Third International Conference on Information Science and Technology
March 23-25, 2013; Yangzhou, Jiangsu, China

Class-wise Information Gain

Pengtao Zhang and Ying Tan

***Abstract*—This paper proposes a new feature-goodness criterion named class-wise information gain (CIG). The CIG is able to measure the goodness of a feature for recognizing a specific class, and further helps to select the features with the highest information content for a specific class. In order to confirm the effectiveness of the CIG, a CIG-based malware detection method is proposed. Eight groups of experiments on three public malware datasets are carried out to evaluate the performance of the proposed CIG-based malware detection method through cross-validation. Comprehensive experimental results suggest that the CIG is an effective feature-goodness criterion, and the proposed CIG-based malware detection method is effective to detect malware loaders and infected executables. This method outperforms the information gain (IG)-based malware detection method for about 26% in detecting infected executables, without decrease in detecting malware loaders, while its memory requirement is about 60% less than that of the IG-based malware detection method empirically.**

I. Introduction

FEATURE selection methods try to select a subset of relevant features for building robust learning models, most of which make use of feature-goodness criteria to obtain a desirable feature set, named detecting feature (DF) set in this paper. There are several criteria which are wildly used in the machine learning community, such as information gain (IG), document frequency, mutual information, χ^2 statistic and term strength [1]. Many feature selection methods have been applied into the anti-malware field in the last decade.

Malware is a general term for all the malicious code that is a program designed to harm or secretly access a computer system without the owners' informed consent [2], such as computer virus, Trojan and worm. It has been one of the most terrible threats to the security of the computers worldwide [3]. Malware is present generally in two forms: malware loader and infected executable. A malware loader is a pure malware only containing malicious code. An infected executable is a combination of a malware loader and a benign program. It is one of the mostly used evading techniques of malware, and is the main form of malware in the wild empirically.

Schultz *et al.* proposed a data-mining framework to detect unseen malware [4], laying a good foundation for the application of machine learning techniques in anti-malware field.

Kolter *et al.* proposed a technique to detect malware based on the relevant N-Grams selected by using the IG [5], and achieved good results. Later this technique is extended to classify malware based on the function of their payload [6]. However, the IG only measures the goodness of a feature for a classification problem, ignoring the class information of the feature, so the classes of the features selected by using the IG are unknown, and the distribution of the features of different classes in the DF set is out of control. When these features are used to detect infected executables, the performance of this technique would drop down dramatically.

A new feature selection criterion, class-wise document frequency, was proposed by Reddy *et al.* [7]. Their experimental results showed that the class-wise document frequency outperformed the IG in the feature selection process. They guessed the reason might be most of the relevant N-Grams selected by using the IG came from benign programs, but they did not give a clear answer. This guess is supported by the experimental results in this paper. Furthermore, they did not realize that the features of benign programs were useless to detect infected executables.

Stolfo *et al.* made use of N-Grams to identify file types [8] and later to detect stealthy malware [9], [10]. Tabish *et al.* proposed a malware detection model using statistical analysis of the byte-level file content [11], which neither memorizes specific strings appearing in the file content nor depends on prior knowledge of file types. Many new malware detection methods have been proposed recently, for details, please refer to [12], [13], [14], [15].

This paper proposes a new feature-goodness criterion named class-wise information gain (CIG) and a novel CIG-based malware detection method. The CIG is able to measure the goodness of a feature for recognizing a specific class, and further helps to select the features with the highest information content for a specific class. These features are considered to be able to recognize their corresponding classes. The proposed CIG-based malware detection method makes use of the malware features selected by using the CIG to detect malware. The experimental results demonstrate that the CIG is an effective feature-goodness criterion, and the proposed CIG-based malware detection method is effective to detect malware loaders and infected executables.

The remainder of this paper is organized as follows. Section II introduces the CIG in detail. Section III describes the proposed CIG-based malware detection method. The detailed information of the experiments are presented in Section IV. Finally, we conclude the paper with a detailed discussion.

II. Class-wise Information Gain

A. Definition Of Generalized Class

Definition 1 If a feature f appears in a class C and does not appear in any other classes, we define feature f as a special

This work was supported by the National Natural Science Foundation of China (NSFC) under grant number 61170057 and 60875080.

