HRS: A Hybrid Framework for Malware Detection

Zhentan Feng, Shuguang Xiong, Deqiang Cao, Xiaolu Deng, Xin Wang,
Yang Yang, Xiaobo Zhou, Yan Huang, Guangzhu Wu

Baidu Research Lab of Softwares, Baidu Inc.
Baidu Campus, No. 10, Shangdi 10th Street Haidian District, Beijing, China
{fengzhentan, xiongshuguang, caodeqiang, dengxiaolu, wangxin24,
yangyang16, zhouxiaobo, huangyan03, wuguangzhu}@baidu.com

ABSTRACT

Traditional signature-based detection methods fail to detect unknown malwares, while data mining methods for detection are proved useful to new malwares but suffer for high false positive rate. In this paper, we provide a novel hybrid framework called HRS based on the analysis for 50 millions of malware samples across 20,000 malware classes from our antivirus platform. The distribution of the samples are elaborated and a hybrid framework HRS is proposed, which consists of Hash-based, Rule-based and SVM-based models trained from different classes of malwares according to the distribution. Rule-based model is the core component of the hybrid framework. It is convenient to control false positives by adjusting the factor of a boolean expression in rule-based method, while it still has the ability to detect the unknown malwares. The SVM-based method is enhanced by examining the critical sections of the malwares, which can significantly shorten the scanning and training time. Rigorous experiments have been performed to evaluate the HRS approach based on the massive dataset and the results demonstrate that HRS achieves a true positive rate of 99.84% with an error rate of 0.17%. The HRS method has already been deployed into our security platform.

Keywords

Malware Detection, Malware Class Distribution, Data Mining, Machine Learning, Antivirus Engine

1. INTRODUCTION

Malware (malicious software) detection is a primary problem in computer security field. Traditional commercial antivirus products usually rely on signature-based methods [6], which need local signature databases to store patterns extracted from malwares by experts. However, more and more polymorphic malwares could be manufactured easily due to the sophisticated technology tools for encryption or packing. For instance, the number of unknown malicious executables

collected by our antivirus platform reaches nearly 1,000,000 per day. This leads to much storage space for the signatures, and much labor work to study malicious files.

Moreover, the update process for signature database introduces extra delay which increases the possibility of computer infection risks. In order to overcome these problems, Cloud Antivirus solutions based on cloud computing are proposed by some companies, such as KSN [1], Panda Cloud Antivirus [2]. It has been proved that the approaches can reduce the signature databases size and response time for new malwares. Nevertheless, there is no particularly differences between Cloud Antivirus and traditional signature-based approaches essentially except for introducing much more powerful computing and storage resources in cloud systems. In recent years, data mining detection methods have been proposed for detecting unknown malicious executables, such as SVM, Artificial Neural Network(ANN) [18], Association Mining [22], etc. These methods can detect unknown malwares with a high detection ratio, at the expense of a high false positive rate.

Could we find a solution which can achieve a good detection ratio for unknown malwares while keeping reasonable low false positive rate as signature-based method? A hybrid framework called HRS is proposed in this paper based on the analysis of 50 millions malware across 20,000 malware classes. According to Figure 1, the samples are roughly divided into three groups by the sizes of the malware classes. First, there are about 85% malware classes containing less than 100 samples and the total number of the files only account for 0.02%, and we consider these classes as the **light-weight group**. For another, about 24% malwares disperse in 15% malware classes with the size from 100 to 100,000, which are called by **mid-weight group**. At last, we find that nearly 76% malwares fall into the 0.1% classes with the size over 100,000, which belong to the **heavy-weight group**.

Different detection models are employed for different groups. For the light-weight group, it is difficult to apply heuristic methods because the number of the files in the class is insufficient for training. The hash-based method, considered as one simple method of signature-based approaches, is employed to build the detection model. For the mid-weight group, we provide a rule-based method based on boolean expressions, which is the core component of the HRS framework, by which we can achieve a good detection ratio while the error ratio can be controlled at a low level. For the heavy-weight group, we use an SVM-based method to train the model in order to obtain a high detection ratio.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).

IWSPA '15, March 4, 2015, San Antonio, Texas, USA.

ACM 978-1-4503-3341-3/15/03.

<http://dx.doi.org/10.1145/2713579.2713585>.

Figure 1: The CDF of malwares and classes.

Specifically, our contributions of the paper are as follows:

- **Hybrid Detection Framework (HRS):** The framework consists of three different detection methods (hash-based, rule-based and SVM-based), which are trained by separated data sets. The detailed detection workflows and performance analysis are also addressed in the following sections. Experiment evaluation demonstrates that the detection ratio of HRS is nearly the same as the data mining method, while the error ratio is much lower than the latter.
- **Rule-based method:** A rule is trained from the malware and benign files for each class in the mid-weight group, which is a boolean expression. In order to control the error ratio, the file is malicious if all the features in an element of the rule are true during the detection process. At the same time, the method has heuristic ability to detect unknown files.
- **SVM-based method relying on critical locations:** After examining the samples carefully, we provide an SVM-based method for the heavy-weight group based on critical locations, which contain the features that can distinguish the malware and benign files. By using the method, the performance of the training and scanning procedures can be significantly improved.
- **Experiments on massive dataset:** We also provide rigorous experiments to evaluate the methods on the dataset with 100,000 malwares and 100,000 benign files from our antivirus platform. The results in parameter tuning and performance comparison for the methods are also provided.

