

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. Current virus detection approaches greatly require preknowledge of virus signatures, such as anti-virus tools. In this paper, we present a new immunity based virus detection approach. This approach collects arguments of process calls instead of sequence of process which obtain more information of process, and 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 2 experiments to evaluate the performance of the approach, and the detection rate achieved is 0.7 which proved this approach can cope with virus.

Keywords

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

1. Introduction

As computer virus is spreading faster and seriously threatens computer system than ever before, how to detect virus is researched intensively. Classic virus detection technologies aim to find signatures of viruses which can be defined as the characteristics of infected files. But it is hard to cope with unknown viruses owning new signatures and with viruses which obfuscate their signatures.

The problems found in computer systems are quite similar to ones in Human Immune System (HIS). When encountered with unknown viruses, HIS can adaptively produce detectors and kill these viruses. Inspired by HIS, Artificial Immune System (AIS) [1] is considered as a new method to defeat spreading viruses. Self-nonsel model is utilized to detect virus behaviors which are more constant than virus signatures. Virus experiments based on AIS against unknown viruses demonstrate that AIS can tackle these viruses.

However, mainly focusing on the sequence of process calls, classic virus detecting approaches with AIS are far from gathering enough information of virus behaviors. Same sequence of process calls might cause totally different effect due to different arguments. Moreover, when detectors are formed after training stage, they can not adjust

to actual test result. Therefore, their performance will not be improved.

In this study, we present the architecture and algorithm of Immunity based virus detection with process call arguments and user feedback aiming to detect unknown virus accurately and adaptively. The main contributions of our work include collecting process call arguments as training data and add user feedback to the testing stage to amend the attributions of detectors. 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 examined, and the threshold between normal files and viruses is moderated by user feedback. Differed from previous approaches adopt 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 innumerable loss. Only in the first half of year 2007, 111,474 new samples of virus have been found, which had infected 75,967,19 computers in China.[22] To reply to this severe situation, main computer security companies have released their anti-virus software. These software mainly rely on classic signature based virus detection that detects new virus.

Classic signature based virus detection look for the existence of sequence of process calls, and use a classifier to distinguish the virus files from normal files. Through extracting the signature, a virus is identified uniquely by anti-virus software. With this method, the detection rate of known virus is acceptable. However, this method does not

perform well to detect unknown virus as the signature of these viruses have not been stored in the signature base. To keep up with the increasing number of viruses, anti-virus software have to frequently update to newest signature base which cost time and band-width. Moreover, new techniques are employed by virus to escape from detection such as code obfuscation and self-evolving. It means that one virus might generate several signatures. Thus, the size of signature base would increase gradually and find 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 detection of 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 the process, tolerant detectors are provided, and they detect unknown antigens which fail to react to detectors correctly.

The algorithm contains two steps. In first step. First, the training step, normal samples of self are represented as 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 reached 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 is proposed by Hofmeyr et.[3][4][5]. Concepts and mechanisms of HIS are represented 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 as well. 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

Figure 1 illustrates the algorithm with 4 stages. In the data input stage, self samples are collected and processed. The process monitor collects all arguments of the file operations done by process, and converts the arguments to 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 value of each parameter, binary type appears deficient as coding a large value needs a long binary.

In the detectors generating stage, the Real-valued Negative Selection (RVNS) [13] based training algorithm is implemented. All the self samples are considered as hypersphere in R_n space. Affinity of detectors and antigens are 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.

In 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 the user feedback to dynamically regulate the threshold and adjust the threshold to detect different in real environment. Accordingly, the detection rate is improved and the false alarm rate is reduced.

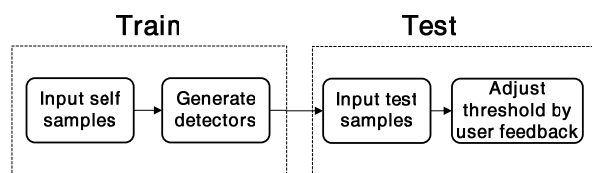


Figure 1. Algorithm process

3.2 The representation 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 are sorted. Virus behaviors on Windows XP platform are shown as following.

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

Hijack File: When host is infected, the virus replaces certain part of system files or inserts itself into certain system files. After that, the virus code will be executed before the application code when operating system accesses the file, and then the right of execution will be left to application.

Copy Files: Virus copies files which it self-extracts to the system folders. So 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, it will be executed when host starts up.

Between all these operations, manipulations of files are the basic 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 are always different. Take WORD of Microsoft co. as an example: this application also modifies registry and system files. So we can not simply rely on finding out suspicious sequences to distinguish viruses from normal applications. This paper takes into account arguments of process calls as the important characteristics of viruses behaviors, and treats the file operations as the monitoring objects and the arguments as the elements of one operation [15] [16].