P Zhang and Y Tan are with the Key Laboratory of Machine Perception (MOE), Peking University, Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, CHINA (email: pengtaozhang@gmail.com, ytan@pku.edu.cn). Y Tan is the corresponding author.

978-1-4673-2764-0/13/$31.00 ©2013 IEEE

feature (SF) of class C, and the special class (SC) of feature f as class C.

The SC is the traditional class. The SF of class C is able to recognize class C. For example, the malware signatures are the SFs of the malware and believed to be able to recognize malware.

Definition 2 If a feature f tends to appear in a class C and not appear in any other classes, we define feature f as a generalized feature (GF) of class C, and the generalized class (GC) of feature f as class C.

The GF of class C is able to recognize class C too. It regards a sample containing itself as a sample of class C. The more a feature f tends to occur in class C and be absent from other classes, the better the feature f is for recognizing class C. According to the above two definitions, a SF of class C is also a GF of class C. The class of a feature mentioned in this paper is the GC of the feature.

B. Definition Of Class-wise Information Gain

The class-wise information gain (CIG, for short) proposed in this paper is defined as

$$CIG(f,C_i) = P(v_f=1,C_i)log\frac{P(v_f=1,C_i)}{P(v_f=1)P(C_i)} + \sum_{C_j\in\{C_i\}\bigwedge i\neq j} P(v_f=0,C_j)log\frac{P(v_f=0,C_j)}{P(v_f=0)P(C_j)} \quad (1)$$

where f is a candidate feature, and C_i is a class. v_f is the value of feature f. v_f = 1, if feature f occurs in a sample and v_f = 0, otherwise. $P(v_f, C_i)$ is the probability that feature f in class C_i has the value v_f. $P(v_f)$ is the probability that feature f in a training set takes the value v_f. $P(C_i)$ is the probability of the training samples belonging to class C_i.

There are two terms in Eq. 1. The first term and the second term, respectively, denote the statistical information of feature f occurring in class C_i and being absent from other classes. The more a feature f tends to occur in class C_i and be absent from other classes, the larger $CIG(f,C_i)$ is. According to Definition 2, the $CIG(f,C_i)$ is considered to be able to measure the goodness of feature f for recognizing class C_i, and further help to select the features with the highest information content for class C_i, which are the GFs of class C_i.

The value v_f of feature f in class C_i and C_j is related to the class of feature f by using $P(v_f=1,C_i)$ and $P(v_f=0,C_j)$ in Eq. 1. In this way, the class information of feature f is imported into the CIG. On this basis, the $CIG(f,C_i)$ measures the discrimination and class information of feature f at the same time. The more information content a feature f brings in for recognizing class C_i, the larger the $CIG(f,C_i)$ is.

In two-class classification problems, $CIG(f,C_0) + CIG(f,C_1) = IG(f)$, where C_0 and C_1 denote the two classes, and $IG(f)$ is the information gain of feature f. In this case, the CIG could be understood as the disassembly of the IG in the perspective of the classes. Hence the CIG is considered to be able to select the features with the highest information content for a specific class by introducing class information into the IG. It is much finer than the IG and also extends the IG greatly.

The CIG is able to be applied to recognition problems directly to select features for a specific class. What is more, the proportions of the features of different classes in the DF set could be balanced or adjusted freely by using the CIG, so it could replace the IG to get more stable or effective models in some classification problems.

C. Analysis Of Class-wise Information Gain

Let us analyze the CIG in a two-class problem: the malware detection problem. In this case, Eq. 1 is further rewritten as

$$CIG(f,C_B) = P(v_f=1,C_B)log\frac{P(v_f=1,C_B)}{P(v_f=1)P(C_B)} + P(v_f=0,C_M)log\frac{P(v_f=0,C_M)}{P(v_f=0)P(C_M)} \quad (2)$$

$$CIG(f,C_M) = P(v_f=0,C_B)log\frac{P(v_f=0,C_B)}{P(v_f=0)P(C_B)} + P(v_f=1,C_M)log\frac{P(v_f=1,C_M)}{P(v_f=1)P(C_M)} \quad (3)$$

where C_B and C_M denote benign program and malware, respectively.

