Intelligent File Scoring System for Malware Detection from the Gray List

Yanfang Ye*
Dept. of Computer Science
Xiamen University
Xiamen, 361005, P.R. China
yeyanfang@yahoo.com.cn

Tao Li
School of Computer Science
Florida International University
Miami, FL 33199
taoli@cs.fiu.edu

Qingshan Jiang
Software School
Xiamen University
Xiamen, 361005, P.R. China
qjiang@xmu.edu.cn

Zhixue Han
Dept. of Computer Science
Xiamen University
Xiamen, 361005, P.R. China

Li Wan
Anti-virus laboratory
KingSoft Corporation
Zhuhai, 519000, P.R.China
wanli@kingsoft.com

ABSTRACT

Currently, the most significant line of defense against malware is anti-virus products which focus on authenticating valid software from a white list, blocking invalid software from a black list, 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. In this paper, we develop an intelligent file scoring system (IFSS for short) for malware detection from the gray list by an ensemble of heterogeneous base-level classifiers derived by different learning methods, using different feature representations on dynamic training sets. To the best of our knowledge, this is the first work of applying such ensemble methods for malware detection. IFSS makes it practical for virus analysts to identify malware samples from the huge gray list and improves the detection ability of anti-virus software. It has already been incorporated into the scanning tool of Kingsoft's Anti-Virus software. The case studies on large and real daily collection of the gray list illustrate that the detection ability and efficiency of our IFSS system outperforms other popular scanning tools such as NOD32 and Kaspersky.

*The author is also affiliated with Anti-virus laboratory, KingSoft Corporation.

KDD'09, June 28–July 1, 2009, Paris, France.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; D.4.6 [**Operating System**]: Security and Protection - Invasive software

General Terms

Algorithms, Experimentation, Security

Keywords

malware detection, gray list, ensemble

1. INTRODUCTION

1.1 Malware Detection from the Gray List

Malware is a generic term [5] to denote all kinds of unwanted software(e.g., viruses, backdoors, spyware, trojans and worms). 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 asNOD32, Kaspersky 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.

In order to capture as many malware samples as possible, besides authenticating valid software from a white list and blocking invalid software from a black list 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 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. 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 very 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 and detect malware samples.

Over the last few years, many research efforts have been conducted on developing intelligent malware detection systems [19, 30, 33, 26]. In these systems, the detection process is generally divided into two steps: *feature extraction* and *categorization*. In the first step, various features such as Application Programming Interface (API) calls and program strings are extracted to capture the characteristics of the file samples. In the second step, intelligent techniques such as decision trees are used to automatically categorize the file samples into different classes based on computational analysis of the feature representations. These intelligent malware detection systems are varied in their use of feature representations and categorization methods. For example, IMDS [33] performs association classification on Windows API calls extracted from executable files while Naive Bayes methods on the extracted strings and byte sequences are applied in [26].

1.2 Contributions of The Paper

Different feature representations and categorization methods have their own advantages and limitations in malware detection. None of the single feature set can immune or resistant to mimicry designed to confuse the anti-virus software as different feature representation typically capture different characteristics of file samples. For example, API calls typically reflect the behavior of program code pieces while program strings consist of reused code fragments, author signatures, files names and system resource information. On the other hand, different categorization methods have different strengths and may excel at different situations. A natural question arises: can we combine different feature representations and categorization methods to improve the performance of malware detection?

Previous research has shown that ensemble methods, by combining multiple input systems, are a popular way to overcome instability and increase performance in many machine learning tasks, such as classification, clustering and ranking [9, 11]. In this paper, we develop an intelligent file scoring system (IFSS for short) for malware detection from the gray list by an ensemble of heterogeneous base-level classifiers derived by different learning methods, using different feature representations on dynamic training sets. To the best of our knowledge, this is the first work of applying such ensemble methods for malware detection.

Our IFSS system has the following major traits:

- *Diverse feature representations:* Two sets of extracted features, API calls and interpretable string, are used in our system. API calls reflect the behavior of program code pieces and the interpretable strings carry semantic interpretations and reflect an attacker's intent and goal. For example, (1) the API call like"GetVersionExA" in "KERNEL32.DLL" actually executes the function of obtaining extended information about the version of the recently running operating system; (2) the string of "$<script\ language='javascript'>window.open('readme.eml')$" always exists in the worms of "Nimda" and implicates that they try to infect the scripts.

- *Dynamic training sets:* Note that malware techniques are constantly evolving and new malware samples are produced on a daily basis. To account for the temporal trends of malware writing, our IFSS system makes use of two different datasets for training purpose: DB_T1 which consists of file samples from the historical data collection and DB_T2 which contains most recent file samples. The training sets are dynamically changing to include new samples while retaining the characteristics of historical data. In addition, training on different training sets also helps to increase the diversity of individual classifiers.

- *Heterogeneous base classifiers:* Associative classifiers and support vector machines have been chosen as our base classifiers. Both classifiers have been successfully used in malware detection [33, 19] and have distinct properties.

- *Human-in-the-Loop:* Our system provides a user-friendly mechanism for incorporating the expert knowledge and expertise of virus analysts. It should be pointed that in many cases, classifying a file sample from the gray list as malware will still be the prerogative of virus analysts. Our IFSS system uses a simple voting scheme to combine the prediction of individual classifiers and produces a file scoring list which is simple for virus analysts to interpret and understand. Virus analysts can then look at the top ranked file samples and manually authenticated and rejected those samples. New labeled samples can then be used to update the training sets.

