A Hierarchical Artificial Immune Model for Virus Detection

Wei Wang, Pengtao Zhang, Ying Tan, and Xingui He

***Abstract*— As viruses become more complex, existing anti-virus methods are inefficient to detect various forms of viruses, especially new variants and unknown viruses. Inspired by immune system, a hierarchical artificial immune system (AIS) model, which is based on matching in three layers, is proposed to detect a variety of forms of viruses. In the bottom layer, a non-stochastic but guided candidate virus gene library is generated by statistical information of viral key codes. Then a detecting virus gene library is upgraded from the candidate virus gene library using negative selection. In the middle layer, a novel storage method is used to keep a potential relevance between different signatures on the individual level, by which the mutual cooperative information of each instruction in a virus program can be collected. In the top layer, an overall matching process can reduce the information loss considerably. Experimental results indicate that the proposed model can recognize obfuscated viruses efficiently with an averaged recognition rate of 94%, including new variants of viruses and unknown viruses.**

I. Introduction

Since the first malicious executable code appeared in 1981, computer viruses have been evolving with the rapid development of computer environments such as operating system, network, etc. Virus techniques, for instance, junk code insertion and code transposition, have become obfuscated and traditional scanning detection is less effective and efficient to detect them [1]. Many researchers proposed various heuristic detection methods, including artificial immune system, to improve the effectiveness of virus detection [2].

The natural immune system is a dynamic, adaptive and distributed learning system. It protects organisms against antigen invasion by distinguishing foreign antigens (pathogens and tumor cells) from organisms' own healthy cells and tissues and eliminating foreign antigens. Similarly, the functionality of computer security systems is to recognize and eliminate virus, so that the natural immune system has provided with an inspiration to develop such kind of anti-viral systems [3].

In this paper, we analyzed the advantage and disadvantage of some approaches that are based on negative selection mechanism in the artificial immune system (AIS). Furthermore, we proposed an improved approach which is characterized of the generation of detectors under supervision. Different from previous approaches, the main contribution of this work is to collect correlation of instructions within a virus program. This AIS-based model uses training set as a guide to generate a candidate virus gene library. This candidate gene library is then upgraded to a detecting virus gene library by deleting all the non-viral information with negative selection in artificial immunity. This allows legal programs in the training set to be memorized. Because the detecting gene library stores virus samples at individual level and makes use of the relevance of different genes in a sample, it enables this model to compare genes on gene level, analyze suspicious program on individual level and ultimately to make the classification decision by detecting the entire gene library.

We firstly discuss the related work in Section 2 briefly, then propose an efficient virus detection AIS model in detail in Section 3, and finally compare simulation results and give our conclusion in Section 4 and 5, respectively.

Authors are with Key Laboratory of Machine Perception, Ministry of Eduction, Peking University, and with Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China. Prof. Y. Tan is corresponding author (Phone: +86-10-62767611, Email:ytan@pku.edu.cn).

II. Related Work

Based on ideas from immunology, Forrest *et al* [4] proposed the initial Negative Selection Algorithm (NSA). The algorithm can recognize self and non-self without reference to any particular information of the non-self set, especially suitable for computer fault diagnosis in an unknown time-varying environment like virus detection. But the model has a high computational cost. The number of detectors is exponentially related to the size of self set. D'haeseleer improved the NSA [5], make the number of detectors linearly related to the self set. But it still needs too many detectors, and they are also not guided generated detectors. Due to above reasons, it is a key problem how to generate effective detectors in a quick way.

Edge *et al* [7] developed a retrovirus-inspired algorithm for virus detection and optimization. The model used a random antibody initiation and the learning phase is further decomposed into two distinct parts that are trained for positive selection and negative selection, respectively. Positive selection gives antibody the ability to detect a virus while negative selection ensures that the antibody does not match self. The antibodies constantly evolve using affinity maturation to identify new viruses. They can die after a specified period of time, thus keeping the number of antibodies to a minimum in order to give a better performance. Although this work realized self-evolution, random antigen initiation would still need a long time for training. In addition, the model was only tested with virtual data and so it needs further validation.

