INTELLIGENT MALWARE DETECTION SYSTEM

Sandeep B. Damodhare*

Prof. V. S. Gulhane**

Abstract: *Malicious programs spy on users' behavior and compromise their privacy. Unfortunately, existing techniques for detecting malware and analyzing unknown code samples are insufficient and have significant shortcomings. We observe that malicious information access and processing behavior is the fundamental trait of numerous malware categories breaching users' privacy (including key loggers, password thieves, network sniffers, stealth backdoors, spyware and root kits), which separates these malicious applications from benign software. Commercial anti-virus software is unable to provide protection against newly launched ("zero-day") malware. In this dissertation work, we propose a novel malware detection technique which is based on the analysis of byte-level file content. The proposed dissertation work will demonstrate the implementation of system for detection of various types of malware.*

*ME Student, Dept of IT, SIPNA's College of Engineering & Technology, Amravati (MS) INDIA

**Associate Professor, Dept of CSE, SIPNA's College of Engineering & Technology, Amravati (MS) INDIA

INTRODUCTION:

Malicious software (i.e., Malware) creeps into users' computers, collecting users' private information, wrecking havoc on the Internet and causing millions of dollars in damage. Malware detection and analysis is a challenging task, and current malware analysis and detection techniques often fall short and fail to detect many new, unknown malware samples. Current malware detection methods in general fall into two categories: signature-based detection and heuristics based detection. The former cannot detect new malware or new variants. The latter are often based on some heuristics such as the monitoring of modifications to the registry and the insertion of hooks into certain library or system interfaces. Since these heuristics are not based on the fundamental characteristics of malware, they can incur high false positive and false negative rates. For example, many benign software access and modify registry entries. Hence, just because an application creates hooks in the registry does not mean that it is malicious (i.e., the application could be a useful system utility). Furthermore, to evade detection, malware may attempt to hook library or system call interfaces that the detector does not monitor. Even worse, since many rootkits hide in the kernel, most such heuristics-based detectors cannot detect them as they do not necessarily modify any visible registry entries or library or system call interfaces.

Malware is software designed to infiltrate or damage a computer system without the owner's informed consent (e.g., viruses, backdoors, spyware, trojans, and worms) [1]. Numerous attacks made by the malware pose a major security threat to computer users. Hence, malware detection is one of the computer security topics that are of great interest. Currently, the most important line of defense against malware is antivirus programs, such as Norton, MacAfee, and Kingsoft's Antivirus. These widely used malware detection software tools use signature-based method to recognize threats. Signature is a short string of bytes, which is unique for each known malware so that future examples of it, can be correctly classified with a small error rate. However, this classic signature-based method always fails to detect variants of known malware or previously unknown malware, because the malware writers always adopt techniques like obfuscation to bypass these signatures [2]. In order to remain effective, it is of paramount importance for the antivirus companies to be able to quickly analyze variants of known malware and previously unknown malware samples. Unfortunately, the number of file samples that need to be analyzed on a daily basis is constantly increasing [3]. According to the virus analysts at Kingsoft Antivirus Laboratory, the "gray list" that is needed to be analyzed per day usually contain more than 70000 file samples. Clearly, there is a need for an automatic, efficient, and robust tool to classify the "gray list."

LITERATURE REVIEW/RELATED WORK:

Recently, many post processing techniques, including rule pruning, rule ranking, and rule selection have been developed for associative classification to reduce the size of the classifier and make the classification process more effective and accurate [5], [6], [14]. It is interesting to know how these post processing techniques would help the associative classifiers for malware detection. In this paper, we systematically evaluate the effects of the post processing techniques in malware detection and propose an effective way, i.e., CIDCPF, to detect the malware from the "gray list."

***1)Rule Pruning**:* In order to reduce the size of the classifier and make the classification process more effective and accurate, the removal of the redundant or misleading rules is indispensable. There are five popular rule pruning approaches which mainly focus on preventing these redundant or misleading rules from taking any part in the prediction process of test data objects.

