Artificial Immune Clonal Selection Classification Algorithms for Classifying Malware and Benign Processes Using API Call Sequences

Khaled A. Al-Sheshtawi
Information Technology Dept., King AbdulAziz University, Jeddah, Kingdom of Saudi Arabia.

H. M. Abdul-Kader
Information System Dept., Faculty of Computers & Information, Menoufia University, Egypt.

Nabil A. Ismail
Computer Science Dept., Faculty of Computers & Information, Menoufia University, Egypt.

Abstract
Machine learning is an important field of artificial intelligence in which models are generated by extracting rules and functions from large datasets. Machine learning includes a diversity of methods and algorithms such as decision trees, lazy learning, k-nearest neighbors, Bayesian methods, Gaussian processes, artificial neural networks, support vector machines, kernel algorithms, and artificial immune systems (AIS). AIS are computation tools that emulate processes and mechanisms of the biological immune system. AIS use the learning, memory, and optimization capabilities of the immune system to develop computational algorithms for function optimization, pattern recognition, novelty detection, and process control, and classification. There are four main sub fields of research that have emerged in AIS cantered on prominent immunological theories; negative selection algorithms, immune network algorithms, danger theory algorithms, and clonal selection algorithms. In this paper, we will analyze API call sequence of a process to classify it as benign or malicious. We have collected API call traces of real malware and benign processes running on Windows operating system. We will employ eight commonly used clonal selection algorithms: AIRS1, AIRS2, AIRS2 Parallel, CLONALG, CSCA, IMMUNOS-1, IMMUNOS -81, and IMMUNOS -99. We evaluate the accuracy of these algorithms for classifying between malware and benign processes using API call sequences.

Key words:
Artificial immune clonal selection, API call sequence, malware.

1. Introduction

The sophisticated computer malware is becoming a serious threat to the information technology infrastructure, which is the backbone of modern e-commerce systems [1]. Among other malwares, computer worm is a self-replicating computer program. Computer worm is a fully stand-alone program that does not need to be a part of other program in order to propagate. The fact that worms propagate very fast on networks makes them one of the most challenging malwares to intercept. Fast detection of computers infected with worms is critically important on local networks. A recent outbreak of Conficker malware affected more than 9 million computers including those of Ministry of Defence, United Kingdom [2]. This incident has proved that commercial anti-virus software, even with updated malware definitions, are incapable of safeguarding our information technology infrastructure [3]. In [4], the authors have shown that commercial anti-virus software are easily befooled using evasion attempts, such as code obfuscation, encryption and polymorphic transformations. Therefore, security experts are now focusing their attention to robust run-time malware detection techniques that analyze API call sequence of a process to classify it as benign or malicious [3]. We focus on the application of clonal selection algorithms in the task of worm detection within the environment of Microsoft Windows® operating system.

The clonal selection theory quickly attracted the attention of computer scientists, since it appeared as a more flexible alternative to genetic and evolutionary algorithms. The clonal selection theory was originally proposed by Burnet, in order to explain the reinforcement learning of the immune system of mammals. According to Burnet's theory, mammals acquire immunity through mutation, selection, and proliferation of the mature B-lymphocytes. The clonal selection theory is used to explain the basic features of an adaptive immune response to an antigenic stimulus. It establishes the idea that only those cells that recognize the antigens are selected to proliferate. The selected cells are subject to an affinity maturation process, which improves their affinity to the selective antigens.

Inspired by the clonal selection theory, De Castro pioneered the clonal selection algorithm [5] in 2000. After that, many clonal selection based artificial immune algorithms have been proposed. The dynamic CSA (DynamiCS) constructed by Kim in 2002 [6] and polyclonal strategy proposed by Licheng Jiao in 2003 [7] are two of the most outstanding contributions. Lei Wang and Licheng Jiao introduced immune concepts and methods into evolutionary algorithm to form immune evolutionary algorithms [8]. The clonal selection algorithm is with a colony search mechanism in nature, which enables it not easily to get into the trap of local optimization, it can thus converge to the global optimization with a higher probability and higher speed. [9]. We used Weka [10] machine-learning workbench in our analysis.

