A Logic Program Semantics-based Framework for Immune Anomaly Detection

Chengyu Tan[1], Hongbin Dong[2], and Yiwen Liang[3]

[1] School of Computer Science, Wuhan University, Wuhan,430072，P. R. China
nadinetan@163.com
[2] State Key Laboratory of Software Engineering/ School of Computer Science, Wuhan University, Wuhan,430072，P. R. China
hbdong@whu.edu.cn
[3] State Key Laboratory of Software Engineering/ School of Computer Science, Wuhan University, Wuhan,430072，P. R. China
ywliang@whu.edu.cn

Abstract. Some algorithms of Artificial Immune System, such as negative selection algorithm, can not effectively capture semantic information in some complex problem spaces. In fact, many semantics exist in digit antigen space. If we can recognize based on the semantics of antigens, the accuracy of anomaly detection can be improved. In this paper, by the background of system call trails generated by process, we design a systematic framework of artificial immunity applied to process anomaly detection. This paper proposes to descript antigens and immune detectors with first order logic, and construct a time sequence model of immune detector based upon stable model theory in extended logic programming（ELP）. At last, we introduce a training strategy of new immune detector based on genetic inductive logic programming.

1 Introduction

Since Farmer proposed to apply immunization to a computational system at first in 1988, Artificial Immune Systems (AIS) have been verified effective to real world problems. Just as Emma Hart pointed out, application areas of AIS techniques can be broadly summarized as Learning, Anomaly Detection and Optimization.[1]

Many recent approaches of AIS are changed from the research by Stephanie Forrest [2]. The main idea of Forrest's research is negative selection algorithm. This algorithm includes three stages. The first stage is to construct normal behavior set (self). The next is to produce detector set. The last is to detect abnormal behavior (non-self). In Forrest's algorithm (negative selection algorithm), antigens and detectors are 49-bits binary string, and matching rule is r-contiguous. There are some limitations in these approaches. A main limitation is Scalability. It means that in order to insure high detection rate, a large number of detectors must be generated. To some extent, the number of detectors could be unmanageable [3]. Just as Balthrop said, the real reason is not negative selection algorithm. The basic reason is that the match rule

reflects the similarity in the genotype space, not the similarity in the phenotype space [4].

Some approaches, such as a real-valued negative selection algorithm by Fabio González [5], have solved these limitations of binary detector to a certain extent. But more recently, the design focus of many AIS has become more engineering oriented, with less emphasis placed on trying to understand and extract key biological properties [6].

C.Ko,G. Fink and K. Levitt proposed to detect in privileged process at the first time [7]. They found program specification language based on first order predicate to descript codes in relation to security. But this method needs researcher read codes of the programs to confirm the anticipated behaviors of process. So Ko use Inductive Logic Programming (ILP) to generate the descriptions of process's anticipated behavior [8].

Genetic Inductive Logic Programming (GILP) is an automatic programming which combines the approaches of ILP and genetic algorithm (GA) [9].

In this paper, we aim at anomaly detection of LINUX processes. By the background of system call trails generated by processes, we design a systematic framework based on artificial immunity. In this system, we propose to descript antigens and immune detectors with first order logic. Because Extended Logic Programming (ELP) can deal directly with incomplete information, we construct a time sequence model of immune detector with ELP. We use stable model computing, which is the semantic model of ELP (also called answer set), to be the matching rule of new detector and antigen. Based on genetic inductive logic programming (GILP), this paper introduces an evolutionary algorithm of the new immune detector.

2 Systemic Framework of Semantic Anomaly Detection

In this section, we give the systemic framework of semantic anomaly detection shown in figure 1. We use this system to detect the privileged process of LINUX. The main difference between our system and others is the description language and the matching rule of antigens and immune detectors. In order to capture the key semantic information of antigens, we adapt the matching rule to compute stable model semantics. Through calculating that the collection of a detector and an antigen has or does not have stable model, we may say that the detector matches or does not match with the antigen.

The framework presented by figure 1consists of two parts: training offline and detecting online. Components of this framework are just as following.

