Immunity based Virus Detection with Process Call Arguments and User Feedback

Zhou Li, Yiwen Liang, Zejun Wu, Chengyu Tan
College of Computer Science,
Wuhan University,
Wuhan, 430072, P.R. China
lzcarl@gmail.com, ywliang@whu.edu.cn, wuzejun@126.com, nadinetan@163.com

ABSTRACT
Detecting unknown virus is a challenging task. Most of the current virus detection approaches, such as anti-virus tools, require precognition of virus signatures for detection, but they are hard to detect unknown virus. In this paper, we present a new immunity based virus detection approach. This approach collects arguments of process calls instead of the sequence of process, which obtain more information of process, and then utilizes them to train detectors with Real-valued Negative Selection (RVNS) algorithm. In the stage of testing, user feedback is analyzed to adjust the threshold between normal files and viruses. We took two experiments to evaluate the performance of the approach, and the detection rate achieved is 0.7, which proved this approach could cope with unknown virus.

Keywords
Artificial Immune System, Real-valued Negative Selection, Process Call Arguments, Virus Detection, User Feedback

1. INTRODUCTION
As computer virus spreads faster and threatens computer system more seriously than ever before, how to detect virus is researched intensively. Classical virus detection approaches aim to find signatures of known viruses, that are the characteristics of infected files. However, it is hard for these approaches to cope with unknown viruses presenting new signatures and with viruses obfuscating their signatures.

The problems found in computer systems are quite similar to ones in Human Immune System (HIS). When HIS is attacked by unknown viruses, it will adaptively produce detectors and kill these viruses. Inspired by HIS, Artificial Immune System (AIS) [1] is considered as one of the new methods to defeat spreading computer viruses. Among all the AIS models, Self-nonself model is commonly adopted to detect virus. In addition, virus behaviors are considered by AIS as the detecting objects other than virus signatures. Previous virus experiments based on AIS against unknown viruses demonstrate that AIS is capable to find these viruses.

However, mainly focusing on the sequence of process calls and regarding them as the embodied virus behaviors, former virus detecting approaches with AIS are incapable of gathering enough information of virus behaviors. Same sequences of process calls might cause entirely different result due to different arguments. Moreover, when detectors are formed after training stage, they are limited to adjust to actual test result. Therefore, their performance will not be improved in the practical use.

In this study, we present a framework using immunity based virus detection with process call arguments and user feedback aiming to detect unknown virus accurately and adaptively. The main contributions of this work include collecting process call arguments for training data and introducing user feedback to the testing stage. In the training stage, the arguments of normal process calls are used to train detectors by Real-valued Negative Selection (RVNS) algorithm. In the testing stage, normal samples and abnormal samples are tested, and the threshold between normal files and viruses is adjusted by user feedback. Differed from previous approaches adopting Linux platform for experiments, we choose Windows XP platform that are used more common.

Related works and Background knowledge are introduced in next chapter. In Chapter 3, the proposed approach is described. At last, the experiments based on Windows XP platform and future work are discussed in Chapter 4 and Chapter 5.

2 BACKGROUND

2.1 Signature based Virus Detection
"Virus is a program that can infect other programs by modifying them to include a possibly evolved copy of itself."[21] Currently, the spreading of virus has caused innumerous loss. Only in the first half of year 2007,

Bionetics'07, December 10-13, 2007, Budapest, Hungary.

111,474 new samples of virus have been found, which had infected 75,967,19 computers in China.[22] To respond to this severe situation, main computer security companies have released their anti-virus software. These software mainly adopt classic signature based virus detection method to detect virus.

Former signature based virus detection approach looks for the existence of special files or codes, and uses a classifier to distinguish the virus files from normal files [9][23][24]. Through examining the signature, a virus is identified uniquely by anti-virus software. The detection rate of known virus is acceptable of this approach. However, this method does not perform well when detecting unknown virus since the signatures of these virus have not been stored in the signature base yet. To keep up with the increasing number of viruses, anti-virus software have to update to the newest signature base frequently, which cost time and bandwidth. Moreover, new techniques are employed by virus to escape from detection such as code obfuscation and self-evolving. It means that one virus might have several signatures. Thus, the size of signature base would increase gradually and finding a signature of one virus would take more time.

2.2 Artificial Immune based Virus Detection

In order to overcome these disadvantages of signature based virus detection, Artificial Immune based virus detection is proposed recently for detecting computer virus.

