Semantic Set Analysis for Malware Detection

Nguyen Van Nhuong, Vo Yen Nhi, Nguyen Tan Cam, Mai Xuan Phu, Cao Dang Tan

Semantic set analysis for malware detection

Nguyen Van Nhuong[1], Vo Thi Yen Nhi[1], Nguyen Tan Cam[2], Mai Xuan Phu[1] and Cao Dang Tan[1]

[1]University of Science Ho Chi Minh City, Vietnam National University – HCMC, Vietnam
nvnhcmus@gmail.com, vtynhi2001@gmail.com,
mxphu@fit.hcmus.edu.vn, tan@hcmus.edu.com
[2] University of Information Technology, Vietnam National University - HCMC, Vietnam
camnguyentan@gmail.com

Abstract. Nowadays, malware is growing rapidly through the last few years and becomes more and more sophisticated as well as dangerous. A striking malware is obfuscation malware that is very difficult to detect. This kind of malware can create new variants that are similar to original malware feature but different about code. In order to deal with such types of malware, many approaches have been proposed, however, some of these approaches are ineffective due to their limited detection range, huge overheads or manual stages. Malware detection based on signature, for example, cannot overcome the obfuscation techniques of malware. Likewise, the behavior-based methods have the natural problems of a monitoring system such as recovery costs and long-lasting detection time. In this paper, we propose a new method (semantic set method) to detect metamorphic malware effectively by using semantic set (a set of changed values of registers or variables allocated in memory when a program is executed). For more details, this semantic set is analyzed by n-gram separator and Naïve Bayes classifier to increase detection accuracy and reduce detection time. This system has been already experimented on different datasets and got the accuracy up to 98% and detection rate almost 100%.

Keywords: data mining algorithm for classification, x86 instruction set, obfuscation techniques, malware detection, semantic set.

1 Introduction

Today, malware is deployed up faster and faster and has a variety of spreading types. According to the McAfee 2012 statistics [1], "*malware is going unabated, with no sign of slowing down*", new malware samples in 2012 grew 50% annually over 2011. McAfee catalogs covered 100,000 new malware samples every day in 2012, which means 69 new pieces of malware per minute [1]. Moreover, the obfuscation techniques of malware are being developed more diversely and more complicatedly [2]. New malware samples not only use one obfuscation technique but also combine more techniques, such as metamorphic malware [2], which could make the difficulty to malware detection. Many malware detection methods are widely used nowadays, such

as: using signature [4], using data mining for classification [4] or using behavior features [5]. Even though signature-based detection can easily classify malwares, the detection's scope is limited around the well-known malware. Data mining-based detection can simply recognize malware with the high accuracy but this system spends more time on both the training process and the detecting process. Moreover, malware detection based on data mining is in static detection type so it cannot categorize metamorphic malwares. Behavior-based detection is effective to defeat the obfuscation techniques; however, its extracting subsystem is too complicated to deploy. In addition, this method belongs to dynamic detection type so it itself has the natural problems of a monitoring system [3]. For instance, the silent malware is used VMM (Virtual Machine Monitor) detection techniques: IDT check, LDT check, MCW check or Virtual PC Special Instruction [3] to detect its executing environment that is whether virtual or not and then automatically changes to the appropriate behavior.

There are many studies of malware detection, but each of them has its advantages and disadvantages. A new detection method by using semantic set as a feature is proposed to recognize malware. Semantic set is a set of values changed on registers and variable allocated memories when program is executing. With the original hypothesis which explains that "*Two programs are similar to each other if and only if their two semantic sets are also similar to*", therefore, semantic set is an effective behavior for identifying malware. In order to get semantic set, either a monitoring system or a smart tracer tool must be used. In this paper, authors developed an automatic tracer to overcome the limitations of the monitoring system. In the proposed system, semantic set is combined with the n-gram separator and Naïve Bayes classifier to become a more precise malware detection system, which also can detect metamorphic malware. Moreover, this system will give a new perspective on malware detection method: by using the dynamic behavior interpreted by malware's code as an input for the Naïve Bayes classifier [4]. This system will inherit their strong characteristics to be a stronger one.

This paper is organized into sections: Section 2 introduces the related studies in malware detection. Section 3 gives more detail technology about semantic set and its application in malware detection. Section 4 initiates the proposed system and its experimental results. Section 5 gives the conclusions and then introduces some future works.

