ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/220615930

Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list

Article *in* Journal of Intelligent Information Systems · August 2010

DOI: 10.1007/s10844-009-0086-7 · Source: DBLP

CITATIONS
38

READS
371

5 authors, including:

Qingshan Jiang
Chinese Academy of Sciences

204 PUBLICATIONS **3,095** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project National Key Research and Development Program of China View project

All content following this page was uploaded by Qingshan Jiang on 05 June 2015.

The user has requested enhancement of the downloaded file.

Trend Micro, Inc.
EX1017-1

J Intell Inf Syst (2010) 35:1–20
DOI 10.1007/s10844-009-0086-7

Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list

Yanfang Ye · Tao Li · Kai Huang · Qingshan Jiang · Yong Chen

Received: 30 November 2008 / Revised: 15 March 2009 / Accepted: 15 March 2009 / Published online: 6 May 2009
© Springer Science + Business Media, LLC 2009

Abstract Nowadays, numerous attacks made by the malware (e.g., viruses, backdoors, spyware, trojans and worms) have presented a major security threat to computer users. Currently, the most significant line of defense against malware is anti-virus products which focus on authenticating valid software from a whitelist, blocking invalid software from a blacklist, and running any unknown software (i.e., the gray list) in a controlled manner. The gray list, containing unknown software programs which could be either normal or malicious, is usually authenticated or rejected manually by virus analysts. Unfortunately, along with the development of the malware writing techniques, the number of file samples in the gray list that need to be analyzed by virus analysts on a daily basis is constantly increasing. The gray list is not only large in size, but also has an imbalanced class distribution where malware is the minority class. In this paper, we describe our research effort on

Y. Ye
Department of Computer Science, Xiamen University,
Xiamen, 361005, People's Republic of China
e-mail: yeyanfang@yahoo.com.cn

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

K. Huang · Q. Jiang
Software School, Xiamen University,
Xiamen, 361005, People's Republic of China

Q. Jiang
e-mail: qjiang@xmu.edu.cn

Y. Chen
Anti-virus Laboratory, Kingsoft Corporation,
Zhuhai, 519000, People's Republic of China
e-mail: chenyong@kingsoft.com

Trend Micro, Inc.
EX1017-2

building automatic, effective, and interpretable classifiers resting on the analysis of Application Programming Interfaces (APIs) called by Windows Portable Executable (PE) files for detecting malware from the large and imbalanced gray list. Our effort is based on associative classifiers due to their high interpretability as well as their capability of discovering interesting relationships among API calls. We first adapt several different post-processing techniques of associative classification, including rule pruning and rule re-ordering, for building effective associative classifiers from large collections of training data. In order to help the virus analysts detect malware from the imbalanced gray list, we then develop the Hierarchical Associative Classifier (HAC). HAC constructs a two-level associative classifier to maximize precision and recall of the minority (malware) class: in the first level, it uses high precision rules of majority (benign file samples) class and low precision rules of minority class to achieve high recall; and in the second level, it ranks the minority class files and optimizes the precision. Finally, since our case studies are based on a large and real data collection obtained from the Anti-virus Lab of Kingsoft corporation, including 8,000,000 malware, 8,000,000 benign files, and 100,000 file samples from the gray list, we empirically examine the sampling strategy to build the classifiers for such a large data collection to avoid over-fitting and achieve great effectiveness as well as high efficiency. Promising experimental results demonstrate the effectiveness and efficiency of the HAC classifier. HAC has already been incorporated into the scanning tool of Kingsoft's Anti-Virus software.

Keywords Malware detection · Gray list · Class imbalance · Hierarchical Associative Classifier (HAC)

1 Introduction

1.1 Malware detection from the gray list

Malware is a generic term to denote all kinds of unwanted software (e.g., viruses, backdoors, spyware, trojans and worms) (Bayer et al. 2006). Numerous attacks made by the malware have posed a major security threat to computer users. Therefore, malware detection is one of the computer security topics that are of great interest. Currently, the most significant line of defense against malware is anti-virus software products, such as Norton, Dr. Web and Kingsoft's Antivirus. These widely-used malware detection software tools mainly 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 (Sung et al. 2004).

In order to capture as many malware samples as possible, besides authenticating valid software from a whitelist and blocking invalid software from a blacklist using signature-based method, most of the existing anti-virus software products run any unknown software (i.e., the gray list) in a controlled manner. The gray list, containing unknown software programs which could be either normal or malicious, is usually authenticated or rejected manually by virus analysts. Unfortunately, with the devel-

Trend Micro, Inc.
EX1017-3

opment of the malware writing techniques, the number of file samples in the gray list that need to be analyzed by virus analysts on a daily basis is constantly increasing. For example, the gray list collected by the Anti-virus Lab of a large software corporation usually contains more than 100,000 file samples per day. The gray list is not only large in size, but also has an imbalanced class distribution where malware is the rare class. The gray list is also complicated since it contains the variants of known malware and previously unknown malware samples. In order to remain effective, it is of paramount importance for the anti-virus companies to be able to quickly analyze the gray list.

1.2 Contributions of the paper

In order to build automatic, effective, and interpretable classifiers for malware detection from the large, complicated and imbalanced gray list, we have to address the following challenges:

- **How to make the classifier interpretable?** The classifier should generate knowledge or patterns that are easy for the virus analysts to understand and interpret.
- **How to construct an effective classifier to detect malware from the gray list?** How to make the classifier less sensitive to the imbalance and perform well for the large and complicated gray list?
- **How to efficiently build the classifiers?** In our application, we have a total of 16 million labeled file samples available for training: half of them are malware samples and the other half are benign file samples. Sampling techniques are needed to build classifiers for such a large data collection to avoid over-fitting and achieve great effectiveness as well as high efficiency. However, the choices of the class distribution and the size of the training data in the sampling strategy for malware detection are not trivial and need careful investigation.

In this paper, we describe our research effort to address the above challenges. To address the interpretability issue, we choose to build associative classifiers since they can generate rules that are easy for virus analysts to understand and interpret. We first adapt several different post-processing techniques of associative classification, including rule pruning and rule re-ordering, for building effective associative classifiers from large collections of training data. To address the second issue, we develop the Hierarchical Associative Classifier (HAC) to make the classifier less sensitive to the imbalance and thus improve its performance. To deal with the last issue, we empirically examine the sampling strategies for efficiently building classifiers. We study the problems of the choice of class distribution and the size for the sampling strategy.

Our case studies on a large and real data collection, including 8,000,000 malware, 8,000,000 benign files, and 100,000 file samples from the gray list, collected by the Anti-virus Lab of Kingsoft corporation, a top Chinese software company in computer security, word processing and online games, illustrate that: (1) After being scanned by all the popular anti-virus software products, such as Norton AntiVirus and Dr. Web, malware in the gray list still can be effectively detected by our HAC. (2) For detecting malware from the imbalanced gray list, the performance and efficiency of our HAC outperform other classification methods. HAC has already been incorporated into the scanning tool of Kingsoft's Anti-Virus software.

