J Comput Virol (2009) 5:283–293
DOI 10.1007/s11416-008-0108-y

ORIGINAL PAPER

SBMDS: an interpretable string based malware detection system using SVM ensemble with bagging

Yanfang Ye · Lifei Chen · Dingding Wang · Tao Li · Qingshan Jiang · Min Zhao

Received: 1 June 2008 / Revised: 23 October 2008 / Accepted: 5 November 2008 / Published online: 26 November 2008

Abstract Malicious executables are programs designed to infiltrate or damage a computer system without the owner's consent, which have become a serious threat to the security of computer systems. There is an urgent need for effective techniques to detect polymorphic, metamorphic and previously unseen malicious executables of which detection fails in most of the commercial anti-virus software. In this paper, we develop interpretable string based malware detection system (SBMDS), which is based on interpretable string analysis and uses support vector machine (SVM) ensemble with Bagging to classify the file samples and predict the exact types of the malware. Interpretable strings contain both application programming interface (API) execution calls and important semantic strings reflecting an attacker's intent and goal. Our SBMDS is carried out with four major steps: (1) first constructing the interpretable strings by developing a feature parser; (2) performing feature selection to select informative strings related to different types of malware; (3) followed by using SVM ensemble with bagging to construct the classifier; (4) and finally conducting the malware detector, which not only can detect whether a program is malicious or not, but also can predict the exact type of the malware. Our case study on the large collection of file samples collected by Kingsoft Anti-virus lab illustrate that: (1) The accuracy and efficiency of our SBMDS outperform several popular anti-virus software; (2) Based on the signatures of interpretable strings, our SBMDS outperforms data mining based detection systems which employ single SVM, Naive Bayes with bagging, Decision Trees with bagging; (3) Compared with the IMDS which utilizes the objective-oriented association (OOA) based classification on API calls, our SBMDS achieves better performance. Our SBMDS system has already been incorporated into the scanning tool of a commercial anti-virus software.

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

L. Chen
School of Mathematics and Computer Science,
Fujian Normal University, Fuzhou 350108,
People's Republic of China
e-mail: cliphy@sina.com

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

D. Wang
e-mail: dwang003@cs.fiu.edu

Q. Jiang
Software School, Xiamen University, Xiamen 361005,
People's Republic of China
e-mail: qjiang@xmu.edu.cn

M. Zhao
Anti-virus Laboratory, KingSoft Corporation, Zhuhai 519000,
People's Republic of China
e-mail: zhaomin@kingsoft.com

1 Introduction

Malicious executables are programs designed to infiltrate or damage a computer system without the owner's consent, which have become a serious threat to the security of computer systems. New, previously unseen malicious executables, polymorphic and metamorphic malicious executables which are generated by obfuscation techniques are more complex and difficult to detect. According to its propagation methods, a malicious code is usually classified into the following categories [1,10,20]: viruses, backdoors, spyware, trojan horses and worms. Malicious executables do not always exactly fit into these categories and the malicious code combining two or more categories can lead to powerful attacks. For instance, a worm containing a payload can install a back door to allow remote access. Due to the significant loss and damages induced by malicious executables, the malware detection becomes one of the most critical issues in the field of computer security.

Currently, the most important lines of defense against malware are anti-virus programs. These widely-used malware detection software tools use signature-based methods to recognize threats [11,12]. 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. In order to remain effective, it is of paramount importance for the anti-virus companies to be able to quickly analyze variants of known malware and previously unknown malware samples. Unfortunately, the number of samples that need to be analyzed on a daily basis is constantly increasing [2]. The virus analysts at Kingsoft Anti-Virus lab conclude that there are around 70,000 samples, which they called "gray list", needed to be analyzed per day. This clearly reveals the need for an automatic, efficient, and robust tool to classify the "gray list".

1.1 Interpretable string based malware detection

Recently there are a few attempts on applying data mining and machine learning techniques to detect new malicious executables [8,17,24,27,31]. The performance of these techniques critically depends on the set of features used to describe the executables and the classifier [23]. The "interpretable strings" are no longer simple printable strings [17,24], but API calls [27] from Import Table which can reflect the behavior of program code pieces and strings carrying semantic interpretations which can reflect an attacker's intent and goal. The virus analysts at Kingsoft anti-virus lab suggest that the interpretable strings are good static features since they can not only parse the possible behaviors of a malicious executable, but also capture the malware author's intent and goal.

For example, the string of "⟨html⟩ ⟨script language = 'javascript'⟩ window.open('readme.eml')" always exists in the worms of "Nimda" and implicates that they try to infect the scripts. Another example could be the string "'&gameid=%s&pass=%s; myparentthreadid=%d; myguid=%s" which indicate that the attacker intend to steal the password of the online game and send it back to the server. The strings, like "!0&0h0m0o0t0y0" and "*3d%3dtgyhjij", though they are printable, will not be extracted as the features of the file samples, since they can't be semantically interpreted. So far, we have gathered 39,838 executables, of which 8,320 are referred to as benign executables and 31,518 are malicious ones of four different types (i.e., backdoors, spyware, trojans and worms). All of these samples are obtained from Kingsoft anti-virus lab. From these executables we extract 963,556 interpretable strings in total.

In order to develop an automatic and efficient system for malware detection based on interpretable strings, we have to address the following four challenges:

- **Scalability**: The system must be scalable since we have thousands of file samples with hundreds of thousands of strings.
- **Generalization**: Using interpretable strings, each file sample is represented as a (usually sparse) vector in a very high dimensional feature space. Hence the system should be able to deal with *the curse of dimensionality* and generalize well.
- **Efficiency**: Since a virus scanner is usually a speed sensitive application, so the performance of the system should be guaranteed.
- **Effectiveness**: The system should have a high detection rate with low false positive rate.

