ISSN:2320-0790

An International Journal of Advanced Computer Technology

Ensemble-based Malware Detection with Different Voting Schemes

Ms. Jyoti H. Landage[1], Prof. Mr. Mahesh P.Wankhade[2]

[1]PG Student, Sinhgad College of Engineering, India, Maharashtra, Pune-41
[2]Associate Professor, Sinhgad College of Engineering, India, Maharashtra, Pune-41

Abstract: Now a day's computer security is the field which attempts to keep information on the computer safe and secure. Security means permitting things you do want, while preventing things you don't want from happening. Malware represents a serious threat to security of computer system. Traditional anti-malware products use the signature-based, heuristic-based detection techniques to detect the malware. These techniques detect the known malware accurately but can't detect the new, unknown malware. This paper presents a malware detection system based on the data mining and machine learning technique. The proposed method consists of disassemble process, feature extraction process and feature selection process. Three classification algorithms are employed on dataset to generate and train the classifiers named as Ripper, C4.5, IBk. The ensemble method Voting is used to improve the accuracy of results. Here majority voting and veto voting are implemented; the expected output is decided on the basis of majority and veto voting. The decision strategy of veto is improved by introducing the trust-based veto voting. The results of majority voting, veto voting and trust-based veto voting are compared. The experimental results show that the trust-based veto voting can accurately detect known and unknown malware instances better than majority voting and can identify the benign files better than veto voting.

Keywords: Data Mining, Ensemble, Feature Extraction, Feature selection, Machine learning, Malware detection, Majority voting, Trust, Veto Voting .

I. INTRODUCTION

The internet plays an important role in all areas of society from economy to the government. It connects millions of computers, which are vulnerable to attacks by malware. Malware is malicious software that performs any kind of malicious action to a computer system. With the rapid growth of internet applications, malware is one of the most serious threats to the information and computer system security. The annual report on the Internet security threat-2013 by Symantec says "Threat to online security has increased and evolved significantly in 2012, in specific, social media and mobile devices have come under growing attack in 2012" [17]. Traditional malware detection techniques detect the known instances of malware accurately but it can't detect the unknown instances of malware with different forms.

In recent years, some researchers have used the data mining and machine learning algorithms, which show promising results in detecting known and unknown malware. Our purpose is to improve the malware detection accuracy by using ensemble method viz. voting. The proposed methodology consists of disassemble process i.e. sequence of opcodes are extracted as a feature from malicious and benign executable files which is a part of feature extraction process. Three pre-processing techniques are used to produce three datasets using sequence of opcodes with different configuration. Feature selection technique is used to achieve the consistent datasets. Three classification algorithms are employed on these three datasets to generate and train the classifiers named as Ripper, C4.5 and IBk. The ensemble learning algorithm voting is used to improve the detection accuracy. Here majority voting and veto voting are used; the predicted output is decided on the basis of majority voting and veto voting. In veto voting the decision strategy of veto is improved by introducing the trust-based veto voting. Here the concept of trust is used. The definition of trust varies according to application area of the problem being solved. The concept of trust is explained in [19]. The trust especially group trust is used in many applications like social network, internet applications, peer to peer network, mobile ad-hoc network.

The rest of the paper is organized as follows: In Section 2 the related work is discussed. Section 3 and 4 present the overall implementation process of malware detection system along with the different voting schemes by discussing the system architecture. In Section 5 the performance evaluation and results are presented and discussed. Finally section 6 presents the future work and concludes the paper.

II. LITERATURE SURVEY

Malware detection system is used to determine whether a program has malicious intent or not. Two tasks are involved in detection system- Analysis and Detection [3].

Malware analysis is the process of examining the purpose and functionality of malware. The purpose of malware analysis is to gain the knowledge about how a specific piece of malware functions so that security can be built to protect the organizations network. There are three types of malware analysis techniques Static analysis, Dynamic analysis and Hybrid analysis. All these techniques are explained in detail in [6, 8]. Malware detection is used to detect the malware and prevent the computer system from being infected, protecting it from potential information loss. Traditional malware detection techniques can be categorized into signature-based detection, heuristic-based detection and specification-based detection technique. All these techniques are described in detail in [15-16, 20].

