An Immunity-inspired Relocation Method for Unknown Malware Detection

Yu Zhang
College of Information Science and Technology
Hainan Normal University
Haikou, China
bullzhangyu@126.com

Lihua Wu
College of Information Science and Technology
Hainan Normal University
Haikou, China

Ziqiang Luo
College of Information Science and Technology
Hainan Normal University
Haikou, China

Abstract—**Malware is a type of software designed to gain unauthorized access to a computer system. Most malwares have a relocation module that gets the base address of the codes in execution environment to infect other programs, neither do most legitimate programs. Due to this unique characteristic, the malware relocation module can be extracted as an antibody in the immune systems to detect the specific antigens. We present a malware detection method inspired by biology immune system and the structure of malware relocation code. The experiment is conducted and the result shows that this approach not only has relatively higher detection rate of unknown malware than other methods, but also has better capability of self-adaptive and self-learning.**

Keywords—Computer Immune System; Malware Relocation; Unknown Malware detection

I. INTRODUCTION

Malware or malicious software is software designed to disrupt computer operation, gather sensitive information, or gain unauthorized access to a computer system [1, 2]. Malware is a general term used by computer professionals to mean a variety of forms of hostile, intrusive, or annoying software or code. Malware includes computer viruses, worms, trojan horses, spyware, adware, most rootkits, and other malicious programs [3]. In this paper, we will focus on computer viruses. A computer virus is a malware program that, when executed, replicates by inserting copies of itself into other computer programs, data files, or the boot sector of the hard drive. Computer viruses currently cause billions of dollars worth of economic damage each year, due to causing systems failure, wasting computer resources, corrupting data, increasing maintenance costs, etc.

The security researchers have developed a variety of detection techniques to protect computer systems against malwares attack. Those techniques can be categorized broadly into three categories: anomaly-based detection, specification-based detection and signature-based detection [4, 5, 6, 7]. Anomaly-based detection techniques use the knowledge of what constitutes normal behavior to decide the maliciousness of a program under inspection. Specification-based techniques leverage some specification of what is valid behavior to decide the maliciousness of a program under inspection. Signature-based detection uses its characterization of what is known to be malicious to decide the maliciousness of a program under inspection [8, 9, 10]. The fundamental limitation of anomaly-based detection is its high false-positive rate and the time complexity in the training phase. The main limitation of specification-based detection is that it is difficult to specify completely and accurately the entire set of valid behaviors. One of the major drawbacks of signature-based detection is that it cannot detect zero-day attack, for which there is no corresponding signature stored in the database [11].

To improve the performance of currently malware detection, we proposed an immunity-inspired relocation method for malware detection. We have found that the malwares cannot. Malware relocation, which is uncommon in legitimate programs, is vital to the execution of computer viruses allowing them to obtain the base address of the codes, and it cannot correctly execute without relocation [12]. So the malware relocation codes are extracted as antibodies in the immune systems to detect the specific antigens. Experiment was conducted and results show that this method has better efficiency in the detection of known and previously unknown malwares than the others.

In the following sections, we first describe the definition of the relocation gene in section 2. Then we introduce the antigen presenting and the generation of the antibody in section 3. Section 4 shows the implementation and experiment results. We state our conclusion and future work in Section 5.

II. DEFINITION OF THE RELOCATION MODULE

Relocation, which is uncommon in legitimate programs, is important to the execution of malicious codes allowing them to obtain the base address of the codes thus creating computer epidemics and maximizing the effectiveness of the attack. The Relocation module can be regarded as a specific sequence of commands that cause the malware to relocate itself and thus to execute correctly in host programs. The typical self-relocation in the malicious executable codes is shown in Figure 1. Relocation can be accomplished in various ways relying on malware writers' techniques. Since the most sophisticated and versatile viruses are still implemented in assembly language

This work was supported by the National Natural Science Foundation of China under Grant No. 61262077, 61462025; the Natural Science Foundation of Hainan under Grant No. 613161; the National Students' Innovation and Entrepreneurship Training Program under Grant No. 201211658036.