To address these above challenges, in this paper, we develop interpretable string based malware detection system (SBMDS) using support vector machine (SVM) ensemble [9] with Bagging [7], to classify the file samples and predict the exact types of the malware. SBMDS consists of three major modules: Feature parser, SVM ensemble, and malware detector. The feature parser is used to extract the interpretable strings from file samples. SVMs have shown to be effective in classification. They are also able to handle large feature spaces and generalize well. However, SVM is computationally infeasible on our large scale of data, because its training complexity is highly dependent on the size of the data set [35]. Ensemble classifier techniques are attractive in this respect, so we use the SVM ensemble with bagging to address the challenge of scalability. To improve the efficiency, we perform feature selection before applying classifiers to select informative strings related to different types of malware.

1.2 Our contribution and the organization of the paper

Our malware detection is carried out with four major steps: (1) first constructing the interpretable strings by developing a feature parser; (2) performing feature selection to select informative strings related to different types of malware; (3) followed by using SVM ensemble with bagging to construct the classifier; (4) and finally conducting the malware detector, which not only can detect whether a program is malicious or not, but also can predict the exact type of the malware. Our case study on the large collection of file samples collected by Kingsoft Anti-virus lab illustrate that: (1) The accuracy and efficiency of our SBMDS outperform several popular anti-virus software; (2) Based on the signatures of interpretable strings, our SBMDS outperforms data mining based detection systems which employ single SVM, Naive Bayes with bagging, Decision Trees with bagging; (3) Compared with the IMDS which utilizes the objective-oriented association (OOA) based classification on API calls, our SBMDS achieves better performance. Our SBMDS system has already been incorporated into the scanning tool of a commercial anti-virus software.

In summary, our main contributions are: (1) We develop an integrated SBMDS system based on the analysis of interpretable strings, which may be the first attempt using them as signatures for malware detection; (2) Based on such large scale data set of high dimensionality and sparsity, we adapt SVM ensemble with bagging technique to improve the system effectiveness and efficiency; (3) Our SBMDS not only can provide the binary prediction, e.g., whether a program is malicious or not, but also can predict the exact type of the malware; (4) We evaluate our system on a large collection of executables including 8,320 benign samples and 31,518 malicious ones; (5) We provide a comprehensive experimental study on various anti-virus software as well as various data mining techniques for malware detection using our data collection; (6) Our system has been already incorporated into the scanning tool of a commercial anti-virus software.

The rest of the paper is organized as follows. Section 2 discusses the related work. The SBMDS system architecture is described in Sect. 3. We present our data collection and SVM ensemble with bagging for multi-classification methodology in Sects. 4 and 5, respectively. In Sect. 6, we present and discuss the experimental results. Finally, Sect. 7 concludes.

2 Related work

In order to overcome the disadvantages of the widely-used signature-based malware detection method, data mining and machine learning approaches are proposed for malware detection [8,17,24,27,31]. The performance of such methods used for malware detection critically depend on the set of features and the classifier [23].

For feature extraction, there are three prevalent methods: binary profiling, string sequences, and so-called hex dumps [17]. In our previous work [34], we extracted the windows application programming interface (API) execution calls, which can reflect the behavior of program code pieces, as the signatures of the file samples. In this paper, based on our experiences of malware analysis at Kingsoft anti-virus lab, we present that the interpretable strings can be better static features for malware detection, as they include not only API calls from Import Table, but also more important semantic information which can reflect an attacker's intent and goal. Besides the pair of the interpretable strings as mentioned in the previous section, here we give another example. The interpretable string "if exist '%s' goto delete /tmp.bat" always implicates that the malware may generate the ".bat" file to suicide. Compared with other feature extraction approaches, such as sequence of instructions or hex dumps [8], the interpretable strings, including API calls from Import Table and strings carrying semantic interpretation, are the high-level specifications of malicious behaviors and contain the important semantic information which can reflect the attacker's intent and goal. In addition, compared with the feature extraction approach by running the program in a sand-box and focusing on its interaction with the operating system [8], our proposed static feature extraction method is easier and less expensive, but includes more implicit information. More importantly, if we use interpretable strings as the file features, it is not easy for the malware authors to evade our detection. This is because even if they generate the variants of the malware by re-compiling or adopting obfuscation [3] techniques such as polymorphism and metamorphism, it is impractical for them to modify all of the interpretable strings in their programs.

For classification, we developed an intelligent malware detection system (IMDS) in our previous work [34], which adopts OOA mining based classification method based on the analysis of API calls. In [34], we compared our IMDS with other classifiers developed in the previous studies [17,24,31], e.g., the Naive Bayes classifier, J4.8 version of Decision Tree implemented in WEKA [33], and also the SVM implemented in LIBSVM package [14]. The results show that our IMDS applying OOA mining based classification method outperforms other classification approaches in both detection rate and accuracy. However, in this paper, OOA mining based classification method is not suitable for the dataset of high dimensionality and sparseness.

With the advantage of handling large feature space without overfitting, SVM [36] and relevance vector machine (RVM) pioneered by Tipping [29] have shown state-of-art results in classification problems [6,29,36]. RVM is a Bayesian treatment of the sparse learning problem. Though it surpasses SVM when probabilistic outputs or kernel selection come to discussion [18,19], SVM still presents complexity and accuracy preponderance [26]. Since malware detection is usually a speed and false positive sensitive application, we apply SVM as our base classifier.

However, despite the success of SVM in classification, the training complexity of SVM is highly dependent on the size of the data set [22]. Ensemble classifiers are quite popular in many data mining applications due to their potential for efficient parallel implementations and high accuracy. An ensemble of classifiers is a set of classifiers whose individual decisions are combined to classify new examples [9]. Ensemble classifier techniques are attractive in solving the problem of scalability, since robust and accurate classifier aggregations can be learned even if each individual classifier operates on the incomplete training data, i.e., certain training instances are eliminated at random [16]. Many methods for constructing ensembles have been developed [9], while Bagging [7] and Boosting [13] are most popular ensemble learning algorithms [21]. Adaboost [13], as the most popular upgraded algorithm of boosting, employs a weak learner to find a good hypothesis. Unfortunately, the findings in [32] show that Adaboost with SVM does not work well. In this paper, we use the SVM ensemble with bagging technique to solve the problem of scalability.