Manuscript received April 5, 2010
Manuscript revised April 20, 2010

In this paper, we will analyze API call sequence of a process to classify it as benign or malicious. We have collected API call traces of real malware and benign processes running on Windows operating system. We will employ eight commonly used clonal selection algorithms: AIRS1, AIRS2, AIRS2 Parallel, CLONALG, CSCA, IMMUNOS-1, IMMUNOS -81, and IMMUNOS -99. We evaluate the accuracy of these algorithms for classifying between malware and benign processes using API call sequences.

The remainder of this paper is organized as follows. Section 2 provides basic principles of clonal selection theory. Section 3 provides basic principles of clonal selection algorithm. Section 4 reports on the set up of the benchmarking experiments and the results using the eight AIS algorithms. Section 5 discusses the feature selection and extraction approach used, and section 6 concludes the paper.

2. Clonal Selection Theory

The role of the biological immune system is to provide the organisms with an effective mechanism against pathogenic infections. The biological immune system mainly consists of two defensive lines, one is the innate immune system, and the other is the adaptive immune system. These two systems perform the defensive tasks complementarily. The core of the adaptive immune response is the clonal selection theory. When B-lymphocytes encounter antigens, they will activate B-lymphocytes to produce antibody molecules. Because antibody molecules are attached to the B-lymphocytes, sometimes we do not make any distinction between them. Each B-lymphocyte can produce only one kind of specific antibody molecules capable of recognizing and .binding to this specific antigen. And it is this binding that will stimulate the B-lymphocyte to reproduce a cell or cells and later differentiate into plasma cells. This asexual proliferation generates daughter cell or cells named a clone.

The first encountering and response to the antigen is named the primary immune response. After the primary immune response, the immune system can remember the antigens that it has responded and this is called the immune memory. The immune memory can guarantee that the immune system can provide a more effective and rapid response to the antigens that have been detected before. And this is the basis of the vaccination, which makes us get rid of many diseases. The immune memory also implies the learning mechanism during the clonal selection process. When antigens are from the inside of the individual organism, the recognizing B-lymphocytes will be discarded or this will cause autoimmunity. This phenomenon is called immune toleration or self-tolerance. [11]

The main property of the clonal selection theory can be summarized as follows [12]:

1) *Negative selection:* elimination of self-antigens;
2) *Clonal expansion:* proliferation and differentiation;
3) *Monospecificity:* phenotypic restriction;
4) *Somatic hypermutation:* new random genetic changes;
5) *Autoimmunity*.

3. The Clonal Selection Algorithm (CSA)

In the CSA, a candidate solution for the specific problem is called an antigen, which is recognized by the antibody. Each antibody represents a point in the search space, i.e., a possible solution to the problem. A population consists of a finite number of antibodies. Every antibody is evaluated by the evaluation mechanism to obtain its affinity. Based on this affinity and undergoing immune operators, a new population is generated iteratively with each successive population, referred to as a generation.

The CSA has two main computational mechanisms: selection and mutation. In the algorithm proposed by de Castro & Von Zuben, these two mechanisms were fulfilled by taking into account the immune properties: the proliferation and mutation rate are proportional to the antigenic affinity. That is to say, the higher the antigenic affinity, the higher the number of clones generated for each antibody. In the hypermutation operation, the cloned population is subject to an affinity mutation process inversely proportional to the antigenic affinity. The receptor editing includes two steps. In the first step, a given number of new antibodies are generated randomly. In the second step, the generated antibodies are used to refresh the whole population by replacing those antibodies with the lowest antigenic affinity.

The CSA uses three immune operators, i.e., cloning, hypermutation, and receptor editing, to refresh the composition of populations. The cloning operator explores the neighborhood of each point of the search space.

3.1. CSA and Learning

One of the most important characteristics of the immune system is learning. The learning ability of the immune system lies primarily in the clonal expansion [13]. The immune system learns in the clonal selection by the mechanism as follows: the immune system is repeatedly exposure to the antigen, which will activate a mechanism called hypermutation. The hypermutation sometimes will increase the diversity of the antibody molecules. And B-lymphocytes with higher affinity antibodies will have more chances to perform clonal selection. The repeating

mutation and selection make the immune system learn to produce higher affinity antibodies binding with the antigen.

3.2. CSA and Evolution

From the perspective of the Darwin's evolution theory, the clonal selection can be explained as a microevolution in the immune system. During the process of the clonal selection, there have three main features: selection, mutation and diversity. Selection occurs only when the antibody molecules can recognize and bind effectively with the antigen; the reproduction can activate the hypermutation, which will increase the diversity of the antibody molecules. The diversity can ensure the immune system against antigens that the organisms have never seen nor have some correlation with the antigens that have met before. And receptor editing is another important mechanism to realize the receptor diversity. Recent work and studies suggest that editing plays a central role in the B-lymphocyte and T-lymphocyte repertoire development [14].

