A Chromosome-based Evaluation Model for Computer Defense Immune Systems

Zejun Wu, Hongbin Dong, Yiwen Liang, R. I. McKay*

School of Computer / State Key Laboratory of Software Engineering

Wuhan University, Wuhan, P. R. China 430072

* School of Information Technology and Electrical Engineering,

University of New South Wales @ ADFA, Canberra, Australia~~School of Information Technology and Electrical Engineering,~~

~~University of New South Wales @ ADFA, Canberra, Australia~~

Wu_zejun@hotmail.com {ywliang,hbdong}@whu.edu.cn rim@cs.adfa.edu.au

Abstract- The Computer Defense Immune System (CDIS) is an artificial immune system for detecting computer viruses and network intrusions. We present a simple chromosome-based evaluation model for CDIS. In this model, the genotype space is a linear number sequence, and a digital pattern sequence is produced as the phenotype space using a number of mechanisms, including pattern mining and genetic algorithms. We present a range of experiment analyses to show the higher efficiency and stronger immunity of this model, improving the rate of successful prediction in intrusion detection in CDIS. The detectors generated may have higher coverage, hence would impose lower communications and computational loads on systems in which they were incorporated.

1 Introduction

The Computer Defense Immune System (CDIS) is a multilevel and distributed defense system, modeled on the biological immune process (Anchor et al, 2002a) and aimed at signature-based detection of computer viruses and network intrusions. The central part of CDIS is Computational Immune System (CIS) or Artificial Immune System (AIS) (Dasgupta, 1999).

We aim, in the work presented here, to improve the performance of CDIS in three respects:

- Prediction accuracy – if prediction accuracy can be improved, with fewer false negatives and false positives, then computer systems can be better protected
- Computational costs – the primary computational cost of a CDIS system is in detection, since every potential intruder must be checked against the antibody suite. The cost is proportional to the number of detectors, hence methods requiring fewer, more powerful detectors will impose less computational load on the overall system.
- Communications load – future CDIS applications will presumably follow a network model, with a need to distribute up-to-date detectors over the network. The resultant communications load will be proportionate to the number of detectors, hence again, fewer, more powerful detectors are desirable.

CDIS is designed to address two main problems in virus and intrusion detection. The first is that the problem domain is constantly changing, i.e. new defenses are confronted by new threats in a never-ending cycle. The second is that the problem domain space is enormous, i.e. it is difficult to cover limitless unknown space with limited

knowledge. CDIS addresses this by applying affinitive behaviors of the immune system model to detect new attacks. CDIS uses a stochastic search technique in an effort to cover those parts of the search landscape that may be the most fruitful. There has been previous work on developing CDIS for the signature-based detection of ~~signature-based~~ computer viruses and header-based detection of network intrusion packets. Customarily, antibodies, whose role is to identify intrusion, have been produced by a variety of methods, including Decision Trees, Finite State Machines and Artificial Neural Networks (Mitchell, 1997; Anchor et al, 2002b; Bradley et al, 2002; Zengren Yuan, 1999). But there are a number of opportunities for improvement in CDIS, especially in the method of producing antibodies and in cooperative detection. In particular, the antibodies produced by negative selection are imperfectly representational, and the antibodies optimized by stochastic search are imperfectly selective. Moreover the effect of cooperative detection has not been entirely evaluated by quantitative analysis.

Building on the work of Lamont (Anchor et al, 2002a; Anchor et al, 2002b), Dasgupta (Dasgupta, 1999), and Forrest (Hofmeyr et al, 1999; Forrest and Perelson, 1994), we apply a chromosome-based evaluation model to address the problems mentioned above and to boost the immunity and the prediction capability of CDIS. Section II describes the CDIS antibody lifecycle. Section III provides a simple chromosome-based evaluation model, while Section IV presents experiment analysis based on the detection of number sequences. Section V presents some ideas for further work.

2 CDIS Antibody Lifecycle

CDIS, by producing antibodies that can detect signature-based computer viruses and header-based network packets, may remedy the disadvantages of static computer defense systems (Anchor et al, 2002a). The life cycle of a network intrusion antibody in CDIS is shown is Figure 1. The main components of the lifecycle are described in detail below; several potential improvements are included in the description.