The remainder of the papers is organized as follows. Section 2 discusses the related works. Section 3 presents the overall HRS framework and related workflows. Section 4 elaborates the rule-based method, and the SVM-based approach is addressed in Section 5. Section 6 describes the performed experiments and the results. Finally, we conclude the paper and outline the future work in Section 7.

2. RELATED WORK

Malware detection methods generally fall into two categories: static methods and dynamic methods [17]. Since dynamic methods require much time to load and run the sample files, industry communities normally choose signature-based methods which belong to the static method category. However, signature-based methods are limited to detect unknown malwares, since the signatures cannot be extracted without the malware samples. Moreover, most of the malwares are encrypted or packed, hence it requires much time and manpower to find the signatures [3].

In order to overcome these disadvantages, data mining approaches are proposed for malware detection. The data mining methods are usually composed of four steps: data set preparation, feature selection, classification model training and cross-fold validation. N-gram sequence is considered as one popular feature, which is proved to be effective for heuristic detection method [12, 15]. Kolter et al [8] compare different machine learning methods based on the n-grams extracted from the binary file body. Information gain is used to select the top-K features. Through extensive experiments performed in WEKA [19], boosted J48 and SVM are proved to be promising. Similarly, a relevant n-grams of the context is introduced based on the information gain and class-wise document frequency [12]. The Dempster Shafer Theory [23] is employed to combine different classifiers (SVM, decision tree, IBK) for better results. However, the n-gram extraction procedures in the above methods are time-consuming, if the n-grams are selected from the whole file.

Opcodes are regarded as useful in feature selection, which are extracted after disassembly for the executable files. Bilar [4] statistically investigates opcode frequency distribution over different malware class, such as Virus, Worm, Trojan, etc, which indicates rare opcodes could be a stronger predictor. Lakhota et al [9] choose N-perms of opcode as the features and implement a Nearest-Neighbour (NN) algorithm to build a rapid malware classification system called VILO. A detection method based on opcode sequence is proposed by [14], which calculates the frequency of the opcode sequence between the malware and benign files. Weighted Term Frequency (WTF) is employed to select the feature vector, and different models trained by decision tree, SVM, K-nearest neighbours (KNN) and Bayesian networks are compared.

Shabtai et al [16] use TF-IDF on opcode sequence as features and conduct rigorous evaluation with various settings of opcode n-gram patterns over several classifiers: SVM, LR, ANN etc. By investigating the imbalance problem in real life environment, a chronological model updating method is proposed with an accuracy higher than 96% and an error rate of 10%. In [13], a method based on opcode graph similarity is proposed to detect the metamorphic malwares using Hidden Markov Models.

Recently, ensemble and hybrid methods are proposed for malware detection systems. Menahem et al [11] select n-grams, PE field information, and functions in PE as features for different training set and evaluate several ensemble methods. The simulation results demonstrate that Troika [10] ensemble method achieves a better detection ratio with more execution time. An Automatic Classification System (AMCS) is provided by [21], in which two types of instruction vector for feature selection are proposed. A novel method combining static and dynamic features is proposed in [20],

which provides an ensemble method with SVM and Association Rules based on hierarchical taxonomies.

3. HRS OVERVIEW

Enlightened by the distribution of malware classes, we propose a hybrid framework called HRS, which consists of a hash-based method, a rule-based approach, and an SVM-based approach with the purpose to combine their advantages and make it more practical in storage cost, heuristic ability, and false-positive control. The basic idea is training specific models to inspect specific group of malware classes as mentioned above, and using the models successively in classification process. Specifically, in the training process, HRS uses the hash-based approach and the SVM-based approach to cope with the malwares in the light-weight group and the heavy-weight group, resp. For a malware class in the mid-weight group, the rule-based approach trains a rule if the accuracy exceeds a given threshold, otherwise it is handled by the SVM-based approach.

The hash-based approach computes a signature of a file by common hashing functions, such as MD5, which is considered to perform well on the light-weight group of malwares, because it is more convenient to save the signatures than the rules or SVM-models, and the variation ability of the malware class (the size of the class) remains low. However, as the class size increases, the MD5 set increases much faster than the rules and the SVM models. Moreover, the rules and the SVM models have heuristic ability to detect unknown malwares. Hence they are more feasible for detecting malwares in the mid-weight and heavy-weight groups than MD5 sets.