We regard Eq. 2 and Eq. 3 as a function $CIG(f,C)$, $C\in\{C_B,C_M\}$, of two variables $P(v_f=1,C_B)$ and $P(v_f=1,C_M)$. Set $P(v_f=1)=0.5$ and $P(C_B)=0.5$. The two functions are drawn in Figure 1. Furthermore, the $IG(f)$ is plotted in Figure 2 for a comparison.

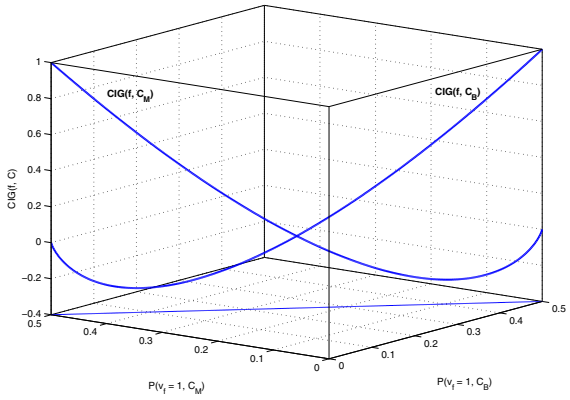

Fig. 1. The $CIG(f,C)$ function

Fig. 1 illustrates that the more a feature f tends to occur in class C_M and be absent from class C_B, the larger the $CIG(f,C_M)$ is, and vice versa. When a feature f gets a larger $CIG(f,C_M)$, its $CIG(f,C_B)$ would be smaller relatively, and vice versa. Furthermore, the $CIG(f,C)$ could be less than 0 which means feature f brings harmful information for recognizing class C.

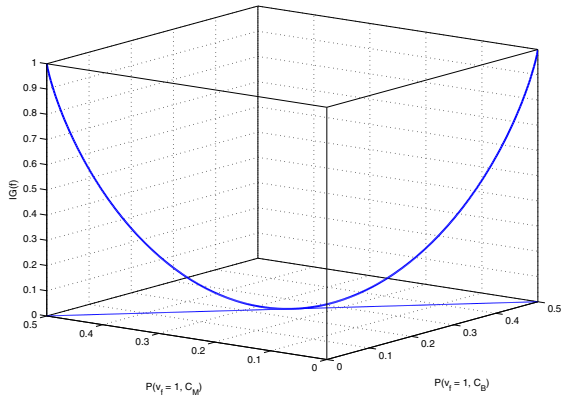

Fig. 2. The $IG(f)$ function

From Fig. 1 and Fig. 2, the features with the highest $IG(f)$ may have larger $CIG(f, C_B)$ with smaller $CIG(f, C_M)$, or smaller $CIG(f, C_B)$ with larger $CIG(f, C_M)$. The distribution of the features with larger $CIG(f, C_B)$ and larger $CIG(f, C_M)$ in the DF set selected by using the IG is up to a specific training dataset. As the IG does not take class information of features into account, it is not able to measure how much information content a feature f brings in for recognizing class C. This is the key problem in the IG.

The CIG proposed in this paper solves this key problem in the IG perfectly by introducing class information of features. The CIG is an effective feature-goodness criterion which is able to measure the goodness of a feature for recognizing a specific class. The proportions of the features of different classes in the DF set could be balanced or adjusted freely by using the CIG, so the CIG could be applied in both classification problems and recognition problems.

III. CIG-BASED MALWARE DETECTION METHOD

The proposed CIG-based malware detection method involves two modules: (1) feature selection module, (2) classification module.

A. Problem Statement

Malware is present in two forms: malware loader and infected executable. Since the infected executable is the main form of malware in the wild empirically, how to detect infected executables effectively becomes one of the most urgent tasks in the anti-malware field.

As an infected executable is a combination of a benign program and a malware loader, it would own all the features of benign programs in theory. In order to detect malware loaders and infected executables effectively, we take the malware detection problem as the malware recognition problem, rather than the classification problem of benign programs and malware. Actually the commercial anti-malware produces have the same view as us. How to select the malware features becomes the key of the malware detection problem.

