

# Artificial Immune System against Viral Attack

Hyungjoon Lee<sup>1</sup>, Wonil Kim<sup>2\*</sup>, and Manpyo Hong<sup>1</sup>

<sup>1</sup> Digital Vaccine Lab, Graduate School of Information and Communication  
Ajou University, Suwon, Republic of Korea  
{prime, mphone}@ajou.ac.kr

<sup>2</sup> College of Electronics and Information Engineering,  
Sejong University, Seoul, Republic of Korea  
wikim@sejong.ac.kr

**Abstract.** Since the first computer virus has been found, scanning detection has been used as a primary method in virus detection systems. As computer viruses and worms become more complex and sophisticated, the scanning detection method is no longer able to detect various forms of viruses and worms effectively. Many anti-virus researchers proposed various detection methods including artificial immune system to cope with these characteristics of viruses and worms. This paper discusses some principle of artificial immune system and proposes artificial immune based virus detection system that can detect unknown viruses.

## 1 Introduction

Since the computer virus first appeared in 1981, it has been evolved as computer environments such as operating system, network, and application program have changed. The frequent evolution of computer virus has forced anti-virus researchers continuous development of new detection method. However most anti-virus systems are still based on scanning detection using signature, since other mechanisms have false positive or detection speed problem in real situations. In this paper, we propose a new virus detection approach that employs the artificial immune system. Artificial immune system is based on human immune system. The human immune system recognizes, memorizes, and responds to virus adaptively. Anomaly detection and adaptability are important properties of artificial immune system. The proposed artificial immune based system applies this property and detects unknown computer viruses.

In the next section, Artificial Immune system will be discussed in detail. In Section 3, the proposed artificial immune based virus detection system will be described. Simulation results and the direction of future research will be discussed in Section 4 and Section 5 respectively.

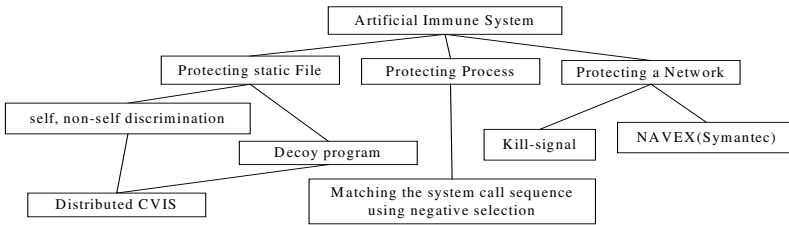
---

\* Author for correspondence +82-2-3408-3795

## 2 Artificial Immune System

Immunologists have traditionally described the human immune system as the problem of distinguishing "self" from dangerous "non-self" and eliminating dangerous non-self. Similarly, the computer security can be viewed as the problem of distinguishing benign program from malicious program such as computer virus or worm. Spafford described a computer virus from a biological point of view by analyzing several characteristic of virus as in artificial life [1]. Since then, many anti-virus researchers have researched on artificial immune system to protect a system from the intrusion including viruses and worms.

Artificial immune system should have anomaly detection capability to defend unknown intrusion. Adaptability is also a necessary property of artificial immune system to learn unknown intrusion and to response learned intrusions quickly. Other properties such as distributability, autonomy, diversity and disposability are required for the flexibility and stability of artificial immune system [14].



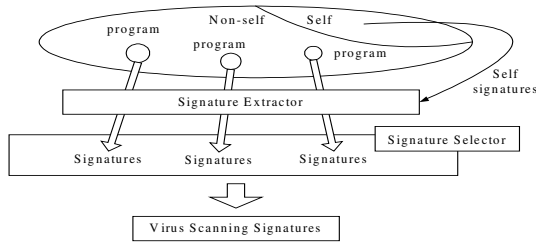
**Fig. 1.** Taxonomy of artificial immune system

Artificial immune system can be divided into three categories according to which computer component will be identified as a protected cell. Figure 1 shows this classification.

