International Journal of Security and Its Applications
Vol. 10, No. 6 (2016) pp.75-86
http://dx.doi.org/10.14257/ijsia.2016.10.6.09

A Detection Framework of Malicious Code Based on Multi-Classifiers Ensemble

Chao Dai*, Jianmin Pang, Feng Yue, Pingfei Cui, Di Sun and Liang Zhu

State Key Lab of Mathematical Engineering and Advanced Computing of China
daichaochn@aliyun.com

Abstract

Malicious code detection is one of the important missions of malicious code analysis. Current researches on the detection of malicious code mostly focused on single classifier, whereas the single classifier is not suitable for the detection based on features of different types. We utilized multi-classifiers ensemble based on fuzzy integral to improve the accuracy of the detection framework. A framework based on the Choquet fuzzy integral was proposed to fuse the analysis results of the base classifiers with different features. And the genetic algorithm was used to obtain the fuzzy measure. Finally, the result of Choquet fuzzy integral was compared to a threshold predefined to determine the maliciousness of binary code. Experiment showed that the framework proposed in this paper could be used to determine the maliciousness of binary code more accurately.

Keywords*: Malicious code, Multi-classifiers ensemble, Fuzzy integral, Fuzzy measure, Genetic algorithm*

1. Introduction

Malicious code detection is one of the important missions of malicious code analysis, which laying solid foundations for situation awareness, threat intelligence and so on. At present, many domestic and foreign organizations have carried out a long and extensive research on the malicious code detection. But they usually focused on a single classifier to distinguish the malicious code and the benign code, which is difficult to guarantee the accuracy of the malicious code detection based on multiple features of different types.

For example, lots of malicious code detection frameworks utilized system calls as a feature to determine the maliciousness of binary code. However, it is not reasonable to completely rely on system calls to counter all kinds of malicious code. The disassembly instruction also plays an irreplaceable role in the process of detection.

First, targets descripted by different features are different. The target of disassembly instruction analysis is the disassembly instruction of the binary code. Whereas the target of system call analysis is the system call extracted from the disassembly instructions. The fewer steps to obtain the target object, the fewer interference. The developers of malicious code could resort to methods such as the conditional jump, the exception of the subroutine and so on to obfuscate the disassembly instructions. In this way, it is difficult to obtain the correct system call from the incorrect disassembly instructions.

Second, implementation methods of malicious code function are different. Many functions of malicious code could be implemented with the assembly language directly without using system calls, such as changing the file structure, inserting code, encoding, decoding and so on. So it is impossible to locate system calls with corresponding functions. Therefore, malicious code detection based on the classification of single feature would lead to false positive.

* Corresponding Author

ISSN: 1738-9976 IJSIA

In view of the situation mentioned above, in this paper we presented a detection framework based on multi-classifiers ensemble. In this way, each feature could be analyzed independently, and could be fused complementarily in the end. The rest of the paper is organized as follows. Section 2 provides a brief overview of the previous work related to the detection of malicious code. Section 3 introduces the basic of fuzzy integral, which plays an important role in the detection framework proposed by us based on multi-classifiers ensemble. Section 4 describes the architecture of static analysis which are sources of base classifiers and the implementation of the ensembled multi-classifiers. Section 5 describes the experiment and the analysis of experiment result. Finally, section 6 concludes our work and discusses the future work.

2. Related Works

Malicious code detection is a hot topic in the field of information security. There are many studies tried to detect malicious code with various classifiers based on different features, such as file structure, disassembly instruction, flow graph and system call.

File structure is an important factor measuring the maliciousness of binary code. With regard to a piece of normal code attached by malicious code, many data structures would be changed, such as the entrypoint of normal code, the segment size, the characteristic of segment *etc*.. Peter Szor listed some suspicious alignment of file structure in his monograph. [1] Which proved that the abnormity in file structure could be used to detect malicious code.