B. Feature Selection Module

The N-Gram is a concept from text categorization, which means *N* continuous words or phrases. Since the smallest unit of an instruction in the Intel Architecture-32 bit set (IA-32) is 1 byte, a gram is defined as a binary string of length 1 byte in this paper. An N-Gram here is defined as a binary string of length *N* bytes. Different from the malware signatures, the N-Gram is short and easy to be matched, so it has the potential ability to detect malware with evading techniques inherently. The number of N-Grams in its feature space is $(2^8)^N$. The N-Grams are taken as the candidate features in this paper, which are the basis of feature selection.

An N-Gram is saved as a three-dimensional vector <content, CIG_B, CIG_M>. The field "content" keeps the string of the N-Gram, while the fields "CIG_B" and "CIG_M" store the values of the $CIG(f, C_B)$ and $CIG(f, C_M)$, respectively.

The procedure of the feature selection module is introduced below. Firstly, the document frequency of every N-Gram is counted by traversing a training set, using a sliding window of length *N* bytes. The sliding window moves forward one byte at a time. The content in a sliding window is an N-Gram. The adjacent two N-Grams have an overlap of $N-1$ bytes, which helps the N-Grams to capture not only strings of length *N* bytes, but also longer strings implicitly. Secondly, we compute the CIG_B and CIG_M for every N-Gram gathered from the training set using Eq. 2 and Eq. 3, respectively. Until now, all the information of the whole N-Grams has been obtained. Finally, the top *M* N-Grams with the highest CIG_B and CIG_M, respectively, are selected to make up the CIG-B and CIG-M, which are the DF sets. The features in the CIG-B and CIG-M are considered to be the features of benign programs and malware, respectively.

It is necessary to mention that the CIG-B is brought in just for a comparison. We will show that the features in the CIG-B are helpless to detect infected executables.

C. Classification Module

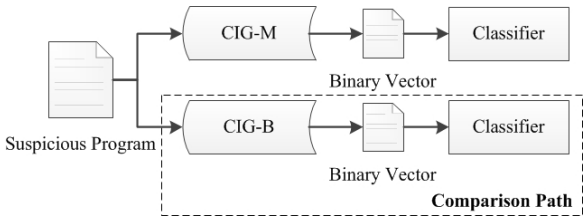

Fig. 3. The flow chart of the classification module

The flow chart of the classification module is shown in Fig. 3. There are two paths for a suspicious program. The first one is mapping the program by the CIG-M which would be used in the real world, while the other one makes use of the CIG-B for the purpose of making a comparison with the first path.

A suspicious program is expressed as a binary vector of length *M* which consists of 0s for absence of the features in the DF set, and 1s for presence of the features in the DF set. A classifier is trained to detect the suspicious programs.

IV. Experiments

A. Datasets

The experiments in this paper are conducted on three public malware datasets: CILPKU08, Henchiri and VXHeanvens datasets. The three datasets and their composition documents can download from www.cil.pku.edu.cn/resources/.

The benign program dataset used here consists of programs in portable executable format from Windows XP and a series of applications, which are the main punching bag of malware.

B. Experimental Setup

The support vector machine (SVM), realized in LibSVM [16], is taken as the classifier of the proposed CIG-based malware detection method in this paper. Other classifiers, such as instance-based learner, and decision tree, could be also used. The area under the receiver operating characteristic curve (AUC) is utilized as the performance evaluation criterion.

In the experiments in Section IV-D, eight groups of experiments are carried out in the three public malware datasets using 5-fold cross validation, and the 95% confidence intervals are computed to look into the stability of the proposed CIG-based malware detection method. Both the CILPKU08 and Henchiri datasets mainly consist of computer viruses, so two experiments are carried out in the two datasets directly, ignoring the categories of malware. There are six categories of malware in the VXHeavens dataset, so we split this dataset into six smaller datasets: backdoor, constructor, Trojan, virus, worm and "Others". The "Others" includes DoS, Nuker, Hacktool and Flooder, while the malware in the other five smaller datasets, respectively, fall into a category. Six groups of experiments are taken in the six smaller datasets.

In all the experiments, there is no overlap between a training set and a test set. That is to say, to a training set, the malware in a test set are unseen malware. This setting increases the reliability of the experiments.