Because both the detection ratio and the error ratio of the rule-based approach are often lower than the SVM-based approach, the rules have a higher priority than the SVM models to reduce false-positives if the detection ratio is acceptable. For a class in the heavy-weight group, it is often difficult to obtain a feasible rule with a high detection ratio and a low error ratio, hence the SVM models are preferred.

The training workflow of HRS is illustrated in Figure 2. HRS needs to update the database used by the Hash Engine, and train rules and SVM-models for the Rule Engines and the SVM Engines, resp. When a new file is determined as malware, if the file belongs to a light-weight class, the hashed value of the file is added into the database of the Hash Engine. Otherwise HRS updates the training file sets including the new file, trains rules and SVM models for the malware class, and selects the rules or the SVM models to inspect the malware class according to the accuracy of the Rule Engine. To avoid frequent updates of the rules and the SVM models, HRS performs a round of training if the number of the new malware files reaches to a given threshold, otherwise the hashed values of the files are temporally put into the database of the Hash Engine.

In classification, HRS combines the advantages of the three approaches. First, by using rules and SVM models to detect malwares in the heavy-weight group, the required storage space could be significantly reduced. For example, given 2,000 classes of malwares, in which each class has 10,000 files on average, the hash-based approach requires about 305MB to store the MD5 (16-byte) values, while the rule-based approach and the SVM-based approach require only 859KB in all, supposing there are 1,000 rules (80-byte), specified for 1,000 classes, and 1,000 SVM models (800-byte) speci-

Figure 2: The Training Workflow of HRS.

Figure 3: The Classification Workflow of HRS.

fied for the other 1,000 classes. Second, HRS remains the heuristic ability to detect unknown malwares relying on the rules and the SVM models, while covering the small and unpopular classes through the hash-based approach. Third, HRS could restrict the false-positive rate at a low level by using the rules. And finally, HRS has good maintainability since the training and classification procedures are highly automatic for the mid-weight and heavy-weight groups.

The classification workflow of HRS is illustrated in Figure 3. The file is successively inspected by a Hash Engine, a set of Rule Engines, and a set of SVM Engines. A Rule Engine or an SVM Engine takes the responsibility for detecting one class of malwares. HRS returns black (malicious) if any engine reports that the file is black, otherwise it returns white (benign).

4. RULE-BASED APPROACH

The rule-based approach consists of two phases: the training phase and the inspecting phase. For each class of the malwares, a **rule** is automatically generated in the training phase. The approach then uses the rules to identify an unknown file in the inspecting phase.

Comparing with traditional signature-based approaches, the approach has the ability to find out new variants or species of malwares like heuristic approaches, because the rule consists of short elements, and perform logic computations, rather than long of the set. Comparing with heuristic approaches, the false-positive rate can be controlled more easily by deleting part of the rules, rather than changing the heuristic models with the risk of bringing new false-positives.

The rest part of the section formulates a coverage problem in the training phase, to which a rule is a solution with the ability to identify malwares, shows the hardness of the problem, presents algorithms for the training phase and the inspecting phase, resp., and analyzes the time and space complexities of the algorithms.

4.1 Notations and Problem Formulation

Let $v(g)$ be a boolean variable representing whether n-gram g appears in a specified file. g covers file f only if g appears in f , and $v(g) = true$, otherwise f is not covered by g , and $v(g) = false$. For a set of file F , define the

coverage of n-gram g on F as $Cov(g, F) = |F_1|/|F|$ where $F_1 = \{f|f \in F, \text{ and } f \text{ is covered by } g\}$. For convenience, the rest part of the paper uses g to represent the boolean variable of n-gram g .

A rule R is a boolean expression composed of n-gram variables and boolean operators $\cup$ and $\cap$. The number of n-gram variables in R is denoted as $size(R)$, and the set of n-grams in R is called the supporting set of the rule, denoted as $supp(R)$. R covers file f only if the values of the n-gram variables on file f make R true. For a set of file F , define the coverage of rule R on F as $Cov(R, F) = |F_1|/|F|$ where $F_1 = \{f|f \in F, \text{ and } f \text{ is covered by } R\}$.

For example, a rule can be expressed as $R = (g_1 \cup g_2) \cap ((g_3 \cup g_4) \cap g_1 \cap g_2)$. In this case, $size(R) = 6$, and $supp(R) = \{g_1, g_2, g_3, g_4\}$.

Definition 1. (The Rule Cover Problem) Given a set of n-grams G , a set of black files B , a set of white files W , a detecting rate r_d , a false-positive rate r_f , and a size limit k , is there a rule R with $supp(R) \subseteq G$ and $size(R) \leq k$ meeting the following requirements: $Cov(R, B) \geq r_d$ and $Cov(R, W) \leq r_f$?

Intuitively, the solution is a good rule with a high detecting rate and a low false-positive rate, i.e., a high coverage on black files and a low coverage on white files. And the size of the rule should be small in order to save storage space and computational cost. It seems not easy to find out such a rule, however, as shown in the next subsection.