From last decade data mining and machine learning is the main focus for detecting the new, unknown malwares and it is the advanced research area for malware detection technique. In 2001 Schultz first introduced the concept of applying the data mining and machine learning algorithm for detection of malware. The result indicated that detection rate of data mining-based detection method is higher than signature based detection method [9]. In 2010 Yi-Bin Lu et al. introduced different ensemble learning algorithms and proposed new ensemble method SVM-AR which offers a better performance in term of accuracy and detection rate [23]. In 2010 Raja Khurram Shazhad detected the spyware by using the data mining and machine learning technique. He used the byte sequences as features. Two different approaches were used as part of feature extraction process for creating two different datasets with different data representation: common feature-based extraction (CFBE) and frequency-based feature extraction (FBFE). The result showed that it achieved the 90.5% overall accuracy with the J48 decision tree algorithm when using n=6 and CFBE feature selection technique [11]. In 2011 he presented the detection of adware by using data mining and machine learning technique. The KNN and SVM were effective when the data is noisy and KNN's performance is improved incrementally when new training data are introduced [12]. In 2012 Asaf shabtai et al. presented the detection of unknown malicious instances by applying the various classification algorithms on opcode patterns and evaluated the number of experiments. He found that the setting of opcode bi-gram, TF, 300 features selected by DF measure outperformed. The performance of decision tree and boosted decision tree was better as compared to NB and boosted NB [1]. The trust model developed the method to automatically compute the trust based on self-experience and recommendation of others [4]. Trust can be computed as +1 or -1, this increased or decreased value can help in determining the level of the trust. A set of inference rules are used to value the trust i.e. $0 \leq trust \leq 1$ and derived value is further used for the decision [4, 14].

The proposed system uses the concept of trust to support the veto and successfully detect the known as well as unknown instances of malware with high accuracy.

III. IMPLEMENTATION DETAILS

We propose a data mining-based malware detection technique, which includes disassembling the malware and benign files during pre-processing. The purpose is to investigate data mining-based detection and extend the idea of heuristic-based malware detection method in order to achieve high accuracy. The entire process is divided into two phases: Training phase and Testing Phase. In training phase training set of benign and malicious files is provided to the system. Each file is disassembled and the opcode is extracted as a feature from it.

In this study IDA pro disassembler is used to extract the opcode. It is most advanced commercial disassembly software. After extraction, feature selection process is employed to remove the noisy, redundant data as a result we get the consistent data to prepare the training dataset. This training dataset is used by data mining algorithm to generate the rule sets i.e. classifiers. These classifiers are used in voting schemes to classify previously unseen executable as malicious or benign. In testing phase, the test data set of new benign and malicious files which did not appear in the training set are pre-processed same as training phase. The test dataset used as input to voting models and test the accuracy of classifiers over unknown executable files. The performance of generated classifiers is evaluated with the help of standard accuracy measures. The system architecture of proposed method is shown in the following Fig.1.

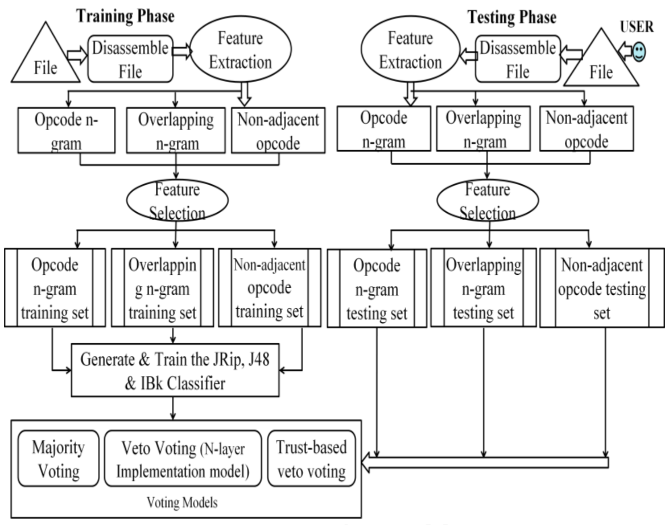


Fig.1. System Architecture [7]

IV. SYSTEM BUILDING BLOCK

The experiment that will be performed consists of four modules which are explained below with their functionality.

A. Dataset construction

We have created a dataset of 500 files out of which 250 are malicious files provided by [21]. To identify the benign files, we used Norton Internet security antivirus. Benign files, including executable files were gathered from machine running the window 7 operating system on our campus. The benign set contained 250 files. The Norton Internet security antivirus was used to verify that these files did not contain any malicious code. Out of 250 benign files some files were downloaded from the website CNET download.com [2].

B. Feature Extraction

Three feature extraction methods are used to extract the opcode as feature and three correspondence data sets are prepared [13]. Three techniques are explained below.

1) **Opcode n-gram:** Dataset is prepared with size of n=2, 3, 4,. . .8. We select the intermediate value as n=2. To understand this process, assume that disassembled file contains the following given data. A couple of character represents an opcode.

11 aa 22 bb 33 cc 44 dd 55 ee

If n=2 then generated n-gram i.e. bi-gram are 11aa 22bb 33cc 44dd 55ee.