2 Related works

From time to time, the malware obfuscation techniques become more and more complicated. Fig. 1 shows the milestone of malware's camouflages [1].

Fig. 1. The milestone of malware's camouflage.

There are three general methods in malware detection: static method, dynamic method and hybrid method.

Sachin Jain and *et al.* [4] used n-hex byte as a signature to detect malware. Instead of using a common string matching algorithm, they used data mining for classifying malware. Classwise Document Frequency [4] is proposed to filter the standard n-hex bytes. It not only reduces the space of n-hex bytes but also supplies a good input for the classifier. However, this method requires more time in the training and in the detecting process, especially on big-sized sample files or big datasets. The selected n-hex bytes can help the classifier to detect malware in the ranges: well-known malwares or simple transformable malwares. With the encrypted malware or advanced transformable malware (polymorphic malware or metamorphic malware), this method is not useful.

Without using the static detection method, Mahboobe Ghiasi and *et al.* [5] developed the DyVSoR system which recognizes malware by the set of registers' values. DyVSoR uses a VMware to monitor the values of some registers as EAX, EBX, ECX and EDX when the program called API/function calls. After a monitoring time, the set of register's values is considered as an input of matching-based behavior. By experimenting, this system has the accuracy up to 96%, higher than Kaspersky [5] at that time. However, DyVSoR can be hardly used widely, because the threshold of monitoring time for a sample file is still an open problem. If the monitoring time is too short, malware cannot perform its action; in contrast, if the monitoring time is too long, it could be ineffective. Besides, the restore time after the monitoring will make the system slower. With the smart malwares, they can detect executing environment by VMM detection techniques [3] and easily change to the proper behavior. They is the death weakness (transparent problem) of any monitoring system.

The dynamic method is able to detect the obfuscation techniques of malware, which static method is unable, but the dynamic method could fail in the silent malware. To resolve that problem, Nguyen Anh M. and *et al.* [3] propose the MAVMM (lightweight VMM) system which can detect silent malware. Authors modify the core of VMM and remove unnecessary modules to perform only one objective: to monitor malware behaviors [3]. This system steps over the problems of monitoring systems and reaches some positive results. For example, the monitoring system status is quickly restored after monitoring a sample file, the monitoring time is fixed and the system is transparent to malwares [3]. Even though there are improvements, it is difficult to deploy widely in reality. Besides that, MAVMM's system needs more supports from hardware such as the virtual technology. In addition, there are some troubles when the system removes unnecessary hardware and software modules to make it more lightweight, but some malwares need network card, SD storage or user information such as email, system information, etc., to run.

Table 1 shows the advantages and disadvantages of these analysis methods in malware detection:

TABLE I. COMPARISON OF MALWARE DETECTION SYSTEM.

Method	*Type*	*Advantage*	*Disadvantage*
3-gram hex bytes and Naïve Bayes [4]	Static detection	Quickly recognize familiar malware.	- Spend more cost on training or detecting process. - Difficult to detect advanced transformable malware.
DyVSoR [5]	Dynamic detection	Detect malware with high accuracy, including transformable malware.	- Problem of monitoring system. - Cannot overcome MM detection techniques.
MAVMM [3]	Advanced dynamic detection	Overcome the natural problem of monitoring system.	- Need supports from hardware and difficult to deploy in reality.

Table 1 proves that the feature based dynamic method can bypass almost all of obfuscation techniques and the static method has higher advantage in detection's time. Therefore, the malware detection method should use those techniques together to improve the accuracy and remove problems of the static and dynamic method. Based on the above ideas, authors developed a malware detection system that uses semantic set as a dynamic feature, and segments this semantic set into set of 3-gram values [4] as an input for Naive Bayes algorithm to quickly identify sample files.

3 Our approach

Before introducing to the proposed system, some basic terms are explained:

- Program has a collection of variables which always change their values when is executed. Those variables are loaded into registers or variable allocated memories to execute, then the change of their values are similar to the change of values on registers and variable allocated memories.
- Set of values changed on any register or variable allocated memory is called semantic string. The set of semantic strings is called a semantic set. The semantic set contains all changed values in a program instance. In this way, the semantic set is appreciated as the dynamic feature of the program.
- Semantic set is considered as a form of dynamic feature which can overcome the weaknesses of the traditional methods, such as hex code signature [4] or API/function call signature [6], [7]. As the result, semantic set is used as an n-gram input of Naïve Bayes classification algorithm [6]. So that, the similarity between two semantic sets are decided by Naïve Bayes classifier.