Qiao Jin *et. al.*, [2] built a binary program classifier that can differentiate packers and identify new packers as they emerge. They focused on the PE-header information of packed binary to obtain the relationship of different packers and PE-header patterns. Based on the observation that different packers have different PE-header styles, and programs generated from the same packer share common properties. They constructed decision tree to classify the packed code and unpacked code. Ashish Saini *et. al.*, [3] regarded the count of suspicious section as one of the features to distinguish malware files from benign ones. Analysis of section information outputted two values for each section: virtual size and raw data size. If the difference between these sizes was huge enough, then the code would be regarded as suspicious. Authors took four classifiers as candidate classifier, including: naïve bayes, multi-layer perceptron neural network, decision tree and J48.

Assembly instruction is the low-level representation of binary code. Each assembly instruction is composed of opcodes and operands. The opcode reveals the operation of each assembly instruction. So the detection relied on the assembly instruction mainly focused on the opcode.

Muazzam Siddiqui [4] *et. al.*, proposed a method using data mining technology to detect malicious code by identifying critical opcode sequence. The author used IDA Pro to obtain the disassembly code, and took the frequency of the opcode sequence appeared in the sample set (including normal code and malicious code) as the main feature selection criteria. They selected 1134 sequences, and calculated the frequency of the above sequences in the sample file. Finally, the author tested the logic regression, neural network, decision tree as a classifier to detect malicious code. Robert Moskovitch *et. al.*, [5] resorted to IDA Pro as well to obtain the disassembly instruction. Then n-gram opcode sequences were extracted from the disassembly instruction. Authors selected *1*-gram, *2*-gram, *3*-gram, *4*-gram, *5*-gram and *6*-gram as the feature respectively. In the end, they tested artificial neural networks, decision trees, naïve bayes, and their boosted versions, BDT and BNB as the classifier respectively. Igor Santos [6] *et. al.*, proposed a method to detect the malicious code variant based on the frequency of opcode. The author first selected the length of the sequence of operations, and then calculated the TF for each length of n. Then the vector $S=(o_1, o_2, o_3, \ldots, o_{n-1}, o_n)$ that indicating the

frequency of operation sequences was calculated. After weighting the frequency of opcode sequence, the vector $v=(wtf_1, wtf_2, wtf_3, \ldots, wtf_{n-1}, wtf_n)$ that indicating the weighted opcode sequence was obtained. For the two programs that will be analyzed, the sequence of opcode were extracted as the feature vector v and u. The detection of malicious code variant was implemented by cosine similarity.

Control flow is an important structure in the compilation process. The control flow graph is a directed graph. The node in the graph indicates the basic block. And the edge in the graph indicates the direction of the transfer to the basic block. The control flow graph characterizes the logic structure of the code, which is useful in the detection of malicious code.

Shahid Alam *et. al.,* [7] proposed ACFG (Annotated Control Flow Graph) and Control SWOD-CFWeight (Window and Difference Flow Weight Sliding) to detect malicious code. ACFG realized a fast matching of control flow chart without affecting the detection accuracy. SWOD-CFWeight alleviated the effect of the change of the frequency of the operation code to some extent. Gao *et. al.,* [8] used a new subgraph isomorphism technique to analyze code control flow. The similarity measure could be realized by means of the subgraph matching, that is, the process of identifying the common subgraph. The similarity measure was obtained by the calculation of maximum common subgraph.

Program always resorts to system calls to request the service of operating system kernel. The operation of malicious code on kernel objects have to be implemented with the help of system call as well. So the system call is an important way to describe the behavior of code.

Surekha Mariam Varghese [9] *et. al.,* represented a specific process as sequences of system calls. The system call frequency table was constructed for each sequence. The frequency of single system call in the trail of process execution indicated the state of specific process. The detection of abnormal behavior which represent malicious code was implemented with bayes model. Zhang Boyun [10] *et. al.,* proposed a method of support vector machine based on rough set for malicious code detection. Through monitoring the implementation of malicious samples, the author tracked API system calls, using sliding window method to extract system call sequence. Due to the large size of the extracted sequence, the author used the rough set theory to eliminate the redundant features. Finally, the support vector machine was utilized to classify the malicious code. Yin Chuanhuan [11] *et. al.,* obtained the system call traces by monitoring process. The normal trail was used to construct a fixed length model. Any deviation from normal mode was considered as an exception. Using sliding window to extract the unique sequence of length k from the system call path, the author chose k=10. The support vector machine with high order Markov kernel was used to detect malicious behavior.