Different from earlier studies, our work is based on a large collection of malware collected at Kingsoft Anti-Virus Lab. To the best of our knowledge, no attempt has been made on interpretable-string feature extraction and using SVM ensemble with bagging for malware detection. It will be interesting to know the performance compared with other feature extraction methods and classifiers.

3 The system architecture

Our SBMDS system is performed directly on Windows portable executable (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. The system consists of three major components: feature parser, SVM ensemble with bagging, and malware detector, as illustrated in Fig. 1.

The functionality of the feature parser is to generate the interpretable strings for each PE file. If a PE file is previously compressed by a third party binary compress tool, it needs to be decompressed before being passed to the feature parser. Then we use the extracted interpretable strings as the signatures of the PE files. After that, a SVM ensemble with bagging is applied to construct the classifiers. To determine whether a PE file is malicious or not and predict the exact type of the malware, we pass the extracted interpretable strings to each classifier in the SVM ensemble and then combine the prediction results by using majority voting. Thus the final report is exported via the malware detector. The detail procedures of data collection, interpretable string generation, and SVM ensemble with bagging for multi-classification methodology are respectively described in the following two sections.

4 Data collection and transformation

As stated previously, we obtain 39,838 executables, of which 8,320 are referred to as benign executables and 31,518 are malicious ones of four different types (e.g., backdoors, spyware, trojans and worms). All of these file samples are provided by Kingsoft Anti-virus lab. Then we develop the feature parser to extract the static features from each PE file, which includes API calls from the Import Table and strings carrying semantic interpretation. The implementation of our feature parser mainly involves the following procedures:

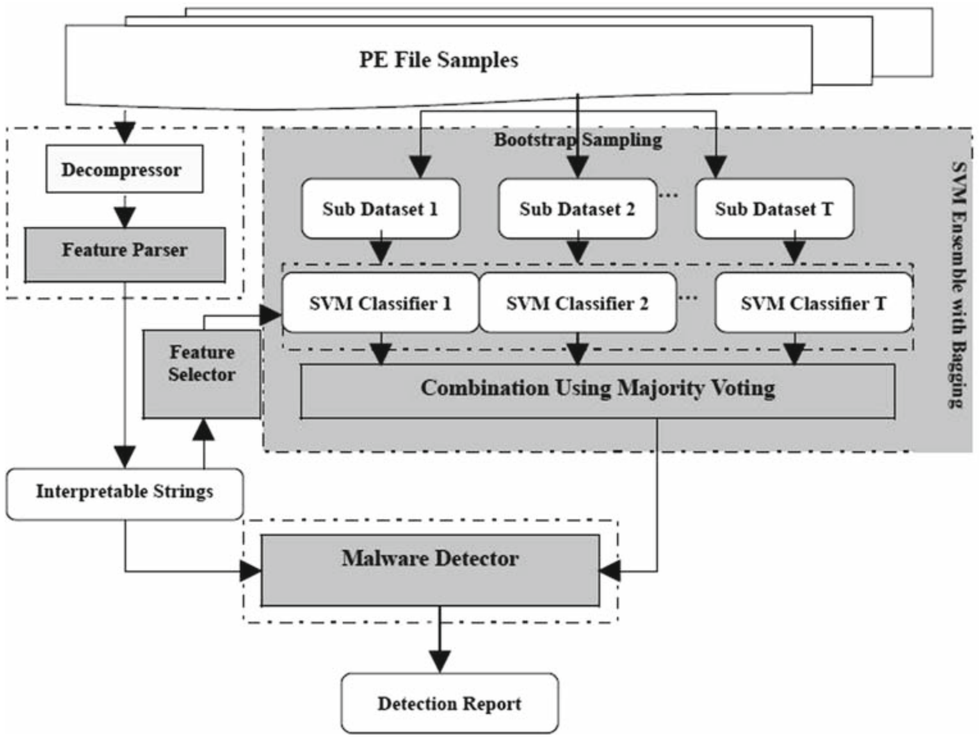
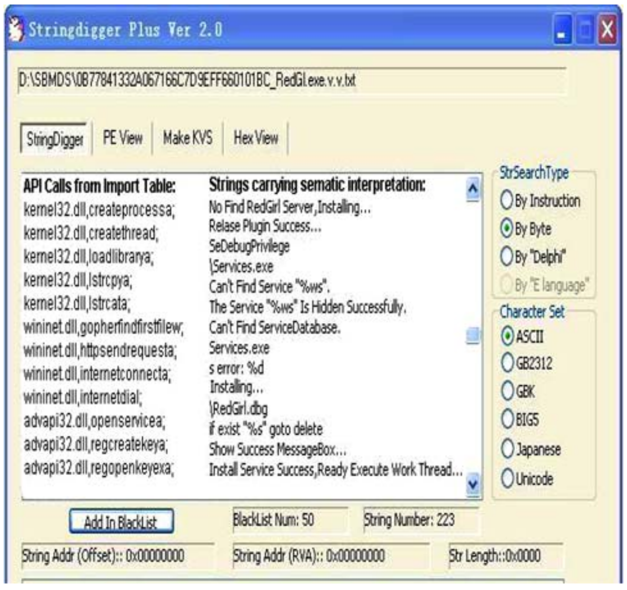


Fig. 1 SBMDS system architecture

1. Verify if the file is a valid PE file.
2. Locate the PE header by examining the DOS header;
3. Obtain the address of the data directory in Image_ Optional_Header;
4. Extract the value of VirtualAddress in the data directory;
5. Find Image_Import_Descriptor structures using the value of VirtualAddress;
6. Check the RVA value in each Image_Import_Descriptor structure found in step 5 to locate the arrays which contain the API function names;
7. Extract API function calls from the Import Table.
8. Read the PE file and if there's a sequence of consecutive bytes belonging to the same Character Set, such as ASCII, GB2312, Big5 and Unicode, then exact them as our candidate interpretable strings.
9. 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.

Figure 2 shows a sample interpretable strings extracted by our feature parser. These strings are extracted from a malware named $Backdoor-Redgirl.exe$. From Fig. 2, we can see the behaviors of the malware and the attacker's intent explicitly.