***A)χ*2 (chi-square) test :**The test is always carried out on each generated rule to find out whether the rule's antecedent is positively correlated with the rule's consequent. It is adopted by classification based on multiple association rules (CMAR) algorithm in its rule discovery step.

B) Redundant rule pruning: This rule pruning method discards specific rules with fewer confidence values than general rules. Several algorithms, such as CMAR [1], [2], and [15], adopt this approach for rule pruning.

C)Database coverage: This pruning approach tests the generated rules against the training dataset, and only keeps the rules, which cover at least one training data object not considered by a higher ranked rule for later classification. This method is created by the classification based on associations (CBA) , and used by CMAR [15], and multiclass classification based on association rule (MCAR) .

D)Pessimistic error estimation: The method works by comparing the estimated error of a new rule. If the expected error of the new rule is lower than that of the original rule, then the original rule will be replaced by the new rule. CBA have used it to effectively reduce the number of the generated rules.

E)Lazy pruning: This method discards the rules, which incorrectly classify training objects and keeps all others. It has been used in [5] for rule pruning.

2) Rule Ranking: Rule ranking plays an important role in the classification process, since most of the associative classification algorithms, such as CBA, CMAR [4], [5], multiclass, multilabel associative classification (MMAC) , and MCAR, utilize rule ranking procedures as the basis for selecting the classifier. Particularly, CBA and CMAR use database coverage pruning approach to build the classifiers, where the pruning evaluates rules according to the rule ranking list. Hence, the highest order rules are tested in advance, and then, inserted into the classifier for predicting test data objects.

For rule ranking, there are five popular ranking mechanisms :

a)confidence support size of antecedent (CSA);

b) size of antecedent confidence support (ACS);

c) weighted relative accuracy (WRA);

d) Laplace accuracy; and

e) $\chi 2$ (chi-square) measure.

CSA and ACS are belonging to the pure “support-confidence” framework and have been used by CBA and CMAR for rule ranking. WRA, Laplace accuracy, and $\chi 2$ measure are used by some associative classification algorithms, such as classification based on predictive association rules (CPAR), to weigh the significance of each generated rule.

3) Rule Selection: After pruning and reordering the generated rules, we can select the subset of the rules from the classifier to predict new file samples. There are three common rule selection approaches :

a) Best first rule: This approach selects the first best rule that satisfies the given data object according to the rule list based on certain rule ranking mechanism to predict the new data object. It is used in CBA for predicting test data objects.

b) All rules: This method collects all rules in the classifier that satisfy the new data object, and then, evaluate this collection to identify its class. CMAR uses weighted $\chi2$ (WCS) testing to predict the class of the new data object.

c) Best *k* rules: Some associative classification algorithms, like CPAR, select the first best *k* rules that satisfy the new data object, and then, make predictions using certain averaging process.

ANALYSIS OF PROBLEM:

1) Systematically evaluate the effects of the post processing techniques in malware detection.

2) Propose an effective way CIDCPF, to detect the malware from the "gray list." CIDCPF adapts several different post processing techniques of associative classification, including rule pruning, rule ranking, and rule selection, for building effective associative classifiers.

3) Improve former malware detection system IMDS and update it to IMDS.

4) Perform cases studies on a large collection of executables including 35000 malicious ones and 15000 benign samples, collected by the Antivirus Laboratory of Kingsoft Corporation.

5) Provide a comprehensive experimental study on various antivirus software as well as various data mining techniques for malware detection using our data collection.

IMPLEMENTATION

SYSTEM ARCHITECTURE

The system architecture of our malware detection is shown in Fig. 1.Basically, the system first uses the feature extractor to extract the API calls from the collected portable executable (PE) files, converts them to a group of 32-bit global IDs as the features of the training data, and stores these features in the signature database.