2.1 Antibody Creation and Negative Selection

Firstly, the system randomly generates a number of

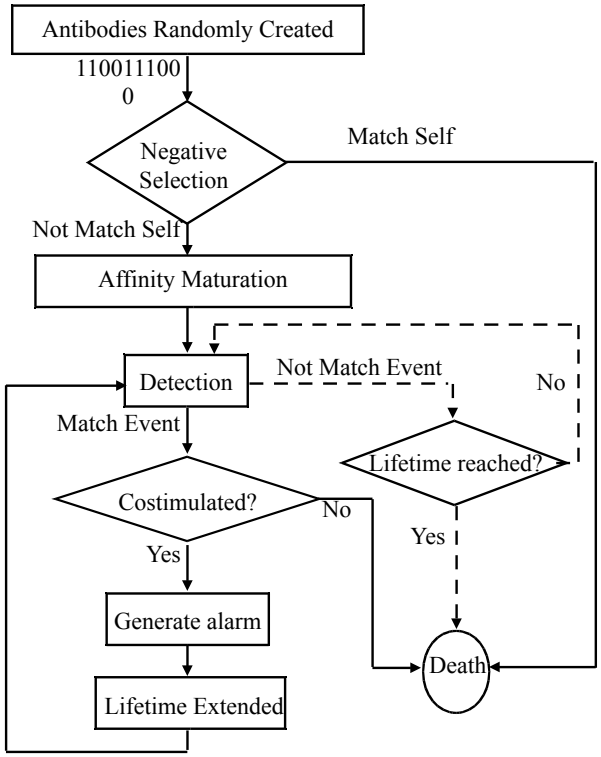

Fig.1. Antibody Lifecycle in CDIS

antibodies, encoded as binary sequences. The length, as well as each digit, of the sequence is stochastic. Each sequence, being regarded as an antibody, is passed through negative selection. The antibody is checked to determine if any individual of the self-training-set matches it (Hofmeyr et al, 1999). If so, the antibody is discarded and another is randomly generated. If not, it passes to the next stage, i.e. Affinity Maturation. The negative selection normally used is a single-direction matching. We propose a more complex bi-directional matching for negative selection, including both the degree of mismatch with the Self-training-set, and the degree of match with the Nonself-training-set, aiming to improve the reliability of negative selection.

2.2 Affinity Maturation

Since antibodies are created randomly, their locations and range parameters are not likely to be as large or general as possible. A stochastic search technique is usually used to optimize the antibodies, improving their recognition capability to the maximum extent possible without matching any individuals in the self-training-set. But this search technique is random and can't ensure the

particularity of the antibodies. Thus, this paper proposes that genetic algorithms, using reproduction, crossover and mutation, be used to evolve the antibodies. The process of evolution of the antibodies, which evolves the antibody-pattern-set and washes out the inferior antibodies, enhances the validity and "affinity" of antibodies. The process of affinity maturation can be regarded as a local search for better antibodies.

2.3 Imperfect Matching Method

In the standard 'perfect matching' method, the object being checked will not be confirmed as an intrusion in the inspection system unless each digit of the object is the same as that of an antibody, the binary sequence mentioned above. In this approach, each antibody can only recognize one incursive antigen, so that the efficiency of recognition is extraordinarily low. The imperfect matching method in CDIS requires that only some, not all, of the digits are the same as the object being checked. If the values of the number sequence are not restricted to 0 and 1, matching can be extended to a requirement that each digit of the object is within a certain ~~appointed~~ nominated range. Thus, the matching method is imperfect in the sense that it matches a space rather than a single point. To further extend the range of recognition, this paper proposes a pattern matching method. This method extracts the most representative pattern set, providing a larger area of coverage.

2.4 Costimulation

Since the self-training-set is not perfectly representative, the signatures produced by the self-training-set may show some errors and produce false predictions, mistaking some normal behaviors for intrusions. To solve this problem, the detection phase is completed by the cooperation of multiple antibodies, known as costimulation in CDIS. So an incoming object is not categorized as an attack unless multiple antibodies detect it. But the effect of cooperative detection has not been entirely evaluated by quantitative analysis. This paper carries out a simple evaluation to look for an affinitive relationship between the object being checked and the excellent antibody set, and to complete the work of cooperative detection. Rather than the solo action of an individual it is the cooperative activity of a whole set of excellent antibodies which represents the generality of the set and increases the reliability of prediction.