- *Simultaneous model construction and testing:* With the dynamic training sets and human-in-the-Loop, our IFSS system performs simultaneous model construction and testing in malware detection, an environment and a task that is constantly evolving over the time. In our IFSS system the following three steps are iteratively conducted on the daily basis: 1) classifiers are first constructed to classify/rank the file samples in the gray list; 2) virus analysts manually analyze these top ranked samples; and 3) labeled samples are then used to dynamically generate new training sets.

All these traits make our IFSS a practical solution for helping virus analysts identify malware samples in the gray list and improving the detection ability of anti-virus software. The case studies on large and real data collections collected by the Anti-virus Lab of Kingsoft corporation illustrate that: (1) After being scanned by all the popular anti-virus software products, such as NOD32 and Kaspersky, malware in the gray list still can be effectively detected by our IFSS. (2) The performance and efficiency of our IFSS outperform other classification methods in detecting malware from the gray list. (3) Our IFSS reduces the number of file samples that need to be analyzed by virus analysts. Our case studies show that the percentage of malware samples in the gray list is about 0.5% while the percentage of malware samples in the top 100 ranked files samples of the file scoring list generated by our IFSS system is 35%. Therefore IFSS can greatly save human labor. As a result, our IFSS has already been incorporated into the scanning tool of Kingsoft's Anti-Virus software.

1.3 Organization of The Paper

The rest of this paper is organized as follows. Section 2 gives an overview of our IFSS system and Section 3 discusses

the related work. Section 4 describes the feature representation and extraction; Section 5 introduces the two base classifiers; Section 6 presents the ensemble framework used in our IFSS system for generating the file scoring list. In Section 7, we systematically evaluate the effects and efficiency of our IFSS system in comparison with other classification methods. In Section 8, based on the daily data collection obtained from Kingsoft Anti-virus lab, we examine the detection ability and efficiency of IFSS in comparison with other popular anti-virus software such as NOD32 and Kaspersky. Finally, Section 9 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 and interpretable strings which carry semantic interpretations and reflect an attacker's intent and goal, we develop the Intelligent File Scoring System(IFSS) to detect malware from the gray list. Figure 1 shows the malware detection procedure of IFSS:

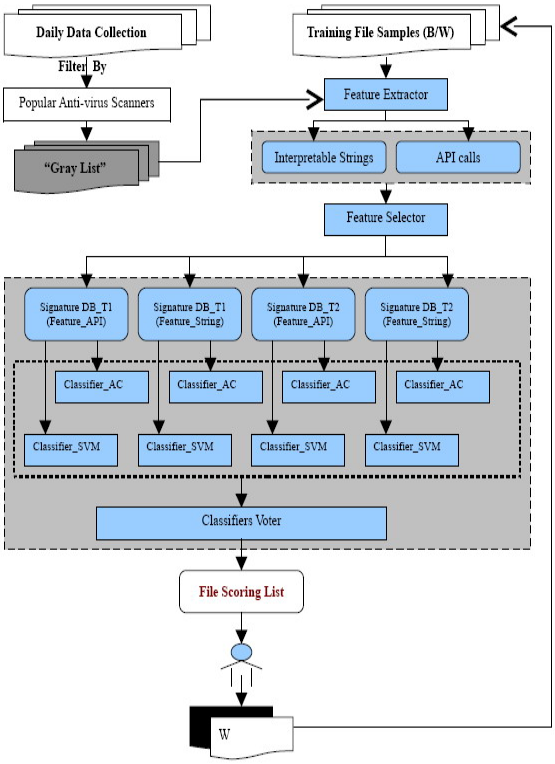
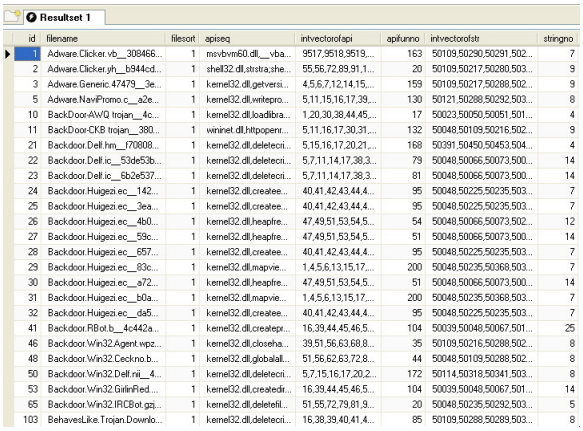


Figure 1: Malware detection flow of IFSS

- **Training:**

  1. Feature Extractor: IFSS first uses the feature extractor to extract the API calls and interpretable strings from the collected Windows Portable Executable (PE) files of "black list" and "white list", 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. A sample signature database is shown in Figure 2, in which there are 8 fields: record ID, PE file name, file type ("0" represents benign file while "1" is for malicious file), called APIs name, called API ID, the total number of called API functions, string ID, and the total number of interpretable strings. The transaction data can also be easily converted to relational data if necessary.


