

CIMDS: Adapting Postprocessing Techniques of Associative Classification for Malware Detection

Yanfang Ye, Tao Li, Qingshan Jiang, and Youyu Wang

Abstract—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). Nowadays, numerous attacks made by the malware pose a major security threat to computer users. Unfortunately, along with the development of the malware writing techniques, the number of file samples that need to be analyzed, named “gray list,” on a daily basis is constantly increasing. In order to help our virus analysts, quickly and efficiently pick out the malicious executables from the “gray list,” an automatic and robust tool to analyze and classify the file samples is needed. In our previous work, we have developed an intelligent malware detection system (IMDS) by adopting associative classification method based on the analysis of application programming interface (API) execution calls. Despite its good performance in malware detection, IMDS still faces the following two challenges: 1) handling the large set of the generated rules to build the classifier; and 2) finding effective rules for classifying new file samples. In this paper, we first systematically evaluate the effects of the postprocessing techniques (e.g., rule pruning, rule ranking, and rule selection) of associative classification in malware detection, and then, propose an effective way, i.e., CIDCPF, to detect the malware from the “gray list.” To the best of our knowledge, this is the first effort on using postprocessing techniques of associative classification in malware detection. CIDCPF adapts the postprocessing techniques as follows: first applying Chi-square testing and insignificant rule pruning followed by using Database coverage based on the Chi-square measure rule ranking mechanism and Pessimistic error estimation, and finally performing prediction by selecting the best First rule. We have incorporated the CIDCPF method into our existing IMDS system, and we call the new system as CIMDS system. Case studies are performed on the large collection of file samples obtained from the Antivirus Laboratory at Kingsoft Corporation and promising experimental results demonstrate that the efficiency and ability of detecting malware from the “gray list” of our CIMDS system outperform popular antivirus software tools, such as McAfee VirusScan and Norton AntiVirus, as well as previous data-mining-based detection systems, which employed Naive Bayes, support vector machine, and decision tree techniques. In particular, our CIMDS system can greatly reduce the number of generated rules, which makes it easy for our virus analysts to identify the useful ones.

Index Terms—Associative classification, malware detection, postprocessing, rule pruning, rule ranking, rule selection.

I. INTRODUCTION

A. Malware Detection

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) [29]. 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 [24]. 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 [19]. According to the virus analysts at Kingsoft Antivirus Laboratory, the “gray list” that is needed to be analyzed per day usually contain more than 70 000 file samples. Clearly, there is a need for an automatic, efficient, and robust tool to classify the “gray list.”

So far, several data mining and machine-learning approaches have been used in malware detection [9], [12], [13], [19], [21], [24], [30], [35], [36]. In our previous paper [35], [36], since frequent itemsets found by association mining represent the underlying profiles (of application programming interface (API) function calls) of malware and benign files, we developed an intelligent malware detection system (IMDS) adopting associative classification method based on the analysis of API calls. We compared our IMDS with other classifiers developed in the previous studies [13], [21], [30] and the results show that our IMDS applying associative classification method outperforms other classification approaches in both detection rate (DR) and accuracy (ACY) [35], [36]. Despite its good performance, IMDS still faces the following two challenges for improving the ACY and efficiency of malware detection: 1) handling the large set of the generated rules to build the classifier; and 2) finding effective rules for classifying new file samples.

Manuscript received November 24, 2008; revised May 6, 2009 and August 11, 2009. First published February 17, 2010; current version published April 14, 2010. The work of Y. Ye and Q. Jiang was supported by the National Science Foundation of China under Grant 10771176 and Guangdong Province Foundation under Grant 2008A090300017. The work of T. Li was supported in part by the U.S. National Science Foundation under Grant IIS-0546280 and in part by multiple IBM Faculty Awards. This paper was recommended by Associate Editor J. Wang.

Y. Ye is with the Department of Computer Science, Xiamen University, Xiamen, 361005, China (e-mail: yeyanfang@yahoo.com.cn).

T. Li is with the School of Computer Science, Florida International University, Miami, FL 33199, USA (e-mail: taoli@cs.fiu.edu).

Q. Jiang and Y. Wang are with the Software School, Xiamen University, Xiamen, 361005, China (e-mail: qjiang@xmu.edu.cn; youyu.wang@gmail.com).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCC.2009.2037978

Recently, many postprocessing techniques [26], 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], [33]. It is interesting to know how these post-processing techniques would help the associative classifiers in our IMDS system for malware detection. In this paper, we adapt different postprocessing techniques of associative classification for improving the ACY and efficiency of the malware detection system. To the best of our knowledge, this is the first paper using postprocessing techniques of associative classification in malware detection. In particular, we systematically evaluate the effects of the postprocessing techniques in malware detection and propose an effective way, i.e., CIDCPF, to detect the malware from the “gray list.” CIDCPF adapts the postprocessing techniques as follows: first applying chi-square testing and insignificant rule pruning followed by using database coverage based on the chi-square measure rule ranking mechanism and pessimistic error estimation, and finally predicting the new file sample by the way of selecting the best first rule. We have incorporated the CIDCPF method into our existing IMDS system, and we call the new system as CIMDS system. Case studies are performed on the large collection of file samples obtained from the Antivirus Laboratory at Kingsoft Corporation and promising experimental results demonstrate that the efficiency and ability of detecting malware from the “gray list” of our CIMDS system outperform popular antivirus software, such as McAfee VirusScan and Norton AntiVirus, as well as previous data-mining-based detection systems, which employed Naive Bayes, support vector machine (SVM), and decision tree techniques. In particular, our CIMDS system can greatly reduce the number of generated rules, which makes it easy for our virus analysts to identify the useful ones.