2) **Overlapping n-gram*:*** In this technique dataset is prepared with introducing two parameters, named as Size and Step. The Size parameter defines the size of n-gram to be extracted and Step parameter defines the number of opcodes to be skipped before extracting the next n-gram. If size=2 and step=1 then generated string will be 11aa aa22 22bb bb33 33cc cc44 44dd dd55 55ee.

3) **Non-adjacent opcode extraction:** Some changes are made to this technique i.e. size parameter is replaced by start-end size parameter. It defines the number of adjacent opcodes to be extracted for start and end of n-gram. The step parameter defines the number of opcodes to be skipped for extracting a new n-gram and Gap size parameter specifies the gap between start and end opcode. If start-end size=2, step=1 and gap=1 then generated string will be 11aabb33 aa2233cc 22bbcc44 and so on.

Above mentioned three feature extraction techniques are used to extract number of possible combination of strings and three correspondence datasets are produced.

C. Feature Selection

In this process the text categorization technique: Term Frequency-Inverse document frequency (TF-IDF) is used. The purpose is to eliminate the redundant, irrelevant, noisy data from the entire large dataset and to selects relevant, consistent features from the entire large feature set as a result reduced feature dataset will be achieved. N-gram is equivalent to term or word in text document. For each term (t) in the vocabulary, its frequency (f) in the single document (d) and in the entire document set (D) is calculated. Weight is assigned to each term. Weight is equal to its frequency (f) in single document (d), such weights are called as term frequency (tf) i.e. frequency of term in document. The frequency (F) of each term is calculated in D, this is called Document Frequency (DF).
The normalized TERM FREQUENCY (TF) is calculated by dividing the frequency of term in document (tf) by the frequency of most frequent term in document [max (tf)]. The value is within the range of [0-1] as refer to (1):

$$TF = \frac{tf}{\max(tf)} \qquad (1)$$

The TF-IDF combines the TF and DF. The formula for TF-IDF is given below:

$$TF - IDF = TF * \log\left(\frac{N}{DF}\right) \qquad (2)$$

N: Number of document in the entire data set.
DF: No. of document (d) in which term (t) appears [10].

The reduced feature dataset is achieved during the feature selection process. The reduced feature dataset is converted into attribute relation file format (ARFF) file format and it is used as input to three classification algorithm named as Ripper (JRip)[22], Decision Tree (C4.5)[5], and K-nearest neighbor (IBk)[18] to generate and train the classifiers. These trained classifiers are used in classification model; this stage is called as training stage.

D. Classification Model

The classification model consists of majority voting model for majority voting, N-layer implementation model for Veto voting, the decision strategy of simple veto is improved by introducing trust-based veto voting. In these models classifiers are used as committee and every classifier gives its own decision. The classifier ensemble learning algorithm Voting is used to reach final prediction. Classifier ensemble performs better than single classifier and helps to improve the detection accuracy.

1) **Majority Voting*:*** The majority voting is simple and effective scheme. It follows democratic rules i.e. the class with maximum number of votes is the outcome. The diagram for majority voting model is shown in following Fig.2.

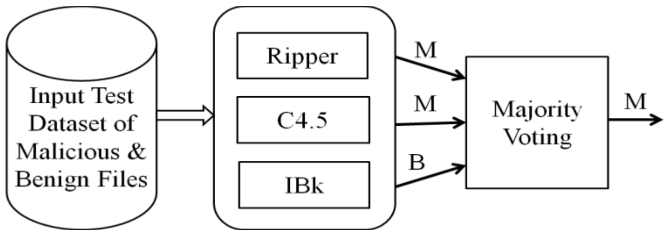


Fig.2. Majority Voting Model

The input to classifiers in the given diagram is the test data set. Every classifier gives its own decision as whether the file is malicious or benign. The outcome of multiple classifiers is combined and final decision is taken on the basis of majority voting [13]. The drawback of majority voting is that if no. of classifiers in the system is an even then it's difficult to make the decision because equal number of votes are given to both benign and malware classes. Another drawback is that the decision taken by majority voting can be wrong if majority of classifiers will classify malware instances as benign. Such drawbacks are overcome by using veto voting schemes.

2) **Veto Voting:** In veto voting, committee gives importance to single expert (classifier) who predicts against the majority. Any vote indicating instances as malware, alone can determine the outcome of the classification task regardless of the count of other votes. The n-layer implementation model is used for veto voting. The model consists of number of n-layers and number of n classifiers.