First, protecting static file category assumes a file including program code as a cell. Forrest proposed a method for change detection using negative selection algorithm [6]. Forrest's artificial immune system randomly generates check code called detector, and eliminates ones that can recognize self (benign program). This process is called negative selection algorithm. All programs or data matched with detectors are indicated as non-self (malicious code). Kephart proposed an artificial immune model that consists of monitoring, scanning and biological immunology [11]. In his model, known viruses are detected by scanner based on weak signature, and unknown viruses are collected and analyzed using decoy program. The decoy program is a dummy program that is not changeable. If decoy program is modified, it is virus infection. Distributed Computer Virus Immune System (CVIS) combined negative selection and decoy program. It uses decoy program to detect the unknown virus and employs negative selection to identify detected unknown virus [16].

Second, protecting process category assumes a process as a cell. Matching the system call sequence is proposed by Forrest [10]. The mechanism of this system is the same as self-nonself discrimination using negative selection. It is different from the first category that this mechanism matches detector with system call sequences, not with program codes.

Third, protecting a network category assumes a host as a cell. Kill-signal mechanism is proposed by Kephart, and a immune system called NAVEX that is proposed by Symantec [11,17]. Both immune systems propagate signatures of detected viruses to protect network connected hosts, as does biological immune system. In biological immune system, when an immune cell detects a virus, the cell spread chemical signal in order to duplicate the same immune cells.



**Fig. 2.** Structure of proposed virus detection system

### 3 Proposed Virus Detection System

Negative selection algorithm and decoy programs are important methods in artificial immune system. Negative selection algorithm is inspired by genetic random-process. But this mechanism has a weakness that come from the difference between human body and computer system. State of human body is stable, whereas that of computer system is not. Negative selection algorithm is suitable in stable system. Idea of decoy program comes from the distributability and diversity of immune system. In the case of decoy program, there is no guarantee that a virus will attack a decoy program. The proposed artificial immune based Virus Detection System (VDS) compromises this weakness, and is suitable for dynamically changing environment such as computer system.

VDS is a signature learning system to detect unknown viruses. VDS classify incoming or changed (patched) programs into legitimate programs and viruses. The process of VDS is consists of the following three steps. In the first step, VDS assumes that all existing programs are legitimate. In the next step, all incoming and changed programs are classified into suspicious programs. Finally, VDS selects virus programs from these suspicious programs using detection method based on virus behavior. Hereafter, we refer to legitimate program as self and suspicious program as non-self.

VDS consists of signature extractor and signature selector as in Figure 2. Signature extractor produces signatures of non-self. The main operation of signature extractor is selecting bit strings that are not matched with any self code at the same position. Therefore, signature extractor produces signature of non-self that never matched with any self. Produced non-self signatures are collected and analyzed by signature selector.

Signature selector compares the similarity of non-self signature with each others. Since viruses tend to infect other programs, if the same non-self signatures are appeared frequently, signature selector classifies those into signatures of viruses, and these signatures are used for virus scanner. Notice that these non-self signatures never

match with any self program. In the case of less frequent signatures, it is assumed as signature of self. We classify the program that contains any one of these self signatures as self. This process provides the adaptable capacity to the proposed VDS. The detailed process of each steps are discussed in the following 3 sections.

### 3.1 Signature Representation

The VDS generates signatures to recognize self and non-self program. The size of signature extraction region, from which we analyze for signature extraction, is always constant. Since most infected file executes virus code at first, most entry pointer of infected program indicates virus code. Therefore, we decide that starting point of signature extraction region is entry point of program. Signature consists of several pairs of offset and bit string. Number of pairs, offset and bit string are selected by signature extractor.

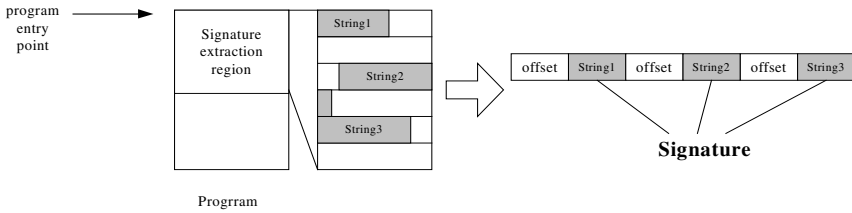


Fig. 3. Signature representation

### 3.2 Signature Extractor

Signature extractor produces non-self program signature that never matches with any self-program signature. This process consists of the following steps.