Through the feature parser, we extract 963,556 interpretable strings including API calls from Import Table and strings carrying semantic interpretation from 39,838 file samples in the database. Now the data is well processed and ready for the further steps to finally achieve the goal of malware detection.



D:\SBMDS\0B77841332A067166C7D9EFF660101BC_RedGl.exe.v.v.txt

StrSearchType: ( ) By Instruction [✓ By Byte] ( ) By "Delphi" ( ) By "E language"

Character Set: [✓ ASCII] ( ) GB2312 ( ) GBK ( ) BIG5 ( ) Japanese ( ) Unicode

```
API Calls from Import Table:        Strings carrying sematic interpretation:
kernel32.dll,createprocessa;        No Find RedGirl Server,Installing...
kernel32.dll,createthread;          Relase Plugin Success...
kernel32.dll,loadlibrarya;          SeDebugPrivilege
kernel32.dll,lstrcpya;              \Services.exe
kernel32.dll,lstrcata;              Can't Find Service "%ws".
wininet.dll,gopherfindfirstfilew;   The Service "%ws" Is Hidden Successfully.
wininet.dll,httpsendrequesta;       Can't Find ServiceDatabase.
wininet.dll,internetconnecta;       Services.exe
wininet.dll,internetdial;           s error: %d
advapi32.dll,openservicea;          Installing...
advapi32.dll,regcreatekeya;         \RedGirl.dbg
advapi32.dll,regopenkeyexa;         if exist "%s" goto delete
                                    Show Success MessageBox...
                                    Install Service Success,Ready Execute Work Thread...
```

BlackList Num: 50 String Number: 223

String Addr (Offset):: 0x00000000 String Addr (RVA):: 0x00000000 Str Length::0x0000

Fig. 2 Interpretable strings sample extracted by feature parser

5 SVM ensemble with bagging for multi-classification

The large scale of data with high dimensionality and sparseness requires proper data mining methods to be applied for malware detection. With the advantage of handling large feature space without overfitting, SVM has shown state-of-art results in classification problems [16,30,35]. However, the training complexity of SVM is highly dependent on the size of the data set, so we propose to use SVM ensemble with bagging technique to solve the problem of scalability. Finally, "one-against-one" approach [14] is used for multi-classification.

5.1 SVM overview

In supervised learning, a learning problem is given training examples of the form $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ for some unknown function $y = f(x)$. The x_i values are typically vectors of the form $< x_{i1}, x_{i2}, \ldots, x_{im} >$ whose components are discrete or continuous valued. These values are also called the features of x_i. The y values are typically drawn from a discrete set of classes $\{1, \ldots, K\}$ in the case of classification or from the real line in the case of regression [9]. Given a training set S, a learning algorithm outputs a classifier. The *classifier* is a hypothesis about the true function f [9]. Given new x values, the classifier predicts the corresponding y values.

Support vector machine is a promising method for data classification and regression [16,30,35]. 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 (the features of the training malware or benign file in our case study) and y_i is the correct output for the i_{th} training example (the type of the file sample, e.g., trojan, worm, backdoor, spyware or benign). Intuitively the classifier with the largest margin will give low expected risk, and hence better generalization.

Though SVM can effectively handle the traditional problem of "Curse of dimensionality" [7], it is not desired for large-scale data mining because the training complexity of SVM is highly dependent on the size of data set. This has become an obstacle to the use of SVM in problem with our large malware data set. In order to deal with such problem, we apply ensemble classifier techniques for malware detection.

5.2 SVM ensemble with bagging

An *ensemble of classifiers* is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new samples [9]. Ensembles are often much more accurate than the individual classifiers that make them up [9].

Though lots of ensembles of classifiers have been proposed [9,15,22,26,32] most of them can be decomposed into two cascaded components. The first component is to create base classifiers with necessary accuracy and diversity, and the second one is to aggregate all of the outputs of base classifiers into a numeric value as the final output of the ensemble classifier. In this paper, we use bagging as the method for generating base classifiers and adopt the majority voting [15] approach to aggregate the independently trained SVMs.

Bagging is the procedure that generates K bootstrap samples from training data, and each one is taken as a subset [7]. A bootstrap sample is generated by uniformly sampling N examples from the training dataset with replacement, where N is the size of training dataset. The algorithm of constructing the SVM ensemble with bagging is shown in Algorithm 1.

```
1. Input: A training set $S = (x_1, y_1), \ldots, (x_n, y_n)$, where
   $x_i \in X$ and $y_i \in Y = (1, 2, \ldots, L)$; $K$: the number of
   base classifiers;

2. For $k = 1, 2, \ldots, K$ do
   (1) Generate a new training set $S^{'}$ = bootstrap sample
   with replacement from $S$;
   (2) Apply the base SVM classifier on $S^{'}$, $C_k : X \longrightarrow Y$;

3. Output the aggregated classifier based on Majority Voting
   approach.
```

Algorithm 1: SVM ensemble with bagging

5.3 SVM ensemble for multi-classification

In the previous sections, we discussed the binary classification. For multi-classification, we use the "one-against-one" approach [14] in which $K(K-1)/2$ classifiers are constructed and each one trains data from two different classes. For training data from the ith and the jth classes, we solve the following two-class classification problem [14]:

$$min_{w^{ij},b^{ij},\varepsilon^{ij}} \frac{1}{2}(w^{ij})^T w^{ij} + C\left(\sum_i (\varepsilon_t^{ij})\right)$$

subject to

$$(w^{ij})^T \phi(x_t) + b^{ij} \geqslant 1 - \varepsilon_t^{ij},\ \text{if } x_t \text{ in the } i_{th} \text{ class;}$$
$$(w^{ij})^T \phi(x_t) + b^{ij} \leqslant -1 + \varepsilon_t^{ij},\ \text{if } x_t \text{ in the } j_{th} \text{ class; } \varepsilon_t^{ij} \geqslant 0.$$

Then we use a voting strategy: each binary classification is considered to be a voting where votes can be casted for all data points x. In the end, a point is designated to be in a class with maximum number of votes. In case that two classes have identical votes, we simply select the one with the smallest index. For our malware dataset, we divide them into to four sets and number 1 to 4 indicates backdoors, spyware, trojans and worms respectively. Benign files are represented as 0.

