Vulnerability Analysis of Immunity-Based Intrusion Detection Systems Using Evolutionary Hackers

Gerry Dozier[1], Douglas Brown[2], John Hurley[3], and Krystal Cain[2]

[1] Dept. of Computer Science & Software Engineering, Auburn University, AL 36849-5347 `gvdozier@eng.auburn.edu`

[2] Dept. of Computer Science Clark-Atlanta University, Atlanta, GA 30314 douglasbrown1982 `KDJCain@aol.com`

[3] Distributed Systems Integration The Boeing Company, Seattle, WA 98124 `john.s.hurley@boeing.com`

Abstract. Artificial Immune Systems (AISs) are biologically inspired problem solvers that have been used successfully as intrusion detection systems (IDSs). This paper describes how the design of AIS-based IDSs can be improved through the use of evolutionary hackers in the form of GENERTIA red teams (GRTs) to discover holes (in the form of type II errors) found in the immune system. GENERTIA is an interactive tool for the design and analysis of immunity-based intrusion detection systems. Although the research presented in this paper focuses on AIS-based IDSs, the concept of GENERTIA and red teams can be applied to any IDS that uses machine learning techniques to develop models of normal and abnormal network traffic. In this paper we compare a genetic hacker with six evolutionary hackers based on particle swarm optimization (PSO). Our results show that genetic and swarm search are effective and complementary methods for vulnerability analysis. Our results also suggest that red teams based on genetic/PSO hybrids (which we refer to Genetic Swarms) may hold some promise.

1 Introduction

Intrusion detection [11,12,13,14,18,19,21,22] can be viewed as the problem of classifying network traffic as normal (self) or abnormal (non-self). Researchers in this area have developed a variety of intrusion detection systems (IDSs) based on: statistical methods [18,19], neural networks [3], decision trees [2], and artificial immune systems (AISs) [1,6,7,11,12,13,15,16,23,26]. One of the primary objectives of machine learning is to develop a hypothesis that has low error and generalizes well to unseen instances [20]. There are two types of error associated with the hypotheses developed by any learning algorithm [19,20]: false positives, known as type I errors, and false negatives, referred to as type II errors. In the context of network security, type II errors represent 'holes' [12] in an IDS.

Since type II errors do exist in IDSs, a dilemma associated with IDS design, development, and deployment is, "Does one try to identify and/or patch holes in advance?" Or "Does one allow the *hackers* to identify the holes and only then try to patch them?" In this paper, we demonstrate how GENERTIA red teams (GRTs), in the form of genetic and particle swarm search [5,17], can be used to discover holes in IDSs. This information can then be used by the designers of IDSs to develop patches or it can be used by an IDS to 'heal' itself. This research is motivated by the fact that cyber-terrorists are now turning towards automated agent-based warfare [14,19,24]. The GRT can be seen as a 'white-hat' hacker agent.

The GRTs presented in this paper are part of a larger system named GENERTIA which contains two sub-systems based on evolutionary algorithms [5]: a GENERTIA blue team (GBT) and a GRT. The objective of the GBT is to design AIS-based IDSs that have high attack detection rates, low error rates, and use a minimal number of detectors. The objective of the GRT is to analyze IDSs and provide feedback to the GBT. Figure 1 shows the architecture of GENERTIA for a host-based IDS. The GBT is used to design an IDS based on input from the network manager. After a preliminary host-based IDS has been designed, the GRT performs a strength and vulnerability analysis of the IDS, based on the input specifications of the network manager. The GRT analysis results in information concerning the relative strength of detectors (labeled as RSD in Figure 1) comprising the IDS as well as a list of vulnerabilities (holes in the IDS). This information is then given to the GBT to be used to re-design the IDS.

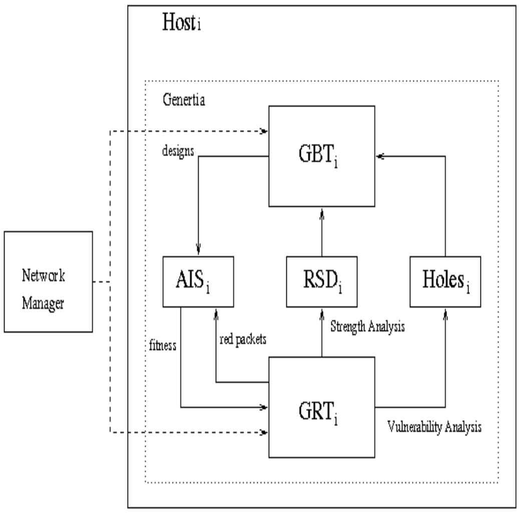

Fig. 1. Architecture of GENERTIA for a Host-Based IDS

2 AIS-Based Intrusion Detection Systems

A number of researchers [1,6,12,13] have developed artificial immune systems (AISs) for networks in an effort to protect them from malicious attacks. A typical AIS-based IDS attempts to classify network traffic as either self or non-self by allowing each host to maintain and evolve a population of detectors. The evolutionary process of these detectors is as follows.

Initially, for each host, a randomly generated set of immature detectors is created. These detectors are exposed to normal network traffic (self) for a user-specified amount of time (where time is measured in packets), $t_{immature}$. If an immature detector matches a self packet then the detector dies and is removed[1]. This process of removing immature detectors that match self packets is referred to as negative selection [11,12].