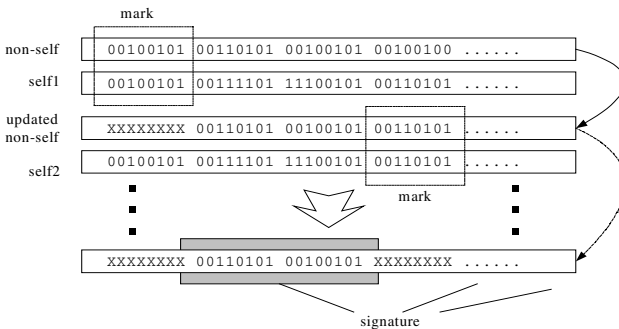


Fig. 4. Generating signature of non-self program

- Step 1: Divide a signature extraction region into several same sized comparison units.
- Step 2: Compare a signature extraction region of non-self program with each one of all self-programs.

- Step 3: If two comparison units on same position are same, mark the comparison unit in signature extraction region of non-self program. Continue this process on each signature extraction region of all self-programs.

In Figure 4, comparison unit is 1 byte. Non-self is compared with self1 and self2. When two comparisons are finished, first unit and fourth unit are marked. After all the comparisons with other self-programs are done, the signature of non-self consists of remained unmatched units. Since all of the comparison units that are the same with self's are marked as useless, the generated signature never matches with any self-programs.

### 3.3 Signature Selector

After we obtain signatures of non-self programs, we need to classify non-self program into virus or normal program. If virus infects other programs, the signature extractor generates signatures from same virus code because infected program is changed by one virus. Therefore, checking frequency of occurrence of the same signature is the same as checking the spread of the same virus code.

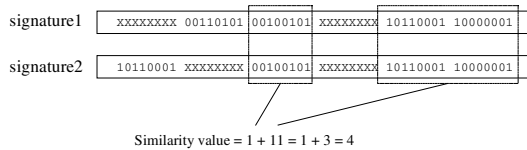


Fig. 5. Similarity between two signatures

Signature selector calculates the similarity values of non-self signatures, as shown in Figure 5. Comparison factors are bit sequence and offset of comparison unit in signature extraction region. If two factors of comparison units are equal, similarity function adds one to similarity value. When consecutive comparison units are equal, similarity value increases. For example, if two compare units are equal and adjacent, similarity value is 3 ( $11_2$ ). In other words, when  $n$  consecutive comparison units are equal, similarity value is  $2^n - 1$ .

Note that signatures of the same programs are more similar than signatures of different programs. In other words, similarity values between signatures of the same program codes are higher than the others. Therefore, VDS can distinguish signatures of the same program codes from signatures of other distinct program codes. The threshold values for classifying signatures of the same and different programs are determined by analyzing similarity values of the entire non-self programs. Determining the threshold values for classifying is discussed in Section 4.

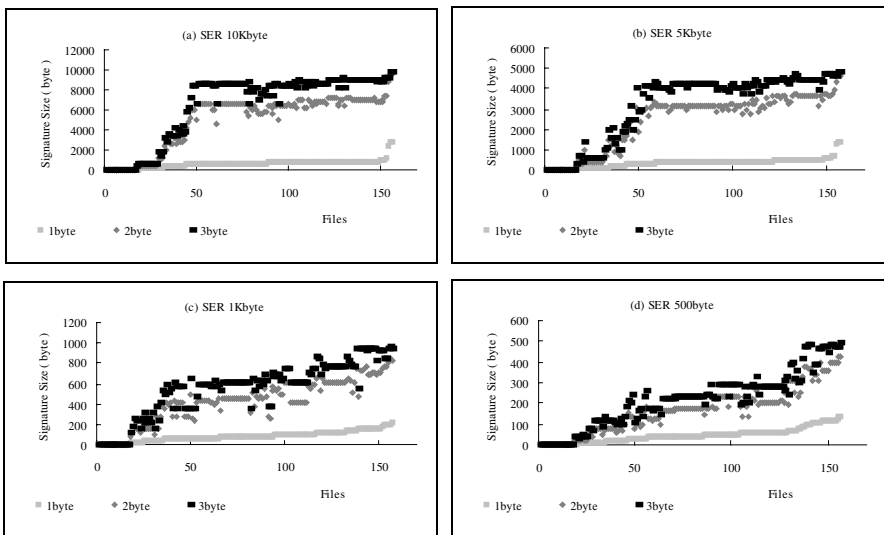
## 4 Simulation

VDS extracts signatures of non-self programs based on self-programs. Signature size and similarity value are important factors in VDS. As the number of self-programs that has the same comparison units at the same position is increased, the size of a