4.2 Problem Hardness

THEOREM 1. *The Rule Cover Problem is NP-Complete.*

PROOF. First, a rule for this decision problem can be validated by computing $Cov(R, B)$ and $Cov(R, W)$, and comparing them with r_d and r_f , resp. All these computation steps require polynomial time in the input size, so the Rule Cover Problem is in NP.

Second, a polynomial-time reduction from the Set Cover Problem to the Rule Cover Problem can be constructed. Let us consider a special case of the Rule Cover Problem, in which $r_f = 1$ and $r_d = 1$, meaning that the white files and the black files are fully covered. For any instance of the Set Cover Problem, we denote the universe, the set, and the size limit as U , S , and t , resp., and construct an instance of the Rule Cover Problem by setting $G = S$, $B = U$, $W = U$, and $k = t$. On one hand, if there is a set $S' \subseteq S$ that can cover U , let a rule $R = \cup g, \forall g \in S'$, and we have $Cov(R, B) = 1 \geq r_d$, $Cov(R, W) = 1 \leq r_f$. Hence R is a solution to the Rule Cover Problem. On the other hand, for any solution R to the Rule Cover Problem, the set $S'' = supp(R)$ can fully cover U because $r_d = 1$, and $|S''| \leq size(R) = t$. So S'' is a solution to the Set Cover Problem.

Because the Set Cover Problem is NP-Complete [7], the Rule Cover Problem is NP-Complete. $\square$

4.3 The Rule Generation Algorithm

The Rule Generation Algorithm includes a heuristic strategy to solve the Rule Cover Problem. In addition, because the n-gram set may be difficult to obtain manually in some cases, e.g., dealing with a large malware class containing thousands of files, the algorithm also includes a strategy to compute the n-gram set.

Algorithm 1 N-gram Selection

INPUT: white files W , black files B of a malware class, white limit l_w , black limit l_b , and the size k of the returned n-gram set.
OUTPUT: a set of n-grams.

- 1: scan B to obtain all the n-grams G , and $\forall g \in G$, calculate $Cov(g, B)$;
 - 2: scan W to calculate $Cov(g, W)$, $\forall g \in G$;
 - 3: remove g from G , if $Cov(g, B) < l_b$ or $Cov(g, W) > l_w$, $\forall g \in G$;
 - 4: sort the n-grams in G by r_{WB} , and return the top- k n-grams;
-

Given a set of white files, and a set of black files, the algorithm automatically computes a rule that can distinguish the two sets of files. The algorithm consists of 3 major steps: N-gram Selection, Factor Selection, and Rule Generation. The first step aims to find out a set of n-grams that can roughly distinguish the black files and the white files. A white limit and a black limit are used to filter out the n-grams with weak classification ability. The rest n-grams are sorted in non-descending order by the white-black rate $r_{WB} = |W_1|/|B_1|$, in which W_1 refers to the set of white files that cover the n-gram, and B_1 is the set of black files covering the n-gram. Only the top- k n-grams are selected, because the n-gram set cannot be too large to be used in the following steps. The details are shown in Algorithm 1.

Taking the n-gram set G as an input, it seems hard to obtain a rule since the Rule Cover Problem is NP-Complete. Furthermore, a scan performed on the white file set which may contain millions of files in practice may consume much time, which increases the difficulty to solve the problem.

Thus we do not focus on exact algorithms searching the entire resolution space. Instead, we present a heuristic strategy by a two-phase-construction manner, which performs only two scans on the white file set. Specifically, two n-grams in G are combined into a factor f , which covers a file only if both the two n-grams covers the file. In this way, both $Cov(f, B)$ and $Cov(f, W)$ for black file set B and white file set W can be restricted at a low level. After that, part of the factors are selected in a greedy way, and combined by $\cup$ operator to construct the rule, which means that if any factor covers a file, the rule covers the file. A white limit is used in each iteration to restrict the increasing speed of $Cov(f, W)$ while the $Cov(f, B)$ increases in the greedy way.

As the first phase for computing a rule, Factor Selection requires an n-gram set G , and outputs a factor set F . The procedure of Factor Selection is similar to that of N-gram Selection, however, a factor in the produced F probably has a much lower coverage on the white files than the n-grams in G . To see the restriction on the false-positive rate, let us consider an example of two n-grams g_a and g_b with $Cov(g_a, W) = Cov(g_b, W) = 50\%$ and $Cov(g_a, B) = Cov(g_b, B) = 0.5\%$. If there is no strong relation between the distributions of the two n-grams, $Cov(f, W)$ would be around 25%, while $Cov(f, B)$ would sharply reduce to around 0.0025%. A white limit is used to filter out the factors in which the two n-grams have a strong relation. The details are shown in Algorithm 2.