Trend Micro, Inc.
EX1017-4

1.3 Content of the paper

The rest of this paper is organized as follows. Section 2 presents the overview of our malware detection system and Section 3 discusses the related work. In Section 4, we describe the post-processing techniques of associative classification that HAC will adopt. In Section 5, a novel Hierarchical Associative Classification method, HAC, is proposed to detect malware from the imbalanced gray list. In Section 6, we empirically investigate the problems of class distribution and sample size in the sampling strategy for classifier construction. In Section 7, we systematically evaluate the effects of our proposed HAC method. We also compare HAC with popular anti-virus software such as Norton AntiVirus and Dr. Web, as well as other classification methods. Finally, Section 8 concludes.

2 System architecture

In this paper, resting on the analysis of Windows API (Application Program Interface) calls which can reflect the behavior of program code pieces, we develop the Hierarchical Associative Classifier (HAC) to detect malware from the large, complicated and imbalanced gray list. Figure 1 shows the malware detection procedure of HAC:

(1) It first uses the feature extractor to extract the API calls from the collected PE files, converts them to a group of 32-bit global IDs as the features of the

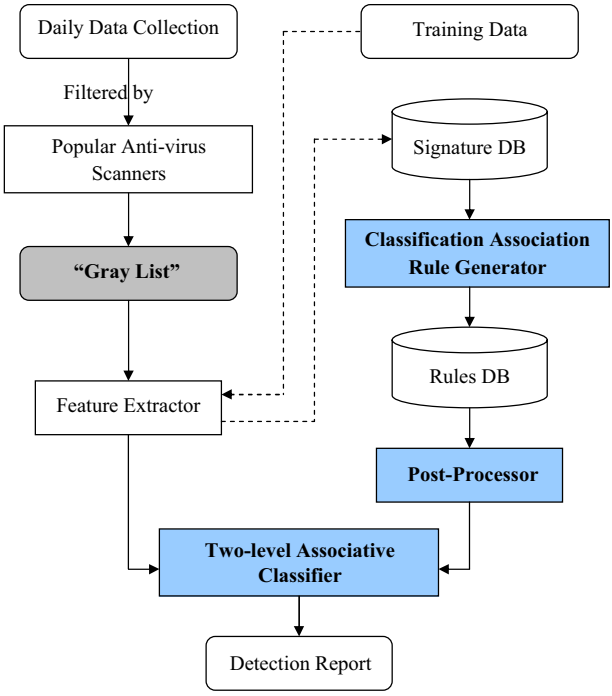

Fig. 1 Malware detection flow of HAC

Trend Micro, Inc.
EX1017-5

Fig. 2 A sample signature database after data transformation

Resultset 1

id	filename	filesort	apiseq	intvectorofapi	apifunno
9198	Adware.Casino.N__27ee8...	1	wsock32.dll;wsock32.dll 13;ws...	5,7,10,12,13,14,15,16,17,1...	158
9199	Adware.DropSpam__29b8...	1	ws2_32.dll;wsasocketa;ws2_3...	1,7,9,14,15,16,17,18,30,32,...	145
9201	Adware.NaviPromo.Gen.3_...	1	kernel32.dll;getconsolemode;k...	6,15,18,43,45,48,53,55,56,...	124
9202	Backdoor.AFCore.Dropper....	1	kernel32.dll;exitthread;kernel32...	2,44,45,81,131,249,258,26...	29
9203	Backdoor.Agent.AAS__93...	1	kernel32.dll;getprocaddress;ker...	1,5,7,8,10,13,14,15,17,18,3...	98
9209	Backdoor.Generic.121907...	1	kernel32.dll;rtlzeromemory;kern...	12,13,39,53,54,55,56,61,63...	74
9210	Backdoor.Hupigon.115462...	1	kernel32.dll;findclose;kernel32...	1,5,7,8,10,11,12,13,14,15,1...	205
9211	Backdoor.Hupigon.115462...	1	kernel32.dll;findclose;kernel32...	1,5,7,8,10,11,12,13,14,15,1...	205
9212	Backdoor.Hupigon.115462...	1	kernel32.dll;findclose;kernel32...	1,5,7,8,10,11,12,13,14,15,1...	205
9231	BehavesLike.Trojan.Firewa...	1	kernel32.dll;globalflags;kernel3...	1,3,4,5,6,8,10,11,12,13,15,...	232
9232	BehavesLike.Win32.Explor...	1	kernel32.dll;tlssetvalue;kernel3...	38,39,40,41,42,43,44,45,46...	57

training data, and stores these features in the signature database. A sample signature database is shown in Fig. 2, in which there are 6 fields: 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.

(2) After data transformation, it then generates the classification association rules from the training signature database.

(3) In the third step, it adapts hybrid post-processing techniques of associative classification to reduce the generated rules.

(4) Finally, it constructs two-level associative classifier to detect malware from the imbalanced "Gray List". This two-level associative classifier can maximize *precision* and *recall* (Chawla et al. 2002) of the minority (malware) class, which can be defined as follows: $precision=\frac{TP}{TP+FP}$, $recall=\frac{TP}{TP+FN}$, where TP is the number of malicious files correctly classified, FP is the number of benign files incorrectly classified as malicious and FN is the number of malicious files incorrectly classified as benign. In the first level, it uses high precision rules of majority (benign file samples) class and lower precision rules of minority class to achieve high recall. Because such rules would avoid misclassifying the minority samples as majority class by the constructed classifier. In the second level, it ranks the minority class files according to their prediction rules and optimizes the precision. As shown in Fig. 3, the first level classifier is displayed with solid line, while the second level boundary, formed by the classifier constructed with the prediction rules of the minority class files according to the mechanism described in Section 5, is displayed with dashed line. As graphically represented

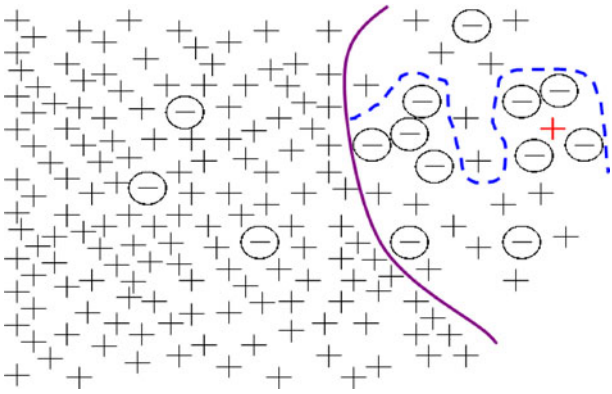

Fig. 3 Graphical representation of gray list and two-level associative classifier boundaries of HAC: *plus sign* represents a benign file sample and *circled dash* represents a malware

Trend Micro, Inc.
EX1017-6

in Fig. 3, the gray list is complicated since it contains the variants of known malware and previously unknown malware samples. In addition, it is also imbalanced and overlapping (i.e., the benign and malware samples are mixed together and can not be easily separated).