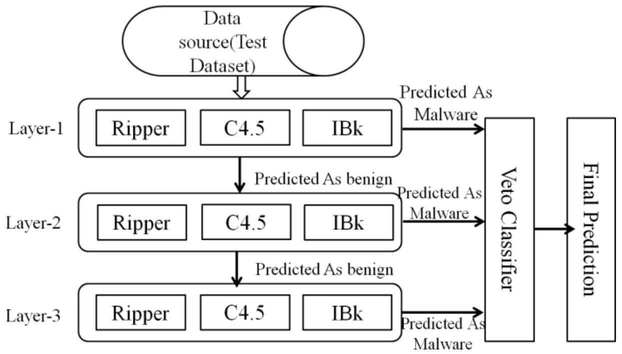


Fig.3. Veto Voting Model [13]

As shown in the Fig.3, we use three layers and three classifiers with different configuration. The veto architecture is configured in two ways. In the first way, there are three layers and at each layer three classifiers are generated viz. Ripper, C4.5, and IBk. All these classifiers are trained using same training dataset and the correspondent test dataset is given as input to the veto architecture.
In the second way, the veto architecture is configured with three layers and each layer has three different classifiers. First layer is configured with classifiers that are trained using opcode n-gram dataset, second layer is configured with classifiers that are trained using overlapping n-gram dataset and lastly the third layer is configured with classifiers that are trained using non-adjacent opcode n-gram dataset and the correspondent test dataset is given as input to each layer. The test dataset of benign and malicious files is given as input to n-layer classification model. In upper layer, the instances that are declared as benign are forwarded to lower layer for reclassification. If at any layer, instance is classified as malware, it is not processed to the next layer; the decision from this layer directly goes to veto classifier. Any classifier predicting instance as malware can act as veto to determine the outcome of final classification task, otherwise instance further proceeds for reclassification and classification results from all layers are given to the veto classifier. This decision strategy of veto is improved by introducing the trust-based veto voting [14].

3) **Trust-based Veto Voting:** In this scheme quantitative representation of trust is used. Trust is quantified as +1 or -1. The positive integer represents trust and negative integer represents the distrust. The increased or decreased value helps in determining the level of trust. In this scheme Local trust, recommended trust and Global trust are calculated and value of global trust is used for deciding the veto.

***a)* Local Trust**

In local trust each algorithm in the system calculates its trust level for other algorithms in the system which means how much algo$_p$ trusts the algo$_q$ in term of predicting the class of an instance, called as local trust. Local trust of algo$_p$ on algo$_q$ is calculated by comparing the predictions (d) of both algorithms with each other and actual class (C) of instance, so from data set of benign and malicious instances, an instance of benign class is given to both algo$_p$ and algo$_q$ for predicting the class of instance. The possible predictions are, both algorithms may predict correct or both algorithms may predict incorrect or any one of the algorithm may predict the correct class. If both algorithms give the same prediction either correct or incorrect, trust is not affected. If algo$_p$ predicts the incorrect class and algo$_q$ predicts the correct class then algop increases the trust level of algo$_q$ with +1. If algo$_p$ predicts the correct class and algo$_q$ predicts the incorrect class then algo$_p$ increases the distrust level of the algo$_q$ with +1. Likewise all the instances in the dataset are given to both algorithms sequentially for the prediction. At the end of process local trust of algo$_p$ on algo$_q$ is calculated by dividing the trust (sat) with the sum of trust (sat) and distrust (unsat). The algorithm for local trust calculation is given below:

Algorithm: Local Trust Calculation [14]

Input: Actual class of instance (C), Prediction of algo$_p$ (d$_p$), Prediction of algo$_q$ (d$_q$)

Output: Local trust of algo$_p$ on algo$_q$ $(t_q : a\lg o_p \rightarrow a\lg o_q)$

```
Repeat
If $(d_p = d_q)$ then
    Move next
End if
If $(d_p \neq d_q)$ then
    If $(d_p = C)$ then
        $Unsat = unsat(a\lg o_{p,} a\lg o_q) + 1$
    Else $(d_q = C)$
        $Sat = sat(a\lg o_{p,} a\lg o_q) + 1$
    End if
End if
```

Until !EOF

$$\left(t_q : a\lg o_p \rightarrow a\lg o_q\right) = \frac{sat\left(a\lg o_p, a\lg o_q\right)}{sat\left(a\lg o_{p,} a\lg o_q\right) + unsat\left(a\lg o_{p,} a\lg o_q\right)}$$

In our system there are three classifiers named as JRip, J48 and IBk. Here we calculate the local trust of JRip on J48 and vice versa, local trust of J48 on IBk and vice versa and local trust of IBk on JRip and vice versa. Local trust shows a unique trust on particular classifier from another classifier. This value is used to calculate the recommended trust.

***b)* Recommended Trust**

The recommended trust is calculated by adding the local trust of all algorithms in the system on that particular algorithm.