2.1 First order predicate and antigens

Stephanie Forrest defined short sequence of system calls to be the process behavior. But in her approaches, system calls did not include arguments, for example, opened files, modified files or not, promoted privileges or not. The intruder can escape the

detection easily through making changes in the intruding means (such as adding some irrelevant system calls in a program in order to improve the matching rate).

In order to overcome this limitation, we think it is necessary to describe the time sequence relation of intrusion behavior. Through tracing the execution of a process, we can get a system calls sequence, and the same time we get the important arguments of system calls. For example, the following short sequence of system calls are given to define a normal behavior (self):

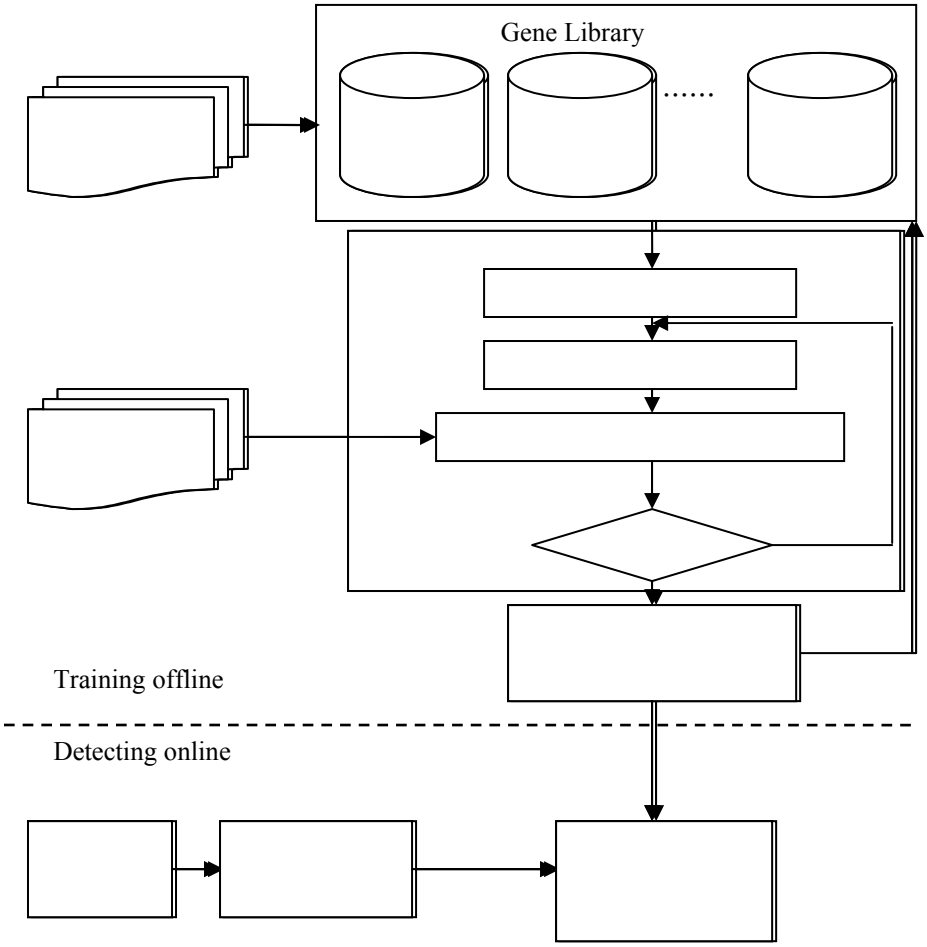


Fig. 1. This shows a figure of the Systemic Framework of Semantic Anomaly Detection

open(f1,m1),read(fd， buf), mmap(ad1,len1,pro1),mmap(ad2,len2,pro2),…

We define two groups of first order predicates to present antigens and immune detectors.

【definition 1】**basic predicates**, the first group predicates relation to system calls and their arguments.

For example, predicate *open(f,m,d)* means that the file *f* is opened with mode *m*, and it appear the d^{th} position in short sequence of system calls.

In our system, we do not use all system call of LINUX in order to reduce the complexity. We just use high dangerous system calls (see table 1). The predicates of this group are defined as following.