6 Experimental results and analysis

We randomly select 9,838 executables from our data collection, of which 2,320 are referred to as benign executables and 1,936 backdoors, 1,769 spyware, 1,911 trojans and 1,902 worms in the training dataset. The rest of the collected executables are used for testing purpose. After filtering the candidate strings, we finally extract 13,448 interpretable strings in total. As not all of the interpretable strings are contributing to malware detection, we rank each interpretable string using Max-Relevance algorithm [34] and choose top 3,000 interpretable strings as the features for later classification. Figure 3 shows that the training accuracy of the base classifier changes slightly when the number of features reaches 3,000. In our SBMDS, 100 SVM base classifiers are constructed. For bootstrapping, we randomly sample 1,230 executables from the training dataset. We train each SVM independently over the replicated training dataset and aggregate the trained SVMs via Majority Voting.

We conduct three sets of validation studies using our collected data obtained from Kingsoft Anti-virus Lab. The first set of study is to compare the abilities to detect the variants of known malware and unknown malware of our SBMDS

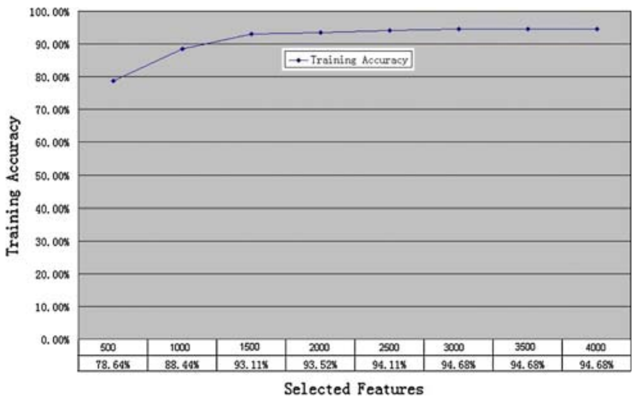


Fig. 3 SVM base classifier detection accuracy changed by features

system with current widely used anti-virus software. The efficiency and false positives by using different scanners have also been examined. In the second set of study, resting on the analysis of interpretable strings, we compare our SBMDS system with other classification based methods. Finally, we compare the SBMDS with our previous IMDS which adopts the OOA mining based classification method based on the analysis of API calls. All the validation studies are conducted under the environment of Windows XP operating system plus Intel P4 1.83 GHz CPU and 1 GB of RAM.

6.1 Comparison of different anti-virus scanners

In this section, we examine the abilities of detecting the variants of known malware and unknown malware of our system in comparison with some of the popular software tools such as Norton AntiVirus, Dr. Web, McAfee VirusScan and Kaspersky Anti-Virus. We use all of their newest versions of the base of signature on the same day (20 February, 2008) for testing. The efficiency and the number of false positives are also evaluated.

6.1.1 Variants of known malware detection

In this experiment, we use 1,000 malware as the test data set, which are not trained by our SBMDS. Several recent Win32 PE malware are included in the test dataset for analysis such as Redgirl, Bancos, MSNBot and Viking. For each malware, we collect their variants generated by the malware authors applying the encryption and obfuscation techniques [4,5], like flow modification, data segment modification and insertion of dead code. These variants of the malware are not included in our training set. Then we compare our system with current most widely used anti-virus software. The results shown in Table 1 demonstrate that our SBMDS system achieves better accuracy than other software in the variants of known malware detection.

6.1.2 Unknown malware detection

In order to examine the ability of identifying new and previously unknown malware of our SBMDS system, we use 500 malware for test. These malware are not simple modifications of well known malware which may be rewritten by the malware authors according to new requirements, like bypassing the signatures of Anti-Virus scanners or modifying part of the malware functions. They are analyzed by the experts in KingSoft Anti-virus lab and their signatures have not been recorded into our training signature database. Comparing with other anti-virus software, our SBMDS system performs the most accurate detection. The results are listed in Table 2.

Table 1 Variants of known malware detection

Software	N	D	M	K	SBMDS
Backdoor.Redgirl	√	√	√	√	√
Redgirl V1	√	×	×	√	√
Redgirl V2	×	√	√	√	√
Redgirl V3	×	√	×	×	√
Spyware.Bancos	√	√	√	√	√
Bancos V1	×	×	√	√	√
Bancos V2	√	√	?	×	√
Bancos V3	√	×	×	×	√
Troj.MSNBot	√	√	√	√	√
MSNBot V1	×	?	×	√	√
MSNBot V2	√	×	√	√	√
MSNBot V3	×	√	×	×	√
Worm.Viking	√	√	√	√	√
Viking V1	×	×	×	?	√
Viking V2	×	×	√	×	√

Remark N Norton AntiVirus, *D* Dr.Web, *M* MacAfee, *K* Kaspersky. In the table, "√" indicates successful detection, "×" indicates failure to detect, and "?" represents only an "alert"; all the scanners used are of most current and updated version

6.1.3 System efficiency and false positives

In malware detection, a false positive occurs when the scanner marks a benign file as a malicious one by error. In this set of experiments, in order to examine the system efficiency and the number of false positives of the SBMDS system, we

Table 2 Unknown malware detection

Software	N	D	M	K	SBMDS
Malware1	×	W	W	×	W
Malware2	D	×	D	T	D
Malware3	×	S	D	×	S
Malware4	×	×	×	×	×
Malware5	×	T	×	D	T
Malware6	×	D	S	×	D
Malware7	W	×	×	×	W
Malware8	×	×	×	×	T
Malware9	×	×	×	D	D
Malware10	×	×	×	×	S
Malware11	T	×	×	×	T
Malware12	×	×	×	W	W
⋮	⋮	⋮	⋮	⋮	⋮
Malware500	S	×	T	×	T
Stat.	332	318	358	395	469
Ratio	66.4%	63.6%	71.6%	79%	93.8%

Remark D Backdoor, *S* Spyware, *T* Trojans, *W* Worms sample 2,000 executables collected by Kingsoft Anti-Virus lab in the test data set, which contains 1,000 malicious executables and 1,000 benign ones including some Windows system files for Simplified Chinese platform and some common software that are easily misclassified by Anti-Virus scanners.