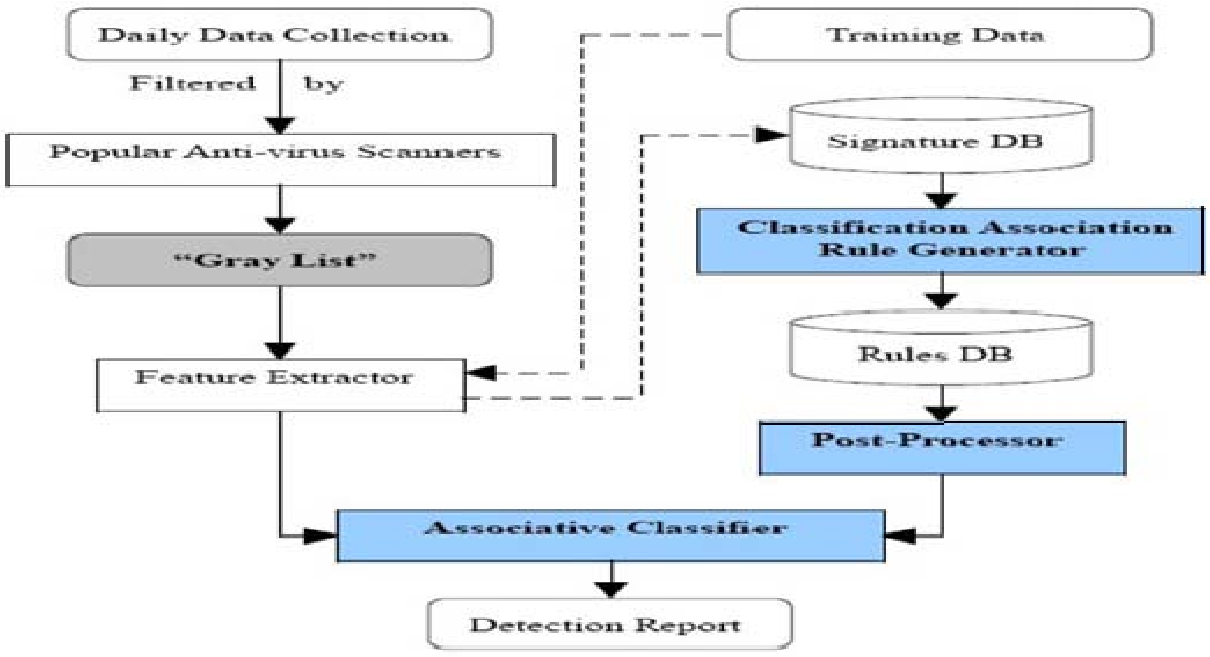


Fig. 1. Flow of malware detection

After data transformation, it then generates the classification association rules from the training Signature database. In the third step, it adapts hybrid post processing techniques of associative classification, including **rule pruning, rule ranking**, and **rule selection** to reduce the generated rules. Finally, it builds the classifier using the rules filtered by the postprocessor to detect malware from the "gray list." We will describe the details of each step in the following sections.

CLASSIFICATION ASSOCIATION RULE GENERATION

Associative classification, as a new classification approach integrating association rule mining and classification, becomes one of the significant tools for knowledge discovery and data mining. It can be effectively used in malware detection , since frequent item sets are typically of statistical significance and classifiers based on frequent pattern analysis are generally effective to test datasets. In this section, we briefly discuss the generation of rules for classification.

A. Data Collection and Transformation

We obtain 50000 Windows PE files of which 15000 are recognized as benign executables and the remaining 35000 are malicious executables. PE is designed as a common file format for all flavors of Windows operating system, and PE malicious executables are in the majority of the malware rising in recent years. All the file samples are provided by the

Antivirus Laboratory of Kingsoft Corporation, and the malicious executables mainly consist of backdoors, spyware, trojans, and worms. Based on the system architecture of our previous malware detection system IMDS , we extract the API calls as the features of the file samples and store them in the signature database.

B. Classification Association Rule Generation

For malware detection in this paper, the first goal is to find out how a set of API calls supports the specific class objectives: class1 = malicious, and class2 = benign.