signature of the non-self program is decreased. In the worst case, VDS cannot generate the signature of particular program, because every comparison unit is marked as useless. The similarity value is used for classifying virus detector from suspicious signatures. In this section, we will discuss several simulation results from various signature extraction region size and comparison unit. The simulation is processed on several parameters as in Table. 1.

**Table 1.** Simulation parameters

| Parameters                       | Variables                                    |
|----------------------------------|--|
| The number of self-programs      | 1385 execution files                         |
| The number of non-self programs  | 160 execution files (3 virus infected files) |
| Signature extraction region size | 500byte, 1Kbyte, 5Kbyte, 10Kbyte             |
| Comparison unit size             | 1byte, 2byte, 3byte                          |

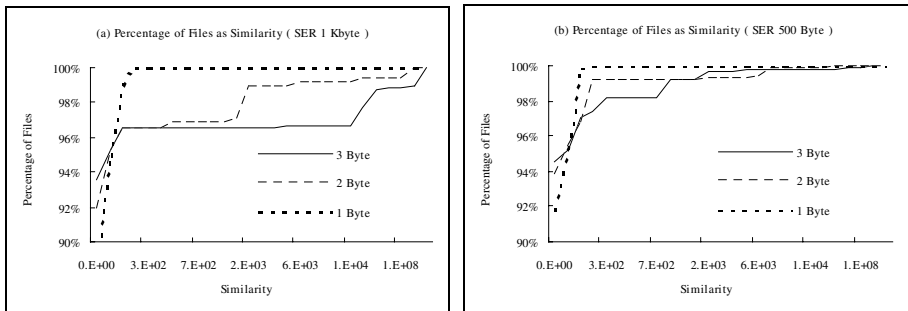


**Fig. 6.** Simulating using various sizes of SER: Graphs show signature size of different Signature Extraction Region (SER) 10Kbyte; 5Kbyte, 1Kbyte, and 500byte. Each graph has three parameters of comparison unit 1byte, 2byte, and 3byte.

The graphs in Figure 6 show the relation between signature size and two parameters; signature extraction region and comparison unit. The more signature extraction region and comparison unit increases, the more signature size increases. Larger signature includes richer information about related non-self program. But, when signature is used for scanning viruses, small signature is effective. Signatures that are larger than 1KB are not feasible for a virus-scanning signature. Moreover, the percentage of files that has zero signature is independent with size of signature extraction region and comparison unit. The number of files with zero signature size is almost 14 (8.75%) in all cases. We chose two effective sizes of signature extraction region; 1Kbyte and 500 byte. We simulated comparison for signature selection about the extracted signatures of non-self programs on these two parameters.

Similarity values between each signature of non-self programs are shown in Figure 7. When signature extraction region is 1Kbyte and comparison unit is 1byte, 88% of signatures of non-self programs have similarity value zero. When signature extraction region is 500byte and comparison unit is 1byte, signatures with zero similarity value are 92% in entire signatures of non-self programs. Since the percentage of actual virus infected file is 1.875%, ideal percentage of signatures that similarity value is zero should be 98.125%. Then, we can classify non-self signatures into virus signatures, whose similarity value is greater than zero. But, the percentage of non-zero signatures is 6% even though the extraction region size is 1Kbyte and comparison unit size is 3 byte. We need to determine threshold value to classify non-self programs into normal programs and viruses.

When SER is fixed, increasing the size of comparison unit results in rapid increase of its similarity value, since consecutive extracted signature is larger. When larger comparison unit is used, the gap of similarity value is larger. When signature extraction region is 1Kbyte and comparison unit is 3byte, we can find easily threshold of similarity value  $1.E+08$  to classify signatures of non-self programs into virus, because ideal percentage of signatures of normal programs is 98.125%. When threshold of similarity value is  $1.E+08$ , three signatures are selected by signature selector. Virus scanner using these signatures can detect virus-infected files.