First, we compare the efficiency of our system with four widely used anti-virus software. The results in Fig. 4 illustrate that our SBMDS system achieves much higher efficiency than other scanners when being executed in the same environment.

The number of false positives by using different scanners are also examined. By scanning 1,000 benign executables which are not trained by our SBMDS, we obtain the results shown in Fig. 5. Figure 5 shows that the false positives by using our SBMDS system are fewer than other scanners. The system files "Netapi32.dll" and "Lsasrv.dll" for Simplified Chinese platform are misclassified by Norton and some common software like "FY-Firewall" is recognized as backdoor by MacAfee. In addition, some security tools and the plug-in of instant message (IM) software like 'QQ", "MSN" and "Yahoo Messager" are also easily recognized as malicious by Anti-Virus scanners.

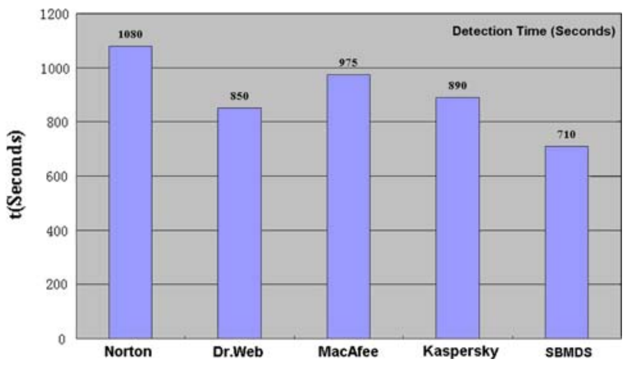


Fig. 4 Efficiency of different scanners

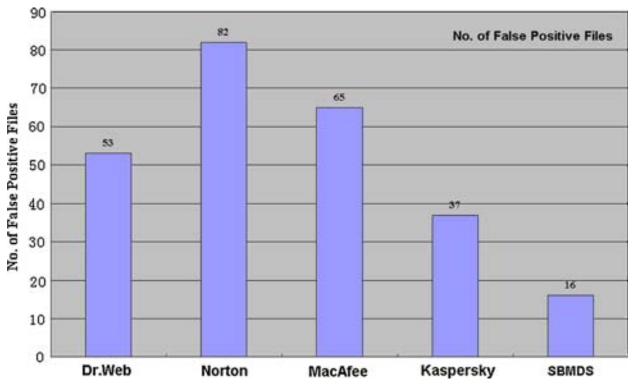


Fig. 5 False positives by using different scanners

6.2 Comparison of different classification methods

In this set of experiments, we use the same dataset as described in Sect. 6 for training, and then select 1,230 file samples from the rest of our data collection to evaluate the performance of each classification method. We compare our SBMDS system with single SVM, Naive Bayes ensemble with bagging, J4.8 version of Decision Tree implemented in WEKA [33] ensemble with bagging. For bootstrapping, we randomly sample 1,230 executables from the training dataset. For each ensemble, 100 base classifiers are constructed and we train each base classifier independently over the replicated training dataset and aggregate the trained base classifiers via Majority Voting.

In this section, we measure the classification performance of different algorithms using F1 measure, which is defined as [25,28],

$$F_1 = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

where recall is the ratio between the number of correct positive predictions and the total number of positive examples; precision is the ratio between the number of correct positive predictions and the number of positive predictions. Based on the F1 score, we use the Micro-F_1 and Macro-F_1 for the comparisons:

1. Micro-F_1 = F_1 over categories and file samples
2. Macro-F_1 = average of with-in category F_1 values

Micro-F1 and Macro-F1 emphasize the performance of the system on common and rare categories, respectively [25,28]. Results shown in Table 3 indicate our SBMDS system achieve the most accurate malware detection.

Table 3 Results by using different classifiers resting on interpretable strings

Algs.	B (%)	D (%)	S (%)	T (%)	W (%)
SSVM	82.28	92.79	84.51	84.16	82.77
BNBayes	95.28	61.71	54.46	55.12	50.42
BJ4.8	97.64	48.20	43.66	53.47	44.96
SBMDS	93.70	93.69	88.26	92.74	92.63
Algs.	Micro-F1 (%)	Macro-F1 (%)			
SSVM	84.72	84.60			
BNBayes	62.25	63.78			
BJ4.8	56.87	57.59			
SBMDS	92.30	92.22			

Remark In the table, *SSVM* single SVM, *BNBayes* Naive Bayes with bagging, *BJ4.8* decision trees (J4.8) with bagging. *B* Benign files, *D* Backdoor, *S* Spyware, *T* Trojans, *W* Worms

6.3 Comparison of SBMDS and IMDS

It may also be interesting to compare IMDS and SBMDS as they use different feature extraction methods. IMDS adopts the OOA mining based classification method resting on the analysis of API calls, while SBMDS is based on interpretable string analysis. Using the same data set described in Sect. 6, we rank each Windows API call using Max-Relevance algorithm, and then choose top 500 API calls as the features for the IMDS. We use the same testing set as described in Sect. 6.2 to evaluate the accuracy of each classifier. Results shown in Table 4 indicate our SBMDS achieve higher accuracy for malware detection.

From Tables 3 and 4, we observe that:

1. Based on the feature extraction method of interpretable strings set, our SBMDS outperforms other classification methods in both detection rate and accuracy.

2. The comparison of SBMDS and IMDS shows that adding strings carrying semantic interpretation as the signatures of the file samples are better than only Win API calls. Figure 6 shows that some malware can be predicted by the SBMDS, while IMDS fails to detect them. For example, one of the popular backdoors named $RedGirlV2007.exe$ is well specified by the strings marked by the red lines in Fig. 6. Parts of the malware can not be recognized by API calls based classifiers, because they may load most of the API calls, thus it is hard to specify the malicious behaviors of the executables. But interpretable strings, as the signatures of the file samples, can be a better representative of each malware, because they include not only API calls, but also more important implicit information which can well reflect the attacker's intent and goal. Table 5 illustrates part of the important relevant strings of the Worms. In Table 5, from our virus analysis experiences, the string of "mailfrom: imissyou@btamail.net.cn" always exists in the worms and implicates that they try to send mails.