Different single classifiers were selected and adapted for malicious code detection based on a certain feature. With regard to the problem discussed in the introduction, it is difficult to deal with out of the weakness of single classifier. Multi-classifiers ensemble is one of the resolutions to achieve better performance.

3. Basic of Fuzzy Integral

For a classification problem accomplished by a number of classifiers, the classification accuracy could be improved by combining the output of multiple classifiers in a reasonable way. [12, 13] Towards combining the output of multiple classifiers, the weighted average is the traditional approach. The weighted average is essentially the Lebesgue integral with respect to the classical additive measure. The measure is defined in the discrete space composed by each classifier, and is determined by the weight of each classifier. This method is based on the assumption that the function of each classifier is independent of each other, so that the joint function of several classifiers is equivalent to the weighted sum of the individual classifiers. However, in many practical

applications, the interaction between a numbers of classifiers is not negligible. The fuzzy measure and fuzzy integral theory can take the non-negative and monotone set function, namely the so-called fuzzy measure to replace the common weight, and fuzzy integral to replace the common weighted average method.

The fuzzy measure and fuzzy integral theory are a continuation of the classical measure theory. The classical measure theory is based on the Lebesgue measure and integral theory proposed by French mathematician Lebesgue. It is found that the additive in the classical measure theory is not easy to be grasped in some practical applications. In 1965, the American control expert Zadeh proposed the concept of fuzzy set, which marked the birth of fuzzy mathematics and resulted in the production of fuzzy measure. In 1974, the Japanese scholars Sugeno first proposed a set of functions which use the monotonous to replace the additive condition, called the fuzzy measure, and defined the integral corresponding to fuzzy measurable.[14] Owing the characteristic of non-additive of fuzzy measure, it not only reflects the importance of the individual classifiers, but also reflects the interaction between the classifiers.

3.1. Fuzzy Measure

The additivity of classical measure is the condition of the concern. In some practical application this condition is too strong, and it limits its application range. In the measurement of subjective or non-repetitive experiments, the non-additivity is inevitable. The main feature of fuzzy measure is non additive.

Let X be a nonempty set, F be a nonempty class of subset of X, and $\mu:\mathrm{F}\to[0,\infty]$ be a nonnegative, extended real valued set function defined on F.

Definition 1. [15] μ is called a fuzzy measure on (X,F) iff

(1) $\mu(\varnothing)=0$ when $\varnothing\in\mathrm{F}$;

(2) $E\in\mathrm{F}$, $F\in\mathrm{F}$ and $E\subset F$, imply $\mu(E)\le\mu(F)$;

(3) $E_n\in\mathrm{F}\ (n=1,...,\infty)$, $E_1\subset E_2\subset\cdots$, and $\bigcup_{n=1}^{\infty}E_n\in\mathrm{F}$, imply

$$\lim_n\mu(E_n)=\mu\left(\bigcup_{n=1}^{\infty}E_n\right);$$

(4) $E_n\in\mathrm{F}\ (n=1,...,\infty)$, $E_1\supset E_2\supset\cdots$, $E_n\in\mathrm{F}\ (n=1,...,\infty)$, $\mu(E_1)<\infty$, and $\bigcap_{n=1}^{\infty}E_n\in\mathrm{F}$, imply

$$\lim_n\mu(E_n)=\mu\left(\bigcap_{n=1}^{\infty}E_n\right).$$

If μ satisfies the condition (1), (2), (3), it is called a lower semicontinuous fuzzy measure. If μ it satisfies the condition (1), (2), (4), it is called an upper semicontinuous fuzzy measure. If $\mu(X)=1$, it is called a regular fuzzy measure.