Rule Generation is the second phase to solve the problem, with an input of factor set F , and an output of a rule by combining part of the factors with operator $\cup$, as illustrated in Algorithm 3. In a straight-forward way, all the factors are put into the rule to obtain the highest coverage on the black files. However, it is unnecessary when a small subset of F

Algorithm 2 Factor Selection

INPUT: white files W , black files B of a malware class, n-gram set G , white limit l_w , black limit l_b , and the size k of the returned factor set.

OUTPUT: a set of factors.

- 1: generate all the factors F derived from G ;
 - 2: scan W and B to calculate $Cov(f, W)$ and $Cov(f, B)$, $\forall f \in F$;
 - 3: remove f from F , if $Cov(f, B) < l_b$ or $Cov(f, W) > l_w$, $\forall f \in F$;
 - 4: sort the factors in F by r_{WB} , and return the top- k factors;
-

Algorithm 3 Rule Generation

INPUT: white files W , black files B of a malware class, factor set F , black limit l_b , and the maximum size k of the returned rule.

OUTPUT: a rule.

- 1: $R = \emptyset$;
 - 2: scan B to obtain a boolean matrix M in which $M[i][j]$ records whether $f_i \in F$ covers $b_j \in B$;
 - 3: **while** $F \neq \emptyset$ and $size(R) < k$ and $Cov(R, B) < l_b$ **do**
 - 4: choose a factor f with the highest coverage on B according to M ;
 - 5: update B and M by removing the files covered by f and deleting the related columns
 - 6: remove f from F , and add F into R ;
 - 7: return R ;
-

has a high coverage, and impractical when the factor set is large, and results in much computation or storage cost in the detecting phase. So a greedy method is used to pickup less factors when meeting the coverage requirements. The algorithm also checks the false-positive rate induced by a factor, and removes the factor from the rule if the rate is higher than the white limit.

4.4 The Algorithm in Classification

The algorithm for inspecting a file is simple: First, it reads the rules, organizes the rules and the n-grams as trees and hash-maps in memory, resp., and scans the file to update the hash-maps according to the appearance of the n-grams. After the scanning, the algorithm computes the boolean value of a rule on the file by traversing the tree of the rule. If any rule covers the file, the algorithm determines that the file is black. If more than one rules that cover the file, the algorithm returns the malware type with the largest number of black files among the malware classes corresponding to the rules.

4.5 Performance Analysis

In practice, people may expect an algorithm for malware detection that takes only the black files and the white files as input, and automatically computes a rule that meets all the requirements. A brutal-force algorithm can obtain a rule by the following steps: (1) enumerate all the subsets of the n-grams appearing in all the files, (2) enumerate all the boolean operators and calculation priorities for each subset to obtain a specific rule, (3) compute the coverage of the rule on both white and black files, and (4) select a rule that meets the coverage requirements. Nevertheless, the algorithm induces too large computation cost to be practical.

Comparing with the brutal-force algorithm, the proposed heuristic algorithm makes it much more easy to obtain a rule: all the three steps of the algorithm run in polynomial-time in the input size, and there are only two scans on the white files, and three scans on the black files.

Furthermore, the algorithm can balance the trade-offs between the detecting/false-positive rates and the computation costs. Specifically, the Factor Selection Algorithm can be performed iteratively to construct new factors with more than two n-grams in each factor, so as to obtain a rule supported by a larger set of factors. It means that the solution space is increased with more types of factors, probably resulting in a rule with a higher detecting rate and a lower false-positive rate, at the expense of more times of scans on the training file sets. For the same reason, the rule-based approach is more flexible, and has a stronger ability to identify malwares than single-ngram-based approaches or multi-ngram-based approaches, where the n-grams are united only by $\cap$ operators.

In inspecting phase, the algorithm performs a lookup operation on the hash-maps for each n-gram in a file. Since each hash-map corresponding to a rule has a size of $O(k)$, supposing there are m rules in total, the lookup operation requires $O(\log(mk))$ time for processing each byte of the file. After the scan, the algorithm requires $O(k)$ time to gather the boolean results from a tree corresponding to a rule, so it will cost $O(mk)$ time to obtain the final result. To save the rules on local disk, $O(mk)$ storage space is required.

5. SVM-BASED APPROACH

The SVM-based approach trains SVM models using white files and black files, then classifies unknown files by the trained models. In the training and classification procedures, a file is transformed to a boolean vector so that the vector can be used by an SVM **trainer** or an SVM **classifier**. This section describes the main procedure of the approach, and presents an optimization strategy to reduce the computation and storage costs.

5.1 The Main Procedure

The SVM-based approach consists of three phases: N-gram Selection, Training, and Classification. The N-gram Selection phase outputs a set of n-grams by scanning the white and black files, denoted as G , which is used to generate a boolean vector for each file. The Training phase takes the vectors of the white and black files as input, and outputs an SVM model. The Classification phase outputs whether a file is black according to the scalar product of the model and the boolean vector of the file.