Among various mechanisms in the biological immune system that are explored by AISs, negative selection, immune network model and clonal selection are still the most important mechanisms. In this paper, we focus on Negative Selection models [11][12].

Negative Selection is a mechanism to train detectors based on the self/non-self discrimination principal in immune system. After this process, tolerant detectors are generated, and they can detect unknown antigens, which fail to react to detectors correctly.

The algorithm contains two steps. At first step, the training step, normal self samples are changed into n-dimensional points, and the algorithm receives them as the input. The detectors are trained to cover nonself space while do not intersect with self space and other detectors. When a detector is mature (go through given generations and still suit for the condition), it is removed from the population, and can be used to detect antigens. This iterative process is finished by generating sufficient detectors to cover given portion of non-self space or reaching given generations. In second step, the testing step, detectors from first step are used to classify samples with normal ones and abnormal ones.

Forrest first proposed a method for change detection using negative selection algorithm, and it aimed to build an intrusion detection system based on the notion of self within a computer system [2].

Building on previous work by the Forrest, A universal architecture of AIS (ARTIS) is proposed by Hofmeyr et.[3][4][5]. Concepts and mechanisms of HIS are implemented in ARTIS such as self-nonself, detectors, self-tolerance, clonal selection. Since then, many AIS based applications adopted ARTIS as their architecture.

The Artificial Immune based virus detection system described by Kephart from IBM focused on the automatic detection of computer viruses and worms [6]. Unlike ARTIS, in purpose of saving time and increasing efficiency, it does not utilize all the mechanisms of HIS, and techniques like blueprint [7] and trap [8] are included in the system.

Currently, there are some researchers combine AIS and signature to build their Artificial Immune based virus detection systems. AIVDS proposed by Hyungjoon Lee extracts signatures from normal files as self in order to train detectors, and signatures are fixed length strings from the head of files. Files infected by viruses should represent different signatures and can be detected [9].

3 APPROACH FOR VIRUS DETECTION

3.1 algorithm process

In this work, a Real-valued Negative Selection (RVNS) algorithm [13] based virus detection approach is implemented. Figure 1 illustrates the algorithm with four stages. In the data input stage, self samples are collected and processed. The process monitor collects value of given arguments of the file operaions done by process, and converts them into self samples for training. We choose real-valued type other than traditionally used binary type to present self samples. Due to the wide range of each parameter and large space for a long binary to code a large value, binary type is deficient in this case.

At the detectors generating stage, all of the self samples are considered as hypersphere in Rn space. Affinity of a detector and an antigen is judged by membership function, which establishes whether a point lies in the shape. This function depends on the shape of detectors, such as hypersphere, and the distance measure. By means of Monte Carlo algorithm, the fitness of detectors can be readily calculated and thus detectors covering sufficient space are produced.

At the testing stage, we first set the threshold - the rate of abnormal file operations to all the file operations- and then evaluate the algorithm with normal and abnormal samples. This stage integrates user feedback to regulate the threshold dynamically in order to detect various virus in real environment. Accordingly, the detection rate is improved and the false alarm rate is reduced.

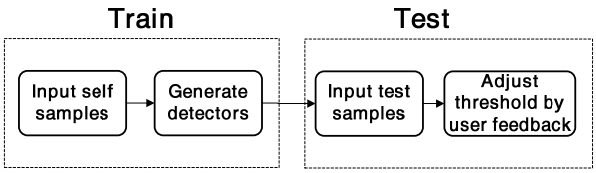

Figure 1. Algorithm process

3.2 Transform of operations

When a host PC is infected by computer virus, virus will try to gain control of the host and commit different kinds of destructions. As the differences of architectures and implementations between various operating systems, general behaviors of viruses vary greatly. Virus behaviors on Windows XP platform are shown as following.

Modify Registry: Add startup registry key at the place such as Run, Runonce, RunonceEx, RunServices. When system starts, virus executes by itself.

Hijack Files: When host is infected, the virus replaces certain content of system files or inserts itself into certain system files. After that, the virus code will be executed before the original code when operating system accesses the file.

Copy Files: Virus copies files it self-extracts to the system folders. Therefore, it can hide the executives.

Modify System Files: Autorun.exe, win.ini, system.ini are vulnerable and are often modified by virus. By adding the commands of virus, malicious code is executed when host starts up.