Definition 2. [15] μ satisfies the λ-rule on F iff there exists

$$\lambda\in\left(-\frac{1}{\sup\mu},\infty\right)\bigcup\{0\},$$

where $\sup\mu=\sup_{\mathrm{E}\in\mathrm{F}}\mu(\mathrm{E})$, such that

$$\mu(\mathrm{E}\bigcup F)=\mu(\mathrm{E})+\mu(F)+\lambda\cdot\mu(\mathrm{E})\cdot\mu(F),$$

whenever

$E\in\mathrm{F}$, $F\in\mathrm{F}$, $E\bigcup F\in\mathrm{F}$, and $E\bigcap F=\varnothing$.

Definition 3. [15] μ is called a λ-fuzzy measure on F iff it satisfies the λ-rule on F and there exists at least one set $\mathrm{E}\in\mathrm{F}$ such that $\mu(\mathrm{E})<\infty$. λ-fuzzy measure is usually denoted by g_λ.

Definition 4. [16] A fuzzy measure μ is called to be *k*-order additive or *k*-additive if its Möbius transform vanishes for any $A\subseteq X$ such that $|A|>k$ and there exists at least one subset *A* with exactly *k* elements such that $\mathrm{m}(A)\neq 0$.

One of the reasons for the wide application of fuzzy integral is that the interaction between different attributes can be expressed by the fuzzy measure values of different attributes.

Let μ be a fuzzy measure on feature set *X*, it is said to be

- additive if $\mu(E\cup F)\le\mu(E)+\mu(F)$ whenever $E\cap F=\varnothing$;
- superadditive if $\mu(E\cup F)\ge\mu(E)+\mu(F)$ whenever $E\cap F=\varnothing$;
- subaddtive if $\mu(E\cup F)\le\mu(E)+\mu(F)$ whenever $E\cap F=\varnothing$.

The superadditive of fuzzy measure indicates that the contribution of two sets of attributes combined together is less than the sum of the two set. That means the two attributes cooperated positively. The subadditive of fuzzy measure indicates that the contribution of two sets of attributes combined together is greater than the sum of the two set. That means the two attributes cooperated negatively.

3.2. Fuzzy Integral

Definition 5.[17] Given a measurable space(X,F), let μ be a fuzzy measure on *X*, and *f* a nonnegative finite measurable function on *X* with range $\{a_1,a_2,\ldots,a_n\}$ where $a_1\le a_2\le\ldots\le a_n\le 1$. The Sugeno fuzzy integral of *f* with respect to μ is defined by

$$(\mathrm{s})\int f d\mu=\bigvee_{i=1}^{n}\left[a_i\wedge\mu\left(\left\{x\middle|f(x)\ge a_i\right\}\right)\right].$$

The Sugeno fuzzy integral is also denoted by $(s)\int f(x)d\mu(x)$ or $(s)\int f d\mu$.

Definition 6.[17] Given a measurable space(X,F), let μ be a fuzzy measure on *X*, and *f* a nonnegative finite measurable function on *X* with range $\{a_1,a_2,\ldots,a_n\}$ where $a_1\le a_2\le\ldots\le a_n$. The Choquet integral with respect to μ is defined by

$$(\mathrm{c})\int f d\mu=\sum_{i=1}^{n}(a_i-a_{i-1})\cdot\mu\left(\left\{x\middle|f(x)\ge a_i\right\}\right),$$

where $a_0=0$, and the Choquet integral is denoted by $(\mathrm{c})\int f(x)d\mu(x)$ or $(\mathrm{c})\int f d\mu$.

4. Detection Framework Based on Multi-classifiers Ensemble

The essence of malicious code detection is the problem of classification, which distinguishes the malicious code from the normal code. Out of the weakness of detection

based on a certain feature, we chose features that complementary with each other to detect malicious code. We assume that there is a classifier for malicious code detection based on each feature. Then the problem of malicious code detection could be induced to the problem of multi-classifiers ensemble.

4.1. Overview of the Detection Framework

The overview of the detection framework was presented in Figure 1. The code sample was statically analyzed to extract corresponding features in different levels. The features used to characterize the code including file structure, disassembly instruction, flow graph and system call. Then Base classifiers are utilized to distinguish the malicious code and the benign code based on each single feature. The output of each base classifier are a probability that the code sample belong to the malicious code. The multi-classifiers ensemble synthesize the outputs of each base classifier to obtain the final decision result.