Given a white file set W , a black file set B , and $U = W \cup B$, for each n-gram g in the whole n-gram space, the N-gram Selection phase computes the Information Gain of g by:

$$IG(g) = \sum_{v \in \{0,1\}} \sum_{C \in \{B,W\}} P(g, C, v) \log \frac{P(g, C, v)}{P(g, v)P(C)} \quad (1)$$

where $P(g, C, v) = |F(g, C, v)|/|U|$, in which $F(g, C, v) = \{f | f \in C, \text{ and } v(g, f) = v\}$, $P(g, v) = |F(g, v)|/|U|$, in which $F(g, v) = \{f | f \in U, \text{ and } v(g, f) = v\}$, and $P(C) = |C|/|U|$. The n-grams are sorted by their information gains in non-ascending order, and only the top- k n-grams are selected. Here the n-grams are sorted by the information gain rather than the white-black rate in the rule-based approach, because in this way, more n-grams in the selected set cover the train files and the file to be inspected, which helps improve the accuracy in both the Training phase and the Classification phase.

For each file $f \in U$, the Training phase scans f to obtain a boolean vector $V(f) = [v(g_0, f), \dots, v(g_n, f)]$, $\forall g_i \in G$. All the vectors and the types of the files are combined in a boolean matrix as the input of the trainer, in which a row is $[v(f), v(g_0, f), \dots, v(g_n, f)]$, $\forall g_i \in G$ where $v(f) = 0$ if $f \in W$, otherwise $v(f) = 1$. The trainer uses the Stochastic Gradient Descent Algorithm (SGD) [5] to obtain a feasible solution.

In the Classification phase, the boolean vector $V(f)$ of the unknown file f is computed as the input of the classifier, which calculates a vector $s(V(f)) = [s_w, s_b]$ according to $s(x) = Wx + b$, where W and b are contained in the model, and s_w and s_b are the scores for f to be black and white, resp. The classifier returns that f is white if $s_w > s_b$, otherwise it returns that f is black.

5.2 Performance Optimization

When using the SVM-based approach to inspect a file, the scanning speed and the consumed storage space are highly related to the scanned file size, and the size of the n-gram set G , since the vector size of a file is the same as $|G|$. The performance thus can be optimized by reducing the scanned file size or the size of the n-gram set. On one hand, although a smaller size of the n-gram set results in a better performance, the accuracy of the approach are probably lowered because some n-grams that have the ability to distinguish part of the black files are excluded from the set. The trade-off can be handled by conducting experiments to obtain a smallest n-gram size while the accuracy remains higher than a given threshold. On the other hand, the scanning speed could be improved by scanning specified critical locations of the file rather than everywhere of the file. Here we give an optimization strategy to obtain such critical locations.

In the optimization strategy, a file is divided into adjacent fragments. Specifically, the algorithm parses the file and obtains its PE structure, which consists of several Sections, such as `.text`, `.code`, and `.data`, etc. Then each Section is divided into adjacent fragments with a fixed length l , i.e., 1024 bytes. The last fragment is allowed to be shorter than the fixed length. A **fragment location** fl refers to the location of a fragment, determined by the Section name and the start position in the Section of a fragment. For instance, $fl = (.text, 5)$ means that the fragment locates at the `.text` Section with an offset of $5l$ bytes.

A **weight** of a fragment location fl , denoted as $w(fl)$, is defined as the number of files in which at least one n-gram appear in fl . Specifically, given the n-gram set G , the algorithm scans each file f in all the training files, and if any n-gram $g \in G$ first appears in a fragment location fl of f , the algorithm increases $w(fl)$ by 1. Note that $w(fl)$ is only increased once if the n-gram appears more than once in fl of the same file, and $w(fl)$ has no relation to a file f if f does not contain the fragment located at fl .

Part of the fragment locations are selected as the critical locations according to their weights, which reflects the dense of n-grams that appears at the locations. The pseudo-code is shown in Algorithm 4.

To inspect a file with the optimization, the SVM-based approach needs to merge all the fragment locations for detecting different class of malwares. The approach then parses the file and obtains its boolean vector by scanning only the merged fragment locations. After that, the classifier takes

Algorithm 4 Location Selection

INPUT: white files W , black files B of a malware class, length of location l , and the size k of the returned locations.
 OUTPUT: a set of fragment locations.

```

1:  $w = \{\}$ ;
2: for  $f \in W \cup B$  do
3:   scan  $f$  to obtain  $G' = \{g \in G, \text{ and } g \text{ covers } f\}$ ;
4:   for  $g \in G'$  do
5:     compute  $fl$  of  $g$  in  $f$  according to  $l$ ;
6:     if  $fl \in w$  then
7:        $w(fl) + 1$ ;
8:     else
9:       add  $fl$  into  $w$ , and let  $w(fl) = 1$ ;
10: sort the fragment locations by their weights in non-ascending
    order, and return the top- $k$  fragment locations;
```

the boolean vector as input, and outputs the classification result.

6. EXPERIMENTS

Extensive experiments are conducted to evaluate the performance of the proposed HRS framework and the algorithms. This section presents the experiment setup, parameter tuning of the proposed approaches, and performance evaluation.

