] Ammar Ahmed E. Elhadi et al., “Malware Detection Based on Hybrid Signature Behaviour Application Programming Interface Call Graph”, American Journal of Applied Sciences, vol. 9, pp. 283-288 , 2016.
- [22] Taegy Kim et al., “Malfinder: Accelerated Malware Classification System through Filtering on Manycore System”, Information Systems Security and Privacy, 2015.
- [23] <https://www.vadesecond.com/en/polymorphic-malware/>
- [24] [https://www.blackhat.com/presentations/bh-usa-08/Hosmer/BH\\_US\\_08\\_Hosmer\\_Polymorphic\\_Malware.pdf](https://www.blackhat.com/presentations/bh-usa-08/Hosmer/BH_US_08_Hosmer_Polymorphic_Malware.pdf)
- [25] <https://blog.malwarebytes.com/threat-analysis/2013/03/obfuscation-malwares-best-friend/>
- [26] Nur Syuhada Selamat et al.,“Polymorphic Malware Detection”, IT Convergence and Security (ICITCS), 6th International Conference, 2016.
- [27] Duaa Ekhtoom et al. “A Compression-Based Technique to Classify Metamorphic Malware”, Computer Systems and Applications (AICCSA), 2016
- [28] Scott Treadwell et al.,”A heuristic approach for detection of obfuscated malware”, Intelligence and Security Informatics, 2009.
- [29] Abhay Kumar Sahoo et al.,”Signature based malware detection for unstructured data in Hadoop”, Advances in Electronic, Computers, Communications (ICAIECC) International Conference, 2014.
- [30] Mila Dalla Preda et al., “A Semantics-Based Approach to Malware Detection”, Volume 42 Issue 1, pp. 377-388, January 2007.
- [31] R. Sekar et al. “A fast automaton-based approach for detecting anomalous program behaviors”, In IEEE Symposium on Security and Privacy, 2001.
- [32] K. Wang et al., “Anomalous payload-based network intrusion detection”, in Proceedings of the 7th International Symposium on (RAID), pages 201–222, September 2004.
- [33] Nir Nissim et al., ”Novel active learning methods for enhanced PC malware detection in windows”, Elsevier, Expert Systems with Applications, vol. 41, Issue 13, pp 5843-5857, 2014.
- [34] Masoud Narouei et al., “DLLMiner: structural mining for malware detection”, Security and Communication Networks, vol. 8, issue 18, 2015.
- [35] Mehadi Hassen et al.,”Malware classification using static analysis based features”, Computational Intelligence (SSCI), 2017.
- [36] M. Shankarapani et al., ”Kernel Machines for Malware Classification and Similarity Analysis”,Neural Networks (IJCNN),The International Joint Conference,2010.
- [37] Mohammad Imran et al.,”Using Hidden Markov Model for Dynamic Malware Analysis: First Impressions”, Fuzzy Systems and Knowledge Discovery (FSKD), 12th International Conference, 2015.
- [38] Andrii Shalaginov et al.,”Automated intelligent multinomial classification of malware species using dynamic behavioural analysis”,Privacy, Security and Trust (PST),14th Annual Conference,2016.
- [39] Oscar Somarriba et al.,”A Collaborative Framework for Android Malware Detection using DNS & Dynamic Analysis”, Central America and Panama Convention (CONCAPAN XXXVII) , 2017.
- [40] Mohamad Fadli et al., ”An Approach for Malware Behavior Identification and Classification”,Computer Research and Development (ICCRD), vol.1, 2011.
- [41] R. J. Mangialardo et al., “ Integrating Static and Dynamic Malware Analysis Using Machine Learning”, IEEE Latin America Transactions ,vol. 13, issue 9, 2015..
- [42] Sean Kilgallon et al., ”Improving the Effectiveness and Efficiency of Dynamic Malware Analysis with Machine Learning”, Resilience Week (RWS), 2017.
- [43] <https://www.toptal.com/machine-learning/machine-learning-theory-an-introductory-primer>
- [44] [https://www.medcalc.org/manual/logistic\\_regression.php](https://www.medcalc.org/manual/logistic_regression.php)
- [45] <http://dataaspirant.com/2017/02/06/naive-bayes-classifier-machine-learning/>
- [46] <https://dataaspirant.com/2017/01/30/how-decision-tree-algorithm-works/><https://www.kaggle.com/c/malware-classification/>
- [47] <http://www.learnbymarketing.com/481/decision-tree-flavors-gini-info-gain/>
- [48] [https://en.wikipedia.org/wiki/Information\\_gain\\_in\\_decision\\_trees](https://en.wikipedia.org/wiki/Information_gain_in_decision_trees)
- [49] <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/>
- [50] [https://en.wikipedia.org/wiki/K-nearest\\_neighbors\\_algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm)
- [51] <http://xgboost.readthedocs.io/en/latest/>
- [52] <https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/>
- [53] <https://www.kaggle.com/c/malware-classification/data>



## BIOGRAPHY OF AUTHORS

	<p><i>Mahendra Deore</i> is working as an Asst. Professor in Computer Engineering Department at MKSSS's Cummins College of Engineering for Women, Pune, India. He was awarded his Master of Technology Degree from Bharati Vidyapeeth Deemed University College of Engineering, Dhankawadi, Pune. He is a research scholar at SGGS Institute of Engineering and Technology, under SRMTUN University Nanded. His areas of interest are Bigdata, Security and Computer Network.</p>
	<p><i>Dr. Uday Kulkarni</i> is working as a Professor, Head in the Department of Computer Science and Engineering at SGGS Institute of Engineering and Technology, Nanded India. He received doctoral degree from Swami Ramanand Teertha Marathwada University, Indian 2002. He is a recipient of a national level gold medal in the Computer Engineering Division for his research paper "Fuzzy Hyper sphere Neural Network Classifier" published in the journal of Institution of Engineers in 2004. He has published more than forty research papers in the field of Neural Networks, Fuzzy Logic and hybrid computing systems in the reputed journals and conferences.</p>