Between all these operations, manipulations of files are the most frequently operations of one application, which can indicate whether an application is malicious. Nevertheless, even if the APIs called by one application are the same, the arguments of the calls differentiate. Take Word of Microsoft co. as an example: this application also modifies registry and system files, though it is a normal application. Thus, we cannot simply rely on finding out suspicious sequences to distinguish virus from normal applications. This paper takes into account arguments of process calls as the important characteristics of viruses behaviors, and regards the file operations as the monitoring objects [15].

In one file operation, there are six pivotal arguments: operation kind, path, file name, success or not, parameter 1, parameter 2 (referred to FileMon [16]). These attributions determine the effect of one file operation. Table 1 shows the fields and descriptions.

Table 1. Details of attributions

Attributions	Field	Description
Operation Kind	{Create, Delete, Write}	The action to one file
Path	String type	The directory of the manipulated file
File Name	String type	The name of the manipulated file
Success or Not	Boolean type	If one file operation is successful, the result is true, else false.
Parameter 1	Integer type	When operation kind is Create, it is the options （Create, Overwrite）. When operation kind is Writing, it is the offset to the file head.
Parameter 2	Integer type	When operation kind is Create, it is the access method (Read, Write, etc.). When operation kind is Writing, it is the length of file writing.

The passages below describe the concrete meaning of Table 1.

Path and filename determine the position of one file. As normal applications hardly modify or delete the system files, they can be important signatures of one application.

Parameter 1 and Parameter 2 reflect the manner one application manipulates the file. Normal applications rarely modify or overwrite specific areas of one file. Hence, they can be other useful elements.

Success or Not is the result of file operation. The inexistence of one file or the restriction to access one file will lead to the failure of one operation. It can indicate whether an operation is legitimate.

Each attribution should be changed into real value type ranged in $[0,1]^n$ as the input of training stage. First, the attributions are changed into Integer type, and then normalized to real-valued type. The technique is described below.

The field of Operation Kind includes Create, Delete, Write, which can be marked as 1, 2, 3.

Path and file name are string type, we sum up the ASCII value of characters included in these two attributions to obtain the integer value. Let c1, c2, c3… cn be the ASCII value of the characters of one string, the value is $\sum_{i=1}^{N} c_i$

Success (true) or not (false) is Boolean type, and are cast into 1 and 0.

Parameter 1 and Parameter 2 store the call parameters of Create and Write. They are already integer type, so they are not changed. Since Delete does not have arguments, the value of parameter 1 and parameter 2 are 0 for Delete.

Then values are normalized to real-valued type ranged in $[0,1]^n$. We denote maxp as the maximal value of one attribute in operation set, and minp as the minimal value of one attribute in operation set, and p is the value of one operation needs to be transformed, and pn is the new value, which is defined as:

pn=(p-minp)/(maxp-minp)

If maxp and minp are the same value, an extra value will add to maxp to avoid divide-zero error.

3.3 Algorithm for detectors generation

The approach developed in this paper uses RVNS to train as many detectors with high fitness as possible. The following sections describe the details of RVNS such as representations of self/nonself, detectors, fitness calculation , generic operators.

3.3.1 Detector Representation

The file operations are cast into points in $[0,1]^n$ after the attributions are extracted and transformed. Likewise, detectors should be represented in $[0,1]^n$. In this approach, we choose hyper-sphere with an n-dimensional center and a radius as the form of detectors.

Besides the representation of one individual, to judge whether a point is in the detector needs a membership function. We choose Minkowski distance function to compute the distance of a point and a detector. Minkowski distance between point x and y is:

$$dist(c,x) = (\sum |c_i - x_i|^n)^{1/n}$$

A point lies in a detector if the distance is less than the radius of the detector.

3.3.2 Fitness Calculation

Fitness is the evaluation of the quality of one individual. Detectors with higher fitness would be selected. As detectors are hyper-spheres in $[0,1]^n$, the fitness of one detector is high when it covers large non-self space and overlaps small space with other detectors. Meanwhile, the detector must have no intersect with self points. The fitness function of detector D is defined as follows:

fitness(D)=effective-coverage(D)-C(m)*m

The effective coverage is the volume that one detector covered while not yet covered by other detectors. Let V(D) be the volume of detector D and OL(D) be the overlap between D and other detectors, then fitness(D)=V(D)-OL(D). The volume of a 2k or 2k+1 dimensional hyper-sphere of radius r is

$$V_{2k}(r) = \frac{\pi^k}{k!} r^{2k} \quad \text{or} \quad V_{2k+1}(r) = \frac{k!}{(2k+1)!} 2^{2k+1} \pi^k r^{2k+1}$$

.