B. Our Contributions

Our main contributions are summarized below: 1) We systematically evaluate the effects of the postprocessing techniques in malware detection. 2) We propose an effective way CIDCPF, to detect the malware from the “gray list.” CIDCPF adapts several different postprocessing techniques of associative classification, including rule pruning, rule ranking, and rule selection, for building effective associative classifiers. 3) We improve our former malware detection system IMDS and update it to CIMDS. 4) We perform cases studies on a large collection of executables including 35 000 malicious ones and 15 000 benign samples, collected by the Antivirus Laboratory of Kingsoft Corporation. 5) We provide a comprehensive experimental study on various antivirus software as well as various data mining techniques for malware detection using our data collection.

The rest of this paper is organized as follows. The related work is discussed in Section II. Section III presents the system architecture and Section IV devotes to the generation of rules for classification. In Section V, we discuss different postprocessing techniques of associative classification, including rule pruning, rule ranking, and rule selection. In Section VI, we systematically evaluate the effects of the postprocessing techniques in

malware detection and propose an effective way to detect the malware from the “gray list.” In Section VII, we compare our CIMDS system with popular antivirus software tools, such as McAfee VirusScan and Norton AntiVirus, as well as previous data-mining-based detection systems, which employed Naive Bayes, SVM, and decision tree techniques. Finally, Section VIII is concluded.

II. RELATED WORK

In order to overcome the disadvantages of the widely used signature-based malware detection method, data mining and machine-learning approaches are proposed for malware detection [9], [13], [21], [24], [30], [35], [36]. Naive Bayes method, SVM, and decision tree classifiers are used to detect new malicious executables in previous studies [13], [21], [30]. 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 [3], [7], [15], [16], [27], [31], [37]. Due to the fact that frequent itemsets (sets of API calls) discovered by association mining can well represent the underlying semantics (profiles) of malware and benign file datasets, associative classification has been successfully used in the IMDS system developed in [35] and [36] for malware detection. However, there is often a huge number of rules generated in a classification association rule mining practice [26], [28]. It is often infeasible to build a classifier using all of the generated rules. Hence, how to reduce the number of the rules and select the effective ones for prediction is very important for improving the classifier’s ACY and efficiency. Recently, many postprocessing 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], [33]. It is interesting to know how these postprocessing techniques would help the associative classifiers for malware detection. In this paper, we systematically evaluate the effects of the postprocessing 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 [6], [11], [14], [18], [20], [23], 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 [22]: 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) [15] 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) [16], and used by CMAR [15], and multiclass classification based on association rule (MCAR) [25]. 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 [16] and [32] 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) [27], 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 [33]: a) confidence support size of antecedent (CSA); b) size of antecedent confidence support (ACS); c) weighted relative accuracy (WRA); d) Laplace accuracy; and e) χ^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 χ^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 [33]: 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 χ^2 (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.

In this paper, we try to make use of each postprocessing technique and find a proper method to build the classifier for our malware detection system.

III. SYSTEM ARCHITECTURE

The system architecture of our malware detection is summarized 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. After data transformation, it then

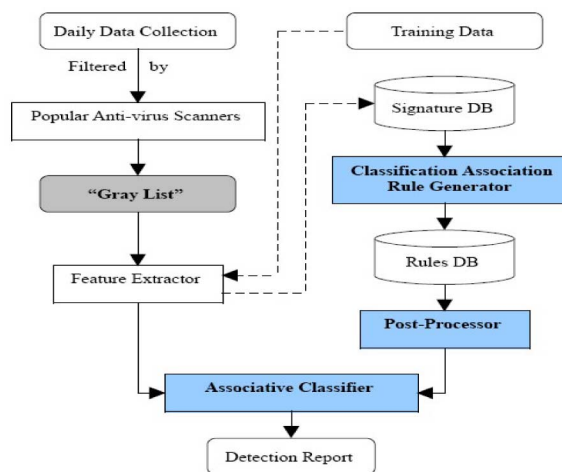


Fig. 1. Flow of malware detection.

generates the classification association rules from the training signature database. In the third step, it adapts hybrid postprocessing 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.

IV. 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 [35], [36], since frequent itemsets are typically of statistical significance and classifiers based on frequent pattern analysis are generally effective to test datasets [8]. In this section, we briefly discuss the generation of rules for classification.