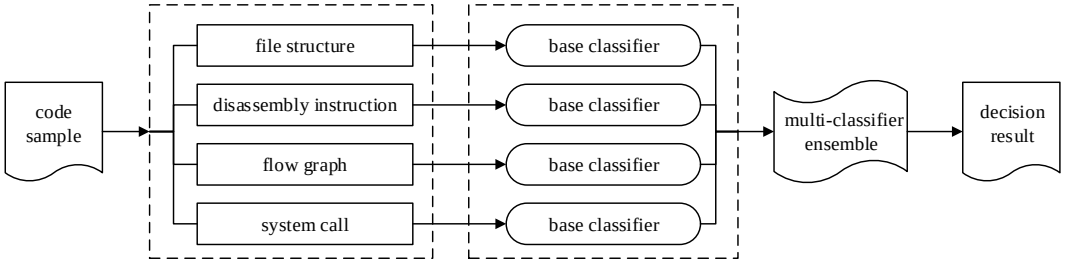

Figure 1. Framework Overview

4.2. Architecture of Static Analysis

The multi-classifiers ensemble depends on the analysis of the code sample. We resorted to static analysis to extract the features of malicious code. Figure 2 is the architecture of static analysis, which is the foundation of the detection framework.

With regard to a code sample, we analyzed its file structure first. File structure is an important data structure of malicious code. Lots of malicious code attached to normal code will change file structure to obtaining the chance to execute before normal code. For normal code, the execution will start from the section of text where the normal code located. Whereas the malicious code attached to the normal code has to extend the section of text or adds a new section. Besides finding a place to lay aside the malicious code, it is necessary to change the entrypoint in the field of file header to point to the malicious code. There are many other characteristic to describe the abnormal in file structure. We utilized a series of flags to mark the abnormality in file structure.

Second, the section of text will be disassembled. The malware behavior analysis was composed of analysis in disassembly level and system-call level. Out of the finer grain, the analysis in disassembly level is indispensable to characterize the behavior of code. However, the effect of existing technologies in disassembly level is not good against the behavior composed of discontinuous instruction sequence. Taking into account the above situation, combined with characteristics of binary code we proposed a method of behavior analysis in disassembly level based on pattern matching with wildcards and gap-length constrains, called CFG-refinedSAIL. The algorithm comparesd the opcode sequences in each basicblock of control flow with the opcode sequences of malware behavior in a refined way to find out the malware behavior in disassembly level.

Intermediate representation is a kind of abstraction of disassembly instruction. Owing the prevailing of obfuscation technology, such as instruction insert, instruction permutation and instruction institution etc., will change the appearance of disassembly instruction. Whereas the intermediate representation will not change substantially. We took the control flow graph as the object of analysis. We followed the method proposed

by Silvio Cesare [18]. We used a distance metric based on the distance between feature vectors. The feature vector was a decomposition of the set of graphs into *q*-gram strings of the high-level source after decompilation.

System call is an important feature to characterize the behavior of the code. The system call extracted by static analysis were constructed as a system call graph. Because the problem of subgraph isomorphic is NP-complete. [19] And with the observation that the system calls were executed in sequence. So we transformed the problem of sub-graph isomorphic to sequence matching. We modified the algorithm of Depth First Searching with backtracking to location the system call pattern.

Based on the feature of different forms, we constructed 4 feature databases to store corresponding malicious feature. The extracted features were compared to the feature stored in corresponding database. Then each base classifier were used to distinguish the malicious code and the normal code separately. We selected the naïve bayes as the base classifier.