This paper uses the IG-A to denote the top *M* features selected by using the IG, which is a DF set. The malware detection method using IG-A proposed in [5], [6] is a IG-based malware detection method and imported for a comparison.

C. Selection Of Parameters

This section is to select the two parameters in the proposed CIG-based malware detection method, i.e., the length of the N-Gram (*N*) and the number of the features in a DF set (*M*).

The dataset used here consists of 1048 benign programs, randomly selected from the benign program dataset, and 1048 computer viruses from VXHeavens dataset. We randomly split the benign programs into two sets with 524 programs for each set, one for training and the other for testing. The same division was done to the viruses. The 524 benign programs and 524 viruses made up test set 1 (referred to as T_1).

Let the computer viruses in the T_1 infect all the benign programs in the T_1 by inserting themselves to the tails of the benign programs. There are three main ways for malware to infect benign programs in the real world: insertion in the head of a benign program, insertion in the tail of a benign program and insertion in the 'cavity' of a benign program ('cavity' means the space which is useless and filled by 0 among different segments in a program). The influences of all the infection ways to the N-Grams of a generated infected executable are very slight, which could be ignored. In this way, we got $524 * 524 = 274576$ infected executables. The infected executables and the benign programs in the T_1 made up test set 2 (written as T_2). In summary, the T_1 includes benign programs and malware loaders, whereas the T_2 includes benign programs and infected executables. All the experiments in this section were taken on this dataset. In the rest of this paper, all the experiments include two test sets: T_1 and T_2, where the T_2 are generated from the T_1 in the same way.

Here we optimize the two parameters using the grid search method, where $N = 2, 3, 4$ and $M = 100, 200, ..., 1000$. We do not try larger *N*. As we know, the space of N-Grams is exponential with respect to the value of *N*: $(2^8)^N$. For example, the space of 5-Grams is 2^{40}. A *1 MB* program contains about 2^{10} 5-Grams at most when every 5-Gram in it is different from the others. When the program was mapped into the 5-Gram space, we found the grams contained in the program were comparatively sparse. What is more, when the *N* is set to a larger value, an N-Gram turns to be more specific. However, what we want are generic grams which could cover as much space as possible with larger information content. Due to the above reasons, we just set $N = 2, 3, 4$. In addition, Kolter *et al.* also set N = 4 [5], [6].

According to the experimental results, we set $N = 4$ and $M = 400$ as the optimal parameters in the proposed CIG-based malware detection method. From the perspective of *M*, the influenc e of *M* to the results was not significant regardless of the values of *N*. Considering the different values of *N*, the performance of the proposed CIG-based malware detection method with $N = 4$ outperformed those when *N* got other values. The experimental results of 4-Gram are shown in Fig. 4.

An interesting result is that the methods using IG-A and CIG-B both show excellent AUCs on the T_1, whereas very bad AUCs on the T_2. However, the method using CIG-M performs very well on both the T_1 and T_2. Section IV-D showed the similar results. The detailed discussions will be given in Section V.

D. Experimental Results

The experimental results of the malware detection methods using IG-A, CIG-B and CIG-M on the three public malware datasets were given in detail, respectively, in Table I, II, III. The results with bold font indicate the optimal AUCs in the three methods.

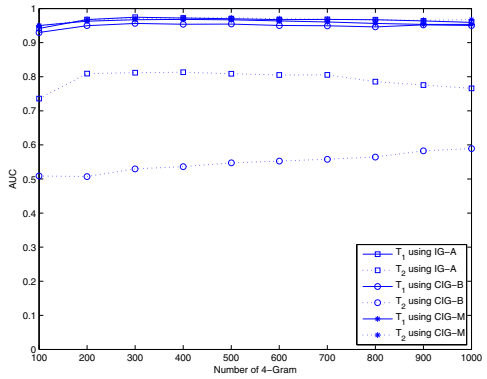

Fig. 4. The experimental results of 4-Gram

TABLE I
THE AUCS OF THE METHOD USING IG-A

Experiment	T_1	T_2
CILPKU08	**0.9992± 0.0009**	0.5008± 0.0005
Henchiri	**0.9990± 0.0009**	0.5011± 0.0008
Backdoor	0.9792± 0.0028	0.5335± 0.0050
Constructor	0.9794± 0.0153	0.9098± 0.0195
Trojan	**0.9642± 0.0054**	0.7007± 0.0516
Virus	0.9770± 0.0043	0.8292± 0.0134
Worm	**0.9708± 0.0064**	0.7706± 0.0080
Others	**0.9675± 0.0070**	0.7821± 0.0130