A. Data Collection and Transformation

We obtain 50 000 Windows PE files of which 15 000 are recognized as benign executables and the remaining 35 000 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 [35], we extract the API calls as the features of the file samples and store them in the signature database. As shown in Fig. 2, there are six fields in the signature database, which are record ID, PE file name, file type (“0” represents benign file, while “1” is for malicious file), called APIs name, called API ID and the total number of called API functions. The transaction data can also be easily converted to relational data if necessary. Now the data is ready for classification association rule generation.

id	filename	filesort	apiseq	intvectorofapi	apifunno
9198	Adware. Casino. N_27e	1	wsock32.dll, wsock32	5, 7, 10, 12, 13,	158
9199	Adware. DropSpam_29b	1	ws2_32.dll, wsasockl	1, 7, 9, 14, 15, 1	145
9201	Adware. NaviPromo. Gen	1	kernel32.dll, getcoi	6, 15, 18, 43, 45	124
9202	Backdoor. AFCore. Drop	1	kernel32.dll, exit	2, 44, 45, 81, 13	29
9203	Backdoor. Agent. AAS_	1	kernel32.dll, getpr	1, 5, 7, 8, 10, 13	98
9209	Backdoor. Generic. 121	1	kernel32.dll, rtlzei	2, 13, 39, 53, 5	74
9210	Backdoor. Hupigon. 115	1	kernel32.dll, findc	1, 5, 7, 8, 10, 11	205
9211	Backdoor. Hupigon. 115	1	kernel32.dll, findc	1, 5, 7, 8, 10, 11	205
9212	Backdoor. Hupigon. 115	1	kernel32.dll, findc	1, 5, 7, 8, 10, 11	205
9231	BehavesLike. Trojan. F	1	kernel32.dll, globa	1, 3, 4, 5, 6, 8, 1	232
9232	BehavesLike. Win32. Ex	1	kernel32.dll, tlssc	38, 39, 40, 41, 4	57
9233	BehavesLike. Win32. Ex	1	kernel32.dll, tlssc	38, 39, 40, 41, 4	57

Fig. 2. Sample data 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: $class_1 = \text{malicious}$, and $class_2 = \text{benign}$.

- 1) Support and confidence: Given a dataset DB, let $I = \{I_1, \dots, I_m\}$ be an itemset and $I \rightarrow \text{class}(os, oc)$ be an association rule whose consequent is a class objective. The support and confidence of the rule are defined as follows:

$$os = \text{supp}(I, \text{class}) = \frac{\text{count}(I \cup \{\text{class}\})}{|DB|} \times 100\%$$

$$oc = \text{conf}(I, \text{class}) = \frac{\text{count}(I \cup \{\text{class}\})}{\text{count}(I, DB)} \times 100\%$$

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

- 2) Frequent itemset: Given mos as a user-specified minimum support. I is a frequent itemset/pattern in DB if $os \geq mos$.
- 3) Classification association rule: Given moc as a user-specified confidence. Let $I = \{I_1, \dots, I_m\}$ be a frequent itemset. $I \rightarrow \text{class}(os, oc)$ is a classification association rule if $oc \geq moc$.

Apriori and FP-Growth algorithms can be extended to associative classification [15], [16]. For rule generation, we use the OOA_Fast_FP-Growth algorithm proposed in [35] to derive the complete set of the rules with certain support and confidence thresholds, since it is much faster than Apriori for mining frequent itemsets. The number of the rules is also correlated to the number of the file samples.

As an example, we sample 35 000 records as the training dataset of which 21 000 records are malicious executables and 14 000 records are benign executables. Based on the label dataset, we generate 8424 rules with the minimum support and confidence as 0.25 and 0.7, respectively for the malicious class, while just 31 rules are derived with the minimum support and confidence as 0.18 and 0.5, respectively for the benign class (in our paper, we set these thresholds after discussion with the virus analysts). Part of the generated rules is shown in Fig. 3. Take

Id	Rules	Os	Oc
4548	177,144,102,58,56,	0.259602	0.746108
4549	154,144,129,56,51,	0.259602	0.800243
4550	225,144,58,56,51,	0.259602	0.771889
4551	593,173,144,130,129,	0.259602	0.948201
4552	177,154,144,129,51,	0.259504	0.798001
4553	225,144,56,51,6,	0.259504	0.772501
4554	225,58,56,51,6,	0.259504	0.765543
4555	144,137,111,58,6,	0.259504	0.718963
4556	239,177,173,156,149,	0.259405	0.906401
4557	239,173,156,149,144,	0.259405	0.90765
4558	239,237,149,144,135,	0.259405	0.948847
4559	238,177,173,156,149,	0.259405	0.906401
4560	238,173,156,149,144,	0.259405	0.90765
4561	238,237,149,144,135,	0.259405	0.948847
4562	239,238,173,156,149,	0.259405	0.906401
4563	206,177,176,173,130,	0.259405	0.871608
4564	177,173,102,58,6,	0.259405	0.776533
4565	177,173,156,154,144,	0.259405	0.789332
4566	177,173,154,58,51,	0.259405	0.815227
4567	225,177,58,51,6,	0.259405	0.755161
4568	177,173,154,56,51,	0.259405	0.81903
4569	593,177,147,130,123,	0.259405	0.954694
4570	177,144,140,129,100,	0.259405	0.863041
4571	147,144,140,129,100,	0.259405	0.861347
4572	206,173,147,144,129,	0.259307	0.822555
4573	239,177,173,154,144,	0.259307	0.782234
4574	238,177,173,154,144,	0.259307	0.782234
4575	239,238,173,154,144,	0.259307	0.78177
4576	154,129,58,51,6,	0.259307	0.804461
4577	225,177,173,129,6,	0.259307	0.793311