In one file operation, there are six pivotal arguments: operation kind, path, file name, success or not, parameter 1, parameter 2. These attributions can determine the effect of one file operation. The field and description are shown in Table 1.

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 file manipulated
File Name	String type	The name of the file manipulated
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.

Path and filename determine the position of one file. As some system files are hardly modified or deleted by normal application, they can be important signatures of one application.

Parameter 1 and Parameter 2 reflect the manner one application manipulates the file. Specific areas of one file are rarely modified by normal applications and some files can not even be overwritten. Hence, they can be other accessory 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 RVNS. First, the attributions are changed into Integer type and then normalized to real value type.

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

Path and file name are string type, and summing up the ASCII value of characters included in these two attributions will obtain the integer value. Let $c_1, c_2, c_3, \dots, c_n$ be the ASCII value of the characters of one string, the value is

$$\sum_{i=1}^N c_i$$

Success or not is Boolean type, true and false can be cast to 1 and 0.

Parameter 1 and Parameter 2 of Create and Write are already integer type, so they do not need to be changed. Since Delete does not have arguments, the value of parameter 1 and parameter 2 are 0 for Delete.

Then values are changed into real value 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 defined as:

$$pn=(p-\text{minp})/(\text{maxp}-\text{minp})$$

If maxp and minp are the same value, maxp should be added an extra value. The attributes are cast into the field of $[0,1]^n$ after the normalization.

3.3 Algorithm for detectors generation

The approach developed in this paper uses RVNS to train the detectors that can cover as large nonself space as possible. The details of RVNS such as representations of self, nonself, detectors and fitness calculation and generic operators will be described in the following sections.

3.3.1 Detector Representation

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

Besides the representation of one individual, a membership function is also needed to judge whether a point is in the detector. Minkowski distance is used to represent the distance of point and detector. A point is in detector if the distance is less than the radius of detector. Minkowski distance between point x and y is:

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

3.3.2 Fitness Calculation

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

$$\text{fitness}(D)=\text{effective-coverage}(D)-C(m)*m$$

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 $\text{fitness}(D)=V(D)-OL(D)$. The volume of an 2k or 2k+1 dimensional hyper-

sphere of radius r is $V_{2k}(r) = \frac{\pi^k}{k!} r^{2k}$ or $V_{2k+1}(r) = \frac{k!}{(2k+1)!} 2^{2k+1} \pi^k r^{2k+1}$

However, the overlap between detectors is not easily calculated. Traditional AISs commonly choose Euclidean distance as a key factor to calculate the distance which is feasible with binary space. There is no problem with calculating the overlap between n-dimensional shapes. Moreover, the arithmetic complexity of geometry overlap calculating algorithm rises sharply when the dimension of space increases.

In order to evaluate the overlap, Monte Carlo technique [18] is used. 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, and the volume of the overlap is $\text{sump}/N*1$. OL (D) can be calculated between D and other detectors using Monte Carlo technique. The arithmetic complexity of overlap calculating is $O(N)$, and the cost is relatively smaller consequently.

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

The genetic algorithm utilized to train detectors is shown below. When the coverage reaches the desired coverage and desired generation is obtained, the algorithm would be terminated. Roulette Wheel Selection is used to select best individual, two-point crossover and bit-flip mutation are used in crossover and mutation. After the training, certain number of detectors will be generated and can be used in the testing stage.

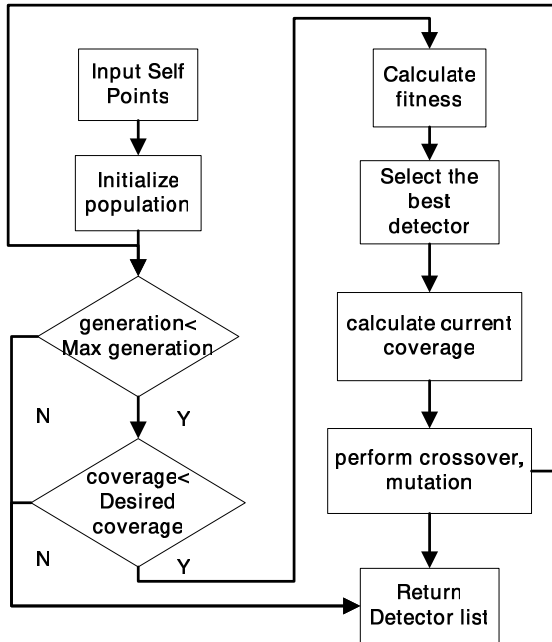


Figure 2. Process of Algorithm

3.4 User Feedback

The objective of user feedback in testing stage is to improve the detection rate and reduce false alarm rate. The threshold of normal files and viruses is defined as the rate of abnormal file operations to all the file operations. Before testing, a given threshold is set by previous knowledge, the rate of abnormal file operations to all the file operations is calculated when a test sample taken into account. Since the distribution of the rate between normal files and virus varied greatly, adjusted threshold would enable detectors to perform more accurately. When one detection is carried out, the threshold is moderated based on user's opinion about previous result and current result. Users are not required to give opinions to all the detections result but just several detections.