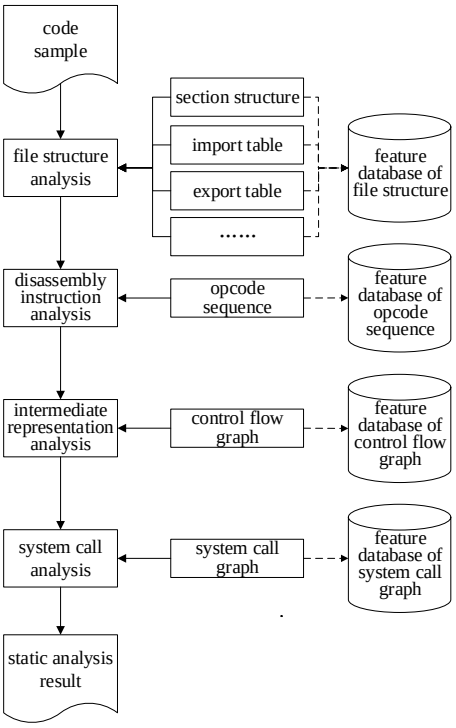

Figure 2. Procedure of Static Analysis

4.3. Fuzzy Measure Calculation based on Genetic Algorithm

The calculation of fuzzy measure is the key point in the application of fuzzy integral. The algorithm of neutral network, quadratic programming and genetic algorithm all could be used to determine the fuzzy measure. The genetic algorithm is a stochastic global search method that mimics the metaphor of natural biological evolution. In the detection framework proposed in this paper, we selected the genetic algorithm to search the optimal solution of fuzzy measure μ. The genetic algorithm is composed of 4 elements.

Element 1: Chromosome encoding and decoding.

In the genetic algorithm, each problem solver is encoded as a chromosome. A position, or a set of positions in a chromosome is encoded as a gene. In the detection framework of malicious code based on multi-classifiers ensemble, the fuzzy integral is corresponding to a chromosome. And the value of the set function at each set in fuzzy measure is corresponding to a gene. We utilized binary encoding to transform a gene into a string of

bits. Let n be the size of feature set, so each chromosome is composed of 2^n-2 genes which represents a nonnegative monotone set function. The other 2 sets are the empty set and the universal set. Out of there are 4 types of features we used, there are 14 genes.

The length of bit string representing each chromosome relies on the precision of fuzzy measure. In this paper, we set the precision of the precision of fuzzy measure to be 0.001. The range of fuzzy measure was restricted to [0, 1]. According to the binary decoding method of genetic algorithm we needed 10 bits to present each gene at least. As there are 14 genes needed to be represented, the length of each chromosome should be 140 bits.

Element 2: Fitness function.

The Fitness function is required in the genetic algorithm to assign a score of fitness to each chromosome in the population. The fitness values calculated by fitness function are then used in the process of natural selection to choose the potential solutions which will continue on to the next generation. In this paper the fitness function of chromosomes was calculated by

$$\frac{1}{1+e} \text{ for simplicity.}$$

Element 3: Genetic operators.

The genetic algorithm constructs a new generation chromosome from an old one following three steps: selection, crossover and mutation.

(1) Selection. Chromosomes are selected from the population to be parents to crossover. The fitter the chromosome, the more times it is likely to be selected to reproduce. We selected rws (roulette wheel selection) as the selection operator to produce new chromosomes as offspring.

(2) Crossover. The crossover operators recombine pairs of individuals with crossover probability to produce offspring. We utilized the one-point crossover as the crossover operator. If one crossover point is selected, string of bits from beginning of chromosome to the crossover point is copied from one parent, the rest is copied from the second parent.

(3) Mutation. This operator randomly flips some of the bits in a chromosome. Given that mutation is generally applied uniformly to an entire population of bit string, it is possible that a given binary string may be mutated at more than one point.

Element 4: Parameters for genetic algorithm

There are some parameters need to be defined in the calculation, including: population size, number of generations, crossover probability and mutation probability.

Population size indicates the number of chromosomes in population (in one generation). If the population size is not large enough, the genetic algorithm has a few possibilities to perform crossover and only a small part of search space is explored. On the other hand, if the population size is too large, the algorithm slows down.

Number of generations indicates when the genetic algorithm stops. As the number of generation increases, the individuals in the population get closer together.

Crossover probability indicates how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome.

Mutation probability indicates how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover without any change. If mutation is performed, part of chromosome is changed.