Resultset 1

id	filename	filesort	apiseq	intvectorofapi	apifunno	intvectorofstr	stringno
1	Adware.Clicker.vb__308468...	1	msvbvm60.dll,__vba...	9517,9518,9519,...	163	50109,50290,50291,502...	7
2	Adware.Clicker.yh__b944cd...	1	shell32.dll,strstra;she...	55,56,72,89,91,1...	20	50109,50217,50280,503...	9
3	Adware.Generic.47479__3e...	1	kernel32.dll,getversi...	4,5,6,7,12,14,15,...	159	50109,50217,50288,502...	9
5	Adware.NaviPromo.c__a2e...	1	kernel32.dll,writepro...	5,11,15,16,17,39,...	130	50121,50288,50292,503...	8
10	BackDoor-AWQ trojan__4c...	1	kernel32.dll,loadlibra...	1,20,30,38,44,45,...	17	50023,50050,50051,501...	4
11	BackDoor-CKB trojan__380...	1	wininet.dll,httpopenr...	5,11,16,17,30,31,...	132	50048,50109,50216,502...	9
21	Backdoor.Delf.hm__f70808...	1	kernel32.dll,deleteci...	5,15,16,17,20,21,...	168	50391,50450,50453,504...	4
22	Backdoor.Delf.ic__53de53b...	1	kernel32.dll,deleteci...	5,7,11,14,17,38,3...	79	50048,50066,50073,500...	14
23	Backdoor.Delf.ic__6b2e537...	1	kernel32.dll,deleteci...	5,7,11,14,17,38,3...	81	50048,50066,50073,500...	14
24	Backdoor.Huigezi.ec__142...	1	kernel32.dll,createe...	40,41,42,43,44,4...	95	50048,50225,50235,503...	7
25	Backdoor.Huigezi.ec__3ea...	1	kernel32.dll,createe...	40,41,42,43,44,4...	95	50048,50225,50235,503...	7
26	Backdoor.Huigezi.ec__4b0...	1	kernel32.dll,heapfre...	47,49,51,53,54,5...	54	50048,50066,50073,502...	12
27	Backdoor.Huigezi.ec__59c...	1	kernel32.dll,heapfre...	47,49,51,53,54,5...	51	50048,50066,50073,500...	14
28	Backdoor.Huigezi.ec__657...	1	kernel32.dll,createe...	40,41,42,43,44,4...	95	50048,50225,50235,503...	7
29	Backdoor.Huigezi.ec__83c...	1	kernel32.dll,mapvie...	1,4,5,6,13,15,17,...	200	50048,50235,50368,503...	7
30	Backdoor.Huigezi.ec__a72...	1	kernel32.dll,heapfre...	47,49,51,53,54,5...	51	50048,50066,50073,500...	14
31	Backdoor.Huigezi.ec__b0a...	1	kernel32.dll,mapvie...	1,4,5,6,13,15,17,...	200	50048,50235,50368,503...	7
32	Backdoor.Huigezi.ec__da5...	1	kernel32.dll,createe...	40,41,42,43,44,4...	95	50048,50225,50235,503...	7
41	Backdoor.RiBot.b__4c442a...	1	kernel32.dll,createpr...	16,39,44,45,46,5...	104	50039,50048,50067,501...	25
46	Backdoor.Win32.Agent.wpz...	1	kernel32.dll,closeha...	39,51,56,63,68,8...	35	50109,50216,50288,502...	8
48	Backdoor.Win32.Ceckno.b...	1	kernel32.dll,globalall...	51,56,62,63,72,8...	44	50048,50109,50288,502...	8
50	Backdoor.Win32.Delf.nii__4...	1	kernel32.dll,deleteci...	5,7,15,16,17,20,2...	172	50114,50318,50341,503...	8
53	Backdoor.Win32.GirlinRed...	1	kernel32.dll,createdir...	16,39,44,45,46,5...	104	50039,50048,50067,501...	14
65	Backdoor.Win32.IRCBot.gzj...	1	kernel32.dll,deletefil...	51,55,72,79,81,9...	20	50048,50235,50292,503...	5
103	BehavesLike.Trojan.Downlo...	1	kernel32.dll,deleteci...	16,38,39,40,41,4...	85	50109,50288,50289,503...	8

Figure 2: A sample signature database after data transformation

  2. Feature Selector: Feature selection is important for many pattern classification systems. As not all of the extracted features are contributing to malware detection, feature selector is used to identify the most representative features.

  3. Base Classifiers: Base classifiers are constructed by applying associative classifier and SVM using different feature representations on different training sets (denoted by DB_T1 and DB_T2). Coupled with the two different feature representations, we have four different combinations of training sets and feature representations: DB_T1 with API calls, DB_T1 with interpretable strings, DB_T2 with API calls, and DB_T2 with interpretable strings. Using the two different classification methods, we thus obtain 8 different base classifiers.

- **Malware Detection from the Gray List**

  The daily collection of file samples is first scanned by the existing popular anti-virus software products. Valid software programs from a white list are authenticated and invalid software programs from a black list are blocked or rejected. The gray list, containing unknown software programs which could be either normal or malicious, is then fed into our IFSS system. After feature extraction and selection, 8 different classifiers are applied to the gray list. A simple voting scheme is used to combine base classifiers and generate a file scoring list. The file score list ranks the input file samples from the gray list. Virus analysts can then look at the top ranked file samples and manually authenticated and rejected those samples. These manually labeled file samples can then be used to update the training sets.

3. RELATED WORK

3.1 Data Mining Methods for Malware Detection

Signature-based methods are widely used in malware detection to recognize threats [12]. A signature is a short string of bytes which is unique for each known malware. However, this classic signature-based method always fails to detect variants of known malware or previously unknown malware. The problem lies in the signature extraction and generation process, and in fact these signatures can be easily bypassed [27]. 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 [19, 26, 30, 7]. The performance of such methods used for malware detection critically depend on the set of features and the classifier [10].

Neural Networks as well as immune system are used by IBM for computer virus recognition [28]. 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 [19, 26, 30]. Recently, associative classification [22], with its ability to utilize relationships among attributes, has been also applied in [33]. Note that the class distribution in the gray list of our collection is quite imbalanced with the malware samples as the minority class. Many accuracy driven classifiers may fail on such a large and imbalanced gray list. For example, neural networks and naive Bayes consistently biased towards the majority class at any given size and prone to treat the minority (malware) class as noise [18]. 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 [18]. Hence in our work, we choose association classifier and SVM as our base classifiers.