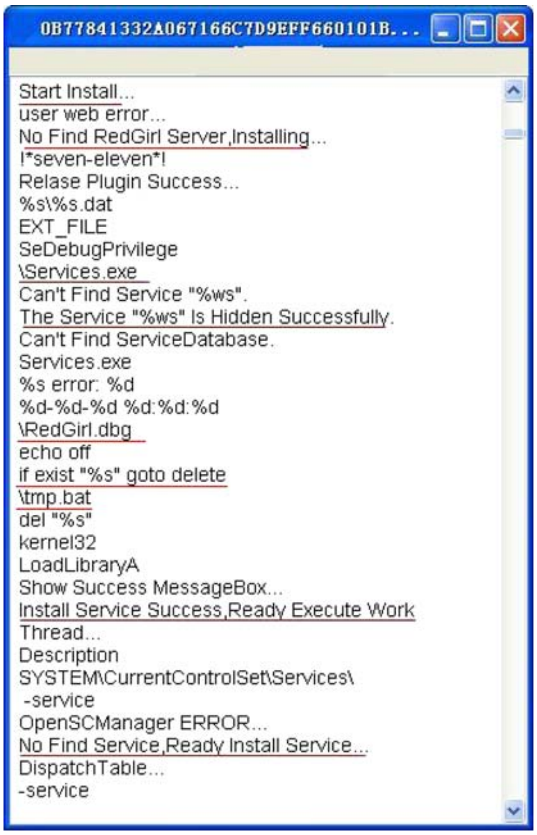
7 Conclusion

In this paper, we describe our research effort on malware detection based on interpretable strings. We develop an integrated SBMDS system consisting of a feature parser, SVM ensemble with bagging classifier and malware detector. First, a feature parser is developed to extract the interpretable strings for each Windows PE file. Then, SVM ensemble with bagging is constructed. Finally through the malware detector, the exact type of the malware can be predicted. SBMDS achieves high performance in scalability, generalization, efficiency and effectiveness, and we summarize our main contributions as follows:

Table 4 Results by using IMDS and SBMDS

Algs.	B (%)	D (%)	S (%)	T (%)	W (%)
IMDS	91.34	90.54	86.85	91.09	89.50
SBMDS	93.70	93.69	88.26	92.74	92.63
Algs.	Micro-F1 (%)	Macro-F1 (%)			
IMDS	90.01	89.91			
SBMDS	92.30	92.22			

Remark In the table, *B* Benign files, *D* Backdoor, *S* Spyware, *T* Trojans, *W* Worms



```
0B77841332A067166C7D9EFF660101B...
Start Install...
user web error...
No Find RedGirl Server,Installing...
!*seven-eleven*!
Relase Plugin Success...
%s\%s.dat
EXT_FILE
SeDebugPrivilege
\Services.exe
Can't Find Service "%ws".
The Service "%ws" Is Hidden Successfully.
Can't Find ServiceDatabase.
Services.exe
%s error: %d
%d-%d-%d %d:%d:%d
\RedGirl.dbg
echo off
if exist "%s" goto delete
\tmp.bat
del "%s"
kernel32
LoadLibraryA
Show Success MessageBox...
Install Service Success,Ready Execute Work
Thread...
Description
SYSTEM\CurrentControlSet\Services\
 -service
OpenSCManager ERROR...
No Find Service,Ready Install Service...
DispatchTable...
-service
```

Fig. 6 Malware sample detected by SBMDS but fail in IMDS

- 1. We develop an integrated SBMDS system based on the analysis of interpretable strings, which may be the first attempt used as the signatures for malware detection.
- 2. Based on such large scale data set of high dimensionality and sparsity, we adapt SVM ensemble with bagging technique to improve the system effectiveness and efficiency.
- 3. Our SBMDS not only can provide the binary prediction, e.g., whether a program is malicious or not, but also can predict the exact type of the malware.
- 4. We evaluate our system on a large collection of executables including 8,320 benign samples and 31,518 malicious ones.
- 5. We provide a comprehensive experimental study on various anti-virus software as well as various data mining techniques for malware detection using the large

Table 5 The important relevant string samples and explanations of the Worms

Interpretable strings	Brief explanation
(1) sendmail	The worms may intend to send emails.
(2) $system\backslash controlset$ 001$\backslash services\backslash$	The worms may intend to load the services.
(3) $\backslash runouce.exe$	The worms of "ChineseHack" intend to delete other worms.
(4) $from : \%s@yahoo$ $.com$	The worms are sending emails.
(5) $hellobtamail.$ $net.cn$	The worms are sending emails.
(6) $mailfrom : imiss$ $you@btamail.net.cn$	The worms are sending emails.
(7) $netsend * mygod$! some one killed %s monitor	The worms are sending emails.
(8) $chinesehacker - 2$	The representative of "ChineseHack" worms.
(9) $< html >< script$ $language =$ "$javascript$" $> window.open($"$readme$ $.eml$")	The worms of "Nimda" try to infect the scripts.

collection of file samples collected by Kingsoft anti-virus lab. The results of validation studies illustrate that: (1) The accuracy and efficiency of our SBMDS outperform several popular anti-virus software; (2) Based on the signatures of interpretable strings, our SBMDS outperforms data mining based detection systems which employ single SVM, Naive Bayes with bagging, decision trees with bagging; (3) Compared with the IMDS which utilizes the OOA based classification on API calls, our SBMDS achieves better performance.

- 6. Our system has been already incorporated into the scanning tool of a commercial Anti-Virus software.

Acknowledgments The work of Y. Ye, L. Chen and Q. Jiang is supported by the National Science Foundation of China under Grant No.10771176. The work of D. Wang and T. Li is supported in part by the US National Science Foundation Under grant IIS-0546280 and by IBM Faculty Awards. The authors would also like to thank the members in Kingsoft Anti-virus lab for their helpful discussions and suggestions.