**Fig. 7.** Percentage of files as similarity: In the case of SER 1Kbyte and 500byte, Percentage of files is showed. Each graph has three parameters of comparison unit 1byte, 2byte, and 3byte.

## 5 Conclusion

Artificial immune system is based on distinguish “self” and “non-self” like biological immune system. But previous artificial immune system is not feasible for dynamic computer system, especially for negative selection algorithm. In this paper, we proposed the artificial immune based Virus Detection System (VDS) that is feasible for dynamic computer system. VDS is a signature learning system to detect unknown virus. VDS produces signatures of non-self from suspicious program, and classify them into self-program and virus. Since virus tends to infect other programs, if similar non-self signatures are appeared frequently, VDS classifies those into signature of viruses for virus scanning. Therefore, VDS detect adaptively unknown viruses

In the simulation of VDS with 1Kbyte of the Signature Extraction Region (SER) and 3byte of the comparison unit, 94% of extracted signatures were completely

different. In other words, their similarity values are zero. The remaining 6% signatures including virus signatures had distinguished similarity values. Especially, the 2% virus signatures had relatively high similarity values. Therefore, proposed VDS can classify suspicious non-self programs into normal programs and viral programs. Using threshold of similarity value, VDS can select virus signatures. Virus scanner using these signatures can detect virus-infected files correctly.

## References

- [1] Eugene H. Spafford, Computer Viruses as Artificial Life, in *Artificial Life II*, Addison-Wesley (1992).
- [2] Richard Marko, Heuristics: Retrospective and Future, *Virus Bulletin Conference* (2002)
- [3] Taras Malivanchuk, The WIN32 worms: Classification and Possibility of Heuristic detection, *Virus Bulletin Conference* (2002)
- [4] Vesselin Bonchev, Macro and Script Virus Polymorphism, *Virus Bulletin Conference*, (2002)
- [5] Gabor Szappanos, Are There Any Polymorphic Macro Viruses at All?, *Virus Bulletin Conference*, (2002)
- [6] Stephanie Forrest, Self-Nonself Discrimination in a Computer, *IEEE Symposium on Research in Security and Privacy* (1994)
- [7] J. P. Anderson, Computer Security threat monitoring and surveillance, Technical report, (1980)
- [8] Dorothy E. Denning , An Intrusion-Detection Model, *IEEE* (1986)
- [9] David Wagner and Drew Dean, Intrusion Detection via Static Analysis, *IEEE* (2001)
- [10] S. Forrest, S. Hofmeyr, A. Somayaji, and T. Longstaff, A Sense of Self for Unix Processes, *IEEE* (1996)
- [11] Jeffrey O. Kephart, A Biologically Inspired Immune System for Computers, High Integrity Computing Laboratory IBM Thomas J. Watson. *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems* (1994)
- [12] Kim, J., Bentley, P., (1999), The Human Immune System and Network Intrusion Detection, submitted to EUFIT'99
- [13] Digging For Worms, Fishing For Answers, (2002)
- [14] Dipankar Dasgupta and Nii Attoh-Okine, Immunity-Based Systems : A survey, *IEEE International Conference* (1997)
- [15] Anil Somayaji, Steven Hofmeyr and Stephanie Forrest, Principles of a Computer Immune System, (1997)
- [16] Robert E. Marmelstein, David A. Van Veldhuizen and Gray B. Lamont, A Distributed Architecture for an Adaptive CVIS
- [17] The Digital Immune System, Symantec technical report
- [18] Stephanie Forrest, Anil Somayaji and David H. Ackley, Building Diverse Computer Systems, *IEEE* (1997)
- [19] Dipankar Dasgupta and Stephanie Forrest, Novelty Detection in Time Series Data Using Ideas from Immunology, *International Conference on Intelligent System* (1999)
- [20] Dipankar Dasgupta, Artificial Neural Networks and Artificial Immune System : Similarities and Differences, *IEEE* (1997)
- [21] Okamoto, T. and Ishida, Y., Multiagent Approach Computer virus: An Immunity-Based System, *AROB* (1999)