3 Related work

3.1 Data mining methods for malware detection

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 (Kolter and Maloof 2004; Schultz et al. 2001; Sung et al. 2004; Wang et al. 2003; Christodorescu et al. 2007; Ye et al. 2007). Neural Networks as well as immune system are used by IBM for computer virus recognition (Tesauro et al. 1996; Kephart et al. 1993, 1997). Naive Bayes, Support Vector Machine (SVM) and Decision Tree classifiers are used to detect new malicious executables based on small data collection in the previous studies (Kolter and Maloof 2004; Schultz et al. 2001; Wang et al. 2003). Associative classification (Liu et al. 1998), as a new classification approach integrating association rule mining and classification, becomes one of the significant tools for knowledge discovery and data mining (Thabtah 2007). The algorithms of associative classification, such as CBA (Liu et al. 1998), CMAR (Li et al. 2001), CPAR (Yin and Han 2003), perform well for classification tasks. Thus associative classifier can be effectively used in malware detection (Ye et al. 2007). In Ye et al. (2007), it is demonstrated that association classifier outperforms other classifiers, like Naive Bayes, Support Vector Machine (SVM) and Decision Tree, based on the large data collection for malware detection. However, in our application, the situation is much more complicated: Firstly, the labeled data set and the gray list of our collection are very large; More importantly, the class distribution in the labeled data set is balanced, while the gray list is quite imbalanced. Hence, the accuracy driven classifiers may fail for such large and imbalanced gray list. For example, neural networks consistently biased towards the majority class at any given size and prone to treat the minority (malware) class as noise and therefore to disregard it (Japkowicz and Stephen 2002). Decision trees algorithms (C4.5) are also not performing well in the presence of imbalance: they might lose big parts of the minority (malware) class at the pruning phase or lead to the trees of large size and over-fitting of the minority class (Japkowicz and Stephen 2002). Though associative classification method performs well on large data collection, it needs to be further investigated for dealing with class imbalanced problem.

3.2 Methods for dealing with class imbalance

Solutions to the class imbalance problem were proposed both at the data and algorithmic levels (Visa and Ralescu 2005): different forms of re-sampling techniques at the data level and effective classification methods at the algorithmic level. However, there seem no existing methods have been testified that they can be directly applied to our application: 16 million balanced labeled data samples available for learning, while around 100,000 imbalanced data from the gray list with high complexity and

Trend Micro, Inc.
EX1017-7

overlapping for prediction per day. Here we briefly review the existing methods for dealing with the class imbalance issue.

Re-sampling For re-sampling, one of the most common assumptions in the design of data mining or machine learning algorithms is that the training data consist of examples drawn independently from the same underlying distribution as the testing data (Zadrozny 2004). If this were true, then we could adopt under-sampling (eliminating data from the majority class) or over-sampling (duplicating data of the minority class) technique (Weiss 2004) to make the distribution of the training set the same as the gray list. However, the assumption is violated according to several studies. It has shown that for the classifiers like decision trees, neither a 1:1 (that is balanced) learning distribution, nor the same distribution as the testing set are the best for the learning task (Weiss and Provost 2003). As a result, the problems of class distribution and sample size in the sampling strategy for classifier construction need to be investigated for our real application.

Classification Methods Fuzzy based classifier, proposed by Visa and Ralescu (2003), may be less sensitive to the class imbalance by considering a relative frequency to the class size. However, it is affected by high complexity and data overlapping. A piecewise linear separable SVM classification algorithm, presented by Kamei and Ralescu (2003), can perform well for imbalanced learning, but is seriously affected by the loss of information. If we change our balanced labeled data to imbalanced distribution, it may result in loss of information by under-sampling or increasing the training size by over-sampling, both of which greatly compromise the performance of this piecewise linear separable SVM classifier. By contrast, if we keep balanced learning and imbalanced prediction, Support Vector Machine (SVM), as an overall accuracy driven classifier, still results in low recall of minority (malware) class.

Cost-sensitive Learning Recently, cost-sensitive learning methods have been developed for dealing with class imbalance (Elkan 2001; Drummond and Holte 2003). These methods exploit the fact that the value of correctly identifying the positive (rare) class outweighs the value of correctly identifying the common class (Weiss 2004). However, the problem with these approaches is that specific cost information is rarely available, since it is hard to access the error costs and class distribution even by human experts.

4 Associative classifier construction

In this paper, our effort is based on associative classification (Liu et al. 1998) for the reasons that (1) it has been testified that associative classifier outperforms other traditional classification methods, such as Naive Bayes, Support Vector Machine (SVM) and Decision Tree, for malware detection based on large data collection (Ye et al. 2007); (2) it can discover interesting relationships among API calls that are explicitly related to malware/benign file class, and generate the rules which are easy for the virus analysts to understand and interpret. However, besides rule generation approach, post-processing (Thabtah 2007) of associative classification is also very important for improving the accuracy and efficiency of the classifiers. In this section,

Trend Micro, Inc.
EX1017-8

we adapt the hybrid post-processing techniques to reduce the generated rules for associative classifier construction.

4.1 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: $class_1 = Malicious$, and $class_2 = Benign$.

- **(Support and confidence)** Given a dataset DB, let $I = \{I_1, ..., I_m\}$ 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 are defined as:

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

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

  where the function $count(I \cup \{Class\})$ returns the number of records in the dataset DB where $I \cup \{Class\}$ holds.
- **(Frequent itemset)** Given mos as a user-specified minimum support. I is a frequent itemset/pattern in DB if $os \geq mos$.
- **(Classification association rule)** Given moc as a user-specified confidence. Let $I = \{I_1, ..., I_m\}$ be a frequent itemset. $I \rightarrow Class(os, oc)$ is a classification association rule if $oc \geq moc$.

Apriori (Agrawal and Imielinski 1993) and FP-Growth (Han et al. 2000) algorithms can be extended to associative classification (Li et al. 2001; Liu et al. 1998). In general, FP-Growth algorithm is much faster than Apriori for mining frequent item sets. In our work, we use FP-Growth algorithm to conduct the classification association rule mining.