9455 rows fetched in 0.2935s (0.0511s)

Fig. 3. Generated classification association ruleset.

one of the generated rules, for example,

$$R1: (219, 181, 180, 120, 19) \rightarrow \text{class} = \text{malicious}(os = 0.26, oc = 1)$$

where os and oc represent the support and confidence, respectively. After converting the API IDs to API names, the rule becomes

$$(\text{ADVAPI32.DLL,Createservicea}; \text{USER32.DLL,Callnexthookex}; \text{KERNEL32.DLL,Openprocess}; \text{Createtoolhelp32snapshot}; \text{Process32first}) \rightarrow \text{class} = \text{malicious}(os = 0.26, oc = 1).$$

With the os and oc values, we know that this set of API calls appears in 9100 malware samples and not in any benign files. Rules like $R1$ with both high support and confidence can be used for determining a new file sample is malicious or not. For instance, if a new file sample includes the following API calls: ADVAPI32.DLL, Createservicea; USER32.DLL, Callnexthookex; KERNEL32.DLL, Openprocess; KERNEL32.DLL, Createtoolhelp32snapshot; and KERNEL32.DLL, Process32first; then it can be predicted as malicious.

In fact, many classification methods (such as associative classifiers) have been developed to construct classifiers based on association rules [3], [7], [15], [16], [27], [31], [37]. In our paper, based on the complete set of the rules generated by the malware detection rule generator, IMDS applied the technique of CBA-CB [16] to build a classifier as the malware detection module to predict new file samples [35]. Though it achieved good performance, it is interesting to know how postprocessing techniques, including rule pruning, rule ranking, and rule selection, would help the associative classifiers in the IMDS system for malware detection. Note that the associative classifier is a supervised

method that generates the rules to capture the characteristics of file samples based on known malicious and benign samples.

V. 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 in Section II: 1) χ^2 (chi-square) testing to measure 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 [17], [38], these insignificant rules should be removed.

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

We use χ^2 measure [22], 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

$$R1: A \rightarrow r\text{-class (supp} = s_1, \text{conf} = c_1)$$

$$R2: AB \rightarrow r\text{-class (supp} = s_2, \text{conf} = c_2)$$

where A, B are the frequent itemsets ($A \cap B = \phi$) 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 $c_1 \geq c_2$, namely the confidence of $R1$ is not greater than its ancestor $R2$, then $R2$ is insignificant and can be pruned.

If $c_1 < c_2$, we set up the hypothesis H_0 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 χ^2 measure to test the significance of the deviation from the expected values. Let f_0 be an observed frequency and f be an expected frequency. The χ^2 value is defined as: $\chi^2 = \sum (f_0 - f)^2 / f$. Given a certain threshold value (e.g., 3.84 at the

TABLE I
OBSERVED AND EXPECTED FREQUENCIES OF $R2$

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

95% significance level [22]), if the χ^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. The reasons are 1) our proposed insignificant rule pruning approach actually includes redundant rule pruning; and 2) the lazy pruning method may lead to very large classifier [26], which would not fit for the malware detection system since it is time sensitive.

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 [33]. Many associative classification algorithms [4], [5], [15], [16], [25], [27] 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 [33]: CSA, ACS, WRA, Laplace accuracy, and χ^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 [33].

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: $WRA(r) = \text{supp}(r.\text{antecedent}) * (\text{conf}(r) - \text{supp}(r.\text{consequent}))$ [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

$$\text{Laplace}(r) = \frac{(\text{supp}(r.\text{antecedent} \cup r.\text{consequent}) + 1)}{(\text{supp}(r.\text{antecedent}) + c)}$$

where c represents the number of predefined classes [33].

5) χ^2 measure: In associative classification algorithms, if the χ^2 measure between two variables (the antecedent and

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	

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.

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 [33]: 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.

In summary, the postprocessing techniques, which will be studied in malware detection, can be summarized in Table II.

VI. EMPIRICAL STUDY OF POSTPROCESSING TECHNIQUES FOR MALWARE DETECTION

A. Experiment Setup

We randomly select 35 000 executables from our data collection, including 14 000 benign executables, 5255 backdoors, 5245 spyware, 5200 trojans, and 5300 worms in the training dataset. The rest 15 000 executables are used for testing purpose of which 6000 are benign files and 9000 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.¹

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,

¹In our paper, we set the thresholds used in rule mining and rule selection after discussion with the virus analysts. The thresholds are not sensitive to the system, since we adopt the proposed postprocessing techniques to prune and rank the generated rules.

TABLE III
POSTPROCESSING TECHNIQUES ADOPTED BY CBA, CMAR, AND CPAR

Algs.	Pruning	Ranking	Selection
CBA	Database coverage, Pessimistic error estimation	CSA	Best first rule
CMAR	χ^2 testing, Database coverage, Redundant rule pruning	CSA	All rules
CPAR	—	Laplace Accuracy	Best k rule

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 CIMDS. All the experimental studies are conducted under the environment of Windows XP operating system plus Intel P4 1.83 GHz CPU and 1 GB of RAM.