3 Chromosome-based Evaluation Model in CDIS

3.1 Genotype Space and Phenotype Space in CDIS

A gene is a biological concept. In biology, taking a rabbit as an example, exterior characteristics, such as the length of tail, the size of ears and the color of eyes, are determined by different sequences of genes. These exterior characteristics of a rabbit, which are presentations of the non-visible genes, can be directly judged by the rabbit's appearance. Thomas Bäck first introduced the concepts of Genotype Space and Phenotype Space in the applications of Evolutionary Computing (Thoma Bäck et al, 1997). In his opinion, the abstract mathematical objects, such as binary strings, are regarded as a genotype space in which to carry out genetic operations conveniently, and the diverse parameters of the real system, such as energy consumption and output, are considered as a phenotype space. The process of producing phenotype space from genotype space is thus a crucial consideration.

3.2 Chromosome-based Immune process in CDIS

This paper embodies a chromosome-based evaluation model derived from research into the CDIS antibody lifecycle. The process of producing phenotype space from genotype space in CDIS is shown in Figure 2. The main components of the immune process are described in detail as follows.

In this paper, we assume that the supposed object being detected is a binary sequence. Three important algorithms are proposed to look for an appropriate evaluation. The immune process is discussed under the succeeding three headings.

(1) Extract signature, code gene, mine pattern

The binary coded Self-training-set T_S and Nonself-training-set, T_N, are each split into sub-sequences of a given length LEN, forming the Self-subsequence-set C_S and Nonself-subsequence-set C_N, which are viewed as the genotype space. The Self-pattern-set E_S and Nonself-pattern-set, E_N, can be extracted from these subsequence sets by pattern-mining. The algorithm is as follows:

The concepts mentioned above are applied to CDIS in this paper, in which the Self-training-set and Nonself-training-set are viewed as the genotype space in the CDIS antibody lifecycle. The Self-patterns and Nonself-patterns, being treated as chromosomes, are produced by pattern-mining over genotype space. Then through negative selection, antibody patterns with obvious signatures are produced. After an evolutionary process using genetic algorithms, excellent Nonself chromosomes are evolved, which combine and form a lot of sequences of chromosomes, i.e. phenotype space. The evaluation algorithm is thus a method to relate genotype space to phenotype space.

The number subsequences can be used as excellent chromosomes in genotype space. The more characters are similar to the elements in Nonself space and dissimilar to those in Self space, the more the object being checked is considered to be close to Nonself, and vice versa. In this paper, we try to exploit this "affinity" by using an evaluation algorithm. This immune process will be described in detail in section 3.2.

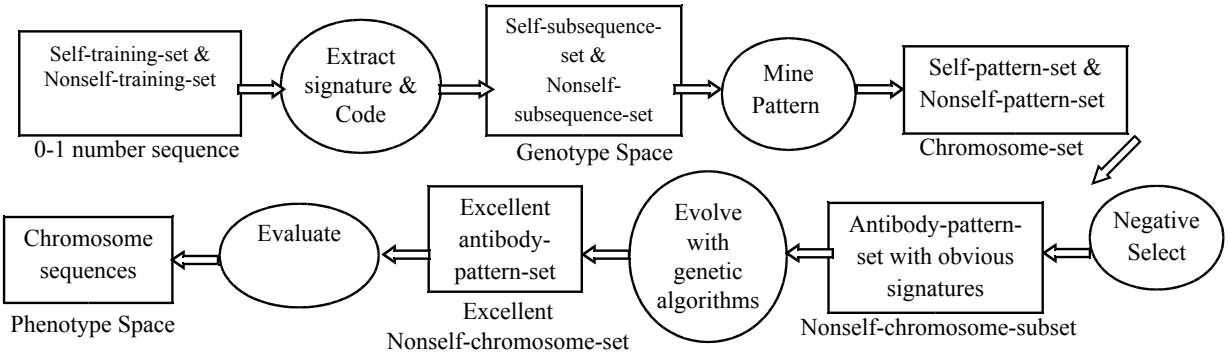

Fig.2. The process of producing Phenotype Space from Genotype Space in CDIS

Algorithm 1: Abstract chromosome from Self or Nonself (taking Self as an example; d_S as an individual in D_S; right as the weight of d_S, which reflects the frequency of d_S's repetitive appearance; RIGHT as a default weight.)