1)Support and confidence: Given a dataset DB, let $I = \{I1, . . . , Im\}$ be an itemset and $I \rightarrow$ class(os, oc) be an association rule whose consequent is a class objective. The support and confidence of the rule.

2)Where the function count ($I \cup \{\text{class}\}$) returns the number of records in the dataset DB where $I \cup \{\text{class}\}$ holds.

3)Frequent item set: Given *mos* as a user-specified minimum support. *I* is a frequent item set/pattern in DB if $os \geq mos$.

4)Classification association rule: Given *moc* as a user specified confidence.

Let $I = \{I1, . . . , Im\}$ be a frequent item set. $I \rightarrow$ class(os, oc) is a classification association rule if $oc \geq moc$. Apriori and FP-Growth algorithms can be extended to associative classification. For rule generation, we use the OOA_Fast_FP-Growth algorithm proposed in to derive the complete set of the rules with certain support and confidence thresholds, since it is much faster than Apriori for mining frequent item sets. The number of the rules is also correlated to the number of the file samples.

POSTPROCESSING TECHNIQUES OF ASSOCIATIVE CLASSIFICATION FOR MALWARE DETECTION SYSTEM

The goal of our malware detection system is to build classifier using the generated rules to classify the new file samples more effectively and accurately, so the postprocessing of associative classification is very important for improving the system's ACY and efficiency.

The postprocessing techniques includes rule pruning, rule ranking, and rule selection.

A.Rule Pruning Approaches

Accompanied with the ability of mining the complete set of the rules, associative classification also has a major drawback that the number of generated rules can be really large and the removal of the redundant or misleading rules is indispensable. Besides the five common rule pruning approaches introduced:

1) $\chi 2$ (chi-square) testing tomeasure the significance of the rule itself;

2) redundant rule pruning to discard the specific rules with fewer confidence values;

3) database coverage to just keep the rules covering at least one training data object not considered by a higher ranked rule;

4) pessimistic error estimation to test the estimated error of a new rule; and

5) lazy pruning to discard the rules incorrectly classifying the training objects, we here propose another rule pruning method before building the classifier, named "insignificant rules pruning."

Since many generated rules are redundant or minor variations of others and their existence may simply be due to chance rather than true correlation these insignificant rules should be removed.

For example, given the rule: "*R1*: Job = yes → Loan = approved (supp=35%, conf=80%)," the following rule: "*R2*: Job=yes, Oversea_asset ≥ 500k→Loan = approved (supp= 32%, conf=81%)" becomes insignificant because it gives little extra information.

We use $\chi 2$ measure , which is based on the comparison of observed frequencies with the corresponding expected frequencies, to test whether the rule is significant w.r.t. to its ancestors. Given two rules generated from the training set *T* consisting of *n* data objects

R1: *A*→ *r*-class (supp = *s*1 , conf = *c*1)

R2: *AB* →*r*-class (supp = *s*2 , conf = *c*2)

where *A*, *B* are the frequent itemsets ($A \cap B = \varphi$) of the generated rules and *r*-class is the class label of *T*. If these two rules have the same class label, then we call *R1* the ancestor of *R2* (or *R2* the descendant of *R1*) [38]. If *c*1 ≥ *c*2 , namely the confidence of *R1* is not greater than its ancestor *R2*, then *R2* is insignificant and can be pruned.

If $c1 < c2$, we set up the hypothesis $H0$ that the two patterns A and B are independent. We then compute the observed and expected frequencies of $R2$ as shown in Table I. We later use $\chi 2$ measure to test the significance of the deviation from the expected values. Let $f0$ be an an observed frequency and f be an expected frequency.

The $\chi 2$ value is defined as: $\chi 2 = \_(f0 - f)/f$.