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 negative (FN): The number of executables mistakenly classified as benign executables.
- 5) DR: $TP / (TP + FN)$.
- 6) ACY: $TP + TN / (TP + TN + FP + FN)$.

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

C. Insignificant Rule Pruning

From Table III, we observe that none of the three algorithms adopts the insignificant rule pruning approach. In this set of

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

Algs.	Used rules	Training Set		Testing Set	
		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.

TABLE V
RESULTS BY USING INSIGNIFICANT RULE PRUNING APPROACH FOR BUILDING
THE CLASSIFIER

Algs.	Used rules	Training Set		Testing Set	
		DR %	ACY %	DR %	ACY %
CBA	5	85.2448	68.083	83.7476	65.5935
CMAR	540	65.8642	64.3195	62.4535	57.5434
CPAR	2,548	62.7461	62.7816	62.2492	62.3183

TABLE VI
RESULTS BY USING DIFFERENT RULE RANKING MECHANISMS IN MALWARE
DETECTION SYSTEM

Algs.	Used rules	Training Set		Testing Set	
		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

experiments, we prune the insignificant rules before building the classifier. From Tables IV and V, the comparisons illustrate that no matter, which algorithm we adopt, the number of rules selected for building the classifier decrease sharply by using insignificant rule pruning approach, while the DR and ACY of the classifiers remain the same or even slightly increase. In addition, the results in Table V illustrate that CBA classifier building method still achieves better performance than the other two for malware detection.

From these two set of experiments, we can conclude that: 1) for rule pruning, using all of the approaches shown in Table II can achieve better performance; and 2) for rule selection, compared with the other two, best first rule selection approach performs best on our dataset. Since the classification rules are unevenly distributed, multiple rules-based prediction may not suitable for our dataset. Thus, in the following experiments, we use χ^2 test, insignificant rule pruning, database coverage, and pessimistic error estimation approaches to prune the generated rules before building the classifier. For prediction, we will adopt the best first rule selection approach to detect the malware.

D. 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 VI illustrate that χ^2 measure rule ranking mechanism performs best.

E. CIDCPF Method

From the aforementioned three set of experiments, we find that building classifier for malware detection by the following way can achieve better performance: applying chi-square testing and insignificant rule pruning followed by database coverage based on the chi-square measure rule ranking mechanism and pessimistic error estimation, and predicting the new file sample by the way of selecting the best first rule. We call this method CIDCPF and incorporate it into our malware detection system CIMDS. As shown in Table VI, the last row is exactly the result of CIDCPF. From Tables IV–VI, we can see that CIDCPF outperforms other classifier building methods on malware detection. In addition, by adapting the proposed postprocessing approach for CIDCPF, we can greatly reduce the number of generated rules. The concise but effective classifier makes our CIDCPF method achieving better performance than others both in detection ability and efficiency. The three rules used in our CIMDS system are listed as follows.

- 1) (Kernel32.dll,GetSystemDirectorya; Kernel32.dll, CreateThread;) \rightarrow class_malicious ($os = 0.345677, oc = 0.933511$).
- 2) (Kernel32.dll,CreateFilea; Kernel32.dll,WriteFile; Kernel32.dll,GetSystemDirectorya;) \rightarrow class_malicious ($os = 0.35779, oc = 0.910983$).
- 3) (Kernel32.dll,CreateFilea; Kernel32.dll,DeleteFilea; Kernel32.dll,CreateThread; Wininet.dll,InternetOpena;) \rightarrow class_malicious ($os = 0.251330, oc = 0.992224$).

Take the third rule for example, the set of those API calls appears in 3770 malware samples while 29 benign files. These rules are very useful for our virus analysts to identify and interpret. By analyzing the API calls in this rule, our virus analysts can conclude that the programs always executing the following functions can be recognized as malware:

- 1) returns a handle that can be used to access the object;
- 2) deletes an existing file from a file system;
- 3) creates a thread to execute within the address space of the calling process;
- 4) initializes an application's use of the WinInet functions.

VII. EXPERIMENT COMPARISONS OF CIMDS WITH OTHER SYSTEMS

In this section, we conduct two sets of experiments to compare our CIMDS 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 CIMDS system, in comparison with some of the popular software tools, such as McAfee VirusScan, Norton AntiVirus, Dr. Web, and Kaspersky AntiVirus. We use fair versions of the base of signature on the same day (19 May, 2008) 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 CIMDS with other classification-based methods, including SVM, Naive Bayes, and Decision tree.