To demonstrate the associative classifier, here we present a case study. We sample 26,160 records as the training set: half of them are malicious executables and the other half are benign ones. 75,887 rules are generated by the associative classifier with the minimum support and confidence as 0.05 and 0.7 respectively for each class. One of the generated rules is:

$$(19,120,180,181,219) \rightarrow class=Malicious(os=0.06,oc=1),$$

where os and oc represent the support and confidence, respectively. After converting the API IDs to API names via our API query database, the rule becomes:

$$(KERNEL32.DLL, Process32first; Createtoolhelp32snapshot; Openprocess; USER32.DLL, Callnexthookex; ADVAPI32.DLL, Createservicea;) \rightarrow class = Malicious(os = 0.06, oc = 1).$$

With the os and oc values, we know that this sequence of APIs appears in 1,570 malware samples and not in any benign files. It can be one of the candidate rules for determining whether a file sample is malicious or not.

4.2 Post-processing for associative classifier construction

Since there is a huge number of rules generated from the training set and it is infeasible to build the classifier used all of rules, post-processing of associative

Trend Micro, Inc.
EX1017-9

classification is also very important for improving the accuracy and efficiency of the classifier. In this section, we mainly investigate the techniques of rule pruning and rule re-ordering.

Rule Pruning Several common rule pruning approaches have been developed for associative classifiers to reduce the generated rules (Baralis and Torino 2002; Bayardo et al. 1999; Toivonen et al. 1995; Li et al. 2001; Liu et al. 1998; Mahta et al. 1996; Thabtah 2007): (1) χ^2 (chi-square) testing (Li et al. 2001) to measure the significance of the rule itself, (2) database coverage (Liu et al. 1998) to just keep the rules covering at least one training data object not considered by a higher ranked rule, and (3) pessimistic error estimation (Liu et al. 1998) to test the estimated error of a new rule. These rule pruning techniques mainly focus on individual rules.

We have used the above three pruning techniques in our application. However, we notice that there are still many "insignificant" rules: some rules are redundant or minor variations of others and some rules simply exist due to chance rather than true correlation (Liu et al. 1999). These "insignificant rules" should be removed. For example, given the rule: ***R1: API_100=yes → class=Malicious (suppcount=35, conf=80%)***, the following rule: ***R2: API_100=yes, API_200=yes → class=Malicious (suppcount=32, conf=81%)*** becomes "insignificant" because it gives little extra information. In our work, we also employ another rule pruning method (Liu et al. 1999), named "insignificant rule pruning", to prune those redundant rules by using χ^2 measure to test whether the rule is significant *w.r.t* to its ancestors. The χ^2 value is defined as $\chi^2 = \sum \frac{(f_0 - f)^2}{f}$, where f_0 is an observed frequency and f is an expected frequency. For the above example, the χ^2 value of *R2* is 0.00395 which is below the threshold value (3.84 at the 95% significance level (Liu et al. 1999)). Therefore, *R2* should be discarded.

Rule Re-ordering Rule re-ordering plays an important role in the classification process since most of the associative classification algorithms utilize rule ranking procedures as the basis for selecting the classifier (Liu et al. 1998; Li et al. 2001; Yin and Han 2003). In particular, CBA (Liu et al. 1998) and CMAR (Li et al. 2001) use database coverage pruning approach to build the classifiers, where the pruning evaluates rules according to the rule re-ordering list. Hence, the highest-order rules are tested in advance and then inserted into the classifier for predicting test data objects. For rule re-ordering, there are five popular mechanisms (Wang et al. 2007): (1) Confidence Support size of Antecedent (CSA), (2) size of Antecedent Confidence Support (ACS), (3) Weighted Relative Accuracy (WRA), (4) Laplace Accuracy, and (5) χ^2 (chi-square) measure. CSA and ACS are belong 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 CPAR (Yin and Han 2003), to weigh the significance of each generated rule. In our work, we adopt hybrid rule re-ordering mechanism by combining CSA and χ^2 to re-order the rules. We first rank the rules whose confidences are 100% by CSA and then re-order the remaining rules by χ^2 measure. Because those rules whose confidences are 100% can make the classifier accurate, while the remaining rules should be considered by the combination of their supports and confidences together.

Trend Micro, Inc.
EX1017-10

Table 1 Results of different post-processing techniques

Classifiers	Rules	Accuracy	Precision	Recall
NP	75,887	58.08%	58.72%	54.35%
CSA	548	76.94%	75.24%	80.32%
ACS	703	67.42%	66.19%	71.23%
WRA	507	63.12%	60.99%	72.83%
Laplace	562	74.79%	73.21%	78.20%
χ^2	539	76.26%	74.15%	80.63%
CSA/χ^2	**534**	**81.33%**	**80.45%**	**82.76%**

NP represents the associative classifier without any post-processing. All other classifiers use the proposed rule pruning techniques followed by the specific rule re-ordering mechanism. The last row presented with bold emphasis shows that the hybrid CSA/χ^2 rule re-ordering method we adopt achieves best results

We use "Best First Rule" (Wang et al. 2007) method to predict the new file samples. We select the first best rule that satisfies the new file sample according to the rule list based on our hybrid CSA/χ^2 rule re-ordering method to predict whether the new case is malware or not.

4.3 Effects of post-processing

In this section, we perform experiments on the dataset obtained in Section 4.1 to demonstrate the effects of post-processing. Experimental results are shown in Table 1. The *NP* row presents the results of the associative classifier described in Section 4.1 which just uses first matched rule for prediction but without any rule pruning and rule re-ordering techniques. All other classifiers use the proposed rule pruning techniques described in Section 4.2 followed by a rule re-ordering mechanism. We use the rule re-ordering mechanism employed in a classifier to denote the classifier. Experimental results demonstrate that the final rule set is concise but effective after rule pruning and rule re-ordering. By adopting the above post-processing techniques, we can reduce the 75,887 generated rules to 534.

5 Hierarchical associative classifier (HAC)

For malware detection from the large and imbalanced gray list, we must try to maximize the *precision* and *recall* of minority (malware) class. Many data mining and machine learning algorithms try to optimize these two competing measures simultaneously (Weiss 2004; Joshi et al. 2001). In order to detect malware from the gray list effectively, we propose a Hierarchical Associative Classifier (HAC) constructing two-level associative classifier to focus on each measure respectively. Figure 4 shows the conceptual flow of the proposed technique of HAC: in the first

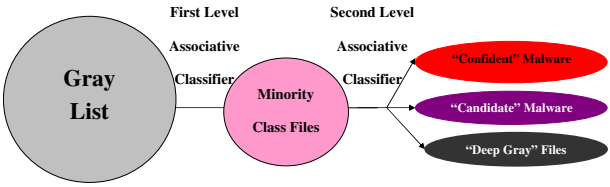

Fig. 4 The proposed HAC method

Trend Micro, Inc.
EX1017-11

level, it uses high precision rules of majority (benign file samples) class and lower precision rules of minority class to achieve high *recall*; in the second level, it ranks the minority class and optimizes the *precision*.