Li has given the same idea of using the concept of affinity maturation in the immune system defined by matching between bit-strings in different files in his book [8]. The matching based on hexadecimal continuous matching rules is well directed to the characters of program files.

III. Proposed Approach

A. Model Architecture

The model is composed of two modules: virus gene library generating module and self-nonself classification module. The first module is used for the training phase, whose function is to generate a detecting gene library to accomplish the training of given data. The second module is assigned as the detecting phase in terms of the results from first module for detection of the suspicious programs.

In biology, it is well known that genetic information is mainly stored in DNA, but not all the fragments in DNA can express useful information. Only gene is a fragment of DNA with genetic information. Gene is made up of several deoxyribonucleotides (ODN).

In this paper, our definitions of some notations are given as follows.

- *DNA*: The whole bit-string of a procedure.
- *Gene*: Virus detector, a fragment of virus DNA, the compared unit for virus detection.
- *ODN*: Every two bytes of a bit-string.

The relation of DNA, gene and ODN is shown in Fig 1.

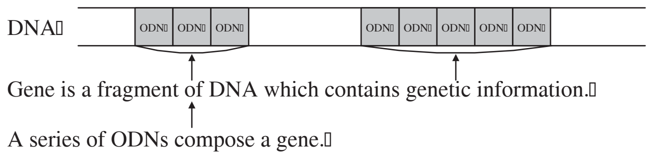
DNA
ODN
ODN
ODN
ODN
ODN
ODN
ODN
ODN
Gene is a fragment of DNA which contains genetic information.
A series of ODNs compose a gene.

Fig. 1. The relationship among DNA, gene & ODN.

The codes of a virus correspond to the DNA in the organism. The small quantity of key codes which perform viral functions are regarded as the genes of a virus. These virus genes are composed of several virus ODNs which are the smallest unit to analyze the virus. An ordered series of ODNs can express one or more program instructions. At this stage, the most important task of the model is to extract the genes of a virus.

B. Virus Gene Library Generating Module

Virus gene library generating module works on the training set consisted of legal and virus programs. The operating principle is shown in Fig 2.

Firstly, this module is to count the ODNs in a DNA of legal and virus programs by a sliding window, respectively, in order to extract ODNs which are regarded as the representative of the virus. A virus ODN library is built by the obtained statistical information. Secondly, the DNAs in virus and legal programs are traversed by the ODNs in the virus ODN library to generate virus candidate gene library and legal virus-like gene library. Finally, according to the negative selection mechanism, we match all the genes in the candidate virus gene library with the genes in the legal virus-like gene library, and delete those genes which appear in both libraries. In such a way, the candidate library is upgraded as the detecting virus gene library.

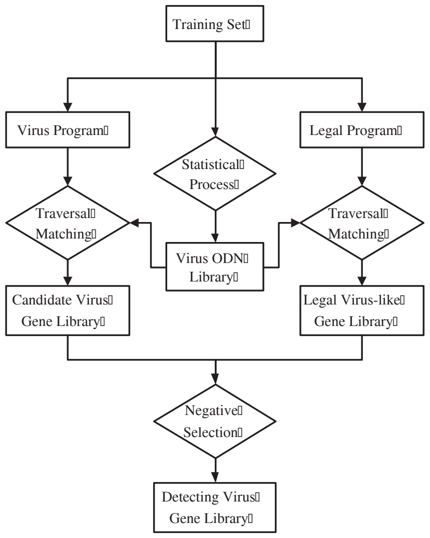

Fig. 2. Virus gene library generating process.

1) Virus ODN library: A sliding window is used to count ODNs in the DNAs to generate this ODN library.

- For the following DNA fragment
  CD21 C307 1FCD 218C C0B8
- Which includes 9 ODNs in total such that
  CD21 21C3 C307 071F 1FCD CD21 218C 8CC0 C0B8

The model can obtain the frequency information of ODNs appeared in the legal and virus programs. In the next step, the model can calculate the degree S^i of which each ODN tends to be more representative of the virus based on the frequency information. When S^i exceeds a chosen threshold, ODN i would be added into the virus ODN library. Here we set this threshold be S_1, called ODN selection threshold. Apparently S_1 is a constant related to the training set. When the training set is fixed, the value S_1 should be constant, but its choice would be variable on different training sets. To choose an appropriate S_1 to make ODNs less but more representative is very important.