4. Clonal Selection Classification Algorithms

One of the goals of this study was to pinpoint the classification algorithm that provides the highest level of detection accuracy. We employed eight commonly used artificial immune classification algorithms: AIRS1, AIRS2, AIRS2 Parallel, CLONALG, CSCA, Immunos-1, Immunos-81, Immunos-99 using Weka [10] machine-learning workbench.

4.1. AIRS - Artificial Immune Recognition System [15]

A resource limited artificial immune system (AIS) for supervised classification, using clonal selection, affinity maturation, and affinity recognition balls (ARBs). The AIRS algorithm has been shown to be a successful classification algorithm for a broad range of machine learning problems. From a data mining point of view, AIRS is a cluster-based approach to classification. It first learns the structure of the input space by mapping a codebook of cluster centers to it and then uses k-nearest neighbor on the cluster centers for classification. The attractive point of AIRS is its supervised procedure for discovering both the optimal number and position of the cluster centers. In AIRS, there are two different populations, the Artificial Recognition Balls (ARBs) and the memory cells. If a training antigen is presented, ARBs (lymphocytes) matching the antigen are activated and awarded more resources. Through this process of stimulation, mutation and selection a candidate memory cell is selected and it is inserted to the memory cell pool if it gives enough information. This process is repeated for all training instances and finally classification takes place by performing a nearest neighbor search on the memory cell population.

AIRS algorithm has features which are listed as follows [16]:

- Generalization: The algorithm does not need all the dataset for generalization and it has data reduction capability.
- Parameter stability: Even though user-defined parameters are not optimized for the problem, the decline of its performance is very small.
- Performance: There has been demonstrated that its performance is excellent for some datasets and totally remarkable.
- Self-regulatory: There is no need to choose a topology before training.

AIRS algorithm has five steps: Initialization, antigen training, competition for limited resource, memory cell selection and classification. The first step and the last step are applied only once, but steps 2, 3, 4 are used for each sample in the dataset.
The classification performance of the AIRS algorithm depends on eight user defined parameters: affinity threshold scalar ATS, clonal rate CR, hypermutation rate HR, number of nearest neighbors kNN, initial memory cell pool size IMPS, number of instances to compute the affinity threshold NIAT, stimulation threshold ST, and total resources TR.

In the AIRS experiments reported here we investigated the effect of the affinity threshold scalar ATS for 25 values between 0.01 and 0.95. The remaining parameters were kept constant, with the following values: CR = 10, HR = 2, kNN = 3, IMPS = 50, NIAT = all, ST = 0.5, and TR = 150. ATS is used to compute a cutoff value for memory cell replacement, and takes values between 0 and 1. A candidate ARB replaces a memory cell if the affinity between a candidate ARB and the best matching memory cell is lower that a threshold. A low ATS value results in a low replacement rate, whereas a high ATS value corresponds to a high replacement rate. Selected results for the AIRS simulations are shown in Table 1, 2, and 3. The best classification are obtained for ATS = 0.4. The AIRS models are stable for the entire range of ATS values examined, with better predictions obtained for small ATS values.

Table 1. AIRS1

AIRS1 [15]	
Classifier Memory Cells	
Correctly Classified Instances	215 (93.4783%)

Incorrectly Classified Instances		15 (6.5217%)
Root mean squared error		0.3489
Detailed Accuracy By Class (Worm | Benign)		
	Worm	Benign
TP Rate	0.993	0.854
FP Rate	0.146	0.007
Precision	0.905	0.988
Recall	0.993	0.854
F-Measure	0.947	0.916

The classification performance of the AIRS2 algorithm [17] depends on eight user defined parameters: affinity threshold scalar ATS, clonal rate CR, hypermutation rate HR, number of nearest neighbors kNN, initial memory cell pool size IMPS, number of instances to compute the affinity threshold NIAT, stimulation threshold ST, and total resources TR. In the AIRS experiments reported here we investigated the effect of the affinity threshold scalar ATS for 25 values between 0.01 and 0.95. The remaining parameters were kept constant, with the following values: CR = 10, HR = 2, kNN = 3, IMPS = 50, NIAT = all, ST = 0.5, and TR = 150. ATS is used to compute a cutoff value for memory cell replacement, and takes values between 0 and 1. A candidate ARB replaces a memory cell if the affinity between a candidate ARB and the best matching memory cell is lower that a threshold. A low ATS value results in a low replacement rate, whereas a high ATS value corresponds to a high replacement rate. Selected results for the AIRS simulations are shown in Table 1, 2, and 3. The best classification are obtained for ATS = 0.2.