3.2 Ensemble Classification

Previous research has shown that ensemble methods, by combining multiple input systems, are a popular way to overcome instability and increase performance in many machine learning tasks, such as classification, clustering and ranking. For example, an ensemble of classifiers is a set of classifiers whose individual predictions are combined in some way (typically by voting) to classify new examples. Generally there are two types of classifier ensemble: 1) Homogeneous ensemble: the base classifiers are constructed using a single learning algorithm, such as decision trees or neural networks [9]. Typically base classifiers are generated by manipulating the training set (as done in boosting or bagging), manipulating the input features, manipulating the output targets or injecting randomness in the learning algorithm [8]. The individual classifiers are then typically combined by voting or weighted voting. 2) Heterogeneous ensemble: the base classifiers are constructed by applying different learning algorithms (with heterogeneous model representations) to a single dataset [24]. More complicated methods such as stacking are used for combining classifiers [32]. In our IFSS system, the base classifiers are constructed by different learning methods (association classification or SVM), using different feature representations (API calls or Interpretable strings) on different training sets (DB_T1 and DB_T2). We expect that our construction of base classifiers would increase their diversity and improve the classification performance. Our work is the first effort on applying such ensemble classifier methods for malware detection.

4. FEATURE EXTRACTION AND SELECTION

Our IFSS system is performed directly on Windows PE code. PE is designed as a common file format for all flavor of Windows operating system, and PE malware are in the majority of the malware rising in recent years. If a PE file is previously compressed by a third party binary compress tool such as UPX and ASPack Shell or embedded a homemade packer, it needs to be decompressed first. We use the dissembler W32Dasm developed by KingSoft Anti-Virus Laboratory to dissemble the PE code and output the assembly instructions as the input for feature extraction.

4.1 Feature Extraction

API Calls: The Windows API execution calls for each benign/malicious executable is generated by a PE parser. Through the API query database, the API execution calls generated by the PE parser can be converted to a group of 32-bit global IDs which represents the static execution calls of the corresponding API functions. For example, the API "KERNEL32.DLL, OpenProcess" executes the function that returns a handle to an existing process object and it can be encoded as 0x00500E16. Then we use the API calls as the signatures of the PE files and store them in the signature database.

Interpretable Strings: The interpretable strings are extracted using a feature parser. The feature parser reads the PE file. If there is a sequence of consecutive bytes belonging to the same Character Set, such as ASCII, GB2312, Big5 and Unicode, then the parser exacts them as our features. Figure 3 shows a sample interpretable strings extracted by our feature parser. These strings are extracted from a malware named $Backdoor - Redgirl.exe$. From Figure 3, we can see the behaviors of the malware and the attacker's intent explicitly.

Since these two sets of features are representation of PE file samples at different semantic levels, we use them for building base classifier respectively.

4.2 Feature Selection

API Calls: As not all of the API calls are contributing to malware detection, we rank each API call using Max-Relevance algorithm [25] and select a set of API calls with the highest relevance to the target class, i.e. the file type, for later classification. Given a_i which represents the API with ID i, and the file type f ("0" represents benign executables and "1" is for malicious executables), their mutual information is defined in terms of their frequencies of appearances $p(a_i)$, $p(f)$, and $p(a_i, f)$ as follows.

$$I(a_i, f) = \int \int p(a_i, f) log \frac{p(a_i, f)}{p(a_i)p(f)} d(a_i)d(f)$$

With this algorithm, we select the top m APIs in the descent order of $I(a_i, f)$, i.e. the best m individual features correlated to the file types of the PE files.

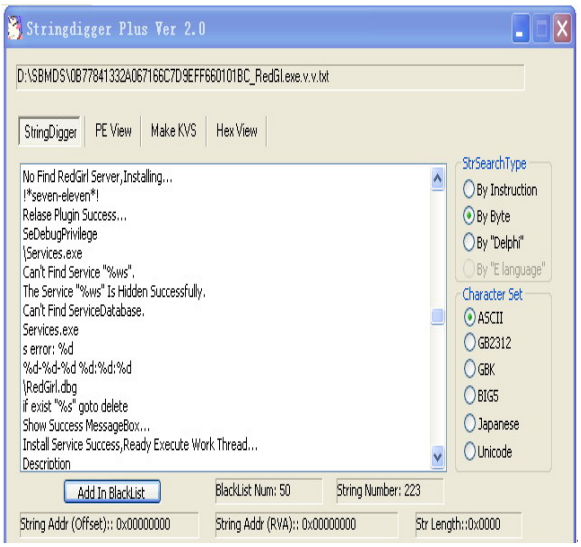


Figure 3: Interpretable strings sample extracted by feature parser

Interpretable Strings: For Interpretable strings, we first use the corpus of natural language to filter the candidate interpretable strings. If the string consists most of the unusual characters which are not in the corpus, like "!0&0h0m0o0t0y0", it will be pruned by our feature parser. We then also apply Max-Relevance algorithm [25] to select a set of the most representative strings for later classification.

5. BASE CLASSIFIERS

In this section, we briefly describe the base classifiers used in our IFSS. We use association classifier and SVM as our base classifiers for the following reasons: 1) The gray list is large and quite imbalanced and many accuracy driven classifiers including neural networks, naive Bayes and decision trees may fail on such a large and imbalanced gray list [18]. On the other hand, association classifier and SVM seems work well on imbalanced datasets. 2) Both associative classification and SVM have been successfully applied in malware detection [33, 19]. In particular, association classification can discover interesting relationships among input features that are explicitly related to malware/benign file class and SVM can identify good classification boundaries for malware detection.