2) Candidate virus gene library: The basic storage block in the virus candidate gene library is virus sample. All the genes in each sample are stored to make different genes in one virus storage and genes in different virus storage separately. This kind of storage mode is called signature storage on individual level in this paper. The gene library mentioned below would apply this storage mode to keep the relevance between different extracted genes in a same virus. Comparison between programs can be made on individual level with integrated information of virus signatures.

The model uses continuous matching to match the virus DNA with ODNs in the virus ODN library. It means, from the first matching position, that a sliding window is employed to move forward until a mismatching happens. Then the number, of which ODNs in the virus ODN library take part in the matching from the beginning to the end, is recorded. If this number is larger than a presenting threshold

TABLE I

R-CONTINUOUS MATCHING

Byte string	Binary string			
68C5	0110	1000	1100	0101
B633	1011	0110	0011	0011

T, the fragment of virus DNA is assigned as a virus gene. Otherwise, the fragment is considered not containing enough information to be a key code of virus or the genes of a virus. It is clear that this method has one ODN unit fault tolerance.

If T is too small, the generated gene does not contain enough information. Too many invalid redundant genes would cause the loss of system performance and effectiveness. If T is too big, some important information would be lost as the matching length is too long. Let T = 3, the minimum gene is 4 bytes long and normally one computer instruction is 1 or 2 bytes, at the moment the gene may contain 1 to 4 instructions which might be regarded as abundant information.

3) Detecting virus gene library: Using the same method for generating the candidate virus gene library, this model can also be used to generate a legal virus-like gene library by matching the legal programs with ODNs in the virus ODN library.

Taking the legal virus-like genes as self, and the candidate virus genes as nonself, the NSA is applied to generate the detecting virus gene library. In the candidate virus library, all the genes that match with the gene in the legal virus-like gene library are deleted. In this way, the gene in the detecting virus gene library does not match with any legal virus-like gene library. Therefore these detectors would be able to distinguish all the legal programs in the training set.

R-continuous matching rules (as shown in Table I) is a popular matching rule in AIS, where binary string coding is used. It is a fuzzy matching method, allowing some faults in matching. We introduce a new T-successive consistency matching (as shown in Fig 3. Gene 1 is a virus gene; Gene 2 is a legal virus-like gene.) for the byte strings of the same mechanism. The rule of T-successive consistency matching is that if there are no less than T successive same ODNs in the two genes and these ODNs belong to the virus ODN library above, the two genes are regarded as a successful matching.

When the candidate virus gene library is generated, the threshold T is set to 3, which means that only 3 or more ODNs connections are required to have enough information to form a gene. So only when two genes contain 3 or more ODNs successive consistency matching, can they be considered as having strong similarity, and regarded as a successful matching.

C. Self-Nonself Classification Module

Repeating the method that generates candidate virus gene library, the ODNs in the detecting virus gene library are used to generate the suspicious virus-like gene library. Then we match virus-like genes in the suspicious program with

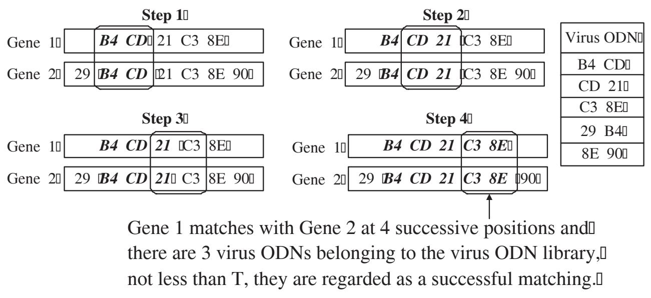
Step 1
Gene 1 B4 CD 21 C3 8E
Gene 2 29 B4 CD 21 C3 8E 90
Step 2
Gene 1 B4 CD 21 C3 8E
Gene 2 29 B4 CD 21 C3 8E 90
Step 3
Gene 1 B4 CD 21 C3 8E
Gene 2 29 B4 CD 21 C3 8E 90
Step 4
Gene 1 B4 CD 21 C3 8E
Gene 2 29 B4 CD 21 C3 8E 90
Virus ODN
B4 CD
CD 21
C3 8E
29 B4
8E 90
Gene 1 matches with Gene 2 at 4 successive positions and there are 3 virus ODNs belonging to the virus ODN library, not less than T, they are regarded as a successful matching.