If the set of all algorithms is S = algo0, algo1,...algon and two subsets S'=algo0 and S"={algo1,algo2,...algon}. The subset S" is having all the algorithms as members in the system except the algorithm algo0 for which recommended trust is calculated. The recommended trust is calculated by using the following formula:

$$RT_q \leftarrow \sum_{n=1}^{n} (\mathrm{t_q : algo_n \rightarrow algo_q}) \ \forall a\lg o_n \in \mathrm{S''} \quad (3)$$

The above formula is used to calculate the recommended trust of JRip, J48 and IBk. The value of recommended trust is normalized to obtain the global trust of that particular algorithm.

c) Global Trust

Basic aim of normalization is to transform the different values on a theoretically standard scale to compare them equally with each other. The normalized global trust value lies in interval of the [0-1] and it is calculated by using following formula.

$$GT_q \leftarrow \frac{RT_q}{\sqrt{\sum_{n=1}^{n} RT_n^2}} \quad (4)$$

d) Veto decision

The veto is decided by using value of global trust. Three classifiers are used in the system named as JRip, J48, and IBk. If two classifiers i.e. JRip and IBk classify the instances as a benign and only one classifier i.e. J48 classifies the instance as a malware then mean of J48 and JRip and IBk is calculated. If the mean of J48 is greater than mean of JRip and IBk then J48 can act as veto and decision will be the prediction of the J48.

$$Veto : GT_{j48} \geq \frac{GT_{Jrip} + GT_{IBk}}{2} \quad (5)$$

In trust-based veto voting every classifier calculates the trust and keeps the trust information locally in a trust table without increasing the processing overhead. The locally stored trust information is used for decision purpose when needed [14].

E. Evaluation Metric

The performance of each classifier is evaluated using following measure.

1) **Overall Accuracy:** Measure number of absolutely, correctly classified instances either positive or negative divided by the entire number of instances.

$$OA = \frac{TP + TN}{TP + TN + FP + FN} \quad (6)$$

2) ***Recall:*** It is also called as True positive rate (TPR). It is the rate of number of positive instances classified correctly.

$$\mathrm{Re}\,call\,/\,TPR = \frac{TP}{TP + FN} \quad (7)$$

3) **False Positive Rate (FPR):** It is number of negative instances misclassified.

$$FPR = \frac{FP}{FP + TN} \quad (8)$$

4) **Precision (P):** It represents the amount of samples classified as malicious that are really malicious.

$$P = \frac{TP}{TP + FP} \quad (9)$$

5) **F1-Measure:** It is the harmonic mean of precision and recall.

$$F1 - measure = \frac{2 * R * P}{R + P} \quad (10)$$

Where

True Positives (TP): The number of malicious samples classified as malicious.

True Negatives (TN): The number of benign samples classified as benign.

False Positives (FP): The number of benign samples classified as malicious.

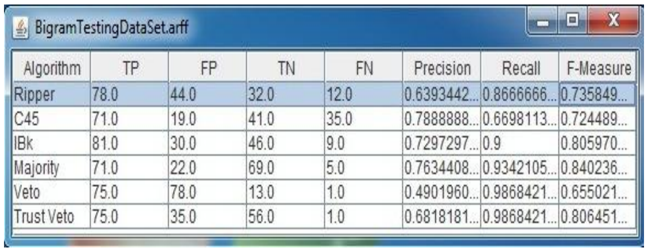
False Negatives (FN): The number of malicious samples classified as benign.

V. EXPERIMENTAL RESULTS

The experiment is evaluated in two ways i.e. veto architecture is configured in two ways. The working procedure of these two ways is explained in section IV.

A. Bi-gram

When we choose bi-gram option, the bi-gram training dataset goes as input to three classification algorithms and three trained classifiers are generated. All these classifiers are present at all three layers. The bi-gram test dataset is selected as input to the voting models. The classification results of all three classifiers and three voting schemes are displayed. The results of classification are shown in the below Fig.4.

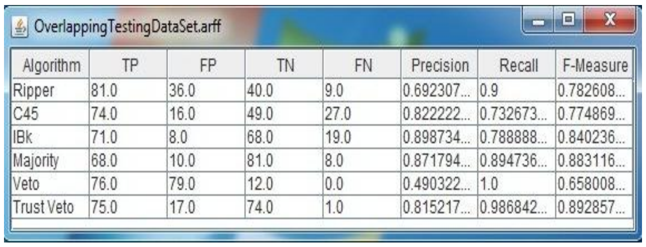

BigramTestingDataSet.arff

Algorithm	TP	FP	TN	FN	Precision	Recall	F-Measure
Ripper	78.0	44.0	32.0	12.0	0.6393442...	0.8666666...	0.735849...
C45	71.0	19.0	41.0	35.0	0.7888888...	0.6698113...	0.724489...
IBk	81.0	30.0	46.0	9.0	0.7297297...	0.9	0.805970...
Majority	71.0	22.0	69.0	5.0	0.7634408...	0.9342105...	0.840236...
Veto	75.0	78.0	13.0	1.0	0.4901960...	0.9868421...	0.655021...
Trust Veto	75.0	35.0	56.0	1.0	0.6818181...	0.9868421...	0.806451...