5.1 First level associative classifier construction

The first level associative classifier strives for high precision rules of majority (benign file samples) class and lower precision rules of minority (malware) class. This mechanism ensures that the classifier can achieve much higher *recall* of minority class. HAC constructs the first level associative classifier by the following steps:

1. generating the rules of majority class whose confidences are 100% and rules of minority class whose confidences are greater than a pre-defined threshold;[1]
2. using the hybrid post-processing techniques to reduce the generated rules and build the classifier;
3. using "Best First Rule" method to predict the gray list.

5.2 Second level associative classifier construction

HAC constructs the second level associative classifier by ranking the minority class files to optimize the *precision* from the following steps:

1. selecting those file samples whose prediction rules have 100% confidences, marking them as "confident" malware and submitting them to the virus analysts for analyzing; (The prediction rule of the file sample is the "Best First Rule" which matches the file sample.)
2. ranking the remaining minority class files in an descending order based on their prediction rules' χ^2 (chi-square) values;
3. selecting the first k user-specified files (k is always set as 200, since a virus analyst can just analyze 200 files per day) from the remaining ranking list and marking them as "candidate" malware;
4. marking the remaining (unselected) files as "deep gray" files.

With the two-level associative classifier building method, HAC first ensures high *recall* of the minority class, then optimizes the *precision*: the "confident" malware are almost actual malware, while the "candidate" malware are also possibly recognized as malicious files by the virus analysts.

6 Sampling strategies for HAC

For our application, there are 16 million balanced labeled samples available for learning, while around 100,000 file samples of the gray list need to be analyzed per day. Sampling techniques are needed to efficiently build classifiers for such a large data collection. Since the data collection is large and the class distribution in the labeled data is not the same as that in the gray list, how to sample the labeled data for training is a very important question for malware detection. In this section,

[1] In our work, we set the threshold to be 70% after discussion with the virus analysts.

Trend Micro, Inc.
EX1017-12

Table 2 Experimental results of AC with different training and testing distributions

%	Test1 1:1	Test2 10:1	Test3 100:1
Train1 (1:1)			
Accuracy	82.15	68.9556	68.6966
Precision	86.4428	18.7779	2.2422
Recall	76.3065	71.7949	71.4286
Train2 (10:1)			
Accuracy	66.3409	93.4726	99.2316
Precision	99.8661	96.3303	75
Recall	32.7514	29.9145	34.2857
Train3 (100:1)			
Accuracy	61.0089	92.5587	99.2316
Precision	100	100	100
Recall	22.0136	18.8034	22.8571

Remarks: "A:B" means the ratio of benign samples vs. malware samples

we empirically investigate the problems of class distribution and sample size in the sampling strategy for classifier construction.

Since it is infeasible to use the whole data collection to conduct empirical studies, we first obtain 26,160 samples for training and 6,958 for testing. Both of the training and testing sets are balanced distribution (i.e., they have equal numbers of malware samples and benign samples). These datasets will be used in our empirical studies in this section.

6.1 Class distribution

One of the most common assumptions in the design of data mining or machine learning algorithms is that the training data consist of examples drawn independently from the same underlying distribution as the testing data (Zadrozny 2004). However, the assumption is violated according to several studies. It has shown that for the classifiers like decision trees, neither a 1:1 (that is balanced) learning distribution, nor the same distribution as the testing set are the best for the learning task (Weiss and Provost 2003). In this section, we empirically investigate the relation between training and testing distributions for malware detection. For comparison purpose, we

Table 3 Experimental results of linear SVM with different training and testing distributions

%	Test1 1:1	Test2 10:1	Test3 100:1
Train1 (1:1)			
Accuracy	87.3239	64.2801	64.0296
Precision	86.5223	15.7824	1.9547
Recall	88.403	67.5287	71.4286
Train2 (10:1)			
Accuracy	71.7304	94.0685	98.8617
Precision	99.5051	87.5776	41.6667
Recall	43.7029	40.4605	44.1176
Train3 (100:1)			
Accuracy	68.2093	94.4343	99.1747
Precision	99.9299	97.1429	80
Recall	36.4404	19.4737	24.176

Trend Micro, Inc.
EX1017-13

use both the Associative Classifier (AC) with post-processing described in Section 4 and Linear SVM (Hsu and Lin 2002) implemented in LibLinear package.

To systematically evaluate the effects of class distributions on classification performance, we vary the class distributions of both the training set and the testing set and measure the classification performance under different conditions. The class distribution is changed from balanced distribution (e.g., 1:1) to 10:1 and 100:1 by using under-sampling technique. For each training distribution, we measure the classifier performance for three different testing distributions. Experiment results using AC and Linear SVM are summarized in Table 2 and Table 3, respectively. We observe that balanced training results in low *precision* of minority (malware) class for imbalanced prediction, while imbalanced training results in low *recall* of minority class. Even the training set has the same distribution as the testing set, the classifiers still can not perform well (e.g., low *recall*) for imbalanced prediction of gray list. From the above analysis of the empirical studies, in order to maximize the *precision* and *recall* of the minority class for imbalanced prediction, we first guarantee the *recall* of the minority class by sampling the training data as balanced distribution, then use our proposed method HAC to maximize the *precision* of the minority class. Besides, there are also another reasons to keep the original class distribution of the balanced labeled data for sampling: (1) In practice, it is hard to know the exact distribution of the daily gray list; (2) Making the training distribution as the testing set by using under-sampling technique may arouse the loss of information.

6.2 Sample size

Here we evaluate the effects of sample size on classification performance. We consider two imbalanced scenarios: Test1 with 10:1 test distribution and Test2 with 100:1 test distribution. We keep the original balanced class distribution and vary the sample size for training by uniformly select 50% and 12.5% of the samples from each class. For each test scenario, we measure the classifier performance for different sample sizes. Experiment results using AC and Linear SVM are summarized in Tables 4 and 5, respectively. The results show that, uniformly sampled subsets can represent/capture the full picture of the whole data collection and achieve almost the same prediction performance for the same testing set. In our application, we will uniformly select 2% file samples of each class from the available 16 million data collection based on created time of the file samples. Because, based on the experience of our virus analysts, each program can have around 50 variants or similar samples.

Table 4 Experimental results of AC under different sample sizes

	Train1	Train2	Train3
	100%	50%	12.5%
Test1 (10:1)			
Accuracy(%)	68.9556	67.4151	66.7102
Precision(%)	18.7779	18.0106	17.8298
Recall(%)	71.7949	71.9889	72.8065
Test2 (100:1)			
Accuracy(%)	68.6966	67.473	66.8184
Precision(%)	2.2422	2.1589	2.1978
Recall(%)	71.4286	71.4286	74.2857

Remarks: "A:B" means the ratio of benign sample vs. malware samples

Trend Micro, Inc.
EX1017-14

Table 5 Experimental results of linear SVM under different sample sizes

	Train1 100%	Train2 50%	Train3 12.5%
Test1 (10:1)			
Accuracy(%)	64.2801	64.6214	67.1802
Precision(%)	15.7824	15.8995	16.8792
Recall(%)	67.5287	66.6208	65.7625
Test2 (100:1)			
Accuracy(%)	64.0296	64.8833	68.6682
Precision(%)	1.9547	2.0016	2.2401
Recall(%)	71.4286	71.4286	71.4286

In summary, in our real application, we keep the natural distribution of the labeled data and uniformly select 2% of the file samples from each class to represent the whole picture of the 16 million data collection.