Fig. 3. T-successive consistency matching.

TABLE II

SIMILARITY VALUE WITH MATCHING LENGTH

Matching length i	1	2	3	...	n
Similarity value	1	3	5	...	$2n-1$

detectors in the detecting virus gene library to get a matching value. If it is larger than a chosen threshold, the program is regarded as a virus; otherwise it is a legal program. (as shown in Fig 4).

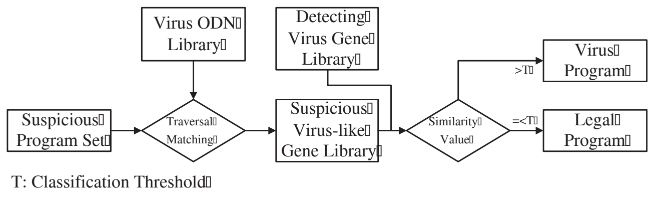

Fig. 4. Self-nonself classification process.

1) Matching degree between two genes: This module still use T-successive consistency matching for two genes' matching. A similarity value R is defined to measure the matching degree between two genes. If two genes are mismatched, the value is set to 0; If two genes are matched successfully, $R_1 = R_2 + R_3$, where $R_i \geq T (i = 1, 2, \cdots)$. We consider that the similarity value R_1 of units matching should be larger than the sum of value of R_2 units matching and R_3 units matching. Suppose that x_i $(i = 1, 2, \cdots)$ is the similarity value of a matching of i units, the following inequalities hold:

$$x_n > x_{n-i} + x_i; \qquad (1)$$

The relation between the similarity value and matching length is shown in Table II.

2) Suspicious program detection: If the suspicious program matches with each virus sample in the detecting virus gene library, the similarity value is calculated. All the values for this program are added together as the similarity value between the program and detecting virus gene library. The pseudocode is shown as follows.

```
initial similarity[M]=0;
initial similarity_indi=0;
for(i=0;i<M;i++)
```

```
        for(j=0;j<N;j++)
            temp=get_match_value();
            if(similarity[i]<temp)
            similarity[i]=temp;
        end
        similarity_indi+=similarity[i];
    end
```

In conclusion, the proposed hierarchical model works through three layers cooperatively when the model detects an incoming suspicious program. In the gene layer, T-successive consistency matching is used to make a fuzzy matching for a good fault toleration. In the individual layer, virus and legal program are compared on the individual level. The interrelated information of instructions is lost as little as possible, hence the model takes the full advantage of the potential relevance between different extracted signatures and recognizes obfuscated viruses effectively and efficiently. Due to the similarity between viruses, it also can detect new variants of known viruses and recognize unknown viruses accurately. Finally classification decision is not a single but an overall behavior in the decision layer which can give a more precise result.

IV. Simulation Results

A. Data Set

We performed experiments on a virus data set "cilpku08" which can be get at the web site *http://www.cil.pku.edu.cn/malware*. We collected 3547 viruses and the viruses were classified to 685 families, based on their properties.

In order to determine the performance and possible advantages of the proposed approach, four classes of experiments are carried on three practical data sets under windows operating system. The first data set contains 538 programs with the self set of 284 legal files and the nonself set of 254 virus files; the second set contains 1815 programs with the self set of 915 legal files and 900 virus files; The third set consists of the second set and 2647 extra virus files, having 4462 files in total.

B. Description of Experiments

In this section, four classes of simulation results are described in detail.

In Tables, "A" denotes the total number of corresponding programs in the certain set; "P" denotes the number of corresponding programs that are correctly recognized; "PR" denotes the correct recognition rate of the corresponding programs, $PR = P/A$; "APR" denotes the average correct recognition rate of the corresponding programs.

$$APR = \frac{P_{legal} + P_{virus}}{A_{legal} + A_{virus}} \tag{2}$$

1) Experiments of class 1: The experiments of class 1 including several tests are carried on the set of 538 files. The set is divided equally and randomly into the training set and the detecting set. The numbers are both 269, but the particular programs in these sets are different. We pitch upon test 1 and test 2 to perform the simulation results in Table III.