TABLE I

OBSERVED AND EXPECTED FREQUENCIES OF R2

Frequency of $R2$	Observed	Expected
Satisfying $R2$	ns_2	ns_2c_1/c_2
Dissatisfying $R2$	$ns_2(1\text{-}c_2)/c_2$	$ns_2(1\text{-}c_1)/c_2$

Given a certain threshold value (e.g., 3.84 at the 95% significance level), if the $\chi 2$ measure is above the threshold value, then we reject the hypothesis and keep $R2$, otherwise, we will accept the assumption and discard $R2$ From the aforementioned six rule pruning approaches, we empirically study four of them for malware detection: the redundant rule pruning and lazy pruning approaches are not used.

b)Rule Ranking Mechanisms

Within the associative classification framework, regardless of which particular methodology is used to generate the rules, a classifier is usually represented as an order list of the generated rules based on some rule ranking mechanisms. Many associative classification algorithms [4], [5], [15], utilize rule ranking procedures as the basis for selecting the classifier during pruning and later for predicting new data objects. As we discussed in Section II, there are five common ranking mechanisms: CSA, ACS, WRA, Laplace accuracy, and $\chi 2$ measure. Here, we give a more detailed introduction.

1) **CSA**: Based on the well-established "support-confidence" framework, CSA first sorts the original rule list based on their confidence values in a descending order. For those rules that share a common confidence value, CSA sorts them in a descending order based on the support values. CSA sorts the rules sharing common values for both confidence and support in an ascending order based on the size of the rule antecedent.

2) **ACS**: Ensuring that "specific rules have a higher precedence than more general rules" [10], ACS considers the size of the rule antecedent as the most significant factor (using a descending order) followed by the rule confidence and support values, respectively .

3) **WRA**: WRA assigns an additive weighting score to each rule to determine its expected ACY. The calculation of the value of a rule *r* is: WRA(*r*) = supp (*r*.antecedent)*(conf (*r*)-supp (*r*.consequent)) [4]. In the rule reordering stage, the original rule list is sorted based on the assigned WRA value in a descending order.

4) **Laplace accuracy**: The principle of Laplace accuracy is similar to WRA. The calculation of the Laplace value of a rule *r* is

Laplace (*r*) =(supp (*r*.antecedent $\cup$ *r*.consequent) + 1)(supp (*r*.antecedent) + *c*)

where *c* represents the number of predefined classes.

5)***χ*2 measure**: In associative classification algorithms, if the *χ*2 measure between two variables (the antecedent and consequent-class of the generated rule) is higher than a certain threshold value, we can conclude that there might be a relation between the rule antecedent and consequent-class, otherwise, it implies that the two variables may be statistically independent. We can order the list of the generated rules in a descending order based on their *χ*2 values. For the aforementioned five rule ranking mechanisms, we empirically study all of them for building the classifier and later for detecting the new malware.

TABLE II ADAPTING POSTPROCESSING TECHNIQUES OF ASSOCIATIVE CLASSIFICATION FOR MALWARE DETECTION SYSTEM

Pruning	Ranking	Selection
χ^2 testing	CSA	Best first
Database coverage	ACS	All
Pessimistic error estimation	WRA	Best k
Insignificant rule pruning	Laplace Accuracy	
	χ^2 measure	

c)Rule Selection Methods

After building the classifier by the techniques of rule pruning and rule ranking, we can select the subset of the rules from the classifier to predict the new file samples. As stated in Section II, there are three common rule selection approaches: best first rule, all rules, and best *k* rules. For our malware detection system, we will also try all of these methods to predict the new file samples and find the best way for malware detection. The postprocessing techniques, which will be perform in malware detection, can be summarized in Table II.

PROPOSED WORK

In the proposed dissertation work intelligent malware detection system will be implemented. The dissertation work will be carried out as follows.

1. Analysis of available malware detection systems.
2. Evaluation of how these systems complement each other to improve detection rates.
3. Implementation of malware detection system for detection of denial of service and backdoor.
4. Analysis of malware detection results.