Table I shows that the method using IG-A performs fairly good on the T_1 and drops down dramatically on the T_2, just like its performance in Section IV-C. The average AUC of the method using IG-A on the T_2 is about 0.69, which is lower than the AUC on the T_1 for about 0.28. Although this method performs terrible on the T_2, it is still better than the random classification as the IG-A consists of the features of both benign programs and malware. The composition of the IG-A will be given in Section V.

The performance of the method using CIG-B on the T_1 is almost as good as that of the method using IG-A from Table II, while it drops down sharply to 0.55 on average in detecting infected executables on the T_2, which is nearly equivalent to the random classification. The results are consistent with those in Section IV-C. It is easy to see that the features in the CIG-B are effective to classify benign programs and malware loaders, but they are helpless to classify benign programs and infected executables. The above results suggest that the features in the CIG-B are the features of benign programs.

Table III demonstrates that the method using CIG-M is able to detect malware loaders and infected executables effectively. Its results on the T_2, $AUC = 0.95$ on average, are almost equivalent to its results on the T_1, $AUC = 0.97$ on average. Hence the features in the CIG-M are considered to be able to recognize malware. The results suggest that the features in the CIG-M are malware features. For the performance reduction on the T_2, the reason should be that the difficulty for detecting

TABLE II
THE AUCS OF THE METHOD USING CIG-B

Experiment	T_1	T_2
CILPKU08	0.9991± 0.0010	0.5008± 0.0005
Henchiri	0.9989± 0.0009	0.5005± 0.0008
Backdoor	**0.9810± 0.0013**	0.5052± 0.0018
Constructor	0.9614± 0.0149	0.5554± 0.0080
Trojan	0.9600± 0.0041	0.5479± 0.0073
Virus	0.9746± 0.0050	0.5773± 0.0063
Worm	0.9611± 0.0129	0.5834± 0.0093
Others	0.9587 ± 0.0052	0.5885 ± 0.0043

TABLE III
THE AUCS OF THE METHOD USING CIG-M

Experiment	T_1	T_2
CILPKU08	0.9827± 0.0023	**0.9811± 0.0022**
Henchiri	0.9822± 0.0029	**0.9812± 0.0040**
Backdoor	0.9669± 0.0084	**0.9601± 0.0095**
Constructor	**0.9818± 0.0145**	**0.9505± 0.0080**
Trojan	0.9506± 0.0069	**0.9328± 0.0081**
Virus	**0.9793± 0.0045**	**0.9747± 0.0077**
Worm	0.9689± 0.0055	**0.9192± 0.0150**
Others	0.9655± 0.0066	**0.9396± 0.0077**

infected executables is larger than that for detecting malware loaders. Actually an infected executable is a latent form of a malware loader, which would help itself escape from the anti-malware scanner.

The results of the methods using IG-A, CIG-B and CIG-M on the T_1 are basically equivalent to each other, whereas the results on the T_2 are significantly different. For detecting infected executables on the T_2, the method using IG-A ($AUC = 0.69$ on average) performs better than the method using CIG-B ($AUC = 0.55$ on average), but much worse than the method using CIG-M ($AUC = 0.95$ on average) definitely. The reason is the difference of the classes of the features in the IG-A, CIG-B and CIG-M.

The features in the CIG-B are considered to be the features of benign programs which are helpless to discriminate infected executables from benign programs, while the features in the CIG-M are believed to be malware features which are able to recognize malware effectively, regardless of the form of malware. Due to the IG does not take the class information of features in to account, the features in the IG-A are mixtures of features of benign programs and malware. According to the above analysis, the features in the CIG-M are considered to be the most suitable features to detect malware, including malware loaders and infected executables. The results suggest that the CIG is able to select the features with the highest information content for a specific class successfully.