Table 2. AIRS2

AIRS2 [17]		
Classifier Memory Cells		
Correctly Classified Instances		220 (95.6522%)
Incorrectly Classified Instances		10 (4.3478%)
Detailed Accuracy By Class (Worm | Benign)		
	Worm	Benign
TP Rate	0.985	0.917
FP Rate	0.083	0.015
Precision	0.943	0.978
Recall	0.985	0.917
F-Measure	0.964	0.946

The classification performance of the AIRS2-Parallel algorithm [18] depends on eight user defined parameters: affinity threshold scalar ATS, clonal rate CR, hypermutation rate HR, number of nearest neighbors kNN, initial memory cell pool size IMPS, number of instances to compute the affinity threshold NIAT, stimulation threshold ST, and total resources TR.

In the AIRS experiments reported here we investigated the effect of the affinity threshold scalar ATS for 25 values between 0.01 and 0.95. The remaining parameters were kept constant, with the following values: CR = 10, HR = 2, kNN = 3, IMPS = 50, NIAT = all, ST = 0.5, and TR = 150. ATS is used to compute a cutoff value for memory cell replacement, and takes values between 0 and 1. A candidate ARB replaces a memory cell if the affinity between a candidate ARB and the best matching memory cell is lower that a threshold. A low ATS value results in a low replacement rate, whereas a high ATS value corresponds to a high replacement rate. Selected results for the AIRS simulations are shown in Table 1, 2, and 3. The best classification are obtained for ATS = 0.2. The AIRS models are stable for the entire range of ATS values examined, with better predictions obtained for small ATS values.

Table 3. AIRS2 – Parallel

AIRS2-Parallel [18]		
Classifier Memory Cells		
Correctly Classified Instances		220 (95.6522%)
Incorrectly Classified Instances		10 (4.3478%)
Root mean squared error		0.3093
Detailed Accuracy By Class (Worm | Benign)		
	Worm	Benign
TP Rate	0.985	0.917
FP Rate	0.083	0.015
Precision	0.943	0.978
Recall	0.985	0.917
F-Measure	0.964	0.946

4.2. Clonal Selection Algorithm (CLONALG) [19]

The CLONALG is an AIS algorithm that gives a central role to the clonal selection theory as proposed by de Castro and Von Zuben. Several mechanisms of the clonal selection are implemented in CLONALG, namely training of a group of memory cells, identification and cloning of the antibodies with the highest recognition power, death of the antibodies with low recognition power, cloning and hypermutation of the antibodies with high recognition power, evaluation and replacement of the clones, generation and preservation of antibody diversity. CLONALG is inspired from the following elements: [16]

- Maintenance of a specific memory set.
- Selection and cloning of most stimulated antibodies.
- Death of non-stimulated antibodies.

- Affinity maturation.
- Re-selection of clones proportional to affinity with antigen.
- Generation and maintenance of diversity.

The classification performance of the CLONALG algorithm depends on six user defined parameters: clonal factor CF, antibody pool size APS, number of generations NG, remainder pool ratio RPR, selection pool size SPS, and total replacements TR. To illustrate the effect of the user defined parameters on the classification performance of CLONALG, we show the influence of the clonal factor CF on the classification. The clonal factor is a scaling factor, with values between 0 and 1, which determines the number of clones generated for each selected antibody. Low values for CF result in a local search, whereas for high values the algorithm generates a larger number of clones that may explore a wider region and result in a higher diversity. The CLONALG experiments for classification were computed for 20 CF values between 0.01 and 0.95. The remaining parameters were kept constant, with the following values: APS = 50, NG = 10, RPR = 0.1, SPS = 20, and TR = 2. The best classification are obtained for CF = 0.9.