5.1 Association Classification

5.1.1 Introduction

For malware detection in this paper, the first goal is to find out how a set of input features (e.g., 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 $oc \geq moc$ where os and oc are the support and confidence of the rule.

Apriori [1] and FP-Growth [13] algorithms can be extended to associative classification [21, 22]. 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.

5.1.2 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 classification is also very important for improving the accuracy and efficiency of the classifier. Rule pruning and rule re-ordering are used for post-processing associative classifier.

Rule Pruning. Several common rule pruning approaches have been developed for associative classifiers to reduce the generated rules [3, 4, 21, 22, 23, 29]: (1) χ^2 (chi-square) testing [21] to measure the significance of the rule itself, (2) database coverage [22] to just keep the rules covering at least one training data object not considered by a higher ranked rule, and (3) pessimistic error estimation [22] 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.

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 [22, 21, 34]. In particular, CBA [22] and CMAR [21] 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 [31]: (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 [34], 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.

We use "Best First Rule" [31] 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.

5.2 Support Vector Machine

Support Vector Machine (SVM) is a promising method for data classification and regression and it has also been successfully used in malware detection [16, 2, 15]. The key to the success of SVM is the kernel function which maps the data from the original space into a high dimensional feature space. By constructing a linear boundary in the feature space, the SVM produces nonlinear boundaries in the original space. The output of a linear SVM is $u = w \times x - b$, where w is the normal weight vector to the hyperplane and x is the input vector. Maximizing the margin can be seen as an optimization problem:

$$\text{minimize } \frac{1}{2}\|w\|^2, \text{ subject to } y_i(w \cdot x + b) \geq 1, \forall i,$$

where x is the training example and y_i is the correct output for the i_{th} training example. Intuitively the classifier with the largest margin will give low expected risk, and hence better generalization.

6. ENSEMBLE CLASSIFIER

Base classifiers are constructed by applying associative classifier and SVM using different feature representations on different training sets (denoted by DB_T1 and DB_T2). Coupled with the two different feature representations, we have four different settings for training base classifiers: DB_T1 with API calls, DB_T1 with interpretable strings, DB_T2 with API calls, and DB_T2 with interpretable strings. Using the two different classification methods, we thus obtain 8 different base classifiers. A simple voting scheme is used to combine base classifiers. For an input file, each base classifier casts a vote for its prediction: i.e., 1 if the input file is predicted to be malicious and 0 otherwise. Therefore after classifier voters, each file sample can obtain a score ranging from 8 to 0. If two file sample have the same score, they will be ranked by their matching association classification rules' χ^2 (chi-square) values [21] in descending order. IFSS system then generates a file scoring list which is a ranked list of all input file samples from the gray list. The file score list is simple for virus analysts to interpret and understand. Virus analysts can then look at the top ranked file samples and manually authenticated and rejected those samples. In practice, each virus analyst can analyze 20 new file samples per day and they pick the top 100 file samples from the file scoring list for manual inspection. These manually labeled file samples can then be used as new training data to improve the system.

7. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we conduct two sets of experimental studies using our data collection obtained from the Anti-virus Lab of Kingsoft to compare our IFSS with other classifiers: (1) The first set of experiments is to investigate the effects of feature selection. (2) In the second set of experiments, we compare our IFSS with different ensemble methods obtained using different combinations of feature representations, training sets and base classifiers. 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 Feature Selection

Identifying the most representative features is critical to improve the performance and efficiency of the classifiers [17, 20]. As not all of the features contributing to malware detection, we rank each API call and interpretable string using Max-Relevance algorithm [25] and select top k API calls and interpretable strings as the features for later classification. We obtain a whole week's data collection(from Jan. 1st, 2009 to Jan. 7th, 2009) from Kingsoft Anti-virus lab to testify the validation of the feature selection method in this set of experiments. We use six days' data collection containing 530,448 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 one day's samples including 89,626 files for testing. There are 7,909 API calls and 32,123 interpretable strings extracted from these file samples. We use *precision*[6] and *recall*[6] of the malware class to evaluate the performance of the classification results, 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. Figure 4 and Figure 5 show that the testing performance of Associative Classifier(AC) changes slightly after the number of API calls reaches 100 and the number of interpretable strings reaches 500. So, we select top 100 API calls and top 500 interpretable strings respectively as the features for later classification.

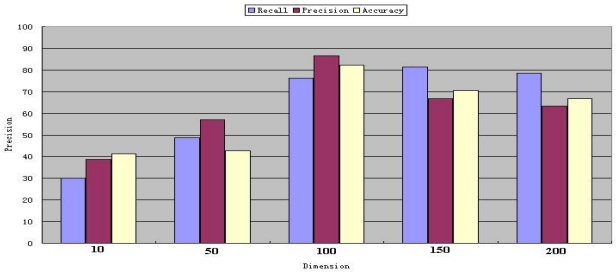


Figure 4: AC performance with different number of API Calls

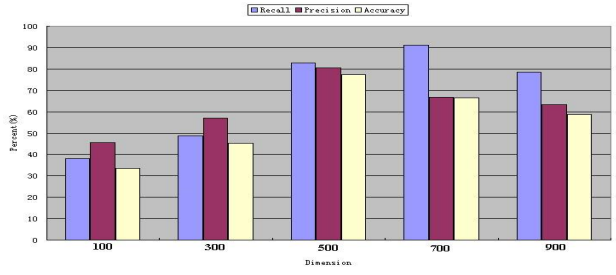


Figure 5: AC performance with different number of Strings

Feature-Classifier-Train	TP	TP+FP	Recall	Precision
C1:API-AC-DB_T1	13	1244	21.31%	1.05%
C2:API-AC-DB_T2	13	896	21.31%	1.45%
C3:STR-AC-DB_T1	30	4173	49.18%	0.72%
C4:STR-AC-DB_T2	9	444	14.75%	2.03%
C5:API-SVM-DB_T1	13	747	21.31%	1.74%
C6:API-SVM-DB_T2	13	568	21.31%	2.29%
C7:STR-SVM-DB_T1	29	3266	47.54%	0.89%
C8:STR-SVM-DB_T2	30	803	49.18%	3.74%