7 Experimental results and analysis

In this section, we conduct three sets of experimental studies using our data collection obtained from the Anit-virus Lab of Kingsoft to compare our HAC with other classifiers and widely-used virus scanners: (1) The first set of study is to identify the most representative features for improving the performance and efficiency of the classifiers. (2) In the second set of experiments, resting on the analysis of API calls, we compare our HAC with other classification based methods, including linear SVM and Associative Classifier (AC). (3) Finally, we apply our HAC for real application to testify its malware detection ability and efficiency of the gray list, in comparisons with some of the popular anti-virus scanners such as Norton AntiVirus, Dr. Web, MaAfee VirusScan and Kaspersky Anti-Virus. All the experimental studies are conducted under the environment of Windows XP operating system plus Intel P4 1.83 GHz CPU and 2 GB of RAM.

7.1 Evaluation of API selection

Identifying the most representative features is critical to improve the performance and efficiency of the classifiers (Jain et al. 2000; Langley 1994). As not all of the API calls are contributing to malware detection, we rank each API call using Max-Relevance algorithm (Peng et al. 2005) and select top k API calls as the features for later classification. Since it is infeasible to use the whole data collection to testify the validation of the feature selection method, in this set of experiments, we obtain 26,160 PE file samples for training (half of them are recognized as benign executables and the other half are malicious executables mainly consisting of backdoors, trojans and worms) and 6,958 samples for test (half of them are benign files and the other half are malicious ones). There are 4,409 API calls extracted from these file samples. Figure 5 shows that the testing performance of Associative Classifier (AC) changes slightly after the number of features reaches 100. So, we select top 100 API calls as the features for later classification.

Trend Micro, Inc.
EX1017-15

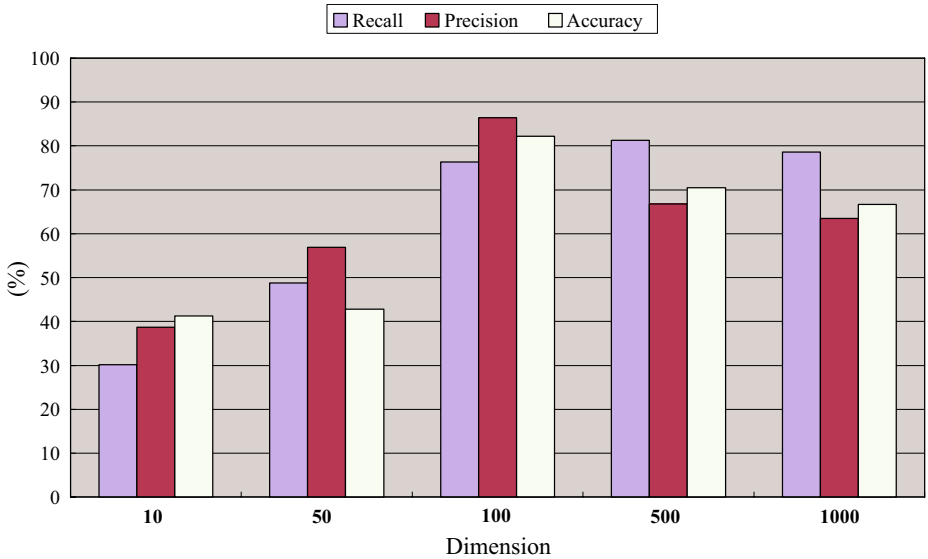

Fig. 5 AC performance with different number of APIs

7.2 Comparisons of different classification methods

In this set of experiments, we evaluate the performance and efficiency of different classifiers: linear SVM (Hsu and Lin 2002) implemented in LibLinear package, Associative Classifier described in Section 4 and HAC, using the dataset described in Section 7.1. We vary the test distribution from 1:1 to 10:1 and 100:1 by using under-sampling technique, to testify each classifier's detection ability for imbalanced prediction. *Precision* and *recall* (Chawla et al. 2002) of minority class are used to

Table 6 Prediction results of different classifiers

	Test1	Test2	Test3
	1:1	10:1	100:1
AC			
Precision(%)	86.4428	18.7779	2.24215
Recall(%)	76.3065	71.7949	71.4286
Linear SVM			
Precision(%)	86.5223	15.7824	1.9547
Recall(%)	88.403	67.5287	71.4286
HAC-Confident			
Precision(%)	99.8833	96.8	80.0
Recall(%)	37.6702	34.4729	45.7143
HAC-Candidate			
Precision(%)	84.6353	77.3684	73.6842
Recall(%)	32.3562	41.8803	40.0

Remarks: "A:B" means the ratio of benign files vs. malware; "HAC-Confident" means that the files marked as "confident" malware by HAC, while "HAC-Candidate" means that the files marked as "Candidate" malware by HAC

Trend Micro, Inc.
EX1017-16

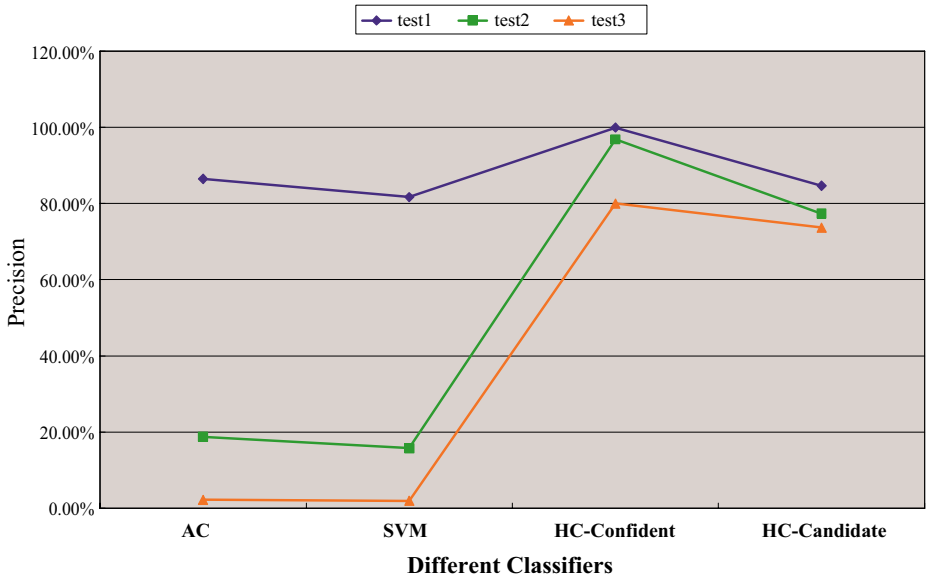

Fig. 6 Malware detection precision of different classifiers

measure the performance of each classifier. From the experiments in Section 6.1, we know that imbalanced training would result in low recall for imbalanced prediction and hence here we only report the experimental results with the balanced training distribution. Table 6, Figs. 6 and 7 show that HAC achieves best imbalanced prediction.