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 to the 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:

$$dt = th - p \quad n = 0,$$

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

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

C. Scene 3 (false negative of normal files)

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

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

$$dt = p - th \quad n = 0,$$

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

L has a similar function as M . 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 to test whether this approach takes effect as the behaviors of the 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]. Word of Microsoft® Office is selected as the monitoring object. The infected document of Word is the media of Macro Virus, and damage is usually done through applications such as Word, Excel, etc. So the behaviors can be readily captured and presented.

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 monitor to one application. This tool can capture all file system activities in real-time. The monitoring scope is confined in process named WINWORD and the operation kind is create, delete and write. The operations are saved to a log file.

Three sets of data are collected to carry out the experiments. The first set is the file operations of normal document which we use to train detectors and specific training cases are shown in Table 2. The second set is the file operations of documents infected by different Macro Viruses which were found in VX Heaven [20], and the list of virus is shown in Table 3. The third set is the file operations of normal documents, details are shown in Table 4. The second and third sets are used to test.

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.

Since the coverage of a ball in a given area with the same radius will be reduced when dimension increased, lots of detectors are needed to be produced in order to cover 6 dimensions space, which is inefficient and costs time. So the dimensions were eliminated to 3 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 synthetic value is $a*10^{nb}+b$.

Another problem is that different file names would influence the result of test, as file name is also an attribution. Therefore, the file name in each case of train or test is set to be constant. Path should be treated similarly as 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).

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 if the ratio which is above 0.15 is abnormal.

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.

(V= virus, N= Normal)

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

The detection rate is 0.5. False positive rate for Macro Virus cases is 1 and false negative rate for normal cases is 0.

B. Experiment 2

This experiment relies on user feedback. In addition, we assign 6 virus samples and 3 normal samples for threshold adjusting with user feedback and 4 virus samples and 2 normal samples for test. Every Two virus samples are followed by one normal sample.

Table 7 shows the threshold change toward each test case. The threshold is set to 0.15 initially. When a case is tested, the threshold is adjusted.

Table 7. Testing result for Macro Virus cases

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)

Threshold is changed to 0.066 after training with user feedback.

Then 4 virus samples and 2 normal samples are used for test. Table 8 shows the detection result for these 6 samples.

Table 8. Testing result

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

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

C. Result Analysis

The detection rate and false negative rate are relatively higher and false positive rate is relatively lower in Experiment 2. Conclusions can be reached 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. Moreover, the adjusted threshold depends on the order of samples. Therefore, the sequence of samples has a great influence on the threshold.
- When the quantity of points is high, it is very likely that a file is infected by virus..
- When opening a normal document, the operations are limited to several kinds. However, at the time saving a document that has been modified, "Write" operations might be different which would influence the testing result.

Though above half cases of Macro Virus are detected, there are some cases not detected. To enhance the testing result, modifying the parameters for training algorithm to get higher coverage, obtaining more samples of Macro Virus, adjusting the initial threshold ratio might be useful.

5 Conclusion

This paper presents an approach for detecting virus with user feedback. The algorithm for training detectors is based on RVNS. In testing stage, we utilize user feedback to adjust threshold. Experiments aimed at detecting Macro Virus were carried out. And the experiments show that this approach can detect Macro Virus while avoid mistaking normal applications for virus without preknowledge of the specific virus and specific application.

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, user feedback play a role in threshold adjusting in the current model, the detectors' generation with user feedback will be considered in next step.

REFERENCES

- [1] Julie Greensmith, and Jamie Twycross, *Immune System Approaches to Intrusion Detection – A Review* Uwe Aickelin, School of Computer Science, University of Nottingham, UK ICARIS 2004, LNCS 3239, pp. 316–329, 2004.
- [2] Stephanie Forrest, Alan S. Perelson, Lawrence Allen, and Rajesh Cherukuri, *Self-nonsel self 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.
- [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-Nonsel Self 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*.
- [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.aspx> Sysinternal FileMon
- [17] *Function Premnmx*, Matlab Help.
- [18] Oak Ridge National Laboratory, *Computational Science Education project. Introduction to Monte Carlo methods*, 1995.
- [19] <http://www.viruslist.com>, *Viruslist*
- [20] <http://vx.netlux.org/>, *VX Heaven*,
- [21] Dr. Frederick B. Cohen. *Computer Viruses Theory and Experiments*
- [22] <http://news.duba.net/report/dbhd/2007/07/04/110797.shtml>, *virus report by Kingsoft, 2007*