References

1. Adleman, L.: An abstract theory of computer viruses (invited talk). In: CRYPTO '88: Proceedings on Advances in cryptology, pp. 354–374. Springer, New York (1990)
2. Bailey, M., Oberheide, J., Andersen, J., Mao, Z.M., Jahanian, F., Nazario, J.: Automated classification and analysis of internet malware. RAID 2007. LNCS, vol. 4637, pp 178–197 (2007)
3. Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. J. Comput. Virol. **2**, 67–77 (2006)
4. Beaucamps, P., Filiol, E.: Metamorphism, formal grammars and undecidable code mutation. J. Comp. Sci. **2**(1), 70–75 (2007)
5. Beaucamps, P., Filiol, E.: On the possibility of practically obfuscating programs towards aunified perspective of code protection. J. Comp. Virol. **3**(1), 2007
6. Bowd, C., Medeiros, F.A., Zhang, Z., Zangwill, L.M., Hao, J., Lee, T., Sejnowski, T.J., Weinreb, R.N., Goldbaum, M.H.: Relevance vector machine and support vector machine classifier analysis of scanning laser polarimetry retinal nerve fiber layer measurements. Invest. Ophthalmol. Vis. Sci. **46**, 1322–1329 (2005)
7. Breiman, L.: Bagging predicators. Mach. Learn. **24**, 123–140 (1996)
8. Christodorescu, M., Jha, S., Kruegel, C.: Mining specifications of malicious behavior. In Proceedings of ESEC/FSE07, pp 5–14 (2007)
9. Dietterich, T.G.: Machine learning research: Four current directions. AI Magaz. **18**(4), 97–136 (1997)
10. Filiol, E.: Computer Viruses: from Theory to Applications. Springer, Heidelberg (2005)
11. Filiol, E.: Malware pattern scanning schemes secure against black-box analysis. J. Comp. Virol. **2**(1), 35–50 (2006)
12. Filiol, E., Jacob, G., Liard, M.L.: Evaluation methodology and theoretical model for antiviral behavioural detection strategies. J. Comp. Virol. **3**(1), 27–37 (2007)
13. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comp. Syst. Sci. **55**(1), 119–139 (1997)
14. Hsu, C., Lin, C.: A comparison of methods for multiclass support vector machines. IEEE Trans. Neural Netw. **13**, 415–425 (2002)
15. Kim, H., Pang, S., Je, H., Kim, D., Bang, S.: Support vector machine ensemble with bagging. SVM 2002, LNCSI, vol. 2388, pp 397–408 (2002)
16. Kolcz, A., Sun, X., Kalita, J.: Efficient handling of high-dimensional feature spaces by randomized classifier ensembles. In: Proceedings of KDD'02 (2002)
17. Kolter, J., Maloof, M.: Learning to detect malicious executables in the wild. In: Proceedings of KDD'04 (2004)
18. Li, Y., Campbell, C., Tipping, M.: Bayesian automatic relevance determination algorithms for classifying gene expression data. Bioinformatics **18**, 1232–1239 (2002)
19. Li, D., Hu, W.: Feature selection with rvm and its application to prediction modeling. AI 2006, LNAI, vol. 4304, pp 1140–1144 (2006)
20. McGraw, G., Morrisett, G.: Attacking malicious code:report to the infosec research council. IEEE Softw. **17**(5), 33–41 (2000)
21. Oza, N.C., Russell, S.: Experimental comparisons of online and batch versions of bagging and boosting. In: Proceedings of KDD'01 (2001)
22. Rangel, P., Lozano, F., Garcia, E.: Boosting of support vector machines with application to editing. In: Proceedings of ICMLA'05 (2005)
23. Reddy, D.K.S., Pujari, A.K.: N-gram analysis for computer virus detection. J. Comput. Virol. **2**, 231–239 (2006)
24. Schultz, M., Eskin, E., Zadok, E.: Data mining methods for detection of new malicious executables. In: Security and privacy, 2001. Proceedings of 2001 IEEE Symposium on 14–16 May, pp 38–49 (2001)
25. Sebastiani, F.: Text categorization. ACM Comput. Surv. **34**(1), 1–47 (2002)
26. Silva, C., Ribeiro, B., Sung, A.H.: Boosting rvm classifiers for large data sets. ICANNGA 2007, Part II, LNCSI, vol. 4432, pp 228–237 (2007)

27. Sung, A., Xu, J., Chavez, P., Mukkamala, S.: Static analyzer of vicious executables (save). In: Proceedings of the 20th Annual Computer Security Applications Conference (2004)
28. Tan, S., Cheng, X., Ghanem, M., Wang, B., Xu, H.: A novel refinement approach for text categorization. In: Proceeding of the ACM CIKM, pp 469–476, 2005
29. Tipping, M.: Sparse bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**, 211–214 (2001)
30. Tsang, I.W., Kwok, J.T., Cheung, P.M.: Core vector machines: Fast svm training on very large data sets. J. Mach. Learn. Res. **6**, 363–392 (2005)
31. Wang, J., Deng, P., Fan, Y., Jaw, L., Liu, Y.: Virus detection using data mining techniques. In: Proceedings of IEEE International Conference on Data Mining (2003)
32. Wickramaratna, J., Holden, S.B., Buxton, B.F.: Performance degradation in boosting. In: Proceedings of the Second International Workshop on Multiple Classifier Systems (2001)
33. Witten, H., Frank, E.: Data mining: Practical machine learning tools with Java implementations. Morgan Kaufmann, Menlo Park (2005)
34. Ye, Y., Wang, D., Li, T., Ye, D.: IMDS: Intelligent malware detection system. In: Proceedings of ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD 2007) (2007)
35. Yu, H., Yang, J., Han, J.: Classifying large data sets using svms with hierarchical clusters. In: Proceedings of KDD'03 (2003)
36. Vapnik, C.C.: Support vector network. Mach. Learn. **20**, 273–297 (1995)