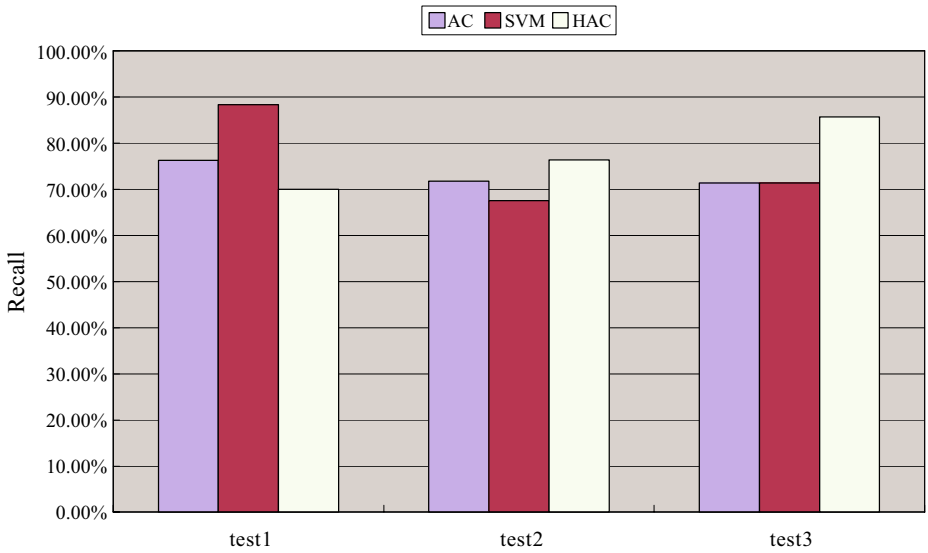

Fig. 7 Malware detection recall of different classifiers. Remarks: the recall of HAC is the recall sum of "HAC-Confident" and "HAC-Candidate"

Trend Micro, Inc.
EX1017-17

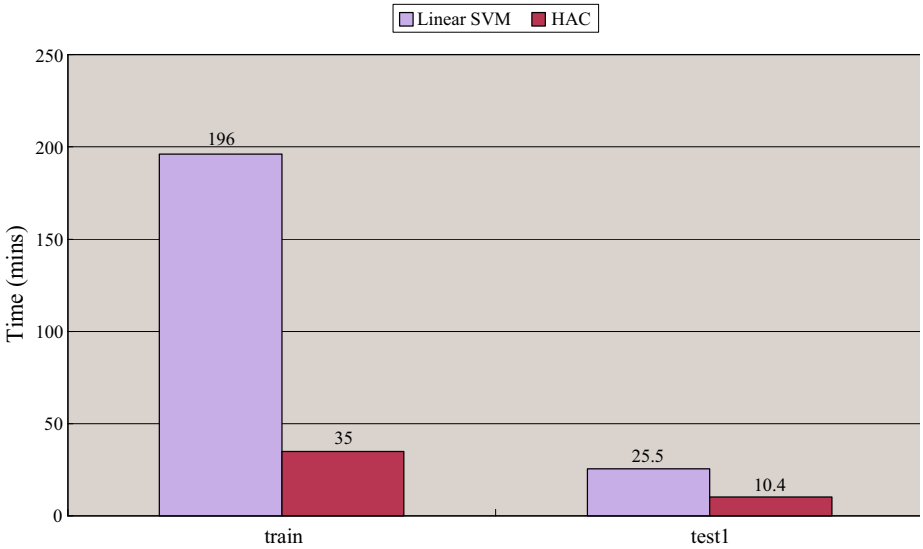

Fig. 8 Detection efficiency of different classifiers

Besides achieving better prediction power than other classification methods, the results shown in Fig. 8 indicate that our HAC also perform better in detection efficiency. HAC detects 6,958 file samples using 10.4 minutes (including feature extraction time). HAC/AC also has another advantage: the generated rules are easy for the virus analysts to understand and interpret.

7.3 Malware detection from the gray list

Our HAC has been incorporated into the scanning tool of Kingsoft's Anti-Virus software. We apply HAC in real applications to testify its malware detection ability and efficiency of the gray list. We randomly select 21,074 unlabeled files from the daily collection of October 9th, 2008, of which 6,123 are recognized as malicious executables by the popular anti-virus scanners such as Norton AntiVirus, Dr. Web, MaAfee VirusScan and Kaspersky Anti-Virus, while the remaining 14,951 files make up the gray list for testing. We use the sampling strategy in Section 6 to select 320,000 file samples from the labeled data collection for training. 133 files are marked by HAC as "confident" malware while 200 files are marked as "candidate" malware. Analyzing by the virus analysts, we find that there are 353 malware in the gray list. The results of the HAC's malware detection ability are shown in Table 7. However, Linear SVM classifies 8,012 files as malicious and AC selects 7,358 samples as malware, resulting very high false positive rates and very low precision values. As a consequence, they can not be applied for detecting malware from the gray list.

Table 7 Detection ability of HAC from the gray list

HAC	TP	FP	Precision (%)	Recall (%)
"Confident"	128	5	96.2406	36.2606
"Candidate"	123	77	61.5	34.8442

Trend Micro, Inc.
EX1017-18

From the above study, we observe that HAC can effectively detect malware from the gray list which cannot be recognized by the widely-used anti-virus scanners. In addition, the detection by HAC can be done very efficiently using a couple of minutes. A virus analyst has to spend 40 days to analyze the above gray list since he/she can analyze 200 file samples per day. The high efficiency of HAC can greatly save human labor.

8 Conclusion

In this paper, resting on the analysis of Windows APIs called by PE files, we develop the Hierarchical Associative Classifier (HAC) to help the virus analysts quickly and efficiently detect the malware from the large and imbalanced gray list with high complexity and overlapping.

Our main contribution can be summarized as follows: (1) We analyze the characteristics of the gray list and find that it is a very large and imbalanced data set with high complexity and overlapping; (2) We perform case studies on a very large data collection including 16 million labeled file samples and investigate the sampling strategy adaptable to the real application; (3) We adapt several post-processing techniques for building concise yet effective associative classifiers and propose a novel classification method, Hierarchical Associative Classifier (HAC) for imbalanced gray list prediction; (4) We provide a comprehensive experimental study and compare our HAC method with other alternative solutions for malware detection; (5) Our HAC method has been incorporated into Kingsoft's Anti-Virus software.