and assembled into executable files, the relocation gene could only be possibly obtained in search of low-level machine codes.

```
Offset              Opcode          Instruction
.text:00401B57      E800000000      call  $+5
.text:00401B5C      loc_401B5C:
.text:00401B5C      5B              pop  ebx
.text:00401B5D      81EB5C1B4000    sub  ebx, offset loc_401B5C
```

Fig. 1. The malware relocation module

The RM (Relocation Module) is usually contained within the code sequence of malware and it could also be dispersed throughout that sequence to become a variety of RM mutations in order to anti-disassemble. Since none of the machine command alone can be considered malicious, only the particular sequences of commands can construct the RM. Therefore, the RM is described using the concept of building blocks, where each block performs a part of the self-relocation. This concept is illustrated in Figure 2. The upper blocks of the self-relocation gene base on the lower blocks. Most of the building blocks involved in malicious self-relocation activity can individually be included in any program for legitimate reasons. Only when integrated into larger structures based on their inter-functional relationships, these building blocks can be regarded as attempts to self-relocate.

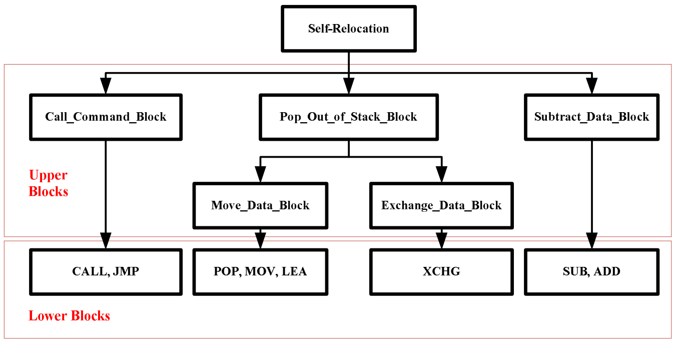

Fig. 2. The RM Structure

The RM consists of such blocks in various ways. Therefore, its structure can be viewed as a regular sentence being built up by concatenating phrases, where phrases are built up by concatenating words, and words are built up by concatenating characters.

Applying such a syntactic approach to describing the RM is to facilitate the recognition of sub-patterns. This suggests the recognition of smaller building blocks first, establishing their relevance and contribution to the relocation, and then considering the next sub-pattern. This process is similar to text analysis, which includes recognizing characters first, then concatenating them into words, and next continue concatenating words into phrases and sentences checking for correct grammar and punctuation. The syntactic description of the RM provides a capability for describing and detecting large sets of complex patterns by using small subsets of simple pattern primitives. It is also possible to apply such a description any number of times to express the basic structures of lots of gene mutations.

Following the concept of syntactic description, the RM structure could be represented using the grammar definition notations [12]:

$$G = \{V_N, V_T, P, S\} \tag{1}$$

where

G is the self-relocation gene,

V_N is the non-terminal variable,

V_T is the terminal variable,

P is the finite set of rules,

S is the starting point of the gene.

Suppose that the RM is represented by the structure above (Figure 1.), the non-terminal variable V_N in the expression above can be defined as:

$$V_N = \left\{ \begin{array}{l} \text{<Self\_Relocation\_Gene>, <Call\_Command\_Block>,} \\ \text{<Pop\_Out\_of\_Stack\_Block>, <Move\_Data\_Block>,} \\ \text{<Exchange\_Data\_Block>, <Subtract\_Data\_Block>} \end{array} \right\} \tag{2}$$

The terminal variable V_T represents the RM sequence:

$$V_T=\{\text{CALL,JMP,POP,MOV, XCHG, SUB, ADD}\} \tag{3}$$

The complete vocabulary $V(G)$ is composed of V_N and V_T, that is, $V(G)= V_N \cup V_T$ and $V_N \cap V_T = \varnothing$.

P, the set of rules, is expressed as $\alpha \rightarrow \beta$, where α and β interconnections in V .

$S \in V_N$ represents the starting point in V_N, which is equal to the < Relocation_Gene >.

A. *Maintaining the Integrity of the Specifications*

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. The Immunity-Inspired RM Detection Method

The malware relocation gene pool through the antigen-presenting is defined as follows.

$$V=\{v|v \in \bigcup_{i=8}^{32} H^i \wedge |v|=i \wedge v=Ap(x \in \mathit{Nonself})\} \tag{4}$$

where

$Ap\ (x)$ is the antigen-presenting function;

v is the virus gene extracted from the virus relocation module, whose length is between eight and thirty-two hexadecimal codes.