However, the overlap between detectors is not easy to calculate. As calculating the intersect volume of two n-dimensional shapes is difficult. Moreover, the arithmetic complexity of geometry overlap calculating algorithm rises sharply when the dimension of space increases.

In this work, Monte Carlo technique, a well-established calculation technique, is used to estimate the volume of overlap. To compute the volume, a set of N random points uniformly distributed in $[0,1]^n$ are generated. Sump is defined as the number of the points which lie both in the areas of detector A and B. Then the volume of the overlap is sump/N*1. The arithmetic complexity of volume calculating is O(N), and the cost is relatively smaller consequently.

When a detector intersects with self points, the fitness subtract m * C(m). m is the self points that lie in the area of detectors. C(m) is the penalty function that prevent detectors from covering self points.

3.3.3 Genetic Algorithm

Figure 2 shows the genetic algorithm used to train detectors. We use Roulette Wheel Selection to select best individual, two-point crossover and bit-flip mutation f crossover and mutation. When the coverage reaches the desired coverage or desired number of detectors have been generated, the algorithm terminates. After the training, a certain number of detectors go into testing stage.

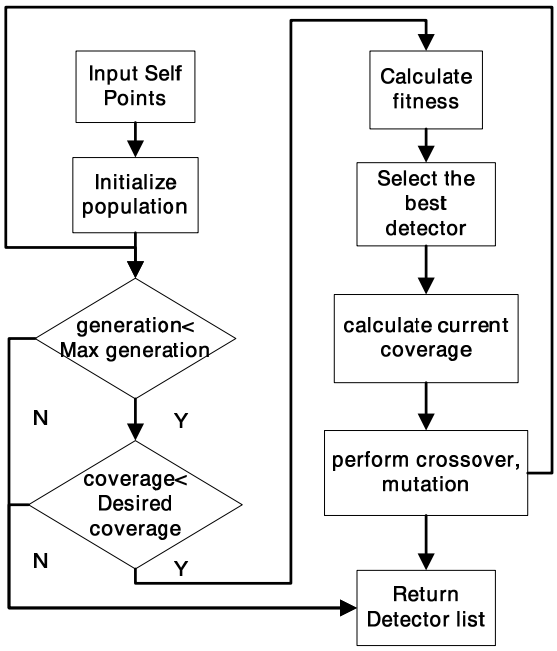

Figure 2. Process of Algorithm

3.4 User Feedback

The objective of user feedback in testing stage is to improve the detection rate and reduce false rate. We define the rate of abnormal file operations to all the file operations as the threshold between normal files and virus. Before testing, a given threshold is set by previous knowledge, and the rate of abnormal file operations to all the file operations is calculated when a sample comes into the testing stage. A test sample is malicious only if the rate is bigger than the threshold. Since the distribution of the rate of normal files and virus vary greatly, rightly adjusted threshold would enable detectors to perform more accurately. When carrying out one test, the threshold is moderated based on user's opinion about previous result and current result. Users are not required to give judgment to all the detections result but just several.

In the real environment, when a detection event happens, system pops up a dialog to inform user and asks user whether this result is correct. User estimates the result based on the information such as infected file and suspicious process delivered by the system. And the threshold will be adjusted based on user response.

There are three kinds of feedback given by user: correct, false positive of virus and false negative of normal files. Different methods of threshold calculation according to three scenes are described as follows:

A. Scene 1 (correct):

There is no need to modify threshold for a correct result, and the threshold is fixed.

B. Scene 2 (false positive of virus)

Let th be the threshold and p be the rate of the virus test case. According to previous experiments, the rate of a normal file is likely to be less than the threshold and the rate of virus is likely to be bigger than the threshold. Thus p is less than th and the new threshold th`=th-dt, (dt>0).

Denote N as the number of all the tested normal cases, n as the number of cases whose rates are between p and th, then:

dt=th-p n=0,

dt=(1-n/N)*(th-p)*M n>0

M is a constant value, which decides the level of threshold change.

C. Scene 3 (false negative of normal files)

th and p are defined the same as Scene 2. But in this circs, p is bigger than th and the new threshold th'=th+dt, (dt>0).