Fig.4. Results of bi-gram test dataset

B. Overlapping

When we choose Overlapping option, the overlapping training dataset is given as input to three classification algorithms and three trained classifiers are generated. All these classifiers are present at all three layers. The overlapping test dataset is selected as input to the voting models. The classification results of all three classifiers and three voting schemes are displayed. The results of classification are shown in the below Fig.5.

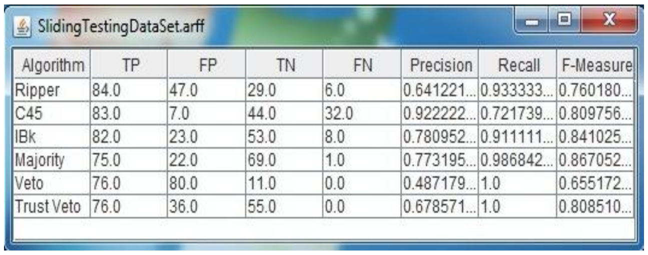

OverlappingTestingDataSet.arff

Algorithm	TP	FP	TN	FN	Precision	Recall	F-Measure
Ripper	81.0	36.0	40.0	9.0	0.692307...	0.9	0.782608...
C45	74.0	16.0	49.0	27.0	0.822222...	0.732673...	0.774869...
IBk	71.0	8.0	68.0	19.0	0.898734...	0.788888...	0.840236...
Majority	68.0	10.0	81.0	8.0	0.871794...	0.894736...	0.883116...
Veto	76.0	79.0	12.0	0.0	0.490322...	1.0	0.658008...
Trust Veto	75.0	17.0	74.0	1.0	0.815217...	0.986842...	0.892857...

Fig.5. Results of overlapping n-gram test dataset

C. Sliding window

When we choose Sliding window option, the sliding window training dataset is given as input to three classification algorithms and three trained classifiers are generated. All these classifiers are present at all three layers. The sliding window test dataset is selected as input to the voting models. The classification results of all three classifiers and three voting schemes are displayed. The results of classification are shown in the below Fig.6.

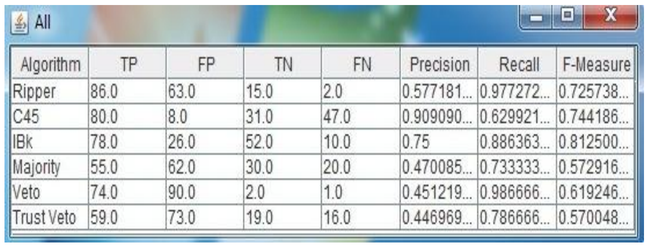

SlidingTestingDataSet.arff

Algorithm	TP	FP	TN	FN	Precision	Recall	F-Measure
Ripper	84.0	47.0	29.0	6.0	0.641221...	0.933333...	0.760180...
C45	83.0	7.0	44.0	32.0	0.922222...	0.721739...	0.809756...
IBk	82.0	23.0	53.0	8.0	0.780952...	0.911111...	0.841025...
Majority	75.0	22.0	69.0	1.0	0.773195...	0.986842...	0.867052...
Veto	76.0	80.0	11.0	0.0	0.487179...	1.0	0.655172...
Trust Veto	76.0	36.0	55.0	0.0	0.678571...	1.0	0.808510...

Fig.6. Results of S. window test dataset

D. All

In this case the veto architecture is configured with three layers and each layer is customized with three different classifiers that are trained using different training datasets. All three test datasets are sequentially given as input to veto architecture and their performance evaluation is shown in the below Fig.7.


All

Algorithm	TP	FP	TN	FN	Precision	Recall	F-Measure
Ripper	86.0	63.0	15.0	2.0	0.577181...	0.977272...	0.725738...
C45	80.0	8.0	31.0	47.0	0.909090...	0.629921...	0.744186...
IBk	78.0	26.0	52.0	10.0	0.75	0.886363...	0.812500...
Majority	55.0	62.0	30.0	20.0	0.470085...	0.733333...	0.572916...
Veto	74.0	90.0	2.0	1.0	0.451219...	0.986666...	0.619246...
Trust Veto	59.0	73.0	19.0	16.0	0.446969...	0.786666...	0.570048...

Fig.7. Results of All test dataset

The overall experimental results are shown in the below Fig.8.