In our future work, we plan to extend our work from the following aspects: (1) Investigate alternative representations of the malware samples, such as the features of call graph and dynamic sequence, to better immune or resistant to mimicry (Wagner and Soto 2002) designed to confuse our detection system; (2) The class of polymorphic behavior is greatly spread and varied to model effectively. Hence modeling normal content might be a more promising defense mechanism than modeling malicious or abnormal content (Song et al. 2007). In our future work, we will further investigate the techniques of whitelist.

Acknowledgements The work of Y. Ye, K. Huang and Q. Jiang is supported by the National Science Foundation of China under Grant No.10771176. The work of T. Li is supported in part by the US National Science Foundation Under grant IIS-0546280 and by a 2007 IBM Faculty Award. The authors would also like to thank the members in the Anti-virus Lab at Kingsoft Corporation for their helpful discussions and suggestions.

References

Agrawal, R., & Imielinski, T. (1993). Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*.

Baralis, E., & Torino, P. (2002). A lazy approach to pruning classification rules. In *Proceedings of the 2002 IEEE international conference on data mining*.

Bayardo, R., Agrawal, R., & Gunopulos, D. (1999). Constraint-based rule mining in large, dense databases. In *ICDE-99*.

Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. *Journal in Computer Virology, 2*, 67–77.

Trend Micro, Inc.
EX1017-19

Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic minority oversampling technique. *Journal of Artificial Intelligence Research, 16*, 321–357.

Christodorescu, M., Jha, S., & Kruegel, C. (2007). Mining specifications of malicious behavior. In *Proceedings of ESEC/FSE07* (pp. 5–14).

Drummond, C., & Holte, C. (2003). C4.5, class imbalance, and cost sensitivity: Why undersampling beats oversampling. In *Proc. of the ICML-2003 workshop: Learning with imbalanced data sets II* (pp. 1–8).

Elkan, C. (2001). The foundations of cost-sensitive learning. In *Proceedings of the seventeenth international conference on machine learning* (pp. 239–246).

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of SIGMOD* (pp. 1–12).

Hsu, C., & Lin, C. (2002). A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks, 13*, 415–425.

Jain, A., Duin, R., & Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*, 4–37.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. In *Intelligent Data Analysis, 6*(5), 429C450.

Joshi, M., Agarwal, R., & Kumar, V. (2001). Mining needles in a haystack: Classifying rare classes via two-phase rule induction. In *SIGMOD 01 conference on management of data* (pp. 91–102).

Kamei, R., & Ralescu, A. (2003). Piecewise linear separanility using support vector machines. In *Proc. of the MAICS conference* (pp. 52C53).

Kephart, J., Sorkin, G., Swimmer, M., & White, S. (1997). Blueprint for a computer immune system. In *Virus Bulletin International Conference*.

Kephart, J., White, S., & Chess, D. (1993). Computers and epidemiology. *IEEE Spectrum, 30*, 20–26.

Kolter, J., & Maloof, M. (2004). Learning to detect malicious executables in the wild. In *Proceedings of KDD'04*.

Langley, P. (1994). Selection of relevant features in machine learning. In *Proceedings of AAAI fall symp.*

Li, W., Han, J., & Pei, J. (2001). CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the 2001 IEEE international conference on data mining (ICDM-01)* (pp. 369–376).

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating classification and association rule mining. In *Proceedings of the fourth international conference on knowledge discovery and data mining (KDD-98)* (pp. 80–86).

Liu, B., Hsu, W., & Ma, Y. (1999). Pruning and summarizing the discovered associations. In *KDD-99*.

Mahta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. In *EDBT-96*.

Peng, H., Long, F., & Ding, C. (2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*, 1226–1238.

Schultz, M., Eskin, E., & Zadok, E. (2001). Data mining methods for detection of new malicious executables. In *Security and privacy, 2001. Proceedings. 2001 IEEE symposium on 14–16 May* (pp. 38–49).

Song, Y., Locasto, M., Stavrou, A., Keromytis, A., & Stolfo, S. (2007). On the infeasibility of modeling polymorphic shellcode. In *CCS '07: Proceedings of the 14th ACM conference on computer and communications security* (pp. 541–551). New York: ACM.

Sung, A., Xu, J., Chavez, P., & Mukkamala, S. (2004). Static analyzer of vicious executables (save). In *Proceedings of the 20th annual computer security applications conference*.

Tesauro, G., Kephart, J., & Sorkin, G. (1996). Neural networks for computer virus recognition. *IEEE Expert, 11*, 5–6.

Thabtah, F. (2007). A review of associative classification mining. *The Knowledge Engineering Review, 22*(1), 37C65.

Toivonen, H., Klemetinen, M., Ronkainen, P., Hatonen, K., & Mannila, H. (1995). Pruning and grouping discovered association rules. In *Mlnet workshop on statistics, machine learning, and discovery in databases* (pp. 47–52).

Visa, S., & Ralescu, A. (2003). Learning imbalanced and overlapping classes using fuzzy sets. In *Proc. of the ICML-2003 workshop: Learning with imbalanced data sets II* (p. 97C104).

Visa, S., & Ralescu, A. (2005). Issues in mining imbalanced data sets—A review paper. In *Proceedings of the sixteen midwest artificial intelligence and cognitive science conference*.

Trend Micro, Inc.
EX1017-20

Wagner, D., & Soto, P. (2002). Mimicry attacks on host-based intrusion detection systems. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security* (pp. 255–264). New York: ACM.

Wang, J., Deng, P., Fan, Y., Jaw, L., & Liu, Y. (2003). Virus detection using data mining techniques. In *Proceedings of IEEE international conference on data mining*.

Wang, Y., Xin, Q., & Coenen, F. (2007). A novel rule ordering approach in classification association rule mining. In *Proceedings of the seventh IEEE international conference on data mining workshops (ICDMW 2007)*.

Weiss, G. (2004). Mining with rarity: A unifying framework. In *SIGKDD Explorations* (Vol. 6, No. 1, pp. 7–19).

Weiss, G., & Provost, F. (2003). Learning when training data are costly: The effect of class distribution on tree induction. *Journal of Artificial Intelligence Research, 19*, 315–354.

Ye, Y., Wang, D., Li, T., & Ye, D. (2007). IMDS: Intelligent malware detection system. In *Proceedings of ACM international conference on knowlege discovery and data mining (SIGKDD 2007)*.

Yin, X., & Han, J. (2003). CPAR: Classification based on predictive association rules. In *Proceedings of the third SIAM international conference on data mining (SDM-03)* (pp. 331–335).

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *Proceedings of 21st international conference on machine learning*.

Trend Micro, Inc.
EX1017-21

View publication stats