P(a1,a2,……,an,d)

P is the system call name (see table 1)

a1,a2,……,an are arguments of system call *p.*

d is the position number in short sequence

Table 1. System calls used in the semantic anomaly detection. These system calls are high dangerous to computer security.

Relation to file system	Relation to process		Others
Open chmod fchmod	setuid	setgid	
chown lchown fchown	setreuid	setregid	
link unlink symlink	setgroups	setfsuid	create-module
rename creat mknod	setfsgid	setresuid	delete-module
mkdir mount	Setresgid	execve	
pivot			

【definition 2】**expanding predicates**, the second group predicates represent the time sequence relation of system calls . We define three predicates just as follows.

AT(SYS1,n)

System call SYS1 is at the n^{th}position

BEFORE(SYS1, SYS2)

System call SYS1 appears in any position in front of system call SYS2

BEFOREN (SYS_1, SYS_2, *n)*

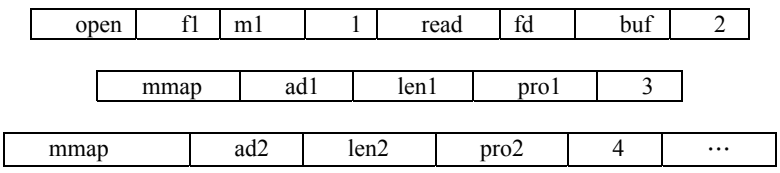
System call SYS1 appears in the n^{th} position in front of system call SYS2

We use basic predicates to present antigens, and to descript immune detectors with basic and expanding predicates. Antigens can be viewed as a set of facts. The example provided above can be described with first-order predicate as following.

open(f1,m1,1),read(fd, uf,2),

mmap(ad1,len1,pro1,3),mmap(ad2,len2,pro2,4),…

The code method of antigens is similar to the method of SIA01 (Supervised Inductive Algorithm version 01). This example can be encoded as the following:



2.2 Gene Library

For reducing the cost of detector generation and improve non-self coverage, we define a gene library in our system. Gene library contains all the atoms that will possibly appear in the immune detector. When we initialize the detector population, we may select atoms from the gene library to generate the detector individuals. It can prevent from producing a large amount of individuals with lower fitness value. For prevent to select the atoms existing conflicts, gene library consists of several sub gene libraries. We can only select an atom from a sub gene library.

Two training sets, self training set and non-self training set, exist in our system. The atoms in gene library come from non-self training set initially. Each atom of each individual in non-self training set joins to gene library.

Through evaluating gradually, gene library will be dynamic. Once a memory detector identifies self incorrectly, it will be deleted from memory detectors population by immune system. Before deleting it, for reserving effective information, its atoms either positive literal or negative literal will joins to gene library. It is benefit for generating more effective detector.

2.3 Time Sequence Model of Immune Detector

In this section, we define a time sequence model of immune detector with ELP. Each detector is a rule described by the predicates defined above. The matching rule between detectors and antigens is stable model computing. A new evolutionary algorithm of time sequence model detector will be introduced in section 3. In this section, we will introduce the phenotype and genotype of time sequence model detector.

Phenotype of detector

The head of rule is defined to be *abnormal*. The phenotype of the detector is as following:

$abnormal{:}\text{-}p_1(x_1,y_1),p_2(x2,y2),\cdots,p_k(x_k,y_k),not\ p_{k+1}(x_{k+1},y_{k+1}),\cdots,not\ p_m(x_m,y_m)$

$m \leq K$, K is a constant that means the maximum number of atoms in rule body. Because each predicate argument is slightly different in number, we describe as two arguments simply here. The experiment can be done according to the actual number of argument.

Genotype of detector

The rule above can be encoded, and its genotype can be shown by the following:

$$\#negative\ p_{k+1}\ x_{k+1}\ y_{k+1}\cdots p_m\ x_m\ y_m\ p_1\ x_1\ y_1\ p_2\ x_2\ y_2\cdots p_k\ x_k\ y_k$$

#negative is the number of negative literals in the rule.