Denote N as the number of the entire tested virus cases, n as the number of cases whose rate is between th and p, then:

dt=p-th n=0,

dt=(1-n/N)*(p-th)*L n>0

L also decides the level of threshold change. Nonetheless, M is relatively bigger as the problem brought by false positive of virus is more severe than false negative of normal files. Based on previous experiments, M and L are set to 1.1 and 0.9.

4 EXPERIMENTS

The primary goal of the experiments is to test whether this approach can detect virus while does not mistake the normal application for virus. We chose Macro Virus as detect object, since the behaviors of their host application are easy to capture. "Macro viruses, as the name suggests, are designed to add their code to the macros associated with documents, spreadsheets and other data files. " [19]. As the approach aimed to run on Windows XP platform, and "the vast majority of macro viruses were designed to spread on the back of Microsoft® Office data files" [19]. We selected Word (with most Macro virus cases) as the monitoring object. Generally, Macro Virus infects document formatted by Word and damages files through Word, Excel, etc.

4.1 Experimental Arguments

The parameters used in the experiments are listed as following: crossover rate = 0.8; mutation rate = 0.8; max generations = 150; population size = 40; desired coverage=0.99; self threshold = 0.1. In the training stage, the terminating condition is that maximum generation reaches 150 or coverage reaches 85%.

4.2 Data preparation

To capture the file operations of one application, we use FileMon[16] as system monitor and collector. This tool can capture all file system activities at real-time. We confined the monitoring scope to process named WINWORD. Meanwhile, the operation kind must be create, delete and write. The operations are saved to a log file.

Since the coverage of a ball in a given area with the same radius reduces when dimension increased, many detectors are needed in order to cover six dimensions space, which is inefficient and costs time. So the dimensions are eliminated to three with the combination of different dimensions. "Operation Kind" and "Success or Not", "Path" and "File", "Parameter 1" and "Parameter 2" are combined separately. Let a be the value of the first argument and b be the value of second argument and nb is the number of digits of b, then the combined value is a*10nb+b.

Another problem is that different file names would influence the result of test, as file name is also an attribution. The file name in each train or test case is set to a constant value. Path is similar processed with file name.

For example, there is one operation with six attributes (CREATE, C:\DOCUME~1\lizhou\LOCALS~1\Temp\~DFD485.tmp, ~DFD485.tmp, S, CREATE, 0013019F) as (Operation Kind, Path, File Name, Success or Not, Parameter 1, Parameter 2), and the filename is changed into A. After the process of integer presentation, normalization and attributes combination, one self point is generated with three real-valued parameters (0.009900990099, 0.038940400873, 0.26787898835).

We used three sample sets to carry out the experiments. The first set is the file operations of normal document, which we used to train detectors.

Table 2 shows the specific training cases. We list ten cases, and each case contains normal file operations. As other operations such as modify the document only modify the data in memory, the case only contains these operations are not used to test.

Table 3 shows the virus test cases. We obtained the entire virus samples from VX Heaven [20]. In addition, we select the virus described in Viruslist [19]. Before starting test, we open the virus infected documents, create another document and save it. Each created document (also infected) is used to test.

Table 4 shows the normal test sets. There are five cases to test. Since the content and structure of one document does not have any impact on the test result. We prepare four normal cases with Macro and one case without Macro.

Table 2. Specific training cases.

Case No	Description
1	Open normal document without Marco.
2	Open normal document with Marco.
3	Save normal file already opened.
4	Save document as template.
5	Run Word.
6	Create a document.
7	Close document in Word.
8	Exit Word.
9	Edit document in Word.
10	Save document newly created

Table 3. List of Macro Virus.

Case No	Virus Name
1	Virus.MSWord.Beast
2	Virus.MSWord.Dub
3	Virus.Multi.Cocaine
4	Virus.MSWord.Inexist.b
5	Virus.MSWord.Mentes
6	Virus.MSWord.Mimir
7	Virus.MSWord.Natas
8	Virus.MSWord.Outlaw
9	Virus.MSOffice.Shiver
10	Virus.MSWord.Titasic

Table 4. Testing cases of the file operations of normal documents

Case No	Description
1, 2	Open normal document without Marco.
3, 4	Open normal document without Macro and save it.
5	Open normal document with Macro and save it.

4.3 Result

After 10 times training, the best training result is the detector list including 150 detectors with coverage 0.87575. This detector list was used for test.

To evaluate the improvement that user feedback takes on the test result. We carried out experiments with or without user feedback.