5. Experiments and Results

In this paper, we selected the operating system Windows 7 (SP1) 32 bit, the processor is Core i3-3.30GHz Intel, the memory is 4.00GB. The sample in the experiment were collected from online.

To evaluate the performance of single classifier and ensemble multi-classifier, we defined 4 concept described below:

FP (false positive): the normal code was classified as malicious.
FN (false negative): the malicious code was classified as normal.
TP (true positive): the malicious code was classified as malicious.
TN (true negative): the normal code was classified as normal.
We chose three indexes to evaluate the performance of the model:

$$FNR = \frac{FN}{TP + FN},$$

$$FPR = \frac{FP}{TN + FP},$$

$$Accuracy = \frac{TP + TN}{TP + FN + FN + FP}.$$

The genetic algorithm was utilized to calculate the fuzzy measure. The selected operators and parameters were listed in Table 1.

Table 1. Operators and Parameters

operator	selection	roulette wheel selection
	crossover	single-point crossover
	mutation	bit flip
parameter	population size	50
	crossover probability	0.7
	mutation probability	0.05
	number of generations	50

The result of fuzzy measure calculated by genetic algorithm was showed in Table 2.

Table 2. Fuzzy Measure

Set	μ	Set	μ
∅	0	$\{f_2,f_3\}$	0.234
$\{f_1\}$	0.224	$\{f_2,f_4\}$	0.654
$\{f_2\}$	0.149	$\{f_3,f_4\}$	0.537
$\{f_3\}$	0.727	$\{f_1,f_2,f_3\}$	0.373
$\{f_4\}$	0.425	$\{f_1,f_2,f_4\}$	0.849
$\{f_1,f_2\}$	0.275	$\{f_1,f_3,f_4\}$	0.913

$\{f_1,f_3\}$	0.317	$\{f_2,f_3,f_4\}$	0.787
$\{f_1,f_4\}$	0.769	$\{f_1, f_2,f_3,f_4\}$	1

The single classifier of each feature was based on naïve bayes. The multi-classifiers ensemble was based on the Choquet fuzzy integral. The comparison was shown in Table 3.

Table 3. Test Result

	FPR	*FNR*	*Accuracy*
Classifier of files structure	8.3%	14.6%	89.1%
Classifier of opcode	6.6%	7.5%	90.4%
Classifier of control flow	12.0%	13.5%	87.3%
Classifier of system call	6.7%	7.3%	90.4%
Ensembled Classifier	1.7%	4.9%	97.1%

The experiment results showed that the accuracy of base classifier of file structure was relatively weak. The false negative ratio of file structure was relatively higher than other. The reason is that there are many detection frameworks resort to file structure to detect malware. Whereas some malicious code don't change the file structure in order not to trigger the alarm of the detection framework.

The accuracy of base classifier of control flow graph was not good as well. The false positive ratio and false negative ratio were higher than others. The reason was ascribe to the insufficiency of control flow graph database.

The accuracy of base classifier of opcode sequence and system call graph were the more effective way to detect malware compared with the other two featurs.

The multi-classifiers ensemble based on Choquet integral achieved best accuracy. Because it utilized the complementary features to detect malware.

6. Conclusion and Future Work

In this paper, we proposed a detection framework to determine the malicious code. The framework took advantage of multi-classifiers ensemble to improve the accuracy of detection. The different features of the code were evaluated by base classifiers first. Then the Choquet fuzzy integral was used to fuse the result of base classifiers. The tests showed that the multi-classifiers ensemble could raise the accuracy effectively.

In order to improve the performance of the detection framework discussed in this paper, there are several issues still need to be researched.

(1) The improvement of base classifiers. In this paper we adopted naïve bayes algorithm as the base classifier to classify a certain feature to malicious or benign. Whereas there are many other algorithms could be used as the base classifier, including neural network, support vector machine, decision tree *etc*.. Whether the algorithms mentioned above could get better performance as base classifiers need to be investigated.

(2) The selection of algorithm for multi-classifiers ensemble. We selected the Choquet integral to fuse the result of base classifiers, and genetic algorithm to compute the fuzzy measure. Whether there are some other algorithm could be used to ensemble the multi-classifiers with better performance need to be researched.