3 Training Strategy of Time Sequence Model Detector

Because model of the detector in this paper is different from traditional models, we need to design new genetic operators, fitness function and matching rule.

3.1 Initialization

The initialization step will consist of the following operations:

1. Produce a random integer M, $1 \leq M \leq K$. M is the number of literals in rule body.
2. Produce M pieces of random integers as the number of sub gene libraries in which we will select atoms. We select an atom randomly from each sub gene library to be selected.
3. Construct a rule A with these atoms selected in step 2.
4. Negative selection. If A can not match any element of self training set, A is joined in the detector population.
5. Repeat step 1 to step 4 until the detector population is full.

3.2 Genetic Operator

We define these operators based on techniques developed in the field of ILP for generalization or specialization of the achieved knowledge. [9] The background knowledge is essential for achieving intelligent behavior. It is knowledge common to several examples.

Selection

We choice roulette wheel selection in this algorithm.

Crossover Operator

We use a restricting one-point crossover which is different from traditional one-point crossover. The cross point must be front of a predicate. Each offspring after restricting one-point crossover must be checked on semantics, and be confirmed to be legal.

Mutation Operators

We define four generalization mutation operators, and four specialization mutation operators. When the fitness value of an individual is lower than a threshold value, the application probability of generalization operators is larger than the application probability of specialization operators.

Generalization Operators: The four generalization mutation operators are according to the principle of climbing generalization to design.
deleteoperator: delete a literal (positive literal or negative literal) from the right side.
Literalsubstitutionoperator1: one or several literals selected from the right side are replaced by one or several literals selected randomly base on the background knowledge. New literals are more generalized than olds.
constanttovariableoperator: a literal is selected from the right side. Then one constant argument of the literal is replaced by a variable.
negativettopositivoperator: if there are negative literals in the right side ,a negative literal is selected . Then it is replaced by its positive literal.
Specialization Operators
plusliteraloperator: a new literal selected from gene library joins to the right side of the individual.
Literalsubstitutionoperator2: one or several literals selected from the right side are replaced by one or several literals selected randomly base on the background knowledge. New literals are more special than olds.
variabletoconsatantoperator: a literal with one or several variable arguments is selected from the right side randomly. Then one variable argument of the literal is replaced by a constant.
Positivetonegativeoperator: a positive literal is selected from the right side. Then it is replaced by its negative literal.

3.3 Matching rule

The matching rule of detector and antigen is to compute the stable model.

For example, detector x is

abnormal-:before(open ,read),read(file1,buf1,3), at(write,6)

Antigen y is

open(file1,mode1,1):-
read(file1, buf1,2):-
mmap(ad1,len1,pro1,3):-
mmap(ad2,len2,pro2):-
open(file2,mode1,5):-
read(file2,buf2,6):-

For computing stable model of the collection of detector and antigen, we need to define some assistance rules presented the reasoning relation among the predicates. Such as,

beforen(p1,p2,m-n):-at(p1,n),at(p2,m),large(m,n)
before(p1,p2):-beforen(p1,p2,x)
at(p,m):-p(a1,a2,…an,m),isdefine(p)

Of course, we need other assistance rules of the closed world assumption to deal with negative information.

If the stable model of the collection of detector x, antigen y and these assistance rules includes the element *abnormal*, we may say that match(x, y) is true, otherwise, match(x, y) is false.

3.4 Fitness Function

The fitness function of individual detector C is defined as following:

$$\text{FITNESS}(C)\begin{cases} = \alpha*N_C-\beta*DIS_C+\gamma*NLIT_C+\delta*NVAR_C & \\ & \text{if } S_C<M \\ =0 \quad \text{otherwise} & \end{cases} \tag{1}$$

N_C =the coverage rate of non-self training set covered by C.
|NONSELF(C)|=the number of non-self training set covered by C.

$$N_C = \sum \frac{|NONSELF(C) \bigcap NONSELF(A)|}{|NONSELF(C)|}. \tag{2}$$

A is any individual besides C in the detector population.
$NLIT_C$: the number of literals in C takes the proportion of the total amount of predicates
DIS_C=total of the distance between C and other individuals in the detector population.
$NVAR_C$: the number of variable in C takes the proportion of the total amount of arguments in C.
α, β, γ and δ are tunable.
S_C=the coverage rate of self training set covered by C
M=the maximum noise that can be tolerated for this algorithm.