Table 4. CLONALG

CLONALG [19]		
Classifier Memory Cells		
Correctly Classified Instances		210 (91.3043%)
Incorrectly Classified Instances		20 (8.6957 %)
Detailed Accuracy By Class (Worm | Benign)		
	Worm	Benign
TP Rate	0.985	0.813
FP Rate	0.188	0.015
Precision	0.88	0.975
Recall	0.985	0.813
F-Measure	0.93	0.886

4.3. Clonal Selection Classification Algorithm (CSCA) [20]

The CSCA was developed by Brownlee, and is formulated as a function optimization procedure that maximizes the number of patterns correctly classified and minimizes the number of patterns incorrectly classified [20-twenty]. CSCA is trained for several generations, and during each generation the entire set of antibodies is exposed to all antigens.

The classification performance of the CSCA immune system depends on six parameters that are set by the user: clonal scale factor CSF, number of nearest neighbors kNN, initial population size IPS, minimum fitness threshold MFT, number of partitions NP, and total generations TG. To demonstrate the influence of the user defined parameters on the CSCA predictions, we evaluate the influence of the clonal scale factor CSF on the classification. CSF is used to increase or decrease the number of clones generated for each antibody, and has a default value of one. Low values for CSF promote a low diversity of solutions, whereas high CSF values increase the diversity of the recognition cells. The CSCA experiments for classification were computed for 16 CSF values between 0.1 and 4. The remaining parameters were kept constant, with the following values: kNN = 3, IPS = 50, MFT = 1.0, NP = 1, and TG = 5. Selected results obtained in the CSCA simulations are shown in Table 5. The best predictions are obtained for CSF = 2.

Table 5. CSCA

CSCA [20]		
Classifier Memory Cells		
Correctly Classified Instances		219 (95.2174%)
Incorrectly Classified Instances		11 (4.7826%)
Detailed Accuracy By Class (Worm | Benign)		
	Worm	Benign
TP Rate	0.97	0.927
FP Rate	0.073	0.03
Precision	0.949	0.957
Recall	0.97	0.927
F-Measure	0.959	0.942

4.4. Immunos [21]

IMMUNOS -81 algorithm is the first AIS-based classification algorithm. IMMUNOS-81 used the artificial T cells to control the production of B-cells. The B-cells would then in turn compete for the recognition of the ‘‘unknowns”. The amino-acid library acts as a library of epitopes (or variables) currently in the system. When a new antigen is introduced into the system, its variables are entered into this library. The T- cells then use the library to create their receptors that are used to identify the new antigen. During the recognition stage of the algorithm T-cell paratopes are matched against the epitopes of the antigen, and then it is created a B-cell that has paratopes that match the epitopes of the antigen [16-6666]. The IMMUNOS -81 represents an instance-based classifier with some similarity to *k*-nearest neighbor classifiers.

Table 6. IMMUNOS-1

Immunos-1 [21]	
Classifier Memory Cells	
Correctly Classified Instances	214

			(93.0435%)
Incorrectly Classified Instances			16 (6.9565 %)
Detailed Accuracy By Class (Worm | Benign)			
	Worm	Benign	
TP Rate	0.985	0.854	
FP Rate	0.146	0.015	
Precision	0.904	0.976	
Recall	0.985	0.854	
F-Measure	0.943	0.911	

Table 7. IMMUNOS-81

Immunos-81 [21]			
Classifier Memory Cells			
Correctly Classified Instances			166 (72.1739%)
Incorrectly Classified Instances			64 (27.8261%)
Detailed Accuracy By Class (Worm | Benign)			
	Worm	Benign	
TP Rate	1	0.333	
FP Rate	0.667	0	
Precision	0.677	1	
Recall	0.1	0.333	
F-Measure	0.807	0.5	

Brownlee improved Immunos-81 algorithm by incorporating elements from other AIS classifiers, such as cloning and hypermutation, to obtain Immunos-99. Immunos-99 has three parameters that control the classification performance: seed population percentage SPP, minimum fitness threshold MFT, and total generations TG. The experiments presented here investigate the influence of the seed population percentage.

SPP represents the percentage of the antigen population from each class that is used as seed for the B-cell population. If SPP = 100% then the initial B-cell population is identical with the antigen population in the same class. The Immunos-99 classifier was trained for 19 values of the SPP parameter, between 0.05 and 0.95, with MFT = 0.5 and TG = 2. Overall, the classifications are lower than those obtained with AIRS, CLONALG, and CSCA, as shown by the selected results presented in Table 8. The best classification are obtained for SPP = 0.6, but such low values are not useful for classification.