EMPIRICAL STUDY OF POSTPROCESSING TECHNIQUES FOR MALWARE DETECTION

A. Experiment Setup

We randomly select 17828 executables from our data collection, including 9721 benign executables, 2255 backdoors, 2245 spyware, 3200 Trojans, and 2021 worms in the training dataset. The rest 17828 executables are used for testing purpose of which 7700 are benign files and 2021 are malicious ones. After filtering some of the worthless API calls, we finally extract 5102 API calls from the training dataset. By using the OOA_Fast_FP-Growth algorithm [35], [36], we generate 31 rules with the minimum support and confidence as 0.18 and 0.5, respectively for the benign class, while 8424 rules are derived with the minimum support and confidence as 0.25 and 0.7, respectively for the malicious class.

To systematically evaluate the effects of postprocessing techniques for malware detection, we conduct the following three sets of experimental studies using our collected data obtained from the Antivirus Laboratory of Kingsoft Corporation. The first set of study is to compare the ACY and efficiency of the three different associative classifier building algorithms: CBA, CMAR, and CPAR [37], when used for malware detection system. Since none of the three algorithms adopt the insignificant rule pruning approach, in the second set of study, we prune the insignificant rules before building the classifier. From these two sets of studies, we will choose the best rule pruning and rule selection methods for malware detection. In third set of experiments, we will compare the five rule ranking mechanisms and find the best ranking method for malware detection. Please note in all the experiments, rule mining, selection, and ranking are performed only within training data. From the three set of experiments, we will propose an effective classifier building method and incorporate it to our improved malware detection system IMDS.

B. Comparisons of CBA, CMAR, and CPAR for Malware Detection

Since the algorithms of CBA, CMAR, and CPAR have been successfully used in associative classification and represent different kinds of postprocessing techniques for building the classifiers, in the first set of experiments, we use them for malware detection and compare their ACY and efficiency. The postprocessing techniques adopted by these three algorithms are listed in Table III.

The datasets described in Section VI-A are used for training and testing. In this paper, we use DR and ACY defined as follows to evaluate each classifier building method.

1) True positive (TP): The number of executables correctly classified as malicious code.

2) True negative (TN): The number of executables correctly classified as benign executables.

3) False positive (FP): The number of executables mistakenly classified as malicious executables.

4) False negtive (FN): The number of executables mistakenly classified as benign executables.

5) DR: *TP/*(*TP* + *FN*).

6) ACY: *TP* + *TN/*(*TP* + *TN* + *FP* + *FN*).

TABLE IV

RESULTS OF CBA, CMAR, AND CPAR CLASSIFIER BUILDING METHOD USED IN MALWARE DETECTION SYSTEM

		Training Set		Testing Set	
Algs.	Used rules	DR %	ACY %	DR %	ACY %
CBA	**21**	**81.5591**	**67.1230**	**79.2140**	**64.1319**
CMAR	619	65.9488	64.4377	62.7397	57.8989
CPAR	8,455	62.7461	62.7779	62.2569	62.3133

Remak: "DR" indicates detection rate and "ACY" indicates accuracy.

The experimental results shown in Table IV indicate that CBA classifier building method performs better than the other two for malware detection.

Comparisons of Different Rule Ranking Mechanisms

In this section, we compare the five rule ranking mechanisms and find the best one for malware detection. Results in Table V illustrate that $\chi 2$ measure rule ranking mechanism performs best.

TABLE V

RESULTS BY USING DIFFERENT RULE RANKING MECHANISMS IN MALWARE DETECTION SYSTEM

		Training	Set	Testing	Set
Algs.	Used rules	DR %	ACY %	DR %	ACY %
CSA	5	85.2448	68.083	83.7476	65.5935
ACS	13	76.1482	64.1485	74.0926	60.5175
WRA	1	70.9056	65.0153	67.0458	57.8313
Lapl	5	85.2448	68.083	83.7476	65.5935
χ^2	3	**89.5245**	**71.3484**	**88.1639**	**67.5049**

Comparisons of Intelligent Malware Detection System (IMDS) With Other Systems

In this section, we conduct two sets of experiments to compare our IMDS system with other malware detection system: 1) the first set of experiments is to examine the abilities of detecting the malware from the "gray list" of our IMDS system, in comparison with some of the popular software tools, such as McAfee Virus Scan, Norton AntiVirus, Dr.Web, and Kaspersky AntiVirus. We use fair versions of the base of signature on the same day (17 May, 2012) for testing. The efficiency by using different scanners have also been examined. 2) In the second set of experiments, resting on the analysis of API calls, we compare our malware detection system IMDS with other classification based methods and Decision tree.