1) T_S is cut into subsequences of length LEN, forming C_S;
2) All elements in C_S are ordered by Gray code to form D_S;
3) RIGHT → d_S.right;
4) Extract pattern to form pattern-set, $E_S=\{0, 1, *\}^+$, from D_S:
   a) While there is a new pattern in D_S;
   If there is only one different digit between two adjacent patterns d_i and d_j then,
   Unite the two patterns into one new pattern e_S, by replacing the different digit with symbol *, and d_i.right + d_j.right → e_s.right, and put e_S into set E_S;
   b) Repeat a) until there is no new pattern in E_S
5) End.

(2) Negative selection, evolution with genetic algorithms

Antibodies, the set A, are created randomly as strings over {0,1,*}. Starting with this set A as an initial population, we apply an elitist genetic algorithm to generate a final population of antibodies; the best member of ~~this~~ the final population is chosen as a member of the "excellent antibody set" M. The process is repeated N times, to give a set of size N. The fitness function used is computed from the matching degree between antibodies and the Nonself-chromosome-set, and the dismatching degrees between antibodies and the Self-chromosome-set, with a weighting given by the weighting parameters x_1 and x_2. In this alogrithm, d_match_self() is a function to compute the degree of mismatch between the antibody and E_S, match_nonself() is a function to compute the degree of match between antibody and E_N, and the matching degree is proportionate to the the number of correct matchings with E. The Genetic algorithm uses Fitness Proportionate Selection, and Reproduction, Crossover and Mutation, with probabilities 0.75, 0.2, and 0.05, respectively. The elite size is ⅓. The algorithm can be written as follows:

Algorithm 2: Negative selection and Genetic Algorithm

1) Initialize an antibody population of size N:
   $A=\{a_1,a_2\ldots\ldots a_n\}$ (a_i consists of 0, 1 and *);
2) Calculate the Fitness of each individual and its selection probability:
   $Fitness_i$=d_match_self(a_i) / x_1
   + match_nonself(a_i) / x_2
   $P_i=Fitness_i / (Fitness_1 + Fitness_2 + \ldots + Fitness_i + \ldots + Fitness_n)$
3) Select two antibodies a_j and a_k with probability proportionate to P_j and P_k and generate two novel antibodies a_j' and a_k' by genetic operators.
4) Repeat 3) N/2 times and generate a new antibody population $A'=\{a_1', a_2' \ldots a_n'\}$;
5) Select best ⅓ individuals from A and ⅔ individuals from A′ to form a new antibody population A″.
6) Repeat from 2) to 5) T times and find antibody m_i , which has the highest value of Fitness, from A″;

7) Repeat from 1) to 6) N times and generate an excellent antibody population as :
$M=\{ m_1, m_2 \ldots\ldots m_n \}$;
8) End.

(3) Evaluation System and Phenotype Space

In order to evaluate the object being checked, we should firstly look for an evaluation function B(X) and threshold values. In this paper a simple evaluation function is adopted to map the object to a space of [0,1]. T_S and T_N are used to confirm the threshold values, such as α and β, which meet that $B(t_S) <\alpha$ and $B(t_N) >\beta$. (t_S is a~~n~~ unit of T_S and t_N is a~~n~~ unit of T_N). The evaluation algrithm with B(X) is written as follows:

Algorithm 3: Evaluation Algorithm
1) Code the object being checked into binary sequence X;
2) Get the excellent Nonself-chromosome-set from Algorithm 2:
$M=\{m_1, m_2 \ldots\ldots m_n\}$;
3) Calculate the evaluation fuction: [1]

$$B(X)=\sum_{i=1}^{n} \frac{f_i}{\sum_{j=1}^{n} f_j} x_i \quad ;$$

4) If B(X)<α, X is a normal activity. If B(X)>β, X is an intrusion and alarm will be raised. If α<B(X)<β, X is an ambiguous activity (If X is an ambiguous activity, artificial arbitration is needed) ;
5) End.