TABLE VII
DETECTION RESULTS FROM THE “GRAY LIST”

Graylist	MacAf	NorAV	DrWeb	KaSky	CIMDS
Sample1	M	-	-	M	M
Sample2	-	-	-	-	-
Sample3	-	M	-	-	-
Sample4	-	M	?	M	M
Sample5	M	-	-	-	M
Sample6	-	-	M	-	-
Sample7	-	-	-	-	M
Sample8	-	-	-	-	M
Sample9	-	-	-	-	-
Sample10	-	-	-	-	M
Sample11	-	-	-	-	-
Sample12	-	-	-	M	-
Sample13	-	-	-	-	M
Sample14	?	M	-	-	M
Sample15	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮
Sample10,000	-	M	-	M	M
Stat.	920	1,320	873	1,280	2,621
TP.	832	1,121	629	1,093	2,307
FPR.	9.57%	15.08%	27.95%	14.61%	11.98%

Remark: MacAf: MacAfee, NorAV: Norton AntiVirus, DrWeb: Dr.Web, KaSky: Kaspersky, Stat.: the total number of the files detected as malware, TP.: the number of the correctly detected files, and FPR: false positive rate, which is rate of mistakenly classified as malware. In the table, “M” indicates the file in the “gray list” is detected as malware. “-” indicates the scanner defaults the file as benign, and “?” represents only an “alert”, all the scanners used are the same day’s updated version.

TABLE VIII
STATISTICAL RESULTS OF DR AND ACY

Scanner	Detection rate	Accuracy
MacAf	27.5954%	90.4348%
NorAV	37.5456%	84.9242%
DrWeb	20.8624%	72.0504%
KaSky	36.2521%	85.3906%
CIMDS	76.5174%	88.0918%

A. 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 10 000 file samples from the “gray list.” Table VII shows the detection results of different antivirus scanners.

Of the 10 000 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 VIII and Fig. 4. From Tables VII and VIII and Fig. 4, we can see that our CIMDS system outperform other antivirus software for malware detection from the “gray list.”

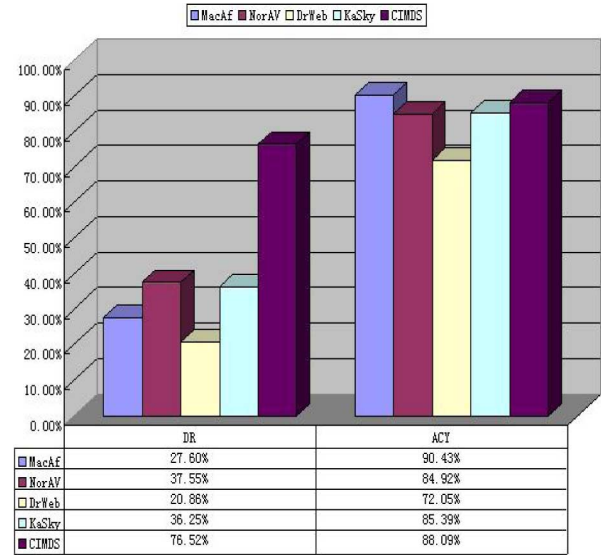


Fig. 4. DR and ACY of different scanners.

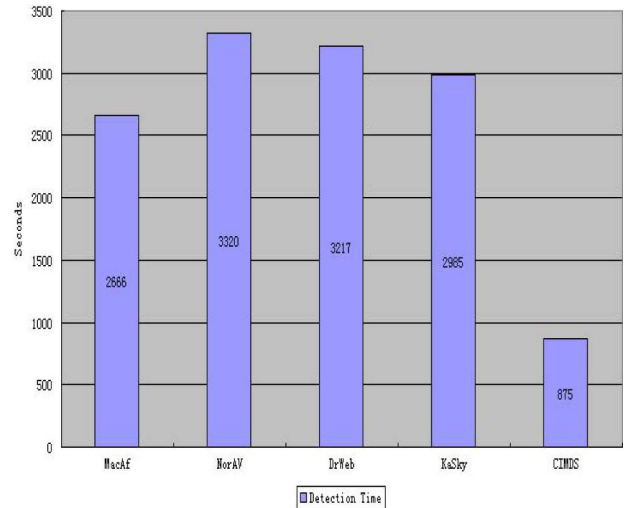


Fig. 5. Detection time of each scanner.

B. Comparisons of the Efficiency of Different Antivirus Scanners

In this set of experiments, in order to examine the efficiency of our CIMDS, we use the same dataset as described in Section VII-A to compare with other antivirus software. For our CIMDS, the detection time includes the time of feature extraction and file prediction. Fig. 5 shows the comparisons of the efficiency of different antivirus scanners. We observe that our CIMDS system achieves much higher efficiency than other scanners when being executed in the same environment due to the concise, but effective classifier as well as effective feature extraction method.

It is infeasible for our virus analysts to manually analyze all of the file samples in the “gray list.” If one expert spends 30 s to perform the analysis for a file sample, it would take him/her three and half days to finish the detection. In comparison, our CIMDS system just uses about 15 min for the accurate detection.