In order to become the input for this classifier, each semantic string is separated into the 3-gram [6] values (a short string contains 3 values) by 3-gram separator [6]. A set of 3-grams is often very large, therefore, it should be reduced by Classwise Document Frequency [6] to only get the 3-gram values, namely:

- The 3-gram values only appear in semantic set of malware or the 3-gram values do not appear in the semantic set of malware.
- The 3-gram values appear in semantic set of malware but their probability of frequencies is very high or very low.

After experiments, 2000 3-gram values is enough to be used as a feature for Naïve Bayes algorithm. Actually, Naïve Bayes algorithm (1) uses all 3-gram value probabilities to identify a sample file:

$$\mathrm{P}(X|C_i) = \mathrm{argmax}\mathrm{P}(C_i).\prod_{k=1}^{n}\mathrm{P}(x_k|C_i) \quad (1)$$

Where $X = \{x_1, x_2, \ldots, x_n\}$ and $P(X|C_i)$ is the probability of sample X in class C_i and $P(C_i)$ is the probability of class C_i (class C contains all attributes which needs classifying). Due to the property of multiplication, the total probability will be zero or very near zero if some 3-gram values' probability is zero or very near zero. It will cause the Naïve Bayes classifier become ineffective. In order to deal with this problem, Maximum Likelihood technique is used to increase the accuracy of semantic set system.

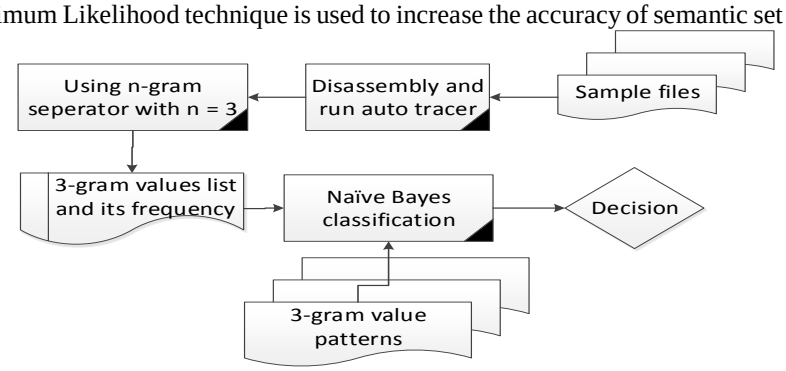

Fig. 2. Diagram of malware detection based on semantic set.

The modified Naïve Bayes algorithm classifies a sample file by giving two values: the MALWARE SCORE (the similarity probability between sample file and malware file) and the BENIGN SCORE (the similarity probability between sample file and benign file). The decision function is shown in the following pseudo code:

```
if ( MALWARE SCORE > BENIGN SCORE)
        File is MALWARE.
else if ( BENIGN SCORE >  MALWARE SCORE and
BENIGN SCORE – MALWARE SCORE <= WARNING THRESHOLD)
        File is WARNING.
else File is BENIGN.
```

Because semantic set is automatically extracted by a tracer tool, so it can overcome the obfuscation techniques of malware. In the following examples, we will give the examples and sequentially analyze the non-effect obfuscation of malware to the change context of semantic set.

Assuming that, there is the original assembly language code:

(1) e6	XOR ax, 0	;ax = 0
(2) 2069 6e	XOR bx, 0	;bx = 0
(3) 206d 6f	ADD ax, 100	;ax = 100
(4) 2e 0d	SUB bx, 10	;bx = −10
(5) 64 65 2e	ADD cx, ax, bx	;cx = ax + bx = 90

The original semantic set is:

$$S_{origin} = [ax = \{0,100\}\ bx = \{0,-10\}\ cx = \{90\}]$$

In turn, the equivalent transform of the semantic set is shown through the following obfuscation techniques (in bold lines):

1−Garbage code insertion.