TABLE III
Experimental results of class 1.

Exp. NO.	The training set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 1	142	142	100%	127	124	97.6%	98.9%
Test 2	142	142	100%	127	124	97.6%	98.9%

Exp. NO.	The detecting set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 1	142	140	98.6%	127	119	93.7%	96.3%
Test 2	142	140	98.6%	127	122	96.1%	97.4%

It can be seen in Table III that two tests have a good correct recognition rate in both testing (98.9%) and detecting set (around 97%). In the training set, the model can perfectly recognize the legal files, with a recognition rate of around 97% for virus files. The trained model can recognize more than 98% unknown legal files and have a recognition rate of around 95% for unknown virus files in the suspicious program set. It is noted that for the experiments time after time, the model have a stable performance independent of any particular program in these sets.

2) Experiments of class 2: The experiments of class 2 is still based on the first data set. Four divisions of training and detecting set were made with ratios of 4:1, 2:1, 1:2 and 1:4, respectively. The results are shown in Table IV.

TABLE IV
Experimental results of class 2.

Exp. NO.	The training set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 3	227	227	100%	203	194	95.60%	97.90%
Test 4	190	190	100%	170	162	95.30%	97.80%
Test 5	94	94	100%	84	82	97.60%	98.90%
Test 6	57	57	100%	51	49	96.10%	98.10%

Exp. NO.	The detecting set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 3	57	56	98.20%	51	47	92.20%	95.40%
Test 4	94	93	98.90%	84	82	97.60%	98.30%
Test 5	190	187	98.40%	170	162	95.30%	96.90%
Test 6	227	224	98.70%	203	193	95.10%	97.00%

The recognition rate of the model does not decrease with the reduction of the training set. In the test 6, the number of files in the training set is much less than that of files in the detecting set. However the correct recognition rate are at 98.7% of the legal files and at 95.1% of suspicious virus files in the detecting set. This indicates that the proposed model can learn enough information in a small data set to get a good results in detecting a much bigger corresponding data set. So, the model has a very strong generalization.

3) Experiments of class 3: The experiments of class 3 is carried on the second set, a bigger data set. The ratios of the training set and the detecting set are 2 to 1, 1 to 1, 1 to 2, respectively. The results are shown in Table V.

The correct recognition rates in this class of experiments

TABLE V

EXPERIMENTAL RESULTS OF CLASS 3.

Exp. NO.	The training set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 7	610	610	100%	600	538	89.70%	94.90%
Test 8	457	457	100%	450	399	88.70%	94.40%
Test 9	305	305	100%	300	279	93.00%	96.50%

Exp. NO.	The detecting set						
	Legal files			Virus files			
	A	P	PR	A	P	PR	APR
Test 7	1305	303	99.30%	300	270	90.00%	94.70%
Test 8	458	450	98.30%	450	400	88.90%	93.60%
Test 9	610	601	98.50%	600	543	90.50%	94.50%

are a little lower than that in the class 1 and 2 , but are still at high levels, with 95% in the training set and around 94% in the detecting set.

All the above results have shown that the proposed model have an excellent stability and generalization.

4) Experiments of class 4: The experiments of class 4 used the third set - the biggest data set with 4462 programs to confirm the model's expansibility. The training set is covered by and much smaller than the detecting set, so that the expansibility and comprehensive ability of the model can be tested.

A outline of the relation between the detecting results and the training set is shown in Figure 5. "○" denotes the correct recognition rate of all the programs on the detecting set; "△" denotes the correct recognition rate of the virus programs; "▽" denotes the correct recognition rate of the legal programs.