Table 8. IMMUNOS-99

Immunos-99 [21]			
Classifier Memory Cells			
Correctly Classified Instances			113 (49.1304%)
Incorrectly Classified Instances			117 (50.8696%)
Detailed Accuracy By Class (Worm | Benign)			
	Worm	Benign	
TP Rate	0.843	0	
FP Rate	1	0.157	
Precision	0.541	0	
Recall	0.843	0	
F-Measure	0.659	0	

Figure -1- below shows the classification accuracy of the eight AIS algorithms

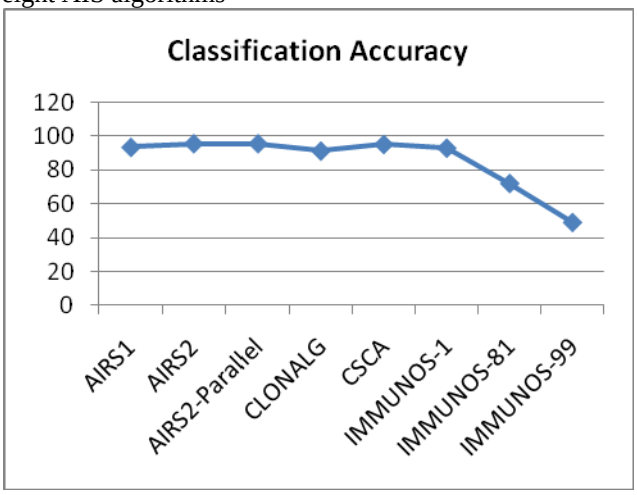

Figure 1. Classification Accuracy

Figure -2- below shows the TP Rate and FP Rate of worm using the eight AIS algorithms

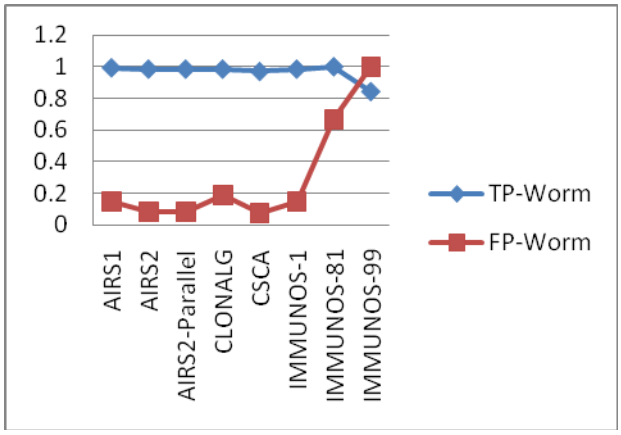

Figure 2 – TP Rate and FP Rate for Worm

Figure -3- below shows the TP Rate and FP Rate of benign using the eight AIS algorithms

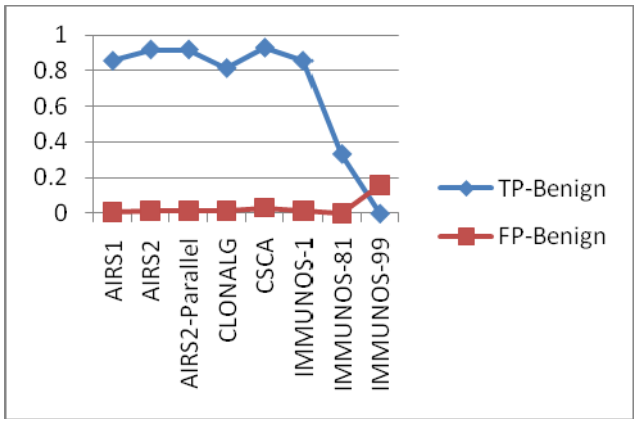

Figure 3 – TP Rate and FP Rate for Benign

5. Feature Selection and Extraction

We used Portable Executable (PE) feature extraction approach. We collected a set of 100 worm and 100 benign Windows executables. All of them are in Win32 portable executable (PE) format. The benign executables are obtained from a freshly installed copy of Windows XP and application installers. The malware executables are obtained from VX Heavens virus collection which is publicly available [22].