The 95% confidence intervals of all the methods are relatively small from Tables I, II, III, which suggest that these results are very stable and believable.

TABLE IV
THE RELATIONSHIP AMONG IG-A, CIG-B AND CIG-M

Experiment	R_B	R_M
CILPKU08	95.85%	0.00%
Henchiri	93.15%	0.00%
Backdoor	93.15%	0.00%
Constructor	44.65%	20.85%
Trojan	54.20%	27.95%
Virus	74.00%	16.35%
Worm	67.25%	13.85%
Others	55.80%	34.05%

TABLE V
THE PROPORTIONS OF 4-GRAMS IN THE EXPERIMENTS

Experiment	r_B	r_M	$r_{B \bigcap M}$
CILPKU08	96.59%	4.07%	0.66%
Henchiri	94.20%	6.69%	0.89%
Backdoor	47.60%	55.65%	3.25%
Constructor	58.37%	44.61%	2.97%
Trojan	43.58%	67.76%	11.35%
Virus	73.35%	32.95%	6.30%
Worm	56.61%	47.19%	3.80%
Others	50.60%	54.50%	5.10%

V. DISCUSSIONS

A. Relationship Among The IG-A, CIG-B And CIG-M

The relationship among the IG-A, CIG-B and CIG-M in the eight groups of experiments is illustrated in Table IV, where the symbols are defined as

$$R_B = \frac{|\text{CIG-B} \bigcap \text{IG-A}|}{|\text{IG-A}|} \tag{4}$$

$$R_M = \frac{|\text{CIG-M} \bigcap \text{IG-A}|}{|\text{IG-A}|} \tag{5}$$

In all the experiments, CIG-B $\bigcap$ CIG-M $= \emptyset$, which suggests that the CIG-B is completely different from the CIG-M. Comparing to the experimental results in Section IV-D, the AUCs of the method using CIG-B are practically equal to 0.5 in detecting infected executables, but fairly good in detecting malware loaders. The results suggest that the features in the CIG-B are the features of benign programs.

From the above analysis, the features in the CIG-M are considered to be malware features. The experimental results in Section IV-D, which suggested that the method using CIG-M performed fairly well in detecting both malware loaders and infected executables, support this deduction.

Table IV illustrates that the IG-A is a mixture of a part of the CIG-B, a part of the CIG-M, and other N-Grams. The distribution of the features of different classes in the IG-A is up to a specific training dataset. In all the experiments, more than 44% features in the IG-A are in the CIG-B which is made up of the features of benign programs, and the percentage of the malware features in the IG-A is less than 35%. This result definitely supports the guess of [7]. The malware features in the IG-A are the keys to make the AUCs of the method using IG-A get greater value in detecting infected executables. However, the features of benign programs in the IG-A tend to recognize an infected executable as a benign program and bring down the AUCs of the method using the IG-A on the T_2 dramatically. The experimental results suggest that the IG cannot select malware features, whereas the CIG proposed in this paper completes this task effectively.

B. Space Complexity

According to the analysis in Section V-A, the features in the CIG-M are the critical features to detect malware loaders and infected executables. The features in the CIG-M are believed to be malware features, which should appear in malware at least once. Hence we just need to handle the features appearing in malware and ignore the features which merely occur in benign programs, in order to obtain the CIG-M. The memory requirement of this kind of malware detection methods mainly comes from keeping a huge number of N-Grams in memory. If we only handle the features appearing in malware, the memory utilized by a plenty of features which just appearing in benign programs could be cut down. The space complexity of the proposed CIG-based malware detection method would drop down sharply. The proportions of 4-Grams in the experiments in Section IV-D are given in Table V, where the symbols are defined as

$$r_B = \frac{|N_B|}{|N_B \bigcup N_M|} \tag{6}$$

$$r_M = \frac{|N_M|}{|N_B \bigcup N_M|} \tag{7}$$

$$r_{B \bigcap M} = \frac{|N_B \bigcap N_M|}{|N_B \bigcup N_M|} \tag{8}$$

where N_B and N_M denote the sets of N-Grams appearing in benign programs and malware, respectively. And the $|N_B|$ and $|N_M|$, respectively, are the numbers of N-Grams in the two sets.