A chromosome sequence of the object being checked matches that of some units in the excellent Nonself-chromosome-set when the object is evaluated as an intrusion by B(X). This means that some common features exist in their phenotype spaces. The simple evaluation function mentioned above is called a chromosome model or an "affinity" model.

Examining the antigen by evaluation function is equivalent to appraising antigen by chromosome sequence. The antigen is confirmed as intrusion whenever some gene signatures of the antigen, which are considered as a chromosome sequence, are similar to that of an antibody. The union of the chromosome sequences is called the phenotype space.

4 Experiment Analysis

4.1 Experimental Data

We generate training sets and testing sets by a stochastic algorithm so as to obtain a range of experimental data. A control experiment is carried out by using different sizes of training and testing sets. 10^3 or 10^4 for training and 10^3 or 6×10^3 for testing. The training set is divided into genes of length 25 digits, i.e. LEN=25. A further control experiment compares the inclusion and omission of pattern mining from the algorithm. The number of generations is 300, i.e. T=300. The size of M is 46, i.e. N=46. The detailed data are shown in Table 1 and Table 2 (TP means the True Positive rate; N-test-set and S-test-set mean the nonself and self training sets respectively; the table gives the mean values over 10 trials, followed by their standard deviations.):

Table 1. Self-training-set and Nonself-training-set of size 10^4.

Exp. 1	Total num	First(pattern mining)			Second(No pattern mining)		
		Detected	Unsure	TP(%)	Detected	Unsure	TP(%)
N-test-set	6000	5318	234	88.6	4827	650	80.5
		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
N-test-set	1000	808	105	80.8	752	153	75.2
		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S-test-set	6000	0	25	0	0	72	0
		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
S-test-set	1000	0	14	0	0	42	0
		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Exp. 2 (First pattern mining) (No pattern mining)

length of gene: 25 number of M: 46 Evolutionary generation (T): 300

length of gene: 25 number of M: 46 Evolutionary generation (T): 300

Table 2. Self-training-set and Nonself-training-set of size 10^3.

11. If m_i does not exist in X, then $x_i=0$, otherwise $x_i=1$. f_i is the fitness value of m_i , which can be calculated by formula metioned in Algorithm 2.

4.2 Experimental Results

The results of of these experiments can be compared with the results of similar experiments of (Forrest and Perelson 1994). As in our experiments, they generated random binary sequences to carry out the experiments. Unlike our work, they used a common negative selection algorithm to produce detectors, and an r-contiguous linear conformation algorithm to detect intrusions. Moreover the evaluation mechanism used by Forrest differs from that proposed in this paper. Table 3 compares the results of the two experiments.

Contrastive Results	R-contiguous Exp by Forrest	Exp. 1 in this paper
Size of N-test-set	128	10^4
Number of Detectors	46	46
TP	84.3%	88.6%
Coverage Rate	2.3	192.7

Table 3. Comparative results between Forrest's experiment and Experiment 1 with pattern mining. Coverage Rate means the average number of intrusions detected by each detector.

From the table, our approach in experiment 1 achieves a recognition rate of 88.6% compared to 84.3% in the r-contiguous experiment. Forrest's paper does not quote a standard deviation, so it is not possible to conduct a significance test on these results; however the fact that the difference of the means is more than double the standard deviation in our experiments is highly suggestive of significance. Equally important, Forrest and Perelson's approach covers 128 objects with 46 detectors, giving an average number of intrusions detected per detector of 2.3; by comparison, our approach covers 10^4 objects with 46 detectors, giving an average number of intrusions detected per detector of 192.7. So the detectors produced by our chromosome-based model have far higher coverage. In operational CDIS systems, distribution and execution of detectors will impose significant communications and computational loads. The distribution cost is proportional to the number of detectors, as is the computational cost of detection. Hence it is better to have a few powerful detectors rather than many detectors with narrow scope.

When we consider the results of experiments 1 and 2 in this paper, we may note the following behaviours:

- **Zero false positive rate and lower false negative rate**

In the experiments, the recognition rate on the self-testing-set is zero, i.e. there is a zero false positive rate. This means that the system will give no false alarms. Moreover if we compute the definite errors from the table (i.e. where the system fails to recognise nonself objects), the inclusion of pattern mining reduces the false negative rates of 8.7% and 9.2% to 7.5% and 8.7% respectively. As a result , the system rarely mistakes unknown Nonself objects as known normal activities.