(1) e6	XOR ax, 0	;ax = 0
(2) 90	**NOP**	
(3) 45	**INC ax**	;ax = ax + 1 = 1
(4) 50	**PUSH ax**	;Push ax into stack.
(5) 58	**POP ax**	;Pop ax into stack.
(6) 4f	**DEC ax**	;ax = ax−1 = 1−1 = 0
(7) e6	XOR bx, 0	;bx = 0
(8) 90	**NOP**	
(9) 206d 6f	ADD ax, 100	;ax = 100
(10) 2e 0d	SUB bx, 10	;bx = −10
(11) 03c2	ADD cx, ax, bx	;cx = ax + bx = 90
(12) 83e1 03	**AND dx, 0**	;dx = 0
(13) 03c2	**ADD dx,dx,0**	;dx = dx + 0 = 0

Semantic set with garbage code insertion is:
$S_{garbage} = [ax = \{0,1,0,100\}\ bx = \{0,-10\}\ cx = \{90\}\ dx = \{0,0\}]$. As you see, S_{origin} and $S_{garbage}$ are absolutely similar.

2−Garbage function or unused function insertion.

Garbage function or unused function insertion actually uses garbage code insertion but with a larger scale. Thanks to the above definition of semantic set, this obfuscation technique is still not affected to the original semantic set.

(1) e6	XOR ax, 0	;ax = 0
(2) e6	XOR bx, 0	;bx = 0
(3) 83e1 03	**AND Reg, 0**	;Reg = 0
(4) 83e1 03	**ADD Reg, Reg, 100**	;Reg = 100
(5) Loop:		
(6) f9	**BEQ Reg, 0, (9)**	;if Reg = 0, exit loop.
(7) 2e 0d	**SUB Reg, Reg, 1**	;Reg = Reg −1
(8) GOTO Loop		;continue loop.
(9) 83e1 03	ADD ax, 100	;ax = 100
(10) 2e 0d	SUB bx, 10	;bx = −10
(11) 83e1 03	**ADD cx, ax, bx**	;cx = ax + bx = 90
(12) 83e1 03	**ADD Reg, Reg, 100**	;Reg = 100
(13) e6	**XOR dx, 0**	;dx = 0

```
(14) Sum:
(15) f9          BEQ Reg, 0, (19)        ;if Reg = 0, exit loop.
(16) 83e1 03     ADD dx, dx, 0           ;dx = dx + 0 = dx = 0
(17) 83e1 03     ADD cx, cx, 1           ;cx = cx + 1
(18) GOTO Sum                            ;exit loop.
(19) 2e 0d       SUB cx, cx, 100         ;cx = cx −100 = 90
(20) Exit                                ;end
```

Semantic set with the garbage/unused function insertion is:
$S_{fgarbage}$ = [ax = {**0,100**} bx = {**0,−10**} Reg = {0, 100,99,98,97,...,0,100,99,98,97,…,0} cx = {90, 91, 92, 93, 94, 95,…,190, **90**} dx = {0,0,0,….,0}]

The $S_{fgarbage}$ becomes much larger than S_{origin} but the "shadows" of S_{origin} (bold instruction) still appears in $S_{fgarbage}$. So S_{origin} and $S_{fgarbage}$ are also similar.

3− JUMP insertion and register renaming.

```
(1) e6           XOR eax, 0              ;eax = 0
(2) eb 05        JMP (3)
(3) 2069 6e      XOR ebx, 0              ;ebx = 0
(4) eb 05        JMP (5)
(5) 206d 6f      ADD eax, 100            ;eax = 100
(6) 2e 0d        SUB ebx, 10             ;ebx = −10
(7) eb 05        JMP (10)
(8) 64 65 2e     ADD ecx, eax, ebx       ;ecx = eax + ebx = 90
(9) 83e1 03      AND edx, 0              ;edx = 0
(10) eb 05       JMP (11)
(11) Exit:                               ;end.
```

Semantic set with jump insertion is:
S_{jmp} = [eax = {0,100} ebx = {0, −10} ecx = {90} edx = {0}]. Therefore, S_{jump} and S_{origin} are also similar.

4− Equivalent code replacement:

```
(1) AND ax, 0                            ;ax = 0
(2) AND bx, 0                            ;bx = 0
(3) ADD ax, 1                            ;ax = 100
(4) MUL ax, ax, 100                      ;ax = ax*100 = 1*100 = 100
(5) ADD bx, −10                          ;bx = bx −10 = 0−10 = −10
(6) ADD cx, ax, bx                       ;cx = ax + bx = 100 – 10 = 90
```

Hence, the semantic set is: S_{equi} = [ax = {0, 100} bx = {0, -10} cx = {90}]
S_{equi} is similar to S_{origin}. Consequently, the equivalent code replacement technique is still not affected to the semantic set. This is a special feature, which traditional method cannot perform.