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

A. The generation of the antibody

The acquired immune system that can protect against the specific viruses is generally acquired through vaccination to generate specific antibodies. The antibodies in this study are generated from the extraction of the vaccines in the malware relocation gene pool, which are the malware genes obtained by antigen-presenting. Then, the detectors are generated from the antibodies to detect the malwares. For example, we will generate the detector E8000000005B from the malware gene pool. The detector set is defined as follows.

$$D = \{< d, affinity >| \, d \in \bigcup_{i=8}^{32} H^{i}, affinity \in N\} \tag{5}$$

where

d is the antibody;

affinity is the match between the antigen and the antibody.

B. The detection of unknown malware

After the detectors were generated, they can effectively detect unknown malwares. During the detection, the antigen whose affinity with the antibody is larger than the threshold value will be regarded as a virus. The dynamic evolutional equations of the antibodies and the detection of unknown malwares are as follows.

$$SA(t) = \begin{cases} \varnothing, t = 0 \\ SA(t-1) + SA_{new}(t), t \geq 1 \end{cases} \tag{6}$$

$$SA_{new}(t) = \{v \mid v \in D, \forall y \in Self, < v, y > \notin Match \wedge v.affinity \geq \beta\} \tag{7}$$

$$Match = \{< v, y >| \, v \in D, y \in AG, f_{match}(v.d, AP(y)) = 1\} \tag{8}$$

$$f_{match}(v, y) = \begin{cases} 1, f_{affinity}(v,y)/L_v \geq \alpha \\ 0, otherwise \end{cases} \tag{9}$$

$$f_{\mathrm{affinity}}(\mathrm{v,y}) = \max(\mathrm{x}_1, \mathrm{x}_2, \ldots, x_{|L_v - Ly|+1}) \tag{10}$$

$$\mathrm{x}_{\mathrm{i}} = \sum_{i=1}^{\min(L_v, L_y)} \theta_{ij} \tag{11}$$

$$\theta_{ij} = \begin{cases} 1, v_i = y_{i+j-1}, 1 \leq i \leq |L_v - L_y| + 1, 1 \leq j \leq L_v \\ 0, otherwise \end{cases} \tag{12}$$

where

SA is the specific antibody set;

SA_{new} is the new generated antibody set, whose affinity with self is greater than the threshold value β;

f_{match} is the matching function between antibody and antigen;

$f_{affinity}$ is the affinity function between antibody and antigen;

Match is the set consisting of the antigens and the antibodies matched by the antigens;

θ_{ij} is the affinity value; if there is a matching between an antibody and an antigen, the value is 1, otherwise 0;

Lv is the vaccine length;

Ly is the antigen length.

IV. EXPERIMENTS AND RESULT

Since there is no benchmark data set available for the malware detection unlike intrusion detection, our data sets include 100 viruses collected from the website VX Heavens [13] and 500 benign executables taken from system32 folder in Windows. Some executables were obfuscated with compression, encryption, or both; some were not, but we were not informed which were and which were not. The main goal of the experiment is to test the detection rate of known and unknown viruses and false-positive rate of the normal files. The experimental results are shown in Figure 4. Figure 4 shows that the detection rate of the malicious codes is 98%, the omitting rate is 3%, and the false-positive rate (misidentification of legitimate programs as viruses) is only 3.6%. The good experimental result can own to the self-relocation based detector that detects the lower level machine codes combing with the self-relocation gene database to form upper level blocks, which makes it effectively detect malicious codes with lower false-positive rate.