- **Antibody pattern has higher recognition rate**

The recognition of definite nonself objects is considerably higher when pattern mining is used – 8.1% higher in Experiment 1, and 5.3% in Experiment 2. That is, the antibody-pattern-set has a higher recognition rate and a larger coverage area when pattern mining is used. These differences are significant at the 1% significance level (hetero-scedastic one-tailed Student's T-test).

- **Effect on model from training-set size**

Experiment 1 uses a training set of size 10^4, compared with 10^3 in experiment 2. Comparing the results between experment 1 and experiment 2, we see an increase in detection rate of 5% with no pattern mining , and 7.8% with pattern mining . That is, the immune model is increasingly effective as the size of the training set increases.

5 Conclusions

This paper proposes a new conception of immune chromosomes derived from research on the CDIS antibody lifecycle. We make several suggestions to improve CDIS, notably the use of bi-directional negative selection to produce representational antibodies, the use of genetic algorithms to optimize distinctive antibodies, and the use of a simple evaluation function to do a quantitative analysis of cooperative detection. These approaches are used here to look for a

high-affinity chromosome model. The relationship between genotype space and phenotype space is generated by the evaluation system. Our experimental data have suggested that :

- The chromosome-based evaluation model has high feasibility and immunity, improving the successful prediction rate in intrusion detection system.
- Our process generates vastly more powerful detectors than comparable methods, hence requires dramatically fewer of them. In practical terms, this means that CDIS systems based on this approach will impose significantly lower resource requirements, both computational and communications.

In future work, we will further examine the chromosome model. In particular, we plan to investigate more complex methods for the evaluation system, such as Evolutionary Modeling or multi-level evaluation. Rather than repeated applications of a Genetic Algorithm to generate the excellent antibody set, we plan to use diversity mechanisms so that the elite members of the final GA population can be directly used as the excellent antibody set. We also plan to trial the system in a real-World environment, testing system call sequences (Hofmeyr et al, 1998), and to measure its comparative resource requirements in a realistic system-protection setting.

Acknowledgements

This work was supported by Key Research Grant No. 90204011 from National Natural Science Foundation of China.

Bibliography

Kevin P. Anchor, Paul D. Williams, Gregg H. Gunsch and Gary B. Lamont, (2002a) “The Computer Defense Immune System: Current and Future Research in Intrusion Detection,” in Congress on Evolutionary Computation.

D. Dasgupta, (1999) “Artificial Immune Systems and Their Applications,” Springer-Verlag, New York.

T. Mitchell, (1997) “Machine Learning,” McGraw-Hill, Boston, MA.

K. Anchor, G. Lamont and G. Gunsch, (2002b) “An Evolutionary Programming Approach for Detecting Novel Computer Network Attacks,” in Proceedings of the Congress on Evolutionary Computation, Honolulu, IEEE Press.

D. W. Bradley and A. M. Tyrrell, (2002) “Immunotronics — Novel Finite-State-Machine Architectures With Built-In Self-Test Using Self-Nonself Differentiation,” in Proceedings of the Congress on Evolutionary Computation, Honolulu, IEEE press.

Zengren Yuan, (1999), “Artificial Neural Networks and Applications, ” Tsinghua University Book Concern.

S. Hofmeyr and S. Forrest, (1999) “Architecture for an Artificial Immune System,” Evolutionary Computation, Vol.7 (1), pp.1289-1296.

Thoma Bäck, Ulrich Hammel and Hans-Paul Schwefel, (1997) “Evolutionary Computation: Comments on the History and Current State,” IEEE Trans on Evolutionary Computation, Vol1, No.1, April pp.3-15.

Zhengjun Pan, Lishan KANG, Yuping Cheng, (1998) “Evolutionary Computing,” Tsinghua University Book Concern.

S. Forrest and A. S. Perelson, (1994) “Self-Nonself Discrimination in a Computer,” in Proceedings of Symposium on Research in Security and Privacy, IEEE press.

S. Hofmeyr, S. Forrest And A. Somayaji, (1998) “Intrusion Detection using a Sequence of System Calls,” Journal of Computer Security, Vol.6, pp.151-180.