From Table V, the proportions of 4-Grams appearing in both benign programs and malware, expressed by $r_{B \bigcap M}$, are lower than 12%. It means the intersection set of the 4-Grams appearing in benign programs and malware is comparatively small. Based on this fact, lots of memory can be saved by processing the features appearing in malware, instead of the features in a training set.

Theoretically, if we just consider the N-Grams appearing in malware, $|N_B - N_M|$ N-Grams need not to be analyzed. The percentage of the memory saved is $|N_B - N_M|/|N_B \bigcup N_M|$. In this way, the memory requirement of the proposed CIG-based malware detection method is less than that of the IG-based malware detection method for about 60% empirically. In the experiments in this paper, more than 4 GB memory is saved on average in this way.

In a general problem, the CIG could help to decrease the memory requirement of a CIG-based method dramatically by selecting features sequentially for every class.

VI. Conclusions

We have proposed the CIG by introducing class information into the IG in this paper. As the CIG is able to measure the goodness of a feature for recognizing a specific class precisely, it is much finer than the IG and also extends the IG greatly.

Comprehensive experimental results demonstrate that the CIG is an effective feature-goodness criterion. The proposed CIG-based malware detection method outperforms the IG-based malware detection method for about 26% in detecting infected executables, without decrease in detecting malware loaders, with much less memory.

References

[1] Y. Yang and J. Pedersen, "A comparative study on feature selection in text categorization," in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE*, 1997, pp. 412–420.

[2] Wikipedia. [Online]. Available: http://en.wikipedia.org/wiki/Malware

[3] C. F-Secure, "F-secure reports amount of malware grew by 100% during 2007," 2007.

[4] M. Schultz, E. Eskin, F. Zadok, and S. Stolfo, "Data mining methods for detection of new malicious executables," in *Security and Privacy, 2001. S&P 2001. Proceedings. 2001 IEEE Symposium on*. IEEE, 2001, pp. 38–49.

[5] J. Z. Kolter and M. A. Maloof, "Learning to detect malicious executables in the wild," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '04, 2004, pp. 470–478.

[6] J. Kolter and M. Maloof, "Learning to detect and classify malicious executables in the wild," *The Journal of Machine Learning Research*, vol. 7, pp. 2721–2744, 2006.

[7] D. Reddy and A. Pujari, "N-gram analysis for computer virus detection," *Journal in Computer Virology*, vol. 2, no. 3, pp. 231–239, 2006.

[8] W. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," in *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*. IEEE, 2005, pp. 64–71.

[9] S. Stolfo, K. Wang, and W. Li, "Towards stealthy malware detection," *Malware Detection*, pp. 231–249, 2007.

[10] W. Li, S. Stolfo, A. Stavrou, E. Androulaki, and A. Keromytis, "A study of malcode-bearing documents," *Detection of Intrusions and Malware, and Vulnerability Assessment*, pp. 231–250, 2007.

[11] S. Tabish, M. Shafiq, and M. Farooq, "Malware detection using statistical analysis of byte-level file content," in *Proceedings of the ACM SIGKDD Workshop on CyberSecurity and Intelligence Informatics*. ACM, 2009, pp. 23–31.

[12] Y. Ye, D. Wang, T. Li, D. Ye, and Q. Jiang, "An intelligent pe-malware detection system based on association mining," *Journal in computer virology*, vol. 4, no. 4, pp. 323–334, 2008.

[13] Y. Ye, T. Li, Q. Jiang, and Y. Wang, "Cimds: adapting postprocessing techniques of associative classification for malware detection," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 40, no. 3, pp. 298–307, 2010.

[14] W. Wang, P. Zhang, Y. Tan, and X. He, "A hierarchical artificial immune model for virus detection," in *Computational Intelligence and Security, 2009. CIS'09. International Conference on*, vol. 1. IEEE, 2009, pp. 1–5.

[15] P. Zhang, W. Wang, and Y. Tan, "A malware detection model based on a negative selection algorithm with penalty factor," *SCIENCE CHINA Information Sciences*, vol. 53, no. 12, pp. 2461–2471, 2010.

[16] LibSVM. [Online]. Available: www.csie.ntu.edu.tw/ cjlin/libsvm/