TABLE IX
RESULTS BY USING DIFFERENT CLASSIFIER

Algs.	Training Set		Testing Set	
	DR %	ACY %	DR %	ACY %
Naive Bayes	63.2578	50.9186	60.2230	48.6926
SVM	84.6012	68.2373	83.4510	64.4357
J4.8	56.8759	55.7260	57.2871	56.7716
IMDS	81.5591	67.1230	79.2140	64.1319
CIMDS	89.5245	71.3484	88.1639	67.5049

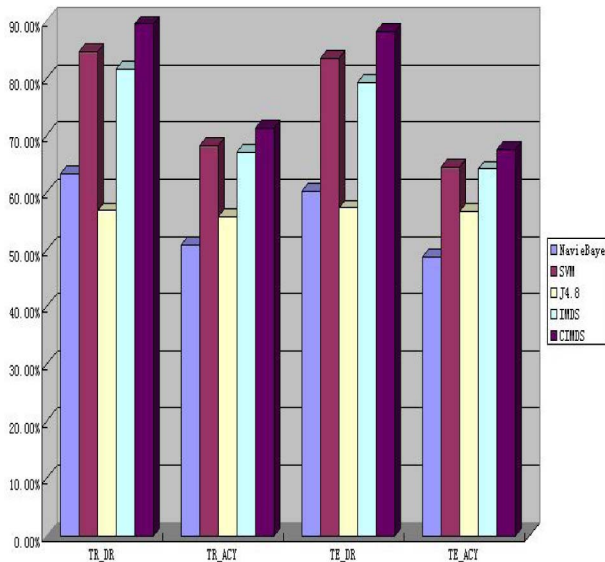


Fig. 6. DR and ACY of different classification methods.

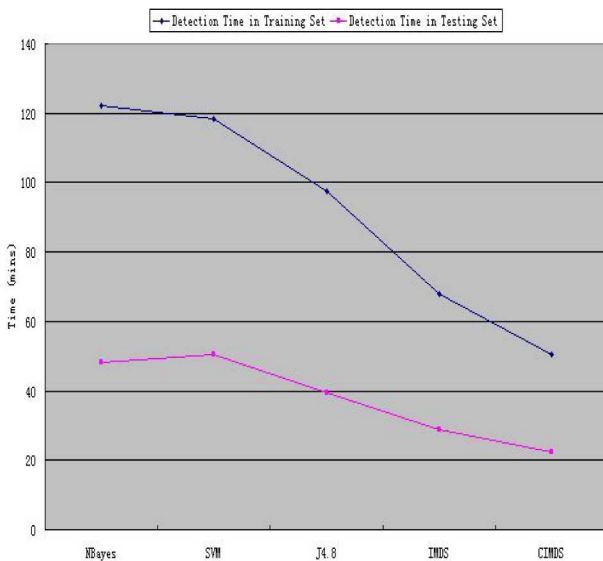


Fig. 7. Detection efficiency of different classification methods.

C. Comparisons of Different Classification Methods

Several data mining techniques, such as Naive Bayes, SVM, and Decision Trees have been applied to malware detection. In this set of experiments, we use the same dataset as described in Section VI-A and compare our CIMDS with Naive Bayes, LIBLINEAR SVM [39] with penalty parameter C set as 1, J4.8

version of Decision tree implemented in WEKA [34] as well as our previous IMDS system. Table IX and Fig. 6 show that our CIMDS system achieves the most accurate malware detection.

Besides achieving higher DR and ACY than other classification methods, the results shown in Fig. 7 indicate that our CIMDS also performs better in detection efficiency. Our CIMDS detects 15 000 file samples using just 22.5 min. The high efficiency results from classifier building method CIDCPF, which makes the reduced ruleset concise, but effective. In addition, our CIMDS system has another advantage over other classification methods, that is, we can generate the rules that are easy for our virus analysts to understand and interpret as shown in Section VI-E.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we systematically evaluate the effects of the postprocessing 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.” To the best of our knowledge, this is the first paper on using postprocessing 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 CIMDS 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. Promising experimental results demonstrate that the efficiency and ability of detecting the malware from the “gray list” of our CIMDS system outperform popular antivirus software, such as McAfee VirusScan and Norton AntiVirus, as well as previous data-mining-based detection systems, which employed Naive Bayes, LIBLINEAR SVM, and Decision Tree techniques.

In our future work, we plan to extend our CIMDS system from the following aspects: 1) collect more detailed information about the API calls, such as their dependencies and timestamps and use it for better malware detection. We will investigate methods such as frequent structure mining to capture the complex relationships among the API calls. 2) Predict the types of malware. Our CIMDS currently only provides binary predictions, i.e., whether a PE file is malicious or not. A natural extension is to predict the different types of malware.

ACKNOWLEDGMENT

The authors would like to thank the members in the Antivirus Laboratory at Kingsoft Corporation for their helpful discussions and suggestions.