A. Experiment 1

This experiment is free of user feedback and Table 5 shows the testing result for Macro Virus cases. The threshold of ratio of nonself points to all the points is fixed to 0.15, and the cases above that threshold are abnormal. The detection rate is 0.5, while the false positive rate is 1. Half of the

cases are not detected, and their results are between 0.05 and 0.15, which is much smaller than the average ratio of abnormal cases.

Table 5. Testing result for Macro Virus cases

Case No	Total points	Nonself points	Ratio	Result (ratio=0.15)
1	5497	5402	0.982	Abnormal
2	35	4	0.114	Normal
3	5681	5449	0.959	Abnormal
4	5454	5446	0.998	Abnormal
5	121	2	0.016	Normal
6	64	9	0.141	Normal
7	5496	5446	0.991	Abnormal
8	12	1	0.083	Normal
9	79	4	0.051	Normal
10	5681	5461	0.961	Abnormal

Table 6 shows the test result for normal cases, the ratio is also 0. 08. No normal case is detected and false negative rate for normal cases is 0.

Table 6. Testing result for normal cases

Case No	Total points	Nonself points	Ratio	Result (ratio=0. 08)
1	49	0	0	Normal
2	16	0	0	Normal
3	61	4	0.066	Normal
4	56	4	0.071	Normal
5	19	0	0	Normal

B. Experiment 2

This experiment relies on user feedback. Similar to Experiment 1, we used cases described in Table 2 for detectors training. In the testing stage, we still used cases described in Table 3 and Table 4. However, threshold is adjusted in six virus samples and three normal samples. The rest our virus samples and two normal samples are tested with the adjusted threshold. Every Two virus samples are followed by one normal sample.

Table 7 shows the threshold change towards each test case. The threshold is set to 0.15 initially. When a case goes through testing, the threshold is adjusted. After testing V1, N2 and V6, the threshold is modified as error detection happens.

Table 7. Testing result for cases to adjust threshold

Case No	Ratio	Adjusted Threshold	Result
V1	0.982	0.15	Abnormal
V2	0.114	0.114	Normal
N1	0	0.114	Normal
V3	0.959	0.114	Abnormal
V4	0.998	0.114	Abnormal
N2	0	0.114	Normal
V5	0.016	0.016	Normal
V6	0.141	0.016	Abnormal
N3	0.066	0.066	Abnormal

(V= virus, N= Normal)

The final threshold is 0.066 after training with user feedback.

Six cases are tested with adjusted threshold. Table 8 shows the detection result for these samples.

Table 8. Testing result for cases with adjusted threshold

Case No	Ratio	Result (ratio=0.15)
V7	0.991	Abnormal
V8	0.083	Abnormal
N5	0.071	Abnormal
V9	0.083	Normal
V10	0.961	Abnormal
N6	0	Normal

(V= virus, N= Normal)

After the two-stage test, the detection rate is 0.7. False positive rate for Macro Virus cases is 0.43 and false negative rate for normal cases is 0.66. The detection rate improves by 0.2 while the false positive rate drops by 0.57 and false negative rate rises by 0.66.

C. Result Analysis

The detection rate and false negative rate are relatively higher and false positive rate is relatively lower in Experiment 2. We can reach conclusions from the result above:

- Adjusting threshold with user feedback is able to provide higher detection accuracy but might increase the possibility of false negative of normal files. As detecting virus is more important to user, the result in Experiment 2 is acceptable.
- When the number of points is high, it is very likely that virus infects a file.
- When opening a normal document, the operations are limited to several kinds. However, at the time saving a

modified document, a large number of different "Write" operations would influence the testing result.

Though over half of the cases of Macro virus are detected, there are some cases not detected. To enhance the system performance, modifying the parameters for training algorithm, obtaining more samples of Macro virus, adjusting the initial threshold ratio might improve the system's performance.

5 CONCLUSION

This paper presents an approach for detecting virus with user feedback. In the training stage, we implement the algorithm for training detectors based on RVNS. In the testing stage, we utilize user feedback to adjust threshold. Finally, we carry out experiments aimed at detecting Macro virus. In addition, the experiments show that this approach can detect virus while avoid mistaking normal applications for virus. Preknowledge of the specific virus and specific application is not required.