Classifier & Voting / Dataset	Ripper						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	78	44	32	12	0.639	0.866	0.735
1-Overlapping	81	36	40	9	0.692	0.9	0.782
2-S.window	84	47	29	6	0.641	0.933	0.760
3-All	86	63	15	2	0.577	0.977	0.725
	IBk						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	81	30	46	9	0.729	0.9	0.805
1-Overlapping	71	8	68	19	0.898	0.788	0.840
2-S.window	82	23	53	8	0.780	0.911	0.841
3-All	78	26	52	10	0.75	0.886	0.812
	C4.5						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	71	19	41	35	0.788	0.669	0.724
1-Overlapping	74	16	49	27	0.822	0.732	0.774
2-S.window	83	7	44	32	0.922	0.721	0.809
3-All	80	8	31	47	0.909	0.629	0.744
	Majority						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	71	22	69	5	0.763	0.934	0.840
1-Overlapping	68	10	81	8	0.871	0.894	0.883
2-S.window	75	22	69	1	0.773	0.986	0.867
3-All	55	62	30	20	0.470	0.733	0.572
	Veto						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	75	78	13	1	0.490	0.986	0.655
1-Overlapping	76	79	12	0	0.490	1.0	0.658
2-S.window	76	80	11	0.0	0.487	1.0	0.655
3-All	74	90	2	1.0	0.451	0.986	0.619
	Trust-based						
	TP	FP	TN	FN	Precision	Recall	F-Measure
0-Bigram	75	35	56	1.0	0.681	0.986	0.806
1-Overlapping	75	17	74	1.0	0.815	0.986	0.892
2-S.window	76	36	55	0.0	0.678	1.0	0.808
3-All	59	73	19	16	0.446	0.786	0.570

Fig.8. Experimental Results

E. Analysis of Results

The experimental results of three voting schemes viz. majority, veto and trust-based veto voting can be discussed in three dimensions by using three measures, i.e. recall, precision, and F-measure. The precision is considered as a complementary measure with the recall. The precision and recall have inverse relationship, if the precision increases, the recall decreases and vice a versa. Therefore, another evaluation measure is required, that can combine the precision and recall. Thus the final evaluation measure is F-measure, which equally weights the precision and recall. The experimental results of classifiers and voting schemes are shown in Fig.8. On the basis of these results, we have analysed the performance of three voting schemes.

1. The low TN of the veto classifier leads to high FP; the low TN is because of veto classifier's inclination towards malware. The low FP of the majority voting leads to high TN; the low FP is because, majority voting is inclined towards benign programs. This indicates that the veto classifier correctly detects malwares while the majority voting correctly detects benign files.
2. The experimental results show that, the veto classifier has a better recall than the majority voting, this implies that the veto classifier reduces the number of misclassified malwares.
3. F-measure of the majority voting is higher than that of the veto classifier, however, the recall for the majority voting, is lower than the veto classifier so it may be said that the veto is a better approach for the malware detection. Thus, the veto classifier has been extended to increase the precision.

On the basis of the experimental results of majority and veto voting, the trust-based veto classifier is applied for combining the decisions of different algorithms trained on the same as well as different representation of data.

4. The trust-based veto voting performs better than the veto voting in terms of TN and reduces the FP.
5. The trust-based veto voting is better than the majority voting in terms of TP and reduces the FN. However, TP of trust-based veto voting is better than the majority voting and less than the veto voting. This indicates that the trust-based veto voting can accurately detect known and unknown malware instances better than majority voting and can identify the benign files better than veto voting; also the precision of trust-based veto voting is higher than veto voting.
6. Recall of the trust-based veto voting is better than the majority voting and less than the veto voting, however, the composite measure F1 of trust-based veto voting is better than veto voting and the majority voting is slightly better than trust-based veto voting.

VI. CONCLUSION AND FUTURE SCOPE

Data mining-based detection method can detect the known as well as unknown malware and ensemble voting improves the accuracy of the results. Static analysis is fast and safe as files are analysed without its execution and it detects the malware accurately. The majority voting can detect benign files more precisely but fails to detect malicious files as accurately as veto voting. On the contrary, the veto voting can detect malicious files more precisely but fails to detect benign files as accurately as majority voting. The trust-based veto voting overcomes the drawbacks of both majority and veto voting, it can accurately detect known and unknown malware instances better than majority voting and can identify benign files better than veto voting. In future, the purpose is to improve the proposed model in three different directions, first is improvement in selection of classifiers for the optimal results, second is detection of different types of malware with multi-class prediction because in the proposed method different types of malware are used and third is automating the process of analysing and predicting the malware.

VII. REFERENCES

[1] Asaf Shabtai, Robert Moskovitch, Clint Feher, Shlomi Dolev and Yuval Elovici. *Detecting unknown malicious code by applying classification techniques on OpCode patterns*, Security Informatics: A Springer open journal [Online], Available:http://www.securityinformatics.com/content/1/1/1, 2012.