6.1 Experiment Setup

In the experiments, all the files are selected from our antivirus platform. We focus on 6 malware classes: Backdoor.Spynet, Trojan-PSW.Facebook, Trojan-PSW.Stealer, Adware.Hao123, Worm.Kolab, and Trojan-Downloader.Was-ki, with the size of 48, 118, 290, 5473, 14305, 114830 files, resp. Most of the black files are selected in training rules and SVM models, while the rest files are treated as unknown files for evaluating heuristic ability. The 100,000 white files are randomly selected from a white file set mainly composed of system files and other files with authentic signatures.

The experiment metrics include True-Positive (TP) rate, False-Positive (FP) rate, scanning speed, consumed storage space, and heuristic ability. The TP rate is the detection ratio, while the FP rate is the portion of the false-positive results over all the scanned white files. The heuristic ability is evaluated by the number of detected unknown files. The experiments run on computers with Linux system, Intel Xeon 12-core CPU at 2GHz, 64GB memory, and 1.5TB hard disk drive.

6.2 Parameter Tuning

In the rule-based approach, the parameters to be tuned are the white/black limit (l_w/l_b) of the n-grams and the factors, and the size of the n-gram set and the factor set.

The Trojan-PSW.Stealer class and Adware.Hao123 class are selected in the experiments. The experiments evaluate the impact of the parameters on the TP/FP rates by varying the white/black limit of the n-grams and the size of the n-gram set, and varying the white/black limit of the factors and the size of the factor set.

The detailed results are shown in Figure 4. For the Adware.Hao123 class with 5473 files, the TP rate is 100% and the FP rate is 0 in all the cases, which implies that the rule-based approach performs excellently on this class. For the Trojan-PSW.Stealer class with 290 samples, the max TP rate is 51.67%, while the FP rate is 0.02%. The results

Figure 4: Parameter Tuning of the Rule-based Approach.

Figure 5: The TP/FP rates of the SVM-based Approach over Variant Training Scales and Portions.

show that the rule-based approach is not very suitable for the Trojan-PSW.Stealer class, since the TP rate cannot exceed 60% in all the cases.

Figure 5 depicts the TP/FP rates of an SVM model over variant size of the training file set and variant portion of the black files over the white files. The TP rate decreases as the portion decreases from 15:1 to 1:15, and a smaller size of the training set results in a sharper decreasing speed. The FP rate also decreases as the portion decreases. The FP rate with 80,000 training files is lower than that with 20,000 or 40,000 files. The results imply that the portion has significant impact on the TP/FP rates, and using more files in training yields a better result in classification.

In selecting n-grams, training SVM models, and classifying unknown files, there are two options to extract n-grams from a file: (1) using the full text or (2) using the critical locations. As shown in Figure 6-(a), the scanning speed has a dramatic increase from 15 file/s to 183 file/s when the

Figure 6: Using Critical Locations in Training and Classification of the SVM-based Approach.

Table 1: Using Critical Locations in N-gram Selection of the SVM-based Approach.

Approach	TP Rate	FP Rate	Scanning Speed
Full	98.66%	3.02%	545 file/s
Critical	98.27%	1.46%	556 file/s

critical fragments with a total size of 10KB are used instead of the full text in classification. As the size of the critical fragments decreases from 10KB to 1KB, the scanning speed increases from 183 file/s to 553 file/s, the TP rate varies over 85%, while the FP rate increases to 23.3%, which implies that the the scanning speed is increased at the expense of decreasing the accuracy of the SVM-based approach. If the critical fragments are also used in training (Figure 6-(b)), the FP rate reduces from 23.3% to 3.02%, while the TP rate remains over 94%. The FP rate can be further reduced by selecting n-grams at critical locations, which means that a larger set of n-grams is computed on full text to obtain the critical locations in advance. The results in Table 1 show that the FP rate decreases from 3.02% to 1.46% with the critical locations of 1KB size.

6.3 Performance Comparison

The performances of the hash-based approach, the rule-based approach, the SVM-based approach, and HRS are evaluated by the TP rate, the FP rate, the scanning speed, the consumed storage space, and the heuristic ability. Each approach needs to scan the black files and 100,000 white files. In HRS framework, the hash-based approach takes responsibility for Backdoor.Spynet and Trojan-PSW.Facebook, the rule-based approach inspects Trojan-PSW.Stealer and AdWare.Hao123, while Worm.Kolab and Trojan-Downloader.Waski are detected by the SVM-based approach.

The overall comparison results of the approaches are shown in Table 2. The hash-based approach has the highest detection rate with an FP rate of zero on the training files, while it fails to detect unknown files, and the required storage reaches as much as 1.8MB, which makes it unpractical. Both the TP rate and the FP rate of the rule-based approach are lower than the SVM-based approach, while the former runs faster than the latter, with a set of rules of only 0.4KB size. The TP rate of HRS is higher than the rule-based approach, while the FP rate is lower than the SVM-based approach. The results imply that HRS achieves a good balance between the TP rate and the FP rate, and the heuristic ability is remained with a relatively high scanning speed.