Acknowledgments

The work presented in this paper was sponsored by the National Natural Science Foundation of China (No.61472447) and the Key Science&Technology Project of Henan Province of China (No.SP10JH13042).

References

[1] P. Szor, "The Art of Computer Virus Research and Defense", Addison-Wesley Professional, **(2005)**.
[2] Jin Q., Duan J., Vasudevan S. and Bailey M., "Packer classifier based on PE header information", Symposium & Bootcamp on the Science of Security, ACM, **(2015)**.
[3] Saini A., Gandotra E., Bansal D. and Sofat S., "Classification of PE Files using Static Analysis", In Proceedings of the 7th International Conference on Security of Information and Networks, ACM, **(2014)**.
[4] Siddiqui M., Wang M. C. and Lee J., "Data mining methods for malware detection using instruction sequences", Proceedings of Artificial Intelligence and Applications, AIA, **(2008)**.
[5] Moskovitch R., Feher C., Tzachar N., Moskovitch R., Feher C., Tzachar N., Berger E., Gitelman M. and Dolev S., "Unknown Malcode Detection Using OPCODE Representation", Proceedings of the 1st European Conference on Intelligence and Security Informatics, Springer-Verlag, **(2008)**, pp. 204-215.
[6] Santos I., Brezo F., Nieves J., Penya Y. K., Sanz B. and Laorden C., "Idea: Opcode-sequence-based malware detection", Engineering Secure Software and Systems, Springer Berlin Heidelberg, **(2010)**, pp. 35-43.
[7] Alam S., Horspool R. N., Traore I. and Sogukpinar I., "A framework for metamorphic malware analysis and real-time detection", Computers & Security, vol. 48, **(2015)**, pp. 212–233.
[8] Gao D., Reiter M. K. and Song D., "Binhunt: automatically finding semantic differences in binary programs", Lecture Notes in Computer Science, 5308, **(2008)**, pp. 238-255.
[9] Varghese S. M. and Jacob K. P., "Process Profiling Using Frequencies of System Calls", Proceedings of the The Second International Conference on Availability, Reliability and Security, IEEE Computer Society, **(2007)**, pp. 473-479.
[10] Zhang B., Yin J., Tang W. and Hao J., "Unknown Malicious Codes Detection Based on Rough Set Theory and Support Vector Machine", Neural Networks, 2006. IJCNN '06, International Joint Conference on IEEE, **(2006)**, pp. 2583-2587.
[11] Y. Chuanhuan, T. Shengfeng and M. Shaomin, "High-order Markov kernels for intrusion detection", Neurocomputing, vol. 71, no. 16-18, **(2008)**, pp. 3247–3252.
[12] Dietterich T. G., "Ensemble learning. The handbook of brain theory and neural networks", vol. 2, **(2002)**, pp. 110-125.
[13] Dietterich T. G., "Ensemble methods in machine learning", In Multiple classifier systems, Springer Berlin Heidelberg, **(2000)**, pp. 1-15.
[14] Sugeno M., "Theory of Fuzzy Integrals and its Applications", Ph.D. dissertation, Tokyo Institute of Technology, **(1974)**.
[15] Wang Z., Klir G. J., "Fuzzy Measure Theory", Springer US, **(1992)**.
[16] Grabisch M., "k -order additive discrete fuzzy measures and their representation", Fuzzy Sets and Systems, vol. 92, no. 2, **(1997)**, pp. 167-189.
[17] Grabisch M., "The representation of importance and interaction of features by fuzzy measures", Pattern Recognition Letters, vol. 17, no. 6, **(1996)**, pp. 567-575.
[18] Cesare S. and Xiang Y., "Classification of malware using structured control flow", Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing, Australian Computer Society, Inc., vol. 107, **(2010)**, pp. 61-70.
[19] Bachl S., "Isomorphic Subgraphs", Proceedings of the 7th International Symposium on Graph Drawing, Springer-Verlag, **(1999)**, pp. 286-296.