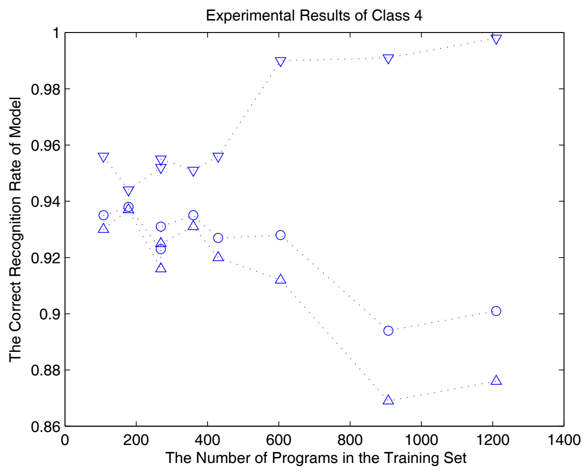

Fig. 5. The correct recognition rates in the class 4.

V. CONCLUSIONS

We have described a novel AIS-based method to overcome three specific shortcomings in traditional AIS models.

- Randomly generating the detectors leading to the bad efficiency;
- poor generalization, poor performance with a big data set;
- Ignoring the relevance between different extracted signatures in one virus.

In our approach, a guided candidate library is made by a priori knowledge. Then a fuzzy matching method is used to dig the similarity between genes. We store different genes in one virus together to keep all the information on the individual level by taking the advantage of relevance between different extracted signatures in the individual. Finally, classification decision is an overall behavior which reduces the information loss to a great extent. The model can effectively and efficiently recognize obfuscated virus, detect new variants of known virus and some unknown viruses. Although these modifications have been made, the model still have its own vulnerabilities. It can not maintain or increase the diversity of the genes in the virus library. Some artificial intelligent algorithms like immune network model or clonal selection algorithm could be used against it in future.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under grant No.60673020 and 60875080, and partially supported by the National High Technology Research and Development Program of China (863 Program), with grant No.2007AA01Z453.

REFERENCES

[1] P. S. Deng, J. Wang, W. Shieh *et al.* "Intelligent automatic malicious code signatures extraction", *IEEE 37th Annual 2003 International Carnahan Conference on Security Technology*, pp. 600-603.

[2] K. P. Anchor, P. D. Williams, G. H. Gunsch *et al.* "The Computer Defense Immune System: Current and Future Research in Intrusion Detection", *Evolutionary Computation*, 2002, pp. 1027-1032.

[3] J. O. Kephart. "A Biologically Inspired Immune System for Computers", in *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, 1994, pp. 130-139.

[4] S. Forrest, A. S. Perelson, L. Allen *et al.* "Self - Nonself Discrimination in a Computer", *Security and Privacy*, Oakland CA, pp. 202-212, 1994.

[5] P. D'haeseleer, S. Forrest, P. Helman. "An immunological approach to change detection: algorithms, analysis, and implications", *Proceedings of IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 110 - 119, May 1996.

[6] H. Lee, W. Kim, M. Hong. "Artificial Immune System against Viral Attack", *ICCS 2004*, *Lecture Notes in Computer Science 3037*, pp. 499-506, 2004.

[7] K. S. Edge, G. B. Lamont, R. A. Raines. "A retrovirus inspired algorithm for virus detection & optimization", *8th Annual Genetic and Evolutionary Computation Conference*, Seattle WA, 2006, pp. 103-110.

[8] T. Li. *Computer Immunology*, Beijing: Publishing house of electronics industry, pp. 187-191, 2004.

[9] D. Dasgupta, N. Attoh-Okine. "Immunity-Based Systems: A survey", *1997 IEEE International Conference on Systems, Man, and Cybernetics*, Computational Cybernetics and Simulation, 1997, pp. 369-374.

[10] P. K. Harmer, P. D. Williams, G. H. Gunsch *et al.* "An Artificial Immune System Architecture for Computer Security Applications", *IEEE Transactions on Evolutionary Computation*, vol. 6(3), pp. 252-280, 2002.

[11] M. D. Preda, M. Christodorescu, S. Jha*et al.* "A Semantics-Based Approach to Malware Detection", *34th Annual Symposium on Principles of Programming Languages*, vol. 42(1), pp. 377-388, 2007.

[12] O. Henchiri, N. Japkowicz, J. Nathalie. "A Feature Selection and Evaluation Scheme for Computer Virus Detection", *Sixth International Conference on Data Mining*, Hong Kong, China, 2006, pp. 891-895.