Comparisons of Detection Results From the Gray List

Since the goal of our improved malware detection system is to help our virus analysts picking up as many malware samples as possible from the "gray list," which consists of millions of executables, we perform the experiments based on the "gray list" in this section. We randomly select 17828 file samples from the "gray list." Table VII shows the detection results of different antivirus scanners. Of the 17828 samples from the "gray list," 4572 are detected as malware by the five scanners in total. These detected results of the scanners should be reviewed by our virus analysts, since they have false positive rate (FPR). The FPR of each scanner results from the disability of recognizing the benign software adopting obfuscation technique in clients, such as the instant message (IM) software “QQ”. Our virus analysts perform analysis on the detected files and find that 3015 of them are correctly detected. We then calculate the DR and ACY for each scanner. The statistical results are shown in Table V. From Tables VI and VII , we can see that our IMDS system outperform other antivirus software for malware detection from the “gray list.”

TABLE VI

Detection Result From The “GRAYLIST”:-

GRAYLIST	MacAF	NorAV	DrWeb	KaSky	IMDS
SAMPLE 1	M	--	--	--	M
SAMPLE 2	--	--	--	M	M
SAMPLE 3	--	M	--	M	M
SAMPLE 4	--	--	--	--	--
SAMPLE 5	--	--	--	--	--
SAMPLE 6	M	--	M	--	M
SAMPLE 7	--	M	--	M	--
SAMPLE 8	--	--	--	--	M
SAMPLE 9	--	--	--	--	--
SAMPLE 10	M	M	--	M	M
-----	--	--	--	--	M
-----	--	--	--	--	--
-----	--	--	--	--	--
-----	--	--	--	M	--
-----	--	--	--	--	M
-----	--	M	--	--	--
-----	--	--	--	--	M
-----	--	--	--	--	--
-----	--	--	--	--	--
	--	--	--	--	--
SAMPLE 17828	--	--	--	--	--
STAT	4918	6692	3718	6462	9721
TP	4448	5683	2679	5494	9721
FPR	09.55%	15.07%	27.86%	14.97%	0%

IMDS => Intelligent Malware Detection System

M => Indicates the file in the "GRAYLIST" is detected as MALWARE.

STAT => Total Numbers of file detected as malware.

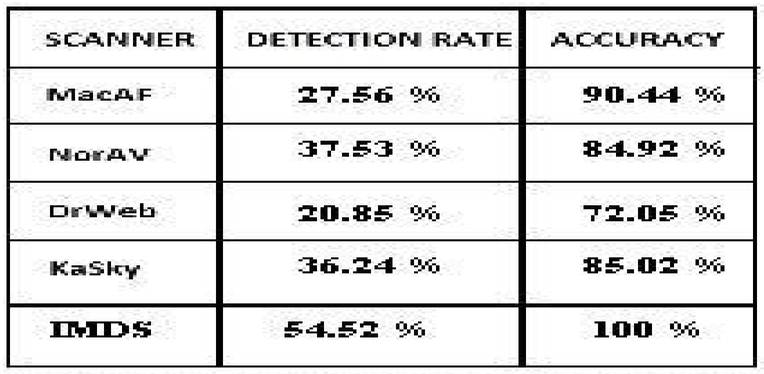
TP => Numbers of correctly detected file.