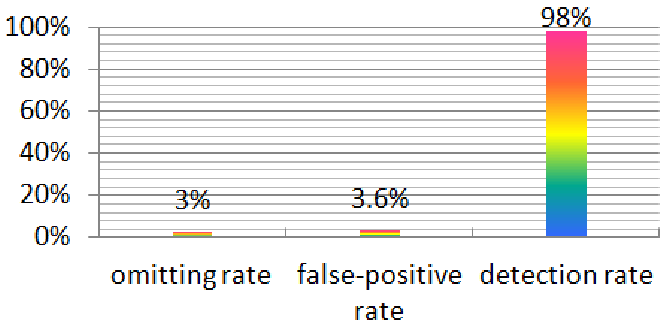

Fig. 4. The experiment results of the proposed method

In order to test the performance of the proposed approach, we conducted the related comparison experiments with the currently most mature antivirus product Kaspersky7.0, 360 security guard, Eset NOD32 and with APIs, which uses API sequence to detect malicious codes.

The comparison experiments results are shown in Figure 5 that the detection rate of our proposed approach is 97%, Kaspersky 88%, 360 security guard 90%, Eset NOD32 95% and APIs 65%. The results indicate that our proposed approach has higher detection rate than the others, and efficiently testify the validity of our proposed approach.

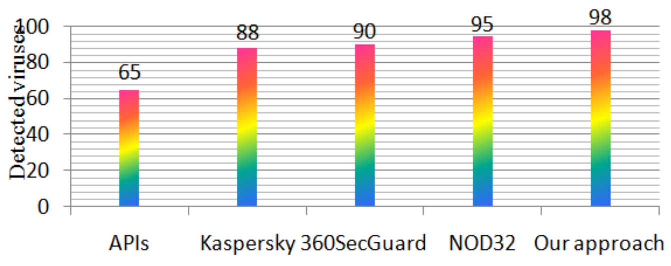

Fig. 5. The comparison experiments results

V. CONCLUSION

We proposed an immunity-inspired relocation method for unknown malware detection. We extract the relocation code from malware and regard it as a malware relocation gene in the computer immune system. One of the primary strengths of the proposed approach is its ability to detect previously unknown malwares with a very low false-positive rate. The experimental results show that the proposed method not only has a high detection rate, low false-positive rate and low omitting rate, but also its efficiency is better than other methods.

REFERENCES

[1] Richard Ford, Eugene H. Spafford, Happy birthday, dear viruses. Science, 2007, vol. 317: 210-211.

[2] Stephen Trilling, Carey Nachenberg. The future of malware. EICAR Proceedings 1999.

[3] Sandeep Kumar, Eugene H. Spafford. A generic virus scanner in C++. Proceedings of the 8th Computer Security Applications Conference, 1992, 210-219.

[4] Nwokedi Idika, Aditya P. Mathur. A Survey of Malware Detection Techniques. http://www.serc.net/report/tr286.pdf, February, 2007.

[5] Yu Zhang, Feng Xia. A Self-Relocation Based Method for Malware Detection. Applied Mechanics and Materials, 220-223, 2012.

[6] V. Skormin, D. Summerville, J. Moronski. Detecting Malicious Codes by the presence of their Gene of Self-Replication, Computer Network Security, Lecture Notes in Computer Science, 2003, vol. 2776.

[7] Douglas Summerville, Victor Skormin, Alexander Volynkin, et al. Prevention of Information Attacks by Run-Time Detection of Self-replication in Computer Codes. Lecture Notes in Computer Science, 2005, vol. 3685: 54 – 75.

[8] D. Ellis, J. Aiken, K. Attwood, and S. Tenaglia. A behavioral approach to worm detection. Proceedings of the 2004 ACM Workshop on Rapid Malcode, 2004, 43–53.

[9] Lansheng Han, Mengxiao Qian, Xingbo Xu, Cai Fu. Malicious code detection model based on behavior association, Tsinghua Science and Technology, 19(5):508-515, 2014.

[10] Asaf Shabtai, Robert Moskovitch, Yuval Elovici, Chanan Glezer. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey, Information Security Technical Report, 14(1):16–29, 2009.

[11] Konrad Rieck, Philipp Trinius, Carsten Willems, Thorsten Holz. Automatic analysis of malware behavior using machine learning. Journal of Computer Security,19(4): 639-668, 2011.

[12] Guojpeng/CVC.GB. The analysis of Win32 PE viruses. 2003, http://www.hynubbs.cn/netstar/news_view.asp?id=61.

[13] VX Heavens. http://vx.netlux.org