3.5 Training algorithm

In this section, a training algorithm of time sequence model detector is given. A negative selected operator is used on each offspring after operating genetic operators in this algorithm. The offspring that does not match any element of self training set can be added to population P(t+1).

Evolutonary algorithm of immune detectors.

```
lp_rules_evolve()
{  pos=nonselfset;
   ruleset=ϕ;
   do{  t=0;
        initialize detector population P(t);
        while(not terminate)
```

```
        {  select two individuals ;
           execute crossover and mutation operators on
              individuals;
           execute negative selected operator to
              offspring;
           add new offspring to population P(t+1);
           t=t+1;
         }
        ruleset=ruleset U P(t);
        pos=pos-{pos covered by the rules in P(t)};
   }while(|pos|>>|nonselfset|);
   ruleset is evolved detector population;
}
```

4 Recognition Process Online

The recognition process is shown as follows.

1. ***Data sensor and transformation.*** Get short sequences of system calls in a running process. Then turn the short sequence into the presentation of first order logic through a predicate converting system.
2. ***Recognition and alarm.*** If a short sequence matches an element of mature detectors population, an abnormal is found. If the number of abnormal in system call sequence of a running process is large than a given threshold value, an intrusion is reported.

When some individuals of mature detectors population can not recognize any abnormal within one period, a different mutation that we call *expanding mutation* will be operated on them based on knowledge update of logic programs. *Expanding mutation* is executed through selecting atoms from self training set or new non-self individuals that are recognized.

5 Conclusion

We design a systemic framework of semantic anomaly detection for LINUX privileged process. Antigens and immune detectors are presented by first order logic. A new detector model defined in this paper is named time sequence model. The matching rule between antigens and detectors is stable model computing of ELP. This method can reflect the similarity of time sequence semantics. Invalid detector individual and gene library can be updated automatic based on knowledge update ability of logic programming.

References

1 Emma Hart1 and Jonathan Timmis2, ApplicationAreas ofAIS:ThePast,ThePresent andThe-Future, ICARIS 2005, LNCS 3627, pp. 483 – 497, 2005.
2 S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonself discrimination in a computer, in Proceedings IEEE Symposium on Research in Security and Privacy, (Los Alamitos, CA), pp. 202~212, IEEE Computer Society Press, 1994.
3 Fabio González, Dipankar Dasgupta Version. Final version appeared in Genetic Programming and Evolvable Machines, 4(4), pages 383-403, Kluwer Acad. Publ., December 2003.
4 J. Balthrop, S. Forrest and M. R. Glickman. Revisting lisys: Parameters and normal behavior. In Proceedings of the 2002 Congress on Evolutionary Computation CEC2002, pages 1045–1050. IEEE Press, 2002.
5 Fabio Gonz´alez, Dipankar Dasgupta and Jonatan G´ome. The Effect of Binary Matching Rules in Negative Selection. GECCO 2003, LNCS 2723, pp. 195–206, 2003.
6 Paul S. Andrews1 and Jon Timmis2, Inspiration for the Next Generation of Artificial Immune Systems. ICARIS 2005, LNCS 3627, pp. 126 – 138, 2005.
7 C.Ko,G. Fink and K. Levitt. Automated detection of vulnerabilities in privileged programs by execution monitoring. In Proceedings of the 10th Annual Computer Security Applications Conference, pages 134–144, December 5–9，1994.
8 Calvin Ko. Logic Induction of Valid Behavior Specifications for Intrusion Detection. In the Proceedings of the 2000 IEEE Symposium on Security and Privacy, Oakland, CA,2000.
9 Gabrilla Kókai Gelog-A System Combining Genetic Algorithm with Inductive Logic Programming B. Reusch(Ed.): Fuzzy Days 2001,LNCS 2206, pp.326-344,2001