Table 1: Detection results of different base classifiers on different training sets using different feature representations. The test data is from the gray list of Jan. 8th, 2009.

Ensemble	TP	Recall	Precision	F1
E1:C1+C2	2	3.28%	2%	0.0248
E2:C3+C4	3	4.92%	3%	0.0373
E3:C5+C6	3	4.92%	3%	0.0373
E4:C7+C8	4	6.56%	4%	0.0497
E5:C1+C2+C3+C4	12	19.67%	12%	0.1491
E6:C5+C6+C7+C8	9	14.75%	9%	0.1118
IFSS:C1-C8	**35**	**57.38%**	**35%**	**0.4348**
Human Expert	2	3.28%	2%	0.0248

Table 2: Detection results of different ensembles. Remarks: We select the top 100 files from the ranking list generated by each ensemble according to the simplest voting and ranking mechanism described in Section 6 and evaluate the performances of different ensembles. For comparison purpose, our virus analysts also select 100 files from the gray list to analyze.

7.2 Comparisons of Different Classification Methods

In this set of experiments, we compare our IFSS with different ensemble methods obtained using different combinations of feature representations, training sets and base classifiers. In particular, we use: (1) API calls and interpretable strings as diverse features, (2) DB_T1 consisting of 491,733 PE file samples obtained from the history data set of Kingsoft Anti-virus lab, and DB_T2 containing 530,448 PE files which is the data collection of the week from Jan. 1st, 2009 to Jan. 7th, 2009, (3) associative classifier described in Section 5.1 and linear SVM [14] implemented in LibLinear package as heterogenous base classifiers, to construct different classifiers. To evaluate the performance of different classifiers and ensembles, we use the gray list of Jan. 8th, 2009 obtained at Kingsoft Anti-virus lab. We randomly sample 10% from the gray list for testing. The test dataset contains 12,365 files, 61 of which are malware samples.

Table 1 shows the detection results of different base classifiers on different training sets using different feature representations. From Table 1, we observe that the precision of each classifier is too low and the number of the file samples misclassified as malware is too large. Obviously, the single classification result is infeasible for real applications. Ensemble classifiers are quite popular in many data mining applications due to their potential for efficient parallel implementations and high accuracy.

Table 2 and Figure 6 show the detections results of different ensembles constructed by different combinations of the feature representations, training sets and base classifiers. In particular, E1-E4 are the ensemble methods constructed by a single classifier with a single feature representation on different training sets; E5-E6 are the ensemble methods constructed by a single classifier with diverse feature representations on different training sets. These methods are typical approaches for constructing ensembles. For comparison purpose, we also include the results of human expert. F1 measure, defined as $F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$, is also used to evaluate the classification performance of different algorithms. From the comparison, we observe that our IFSS outperforms other ensembles as well as human experts.

In addition, the detection by our IFSS can be done very efficiently using a couple of minutes (it uses 21.5 minutes to detect these 12,365 file samples, including feature extraction time). A virus analyst has to spend 5 days to analyze the 100 file samples in the gray list, since he/she can analyze 20 new file samples per day. Our case studies shows that the percentage of malware samples in the gray list is about 0.5% while the percentage of malware samples in the top 100 files samples of the file scoring list generated by our IFSS system is 35%. Because of its high efficiency and effectiveness, our IFSS system makes it practical for human experts to inspect the top rank files.

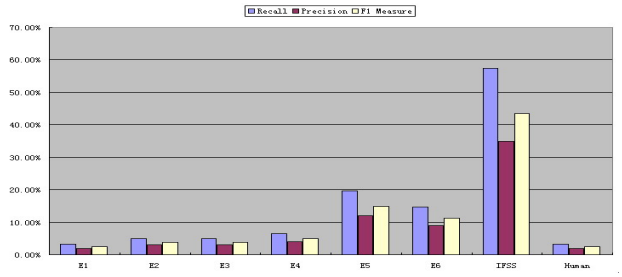


Figure 6: F1 measures of different ensembles based on part of the gray list of Jan. 8th, 2009.

8. REAL APPLICATION OF IFSS

It should be pointed out that our IFSS system for malware detection has already been incorporated into the product of Kingsoft's Anti-virus software. Figure 7 shows the interface of the IFSS system. We call the new scanning tool of Kingsoft's Anti-Virus software which incorporates IFSS system as KS-IFSS. The old scanning tool of Kingsoft's Anti-Virus software is referred as KS. The main purpose of IFSS is to help virus analysts find out malware samples in the gray list on which all other popular scanners fail and to improve the malware detection ability of anti-virus software. Therefore, we apply KS-IFSS in real applications and compare with other popular scanners(including KS) to testify its malware detection ability and efficiency on the daily data collection.