API Monitor 1.5 is used to log only the API call sequences of Windows system-wide processes because some malware in our study use Windows processes like explorer.exe to carry out malicious activities. Therefore, it is not possible for us to exactly pin-point a set of processes for monitoring [3].

API Monitor 1.5 captures these API calls and stores them in *apm* file format [22]. It has an API and a process filter. The API filter gives us the option of filtering unnecessary API calls by category. In the API filter, we can select the calls of following categories:

- Dynamic-Link Libraries
- Processes and Threads
- Memory Management
- Network Management
- Registry
- Socket

Feature selection is the process of identifying relevant features in the dataset and discarding everything else as irrelevant and redundant, it enables the classification algorithms to operate more effectively and rapidly.

We use n-gram analysis for feature extraction. n-gram of a sequence is the normalized frequency histogram of n successive elements of the sequence [23].

In this study we chose a value of n=4 to get sufficient information from the n-grams while incurring reasonable processing overheads.. Each API function is mapped to a unique random variable. We extract the most informative 4-grams from all dataset files by ranking them according to their information gain. The information gain of a feature i is defined as [24]:

$$I(Y ; X) = H(Y) - H(Y \mid X), \qquad (1)$$

where X is an input attribute, Y is a class attribute, H(Y) is the entropy of the class attribute variable Y and H(Y|X) is the conditional entropy of Y with respect to X.

For feature extraction, we check the log file of each executable file for presence or absence of the selected feature. We place 1 if the feature is present and 0 otherwise. Each executable log is mapped to a 500-dimensional binary string.

6. Conclusion

In this paper we explored the feasibility of detecting unknown worm activity in individual computers, at a high level of accuracy using the following algorithms: AIRS1, AIRS2, AIRS2 Parallel, CLONALG, CSCA, IMMUNOS-1, IMMUNOS-81, IMMUNOS-99. Additionally, they all were performed by using 10-fold cross-validation. Experiments were repeated 10 times to produce statistically reliable results.

To examine the possibility of classifying unknown worms, two classes were defined, a worm type consisting of the API call sequence samples and benign API call sequence samples. The training sets had 100 worms. We found that the level of detection accuracy varied within each algorithm as follows: AIRS1=93.4783%, AIRS2=95.6522%, AIRS2-Parallel=95.6522%, CLONALG=91.3043%, CSCA=95.2174%, IMMUNOS-1=93.0435%, IMMUNOS-81=72.1739%, IMMUNOS-99=49.1304%.

In conclusion it can be seen from our study that it is possible to detect malicious activity of worms by looking at the attributes derived from the computer operation parameters derived from API call sequence of a process using AIRS1, AIRS2, AIRS-Parallel, CLONALG, CSCA. On the other hand the place of misclassification and FP rates that are still significantly high in IMMUNOS-1, IMMUNOS-81, and IMMUNOS-99suggests that there are still difficulties related to the detection of the worms using these algorithms.

References

[1] F-Secure Corporation, “F-Secure Reports Amount of Malware Grew by 100% during 2007", Press release, 2007.