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] M. Antonie, O. Zaiane, and A. Coman, "Associative classifiers for medical images," in *Proc. Mining Multimedia Complex Data (LNCS)*, 2003, pp. 68–83.
- [3] B. Arunasalam and S. Chawla, "CCCS: A top-down associative classifier for imbalanced class distribution," in *Proc. KDD-2006*, 2010, pp. 517–522.
- [4] E. Baralis, S. Chiusano, and P. Graza, "On support thresholds in associative classification," in *Proc. ACM Symp. Appl. Comput. 2004*, pp. 553–558.
- [5] E. Baralis and P. Torino, "A lazy approach to pruning classification rules," in *Proc. IEEE Int. Conf. Data Mining 2002*, pp. 35–42.
- [6] R. Bayardo, R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," in *Proc. ICDE-1999*, pp. 188–197.
- [7] H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," in *Proc. ICDE-2008*, pp. 169–178.
- [8] H. Cheng, X. Yan, J. Han, and C. Hsu, "Discriminative frequent pattern analysis for effective classification," in *Proc. ICDE-2007*, pp. 716–725.
- [9] M. Christodorescu, S. Jha, and C. Kruegel, "Mining specifications of malicious behavior," in *Proc. ESEC/FSE-2007*, pp. 5–14.
- [10] 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.
- [11] H. Toivonen, M. Klemetinen, P. Ronkainen, K. Hatonen, and H. Mannila, "Pruning and grouping discovered association rules," in *Proc. Minet Workshop Statist., Mach. Learning, and Discovery Databases*, 1995, pp. 47–52.
- [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.
- [13] J. Kolter and M. Maloof, "Learning to detect malicious executables in the wild," in *Proc. KDD-2004*, pp. 470–478.
- [14] Ng. R. T. Lakshmanan and J. L. Han, "Exploratory mining and pruning optimizations of constrained association rules," in *Proc. SIGMOD-1998*, pp. 13–24.
- [15] W. Li, J. Han, and J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules," in *Proc. IEEE Int. Conf. Data Mining 2001*, pp. 369–376.
- [16] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *Proc. 4th Int. Conf. Knowl. Discovery Data Mining 1998*, pp. 80–86.
- [17] B. Liu, W. Hsu, and Y. Ma, "Pruning and summarizing the discovered associations," in *Proc. KDD-1999*, pp. 125–134.
- [18] M. Mahta, R. Agrawal, and J. Rissanen, "SLIQ: A fast scalable classifier for data mining," in *Proc. EDBT-1996*, pp. 18–32.
- [19] M. Bailey, J. Oberheide, J. Andersen, Z. M. Mao, F. Jahanian, and J. Nazario, "Automated classification and analysis of internet malware," in *Proc. RAID 2007, LNCS*, vol. 4637, pp. 178–197.
- [20] J. R. Quinlan, *C4.5. Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann, 1993.
- [21] M. Schultz, E. Eskin, and E. Zadok, "Data mining methods for detection of new malicious executables," in *Proc. Security Privacy, 2001. Proc. 2001 IEEE Symp.*, May 14–16, pp. 38–49.
- [22] W. Snedecor and W. Cochran, *Statistical Methods*, 8th ed. Iowa City, IA: Iowa State Univ. Press, 1989.
- [23] R. Srikant, Q. Vu, and R. Agrawal, "Mining association rules with item constraints," in *Proc. KDD-1997*, pp. 67–73.
- [24] A. Sung, J. Xu, P. Chavez, and S. Mukkamala, "Static analyzer of vicious executables (save)," in *Proc. 20th Annu. Comput. Security Appl. Conf.*, 2004, pp. 326–334.
- [25] F. Thabtah, P. Cowling, and Y. Peng, "MCAR: Multi-class classification based on association rule approach," in *Proc. 3rd IEEE Int. Conf. Comput. Syst. Appl.*, 2005, pp. 1–7.
- [26] F. Thabtah, "A review of associative classification mining," *Knowl. Eng. Rev.*, vol. 22, no. 1, pp. 37–65, 2007.
- [27] F. Thabtah, P. Cowling, and Y. peng, "MMAC: A new multi-label, multi-class associative classification approach," in *Proc. 4th IEEE Int. Conf. Data Mining 2004*, pp. 217–224.
- [28] R. Topor and H. Shen, "Construct robust rule sets for classification," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 564–569.
- [29] U. Bayer, A. Moser, C. Kruegel, and E. Kirda, "Dynamic analysis of malicious code," *J. Comput. Virol.*, vol. 2, pp. 67–77, May 2006.
- [30] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu, "Virus detection using data mining techniques," in *Proc. IEEE Int. Conf. Data Mining*, 2003, pp. 71–76.
- [31] J. Wang and G. Karypis, "Harmony: Efficiently mining the best rules for classification," in *Proc. SDM-2005*, pp. 205–106.
- [32] K. Wang, S. Zhou, and Y. He, "Growing decision tree on support-less association rules," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Disc. Data Mining*, 2000, pp. 265–269.
- [33] Y. J. Wang, Q. Xin, and F. Coenen, "A novel rule ordering approach in classification association rule mining," in *Proc. 7th IEEE Int. Conf. Data Mining Workshops 2007*, pp. 339–348.
- [34] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. San Francisco, CA: Morgan Kaufmann, 2005.
- [35] Y. Ye, D. Wang, T. Li, and D. Ye, "IMDS: Intelligent malware detection system," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining 2007*, pp. 1043–1047.
- [36] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent pe-malware detection system based on association mining," *J. Comput. Virol.*, vol. 4, pp. 323–334, Jan. 2008.
- [37] X. Yin and J. Han, "CPAR: Classification based on predictive association rules," in *Proc. 3rd SIAM Int. Conf. Data Mining 2003*, pp. 331–335.
- [38] X. Zou, Q. Wang, M. Xiao, and Q. Cai, "Research on post-processing of data mining for electric power dispatching," *Mini-Micro Syst.*, vol. 24, no. 6, 2003.
- [39] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learning Res.*, vol. 9, pp. 1775–1778, 2008.

Authors' photographs and biographies not available at the time of publication.