5− Combine many obfuscation techniques.

```
(1) XOR dx, 0                            ;dx = 0
(2) NOP
(3) JUMP (6)
```

(4) ADD dx, dx, 100 ;dx = 100
(5) **JUMP (12)**
(6) AND ex, 0 ;ex = 0
(7) **INC ex** ;ex = ex + 1
(8) **PUSH ex** ;Push ex into stack.
(9) **POP ex** ;Get ex from stack.
(10) **SUB ex, ex, 1** ;ex = ex −1 = 0
(11) **JUMP (5)**
(12) ADD cx, dx, ex ;cx = dx + ex = 100 – 10 = 90
(13) GOTO Exit
(14) ADD ex, −10 ;ex = −10
(15) Exit
(16) **JUMP (14)**

The related semantic set is: $S_{complex}$ = [dx = {0, 100} ex = {0, 1, 0, -10} cx = {90}]
This semantic set will increase the number of its strings and the number of values in each semantic string but the similarity between S_{orign} and $S_{complex}$ is still not changed. In other words, the transformable code techniques are unlikely to be affected to the similarity between two programs if their internal processing is equivalent.

6− Code encryption.

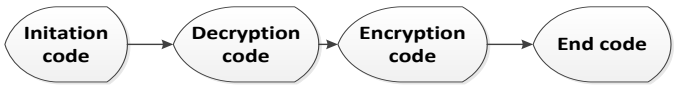

Fig. 3. Diagram of malware encryption.

The malware belonging to this group has 4 basic parts as in Fig. 2. When the program is executed, the decryption part will be executed firstly to decrypt the encryption code. After that, the final program will become a normal program. This technique is used by malware to evade detection and loopholes only at runtime. Therefore, the semantic set of this program will increase some semantic strings of decryption part but the semantic set similarity between original program and encrypted program remains unchanged.

7− Malware used another technique.

Some malware opens itself only during its execution. It implements functions and prepares necessary data to execute itself. A sigmoid function is prepared to activate malware's behavior and this behavior is not shown explicitly in the executable code.
One simple example:

(1) **MOV [0x12345]**, locate_0x101; *preparing code.*
(2) **MOV [0x22345]**, locate_0x102
(3) **PUSH [0x1233]**, locate_0x103
(4) **POP [0x1000]**
(5) **JMP [0x1000]**; *executable code*
(6) **JMP [0x12345]**
(7) **JMP [0x22345]**
(8) **JMP [0x1233]**

In the above code, the commands (1)→(4) prepare data (started address of the build-in functions) for commands (5)→(8) to be executed by jump commands. The semantic set includes two parts: runtime semantic strings of malware and semantic

string of the preparing code. This ensures to detect successfully this malware, otherwise method based on signature such as API/function call or hex bytes cannot do. In spite of using the same feature as DyVSoR system of Mahboobe Ghiasi [5], the proposed method has some new characteristics:

- Instead of using a VMware system to monitor the changed values of the registers after each API/function call [5], the proposed system identifies malware based on the semantic set. The modified Pyew tool [8] is used for disassembly PE (Portable executable) file. Then, all changed values on registers and variables allocated memory will be extracted automatically into semantic set by our tracer tool. In addition, DyVSor uses VMware thus increases processing time for each sample file, but the proposed system limits it.
- DyVSor only traces the values on basic registers, such as: EAX, EBX, ECX, and EDX [5] when the executed program calls API/function. This API/function call is limited by the set of API collected such as network API system, file system API [5]. Hence DyVSoR is not as general as malware detection system based on semantic set. For that reason, those kinds of malware using the external library with different API/function calls as OpenGL API, OpenCV API or encrypted API, etc., will not be detected correctly by DyVSoR system.
- Besides that, DyVSoR uses string matching algorithm [5] while the semantic set system uses Naïve Bayes algorithm to classify sample files. With the long register's value string, the matching algorithm will not work as effectively as Naïve Bayes algorithm, because Naïve Bayes only uses the features extracted from the training process instead of using all semantic string.
- In addition, the semantic set system uses the tracer (running as an interpreter) to extract the semantic set. Thus, it can control the executing of a program and easily prevent the threat action of malware without recovery time like the DyVSoR system. Moreover, the tracer is independent so it can be integrated into other systems.