8.1 Detection Ability of KS-IFSS

In this section, we apply KS-IFSS in real applications to testify its detection ability of the daily data collection. Table 3 illustrates the daily data collection obtained from Kingsoft Anti-virus lab for the week of Jan. 25th, 2009 to Jan. 31st, 2009.

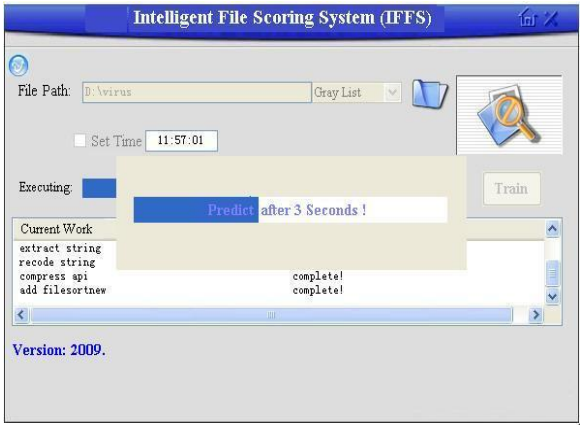


Figure 7: The interface of IFSS system.

Day	Date	All Files	Malware	Benign Files
1	2009/01/25	407,882	42,608	51,595
2	2009/01/26	516,715	44,204	59,245
3	2009/01/27	551,120	44,813	50,297
4	2009/01/28	597,767	47,796	38,982
5	2009/01/29	312,372	49,077	65,113
6	2009/01/30	520,761	57,144	66,022
7	2009/01/31	705,620	55,523	54,489
	Sum	3,612,237	341,165	385,723

Table 3: Daily data collection for the week of Jan. 25th, 2009 to Jan. 31st, 2009.

3,612,237 file samples are collected in total: 2,885,349 of which are gray files, 385,723 of which are benign files, and 341,165 of which are malware samples detected by all of the four anti-virus scanners. For the gray files, we just review the malware samples detected by KS-IFSS. We examine KS-IFSS's malware detection ability and FP(False Positive)rate which is the ratio of benign files misclassified as malicious in comparison with some of the popular scanning tools like NOD32, Kaspersky and KS. For comparison purpose, we use all of the Anti-virus scanners' newest versions of the base of signature on the same day. Table 4 and Figure 8 show that KS-IFSS outperforms other Anti-virus scanners on malware detection ability, since it can detect the malware from the gray list while all the popular scanners fail. Figure 9 shows that KS-IFSS outperforms other Anti-virus scanners on FP(False Positive)rate.

Day	Perf.	KS-IFSS	KS	NOD32	Kaspersky
1	DRate	91.53%	87.33%	81.56%	72.85%
2	DRate	91.98%	87.72%	82.69%	66.09%
3	DRate	91.28%	88.24%	83.42%	65.72%
4	DRate	92.84%	89.21%	84.95%	62.95%
5	DRate	92.89%	89.97%	86.27%	75.47%
6	DRate	93.02%	89.68%	88.66%	79.12%
7	DRate	91.76%	89.40%	85.37%	86.36%

Table 4: Malware detection results of different Anti-Virus Scanners. Remarks: DRate means the detection rate of the Anti-Virus Scanner which is the ratio of malware correctly classified.

8.2 Detection Efficiency of KS-IFSS

In this set of experiments, we compare the efficiency of our KS-IFSS with different Anti-virus scanners. We also use the daily malware data collection, from Jan. 25th, 2009 to Jan. 31st, 2009, described in Section 8.1 to testify the detection efficiency of each Anti-virus scanner. The results in Figure 10 illustrate that KS-IFSS achieves higher efficiency than other scanners when being executed in the same environment.

Figure 8: Comparisons of malware detection ability for different Anti-Virus Scanners.

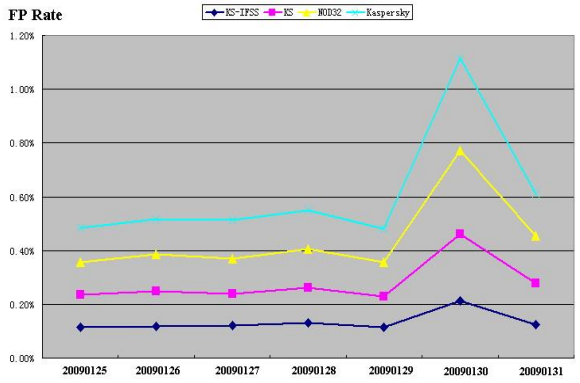
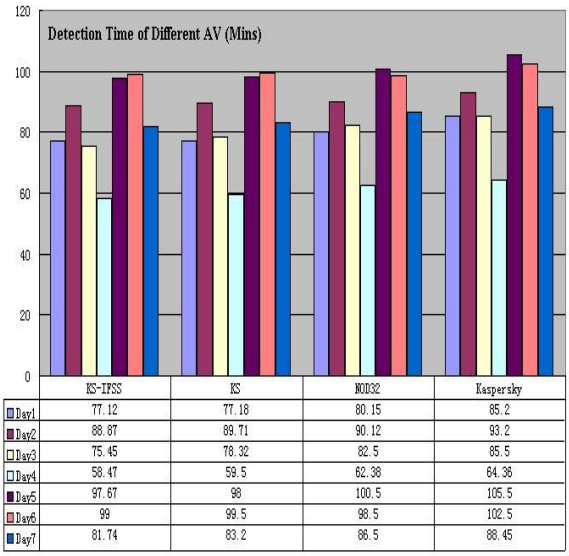


Figure 9: Comparisons of FP rate for different Anti-Virus Scanners.