[2] Symantec, “Internet Security Threat Report", Vol. XIV, 2009.
[3] S. Manzoor, M. Z. Shafiq, S. M. Tabish, M. Farooq “A Sense of `Danger' for Windows Processes”, http://www.nextginrc.org/, 2008.
[4] M. Christodorescu, S. Jha, “Testing Malware Detectors", ACM SIGSOFT Software Engineering Notes, 29(4), ACM Press, pp. 34-44, 2004.
[5] L. N. De Castro, F. J. Von Zuben. “The Clonal Selection Algorithm with Engineering Applications”, Proceedings of Workshop on Artificial Immune Systems and Their Applications, pp.36-37, 2000.
[6] J. Kim and P. J. Bentley, “Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection”, Proceedings of Congress on Evolutionary Computation, pp. 1015-1020, 2002
[7] R. Liu, H. Du, L. Jiao, “Immunity Clonal Strategies”, Proceedings of the Fifth International Conference on Computational Intelligence and Multimedia Applications, 2003:290.
[8] L. Wang, J. Pan, L. Jiao, “The Immune Programming”, Chinese Journal of Computers. 23(8), pp. 806-812, 2000
[9] J. Xian, F. Lang., X. Tang, “A Novel Intrusion Detection Method based on Clonal Selection Clustering Algorithm”, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, pp. 3905-3910, 18-21 August 2005.
[10] K. Liaskos, M. Roper, “Hybridizing Evolutionary Testing with Artificial Immune Systems and Local Search”, Conference on Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International, pp. 211-220, 2008.
[11] Y. Ying, C. Z. Hou, “A Clonal election Algorithm By Using Learning Operator”, Proceedings of the Third International Conference on Machine Learning and Cybernetics, Shanghai, pp. 26-29, August 2004.
[12] N. L. de Castro, J. Timmis, “Artificial immune systems: a new computational intelligence approach”, New York: Springer, London, 2002.
[13] D. Dasgupta, “Artificial Immune Systems and Their Applications”, Germany: Springer-Verlag, Berlin, 1999.
[14] D. Nemazee, K. A. Hogquist, “Antigen receptor selection by editing or downregulation of V(D)J recombination”, Current Opinion in Immunology, Vol. 15, pp. 182-189, 2003.
[15] B. A. Watkins, “A resource limited artificial immune classifier”, Mississippi State University, (Masters Thesis), 2001.
[16] C. Catal, B. Diri, “Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem”, Elsevier, pp. 1040-1058, 2008.
[17] A. Watkins, J. Timmis, L. Boggess, “Artificial Immune Recognition System (AIRS): An Immune-Inspired Supervised Learning Algorithm”, Genetic Programming and Evolvable Machines, vol. 5, pp. 291-317, Sep, 2004.
[18] A. Watkins, J. Timmis, “Exploiting Parallelism Inherent in AIRS, an Artificial Immune Classifier", Proceedings of the 3rd International Conference on Artificial Immune Systems (ICARIS2004), Catania, Italy, pp. 427-438, 2004.
[19] L. N. de Castro, J., V. Zuben, “Learning and Optimization Using the Clonal Selection Principle”, IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems. 2002; 6(3): 239-251.
[20] J. Brownlee. [Technical Report], “Clonal Selection Theory & CLONALG - The Clonal Selection Classification Algorithm (CSCA)”, Victoria, Australia: Centre for Intelligent Systems and Complex Processes (CISCP), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology, 2005 Jan, Technical Report ID: 2-01.
[21] J. Brownlee. [Technical Report], “Immunos-81 - The Misunderstood Artificial Immune System”, Victoria, Australia: Centre for Intelligent Systems and Complex Processes (CISCP), Faculty of Information and Communication Technologies (ICT), Swinburne University of Technology; 2005 Feb; Technical Report ID: 3-01.
[22] API Monitor, available at http://www.rohitab.com/apimonitor.
[23] M. Damashek, “Gauging Similarity with n-Grams: Language-Independent Categorization of Text", Vol. 267, pp. 843-848, Science, 1995.
[24] J.Z. Kolter, M.A. Maloof, “Learning to detect malicious executables in the wild", International Conference on Knowledge Discovery and Data Mining, pp. 470-478, ACM Press, USA, 2004.

Khaled AbdElKhalek Al-Sheshtawi obtained his B.Sc. in Computer Science from King AbdulAziz University, Faculty of Science, Saudi Arabia in 1991 and M.Sc. with Distinction in Computer-based Information Systems from University of Sunderland, School of Computing and Information Systems, United Kingdom in 1998. He is currently a Lecturer in King AbdulAziz University, Information Technology Department since 1999.

Hatem Mohamed Abdul-Kader obtained his B.S. and M.SC. (by research) both in Electrical Engineering from the Alexandria University , Faculty of Engineering , Egypt in 1990 and 1995 respectively. He obtained his Ph.D. degree in Electrical Engineering also from Alexandria University, Faculty of Engineering, Egypt in 2001 specializing in neural networks and its applications. He is currently a Associative professor in Information systems department, Faculty of Computers and Information, Minufiya University, Egypt since 2004. He has worked on a number of research topics and consulted for a number of organizations. He has contributed more than 35+ technical papers in the areas of neural networks, Database applications, Information security and Internet applications.

Nabil A. Ismail is a professor of Computer Science and Engineering. He used to be the Dean of the Faculty of Computers and Information, University of Menoufia, Egypt (2006-2008). Prof. Ismail has obtained his PhD from Durham University, England, in 1983. He is now working as a Professor at the Faculty of Electronic Engineering, Egypt and Al-Baha College of Computer Science, KSA. Prof. Nabil Ismail research interests including computer security, image reconstruction, computer architecture, and constrained-resource ECP.