[2] CNET. Free Software Downloads, [Online], Available: http: // download. Cnet. Com, Nov-2013.

[3] Imtithal A. Saeed, Ali Selamat, Ali M. A. Abuagoub. A Survey on Malware and Malware Detection Systems. *International Journal of Computer Applications,* (0975: 8887), Volume 67, No.16. (April-2013).

[4] Jianqiang Shi, Gregor V. Bochhmann, Carlisle Adams. A trust model with statistical foundation, *School of information technology and Engineering (SITE),* System science, University of Ottawa.

[5] J. R. Quinlan. C4.5: programs for machine learning. *Morgan Kaufmann Publishers* Inc, 1993.

[6] Jyoti Landage, M.P. Wankhade. Malware and Malware Detection Techniques: A Survey. *International Journal of Engineering Research and Technology (IJERT)*, Vol. 2 Issue 12, December-2013 ISSN: 2278-0181.

[7] Jyoti Landage, M.P. Wankhade. Malware Detection with Different Voting Schemes, *COMPUSOFT, An international journal of advanced computer technology (IJACT)*, Vol. 3 Issue 1, January-2014, ISSN: 2320-0790.

[8] Kirti Mathur, Saroj Hiranwal. A Survey on Techniques in Detection and Analyzing Malware Executables. *International Journal of Advanced Research in Computer Science and Software Engineering*, ISSN: 2277-128X, Volume 3, Issue 4, April 2013.

[9] Matthew G. Schultz, Eleazar Eskin, Erez Zadok, and Salvatore J. Stolfo. Data Mining Methods for Detection of New Malicious Executables. *In Proceedings of the Symposium on Security and Privacy,* pp.38-49, 2001.

[10] R. K. Shahzad and N. Lavesson. Detecting scareware by mining variable length instruction sequences. *In Proceedings of the 10th Annual Information Security South Africa Conference (ISSA11),* IEEE, Johannesburg, South Africa, pp.1-8, August 2011.

[11] R. K. Shahzad, S. I. Haider, and N. Lavesson. Detection of spyware by mining executable files. *In Proceedings of the 5th International Conference on Availability, Reliability, and Security, IEEE Computer Society*, pp.295-302, 2010.

[12] R. K. Shahzad, N. Lavesson, H. Johnson. Accurate Adware Detection using Opcode Sequence Extraction., *In Proceedings of the 6th International Conference on Availability,Reliability and Security (ARES11),* Prague, Czech Republic, IEEE, pp.189-195, 2011.

[13] R. K. Shahzad, Niklas Lavesson. Veto-based Malware Detection. *In Proceedings of Seventh International Conference on Availability, Reliability and Security(ARES12)*, Prague, Czech Republic, IEEE, pp.47-54, 2012.

[14] R. K. Shahzad, Niklas Lavesson. , Comparative Analysis of Voting Schemes for Ensemble-based Malware Detection. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, volume: 4, number: 1, pp. 98-117, 2012.

[15] Robiah Y, Siti Rahayu S., Mohd Zaki M, Shahrin S., Faizal M. A., Marliza R., A New Generic Taxonomy on Hybrid Malware Detection Technique. *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 5, No. 1, 2009.

[16] Ronghua Tian. An Integrated Malware Detection and Classification System, Thesis, Changchun University of Science and Technology, Deakin University, August-2011.

[17] Takashi Katsuk. Crisis: The Advanced Malware, Internet security threat report-2013.,Symantec Corporation,Volume 18.

[18] T. Cover,P. Hart. Nearest neighbor pattern classification, IEEE Transactions on Information Theory, vol.13, no.1, pp.21-27, 1967.

[19] T.W. A. Grandison. Trust management for internet applications, Ph.D. dissertation, Imperial College of Science, Technology and Medicine, University of London, 2003.

[20] Vinod P., V. Laxmi, M.S.Gaur. Survey on Malware Detection Methods. *3rd Hackers, Workshop on Computer and Internet Security*, Department of Computer Science and Engineering, Prabhu Goel Research Centre for Computer and Internet Security,IIT, Kanpur,pp.74-79,March,2009.

[21] VX heavens. [Online], Available: http: // www. vxheavens . com / vl . php and http : // malwareurls . joxeankoret . Com / normal.txt.

[22] W. Cohen. Fast effective rule induction. *In Proceedings of 12th International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc., pp.115-123, 1995.

[23] Yi-Bin Lu, Shu-Chang Din, Chao-Fu Zheng, and Bai-Jian Gao. Using Multi-Feature and Classifier Ensembles to Improve Malware Detection. *JOURNAL OF C.C.I.T*,Vol.39, No.2, Nov-2010