FPR => Mistakenly classified as MALWARE.(FALSE POSITIVE RATE)

-- => Scanner default files

TABLE VII

STATISTICAL RESULTS OF DR AND ACY



SCANNER	DETECTION RATE	ACCURACY
MacAF	27.56 %	90.44 %
NorAV	37.53 %	84.92 %
DrWeb	20.85 %	72.05 %
KaSky	36.24 %	85.02 %
IMDS	54.52 %	100 %

STATISTICAL RESULTS OF DR AND ACY

APPLICATION:

Many instances of malware take advantage of features provided by common applications, such as e-mail clients, Web browsers, and word processors. By default, applications often are configured to favor functionality over security. Accordingly, organizations should consider disabling unneeded features and capabilities from applications, particularly those that are commonly exploited by malware, to limit the possible application attack vectors for malware. Organizations should also consider identifying applications that are typical malware propagation methods (e.g., Web browsers, e-mail clients and servers) and configuring them to filter content and stop other activity that is likely to be malicious. Spam is often used for phishing and spyware delivery (e.g., Web bugs often are contained within spam), and it sometimes contains other types of malware. Using spam filtering software on e-mail servers or clients or on network-based appliances can significantly reduce the amount of spam that reaches users, leading to a corresponding decline in spam-triggered malware incidents.

CONCLUSION:

We systematically evaluate the effects of the post processing techniques (e.g., rule pruning, rule ranking, and rule selection) of associative classification in malware detection and propose an effective way, i.e., CIDCPF, to detect the malware from the “gray list.” We are using post processing techniques of associative Classification in malware detection. Experiments on a large real data collection from Antivirus Laboratory at Kingsoft Corporation demonstrate among the most common and popular associative classification building methods, our CIDCPF method achieves better performance on detection ability and efficiency because of its concise, but effective classifier. In addition, our IMDS system, which adopts CIDCPF method for building classifiers can greatly reduce the number of generated rules and make it easy for our virus analysts to identify the useful ones.

REFERENCES:

[1] M. Antonie and O. Zaiane, "An associative classifier based on positive and negative rules," in *Proc. 9th ACM SIGMOD Workshop Res. Issues Data Mining Knowl. Discovery*, 2004, pp. 64–69.

[2] D. Brumley, C. Hartwig, M. G. Kang, Z. Liang, J. Newsome, D. Song, and H. Yin. BitScope: Automatically dissecting malicious binaries. Technical Report CMU-CS-07-133, School of Computer Science, Carnegie Mellon University, March 2007.

[3] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin. Botnet Analysis, chapter Automatically Identifying Trigger-based Behavior in Malware. 2007.

[4] M. Costa, J. Crowcroft, M. Castro, A. Rowstron, L. Zhou, L. Zhang, and P. Barham. Vigilante: End-to-end containment of internet worms. In Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP'05), October 2005.

[5] J. R. Crandall and F. T. Chong. Minos: Control data attack prevention orthogonal to memory model. In Proceedings of the 37th International Symposium on Microarchitecture (MICRO'04), December 2004.

[6] M. Egele, C. Kruegel, E. Kirda, H. Yin, and D. Song. Dynamic Spyware Analysis. In Proceedings of the 2007 Usenix Annual Conference (Usenix'07), June 2007.

[7] P. Ferrie. Attacks on virtual machine emulators. Symantec Security Response, December 2006.

[8] H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," in *Proc. ICDE-2008*, pp. 169–178.

[9] H. Cheng, X. Yan, J. Han, and C. Hsu, "Discriminative frequent pattern analysis for effective classification," in *Proc. ICDE-2007*, pp. 716–725.

[10] M. Christodorescu, S. Jha, and C Kruegel, "Mining specifications of malicious behavior," in *Proc. ESEC/FSE-2007*, pp. 5–14.

[11] F. Coenen and P. Leng, "An evaluation of approaches to classification rule selection," in *Proc. 4th IEEE Int. Conf. Data Mining 2004*, pp. 359–362.

[12] X. Jiang and X. Zhu, "vEye: Behavioral footprinting for self-propagating worm detection and profiling," *Knowl. Inf. Syst.*, vol. 18, no. 2, pp. 231– 262, 2009.