9. CONCLUSION

In this paper, we present an an intelligent file scoring system (IFSS) for malware detection from gray list. IFSS uses an ensemble framework and it has several favorable traits including diverse feature representations, dynamic training sets, heterogeneous base classifiers and human-in-the-Loop. In addition, IFSS performs simultaneous model construction and testing. With these properties, IFSS makes it practical for virus analysts to identify malware samples from the huge gray list and improves the detection ability of anti-virus software.

IFSS has already been incorporated into the scanning tool of Kingsoft's Anti-Virus software. The case studies on large data collections on the the gray list and real daily data collection obtained from the Anti-virus Lab of Kingsoft corporation demonstrate the effectiveness and efficiency of our IFSS system.



	KS-IFSS	KS	NOD32	Kaspersky
Day1	77.12	77.18	80.15	85.2
Day2	88.87	89.71	90.12	93.2
Day3	75.45	78.32	82.5	85.5
Day4	58.47	59.5	62.38	64.36
Day5	97.67	98	100.5	105.5
Day6	99	99.5	98.5	102.5
Day7	81.74	83.2	86.5	88.45

Figure 10: Malware detection efficiency of different Anti-Virus Scanners

Acknowledgments

The authors would also like to thank the members in the Anti-virus Lab at Kingsoft Corporation for their helpful discussions and suggestions. The work of Y. Ye an Q. Jiang is partially supported by the National Science Foundation of China under Grant #10771176 and Guangdong Province Foundation under Grant #2008A090300017. The work of T. Li is supported in part by the US National Science Foundation Under grant IIS-0546280 and by multiple IBM Faculty Awards.

10. REFERENCES

[1] R. Agrawal and T. Imielinski. Mining association rules between sets of items in large databases. In *Proceedings of SIGMOD*, 1993.

[2] A.Kolcz, X.Sun, and J.Kalita. Efficient handling of high-dimensional feature spaces by randomized classifier ensembles. In *Proceedings of KDD'02*, 2002.

[3] E. Baralis and P. Torino. A lazy approach to pruning classification rules. In *Proceedings of ICDM'02*, 2002.

[4] R. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of ICDE'99*, 1999.

[5] U. Bayer, A. Moser, C. Kruegel, and E. Kirda. Dynamic analysis of malicious code. *J Comput Virol*, 2:67–77, May 2006.

[6] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[7] M. Christodorescu, S. Jha, and C. Kruegel. Mining specifications of malicious behavior. In *Proceedings of ESEC/FSE'07*, pages 5–14, 2007.

[8] Thomas G. Dietterich. Machine-learning research: Four current directions. *AI Magazine*, 18(4):97–136, 1997.

[9] Thomas G. Dietterich. Ensemble methods in machine learning. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 1–15.

[10] D.K.S.Reddy and A.K.Pujari. N-gram analysis for computer virus detection. *J Comput Virol*, 2:231–239, November 2006.

[11] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, 2004.

[12] Eric Filiol, Gregoire Jacob, and Michael Le Liard. Evaluation methodology and theoretical model for antiviral behavioural detection strategies. *Journal in Computer Virology*, 3(1):27–37, 2007.

[13] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of SIGMOD*, pages 1–12, May 2000.

[14] C. Hsu and C. Lin. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13:415–425, 2002.

[15] H.Yu, J.Yang, and J.Han. Classifying large data sets using svms with hierarchical clusters. In *Proceedings of KDD'03*, 2003.

[16] I.W.Tsang, J.T Kwok, and P.M.Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6:363–392, 2005.

[17] A. Jain, R. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:4–37, 2000.

[18] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. In *Intelligent data Analysis 6(5)*, pages 429–450, 2002.

[19] J. Kolter and M. Maloof. Learning to detect malicious executables in the wild. In *Proceedings of KDD'04*, 2004.

[20] P. Langley. Selection of relevant features in machine learning. In *Proceedings of AAAI Fall Symp.*, 1994.

[21] W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings ICDM'01)*, pages 369–376, 2001.

[22] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of KDD'98*, pages 80–86, 1998.

[23] M. Mahta, R. Agrawal, and J. Rissanen. SLIQ: A fast scalable classifier for data mining. In *EDBT-96*, 1996.

[24] Christopher J. Merz. Using correspondence analysis to combine classifiers. *Machine Learning*, 36(1-2):33–58, 1999.

[25] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27, 2005.

[26] M. Schultz, E. Eskin, and E. Zadok. Data mining methods for detection of new malicious executables. In *Proceedings. 2001 IEEE Symposium*, pages 38–49, 2001.

[27] A. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables (save). In *Proceedings of the 20th Annual Computer Security Applications Conference*, 2004.

[28] G.J. Tesauro, J.O. Kephart, and G.B. Sorkin. Neural networks for computer virus recognition. *IEEE Expert*, 11:5–6, 1996.

[29] F. Thabtah. A review of associative classification mining. In *The Knowledge Engineering Review, Vol. 22:1*, pages 37–65, 2007.

[30] J. Wang, P. Deng, Y. Fan, L. Jaw, and Y. Liu. Virus detection using data mining techniques. In *Proceedings ICDM'03*, 2003.

[31] Y. Wang, Q. Xin, and F. Coenen. A novel rule ordering approach in classification association rule mining. In *Proceedings of ICDM Workshop*, 2007.

[32] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[33] Y. Ye, D. Wang, T. Li, and D. Ye. IMDS: Intelligent malware detection system. In *Proccedings of KDD'07)*, 2007.

[34] X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proceedings of SIAM International Conference on Data Mining (SDM-03)*, pages 331–335, 2003.