In summary, the proposed system used semantic set as an input for Naïve Bayes classifier. Therefore, this system can inherit the advantages of the semantic set and Naïve Bayes filter. The system has already been experimented on common datasets and got the high accuracy.

4 Implement and experimental result

4.1 Implementation

In order to compare semantic set method with equivalent malware detection methods, the datasets which have the same ratio in number of files and type of files are used:

TABLE II. EXPERIMENTAL DATASET

Dataset number	*Num. of files*	*Type of file*
Dataset 1 [7]	85 files	10 Benign, 21 Virus, 34 Worm, 20 Trojan.
Dataset 2 [5]	155 files	20Backdoor, 20 P2Pworm, 20 Trojan, 20 Worm, 20 Virus and 55 Begin.

Dataset 3 [4]	107 files	51 Malware, 56 Benign
Dataset 4 [6]	79 files	18 Benign, 61 Malware

Besides, those datasets are chosen because of two other reasons:

- Firstly, the datasets include all common very dangerous and many variant malware's types such as: worm, backdoor, trojan, virus and benign files belonging to Windows32 system files.
- Secondly, the number of samples in the datasets for the semantic set system and these methods such as API Graph [7], DyVSoR [5], n-hex byte Naïve Bayes [4] and Structure and Behavior features [6] are similar in ratio. A list of selected malwares is shown in Table III.

TABLE III. CLASIFICATION OF SAMPLE'S TYPE IN EACH DATASET.

No.	*Name of sample file*	*Classification*
1	P2P-Worm-Win32.Agent	Worm
2	Backdoor.ASP.Ace	Backdoor
3	Backdoor.PHP.Agent	Backdoor
4	Trojan.Win32.KillFiles	Trojan
5	Virus.Win32.Delf	Virus
6	Virus.Win32.Seppuku	Virus
7	Worm.Win32.Downloader	Worm
8	Worm.Win32.Viking	Worm
9	System32 folder	Benign

4.2 Experimental result

TABLE IV. THE EVALUATION PARAMETERS TABLE.

No.	***Parameter***	***Meaning***
1	TP	The amount of malware files are recognized as malware.
2	FP	The amount of benign files are recognized as malware.
3	TN	The amount of benign files are recognized as benign.
4	FN	The amount of malware files are recognized as benign.
5	Accuracy	(TN+TP)/(TN+TP+FN+FP)
6	Detection rate	TP/ (TP + FP)

To evaluate the accuracy of the SSSM system, the above parameters are used (Table IV). After that, the datasets mentioned in Table III has been tested, the accuracy of malware detection system is shown in Table V:

TABLE V. DETECTION RESULT OF MALWARE DETECTION SYSTEM BASED ON SEMANTIC SET.

Dataset	*Evaluation*	*Accuracy*	*Detection rate*
1	TN = 9, FN = 1, TP = 75, FP = 0	98.82%	100%
2	TN = 49, FN = 6, TP = 100, FP = 0	96.12%	100%
3	TN = 50, FN = 6, TP = 51, FP = 0	94.39%	100%
4	TN = 17, FN =1, TP = 61, FP = 0	98.73%	100%

The result proves that our malware detection system has ability to detect most of malware samples in datasets. The detection rate reaches 100% and accuracy is up to approximately 98.82%.Therefore, this new method should be paid more attention to by anti-malware researchers. These precision charts reveal the detection result of our method that compares with API/function Graph, DyVSoR system, n-hex byte Naïve Bayes and Structure and Behavior feature system.

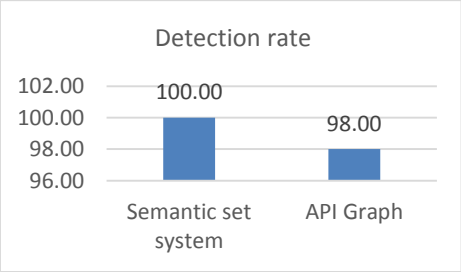

Fig. 4. The detection rate of semantic set system and API Graph system.

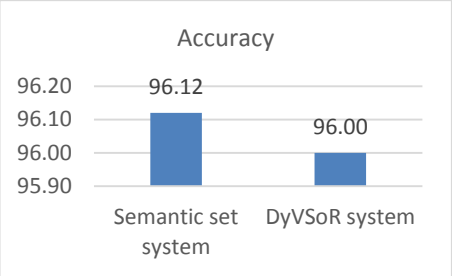

Fig. 5. The accuracy of semantic set system and DyVSor system.