Table 2: Performance Comparison of the Approaches

Approaches	TP Rate	FP Rate	Storage Cost	Scanning Speed	Unknown Files
Hash	100.00%	0.00%	1.8MB	702 file/s	0
Rule	76.10%	0.01%	0.4KB	20 file/s	1826
SVM	99.93%	4.17%	46.9KB	15 file/s	1960
HRS	99.84%	0.17%	17.9KB	18 file/s	2329

7. CONCLUSION AND FUTURE WORK

In this paper, we investigate the distribution of the malware classes and find out that the classes obey the power-law distribution. Enlightened by the finding, we propose a hybrid framework called HRS consisting of a hash-based approach, a rule-based approach, and an SVM-based approach, which combines the advantage of the three approaches. We present a rule-based approach which is more flexible than traditional signature-based approaches, and we also present an optimization strategy to increase the scanning speed by using critical locations. Extensive experiments are conducted to evaluate the performance of the proposed approaches. The experimental results show that HRS significantly reduces the storage space comparing with the hash-based approach, significantly reduces the FP rate than the SVM models, and has a higher heuristic ability than the rule-based method.

In the future, we plan to extend the HRS approach with more classes of malwares, and more sophisticated strategy is one of the possible research directions.

8. REFERENCES

- [1] <http://ksn.kaspersky.com/en>.
- [2] <http://www.cloudantivirus.com/en>.
- [3] Z. Bazrafshan, H. Hashemi, S. Fard, and A. Hamzeh. A survey on heuristic malware detection techniques. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 113–120, May 2013.
- [4] D. Bilar. Opcodes as predictor for malware. *Int. J. Electron. Secur. Digit. Forensic*, 1(2):156–168, Jan. 2007.
- [5] L. Bottou. Stochastic learning. In O. Bousquet and U. von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.
- [6] E. Filiol. Malware pattern scanning schemes secure against black-box analysis. *Journal in Computer Virology*, 2(1):35–50, 2006.
- [7] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Macmillan Higher Education, San Francisco, CA, USA, 1979.
- [8] J. Z. Kolter and M. A. Maloof. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.*, 7:2721–2744, Dec. 2006.
- [9] A. Lakhotia, A. Walenstein, C. Miles, and A. Singh. Vilo: a rapid learning nearest-neighbor classifier for malware triage. *Journal of Computer Virology and Hacking Techniques*, 9(3):109–123, 2013.
- [10] E. Menahem, L. Rokach, and Y. Elovici. Troika - an improved stacking schema for classification tasks. *Inf. Sci.*, 179(24):4097–4122, Dec. 2009.
- [11] E. Menahem, A. Shabtai, L. Rokach, and Y. Elovici. Improving malware detection by applying multi-inducer ensemble. *Comput. Stat. Data Anal.*, 53(4):1483–1494, Feb. 2009.
- [12] D. Reddy and A. Pujari. N-gram analysis for computer virus detection. *Journal in Computer Virology*, 2(3):231–239, 2006.
- [13] N. Runwal, R. M. Low, and M. Stamp. Opcode graph similarity and metamorphic detection. *J. Comput. Virol.*, 8(1-2):37–52, May 2012.
- [14] I. Santos, F. Brezo, X. Ugarte-Pedrero, and P. G. Bringas. Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Inf. Sci.*, 231:64–82, May 2013.
- [15] M. Schultz, E. Eskin, E. 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*, pages 38–49, 2001.
- [16] A. Shabtai, R. Moskovitch, C. Feher, S. Dolev, and Y. Elovici. Detecting unknown malicious code by applying classification techniques on opcode patterns. *Security Informatics*, 1(1), 2012.
- [17] M. Siddiqui, M. C. Wang, and J. Lee. A survey of data mining techniques for malware detection using file features. In *Proceedings of the 46th Annual Southeast Regional Conference on XX*, ACM-SE 46, pages 509–510, New York, NY, USA, 2008. ACM.
- [18] G. Tesauro, J. Kephart, and G. Sorkin. Neural networks for computer virus recognition. *IEEE Expert*, 11(4):5–6, Aug 1996.
- [19] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.
- [20] C. Z. YB Lu, S. Din and B.Gao. Using multi-feature and classifier ensembles to improve malware detection. *Journal of CCIT*, 39(2):57–72, 2010.
- [21] Y. Ye, T. Li, Y. Chen, and Q. Jiang. Automatic malware categorization using cluster ensemble. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 95–104, New York, NY, USA, 2010. ACM.
- [22] 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*, 4(4):323–334, 2008.
- [23] B. Zhang and S. Srihari. Class-wise multi-classifier combination based on dempster-shafer theory. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 2, pages 698–703 vol.2, Dec 2002.