Current work mainly focuses on detecting virus from the log file of the operations done by virus. To fulfill the requirement for detecting virus at real time, it needs to monitor the application when it runs, and the efficiency of the algorithm needs to be considered. Multi-shaped detectors [14] can be used to improve the detection rate and reduce the false negative rate and false positive rate. Other operations such as operations to Registry would be included in the future development of the approach to collect more information of an application. Finally, though user's feedback only plays an important role in threshold adjusting in the current model, we will consider the detectors' generation with user feedback in next step.

REFERENCES

[1] Julie Greensmith, and Jamie Twycross. *Immune System Approaches to Intrusion Detection – A Review ,* ICARIS 2004, LNCS 3239, pp. 316–329, 2004.

[2] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri. Selfnonself discrimination in a computer. *In Proceedings of the 1994 IEEE Symposium on Security and Privacy, page 202*. IEEE Computer Society, 1994.

[3] S. Hofmeyr. *An Immunological Model of Distributed Detection and its Application to Computer Security.* Ph.D. dissertation, Univ. New Mexico, 1999.

[4] S. Hofmeyr and S. Forrest. Immunity by design: an artificial immune system. *Proc. of Genetic Evolutionary Computation Conf.* San Francisco, CA, 1999.

[5] S. Hofmeyr, S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation,* 2000, 8(4):443–473.

[6] J Kephart. A biologically inspired immune system for computers. *In Proceedings of the Fourth International Workshop on Synthesis and Simulation of Living Systems, Artificial Life IV,* pages 130–139, 1994.

[7] J.O. Kephart and W.C. Arnold. Automatic extraction of computer virus signatures. *Proceedings of the Fourth International Virus Bulletin Conference*, St. Helier, Jersey, UK, 1994.

[8] J.O. Kephart, Gregory B. Sorkin, Morton Swimmer, and Steve R. White. *Blueprint for a Computer Immune System. Proceedings of the 1997 International Virus Bulletin Conference*, San Francisco, California, October, 1997.

[9] Hyungjoon Lee, Wonil Kim, Manpyo Hong. Biologically Inspired Computer Virus Detection System. *BioADIT 2004*, LNCS 3141, pp. 153–165, 2004.

[10] Gaurav Tandon and Philip Chan. *Learning Rules from System Call Arguments and Sequences for Anomaly Detection*. Department of Computer Sciences Technical Report CS-2003-20, Florida Institute of Technology, Melbourne, FL, 2003.

[11] E. Hart and P. Ross. Exploiting the analogy between immunology and sparse distributed memories. *Proceedings of the First International Conference on ICARIS*, 2002.

[12] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri. *Self-Nonself Discrimination in a Computer. In Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy,* Los Alamitos, CA: IEEE Computer Society Press, 1994.

[13] F. Gonzalez, D. Dasgupta. Anomaly detection using real-valued negative selection. *Journal of Genetic Programming and Evolvable Machines*, Vol. 4, 2003, 383-403.

[14] Sankalp Balachandran, Dipankar Dasgupta, Fernando Nino, Deon Garrett. *A General Framework for Evolving Multi-Shaped Detectors in Negative Selection,* In the proceedings of IEEE Symposium Series on Computational Intelligence, April 1-5, 2007.

[15] Roberto Battistoni, Emanuele Gabrielli, and Luigi V. Mancini, A Host Intrusion Prevention System for Windows Operating Systems*. ESORICS 2004, LNCS 3193, pp. 352–368, 2004.*

[16]*http://www.microsoft.com/technet/sysinternals/default.mspx* Sysinternal FileMon.

[17] *Function Premnmx.* Matlab Help.

[18] Oak Ridge National Laboratory. *Introduction to Monte Carlo method.* Computational Science Education project, 1995.

[19] http://www.viruslist.com, *Viruslist*

[20] http://vx.netlux.org/, *VX Heaven,*

[21] Cohen, F. Computer Viruses: Theory and Experiments. *Computers and Security,* 6 (1987) 22 – 35.

[22]http://news.duba.net/report/dbhd/2007/07/04/110797.shtml, *virus report by Kingsoft, 2007*

[23] 4. Gryaznov. D. Scanners of the Year 2000: Heuristics. *In Proceedings of the 5th Virus Bulletin International conference,* Boston, Massachusetts (1999) 225–234.

[24] Schultz, M.G., Eskin, E., Zadok, E., Stolfo, S.J. Data Mining Methods for Detection of New Malicious Executables. *In Proceedings of the 2001 IEEE Symposium on Security and Privacy*, Oakland, California (2001) 38–49