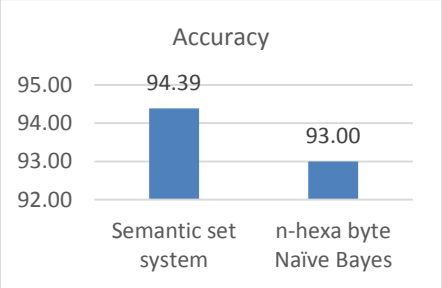

Fig. 6. The accuracy of semantic set system and n-hex byte Naïve Bayes.

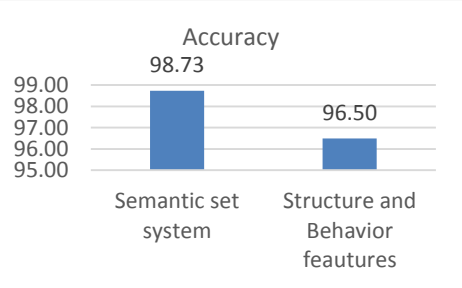

Fig. 7. The accuracy of semantic set system and Structure and Behavior features system.

In addition, the system is able to detect some kinds of malware that are difficult to others. For example, Backdoor.ASP.Agent and Backdoor.PHP.Ace are two malwares that cannot extract API/function call list. As a result, API Graph method [7] cannot be used for them, but semantic set system is able. Moreover, the system can detect the complicated malware as: Worm.Win32.Downloader which has many variations.

5 Conclusion

Today, malwares are deployed faster and faster and their metamorphosis is more and more complicated. Nevertheless, the current detection methods together with some advantages are still limited, so an effective method to classify malware and to be able to overcome the limitations of the current methods becomes more demanded. We propose a new method for malware detection that combines semantic set as a dynamic feature with Naïve Bayes classifier to increase detection accuracy. In addition, the simulating and extracting the malware behavior by our tracer tool help the detection

system reduce the detection time and get rid of the effect of malware behavior to system. Further, our system was experimented on different datasets and gained highly accuracy, which provides a new aspect for anti-malware researchers.

In order to improve the accuracy and realize the semantic set system, some future works are listed:

1. Integrate semantic set with other methods to create a new malware detection that can detect malware on a large scale and high precision.
2. Improve system's performance by combining between parallel CPU and GPU programming in order to build a real-time system for malware detection.
3. Develop tracer tool to be a debugger that extracts values on registers and variable allocated memories more and more exactly and therefore increases the reliability and exactness of malware detection system.

References

1. Infographic: The State of Malware 2013, McAfee Security, http://www.mcafee.com/us/security-awareness/articles/state-of-malware-2013.aspx
2. Rad, Babak Bashari, Maslin Masrom, and Suhaimi Ibrahim.: Camouflage in Malware: from Encryption to Metamorphism. In: International Journal of Computer Science & Network Security (2012).
3. Anh M. Nguyen, Nabil Schear, HeeDong Jung, Apeksha Godiyal, Samuel T. King, and Hai D. Nguyen.: MAVMM: Lightweight and purpose built VMM for malware analysis. In: Computer Security Applications Conference (2009)
4. Jain Sachin and Yogesh Kumar Meena.: Byte Level n–Gram Analysis for Malware Detection. In: pp. 51-59, Computer Networks and Intelligent Computing. Springer Berlin, Heidelberg (2011)
5. Mahboobe Ghiasi, Ashkan Sami, and Zahra Salehi.: DyVSoR: Dynamic Malware Detection Based on Extracting Patterns fromValue Sets of Registers. In: The ISC International Journal of Information Security (2013).
6. Manoun Alazab, Robert Layton, Sitalakshmi Venkataraman, and Paul Watters.: Malware detection based on structural and behavioural features of API calls. In: The Proceedings of the 1st International Cyber Resilience Conference (2010)
7. Elhadi, Ammar Ahmed E., Mohd Aizaini Maarof, and Ahmed Hamza Osman.: Malware Detection Based on Hybrid Signature Behavior Application Programming Interface Call Graph. In: American Journal of Applied Sciences (2012)
8. Pyew Python tool, https://code.google.com/p/pyew/
9. Virus heavens Snapshot, https://archive.org/details/vxheavens-2010-05-18