All immature detectors that survive the process of negative selection become mature detectors. Thus, mature detectors will typically match non-self packets. Mature detectors are also given a user-specified amount of time, t_{mature}, to match m_{mature} non-self packets. This time represents the learning phase of a detector [11,12]. Mature detectors that fail to match m_{mature} non-self packets during their learning phase die and are removed from the detector population. If a mature detector matches m_{mature} non-self packets during its learning phase, a primary response (alarm) is invoked.

Once a mature detector sounds an alarm, it awaits a response from the network administrator to verify that the detector has identified a valid attack on the system. This response is referred to as co-stimulation [11,12]. If co-stimulation does not occur within a prescribed amount of time the mature detector will die and be removed from the detector population. Those detectors that receive co-stimulation are promoted to being memory detectors and are assigned a longer life time of t_{memory}. Memory detectors invoke stronger (secondary) responses when they match at least m_{memory} non-self packets (where usually $m_{mature} > m_{memory}$).

2.1 Advantages Offered by AIS-Based IDSs

There are a number of advantages to using AIS-based intrusion detection. One advantage is that they provide a form of passively proactive protection via negative selection. This enables an AIS-based IDS to detect novel attacks. Another advantage is that AISs are systems that are capable of adapting to dynamically changing environments. As the characteristics of self traffic change over time, a properly tuned AIS will be able to effectively adapt to the dynamically changing definition of self. Finally, the detectors evolved by AIS-based IDSs can easily be converted into Snort or tcpdump filters. This is especially true when constraint-based detectors are used [15,16,26]. These constraint-based detectors are easy to understand and can provide network administrators with important forensic information when new attacks are detected.

[1] Any time a detector is removed from the detector population it is automatically replaced with a randomly generated immature detector. This keeps the size of the detector population constant.

2.2 A Disadvantage of Using AIS-Based IDSs

A disadvantage with the use of AIS-based IDSs[2] is that, at present, there is no way to know exactly what types of attacks will pass through undetected. However, a number of techniques have been developed to reduce the size of potential holes [1,7,12] but none of these can be used to alert the designer or users as to the types of attacks that will go undetected by the AIS. In the next section we demonstrate how quickly a GRT can discover holes in an AIS-based IDS.

3 The GENERTIA AIS

As stated earlier, the purpose of the GBT is to develop an AIS in the form of a set of detectors to catch new, previously unseen attacks. The purpose of the GRT is to discover holes in the AIS. As shown in Figure 1, the AIS communicates with the GRT by receiving ‘red’ packets in the form of attacks from the GRT and returning the percentage of the detector set that failed to detect the ‘red’ packet, referred to as the fitness of the ‘red’ packet. The preliminary results presented in this paper are based on a single host-based IDS. However, these results can be generalized to a network where each host has an instance of GENERTIA running on it.

3.1 Representation of Packets

For our AIS, packets are represented as triples (which we will refer to as data triples) of the form **(ip_address, port, src)**, where **ip_address** represents the IP address of the remote host, **port** represents the port number the receiving host, and **src** is assigned to 0 if the packet is incoming or 1 if the packet is outgoing.

3.2 The Representation and Behavior of Detectors

The AIS maintains a population of constraint-based detectors of the form: ($\boldsymbol{lb_0}$ .. $\boldsymbol{ub_0}$, $\boldsymbol{lb_1..ub_1}$, $\boldsymbol{lb_2..ub_2}$, $\boldsymbol{lb_3..ub_3}$, $\boldsymbol{lb_{port}..ub_{port}}$, $\boldsymbol{src}$)[3], where the first 4 intervals represent a set of IP addresses, the fifth interval represents the lower and upper bounds on the port, and where $\boldsymbol{src}$ is 0 to denote that the detector should be used on incoming traffic or 1 to denote that the detector should be used on outgoing traffic.

[2] Actually, this is the case with all IDSs that operate by building a model of self/non-self.

[3] Notice that detectors based on the above representation can be viewed as constraints. Thus, a detector population can be viewed as a population of constraints where self corresponds to a set of all solutions (data triples) that satisfies the constraints and where non-self corresponds to the set of all solutions that violate at least one constraint. Therefore the intrusion detection problem can be viewed as a distributed constraint satisfaction problem [25].

An *any-r intervals* matching rule [15,16,26] is used to determine a match between a data triple and a detector. That is, if any r numbers representing a data triple fall within the corresponding r intervals of a detector then that detector is said to match the data triple. For the experiments presented in this paper, $r = 3$.

If an immature detector matches a self packet then it is first relaxed by splitting it into two parts based on where the self packet intersects an interval and randomly removing either the lower or the upper part. This relaxation method is based on the the split-detector method [26].

In our experiments, if an immature detector fails to match a self packet after being exposed to $t_{immature} = 200$ self packets then it becomes a mature detector and is given a life time, t_{mature}, that insures its survival for the remainder of an experiment. If a mature detector matches a self packet it is relaxed using the split detector method. For our experiments, the values for m_{mature} and m_{memory} were set to 1.

4 Genetic and Swarm-Based Red Teams

The GRTs compared in this paper come in the form of a steady-state GA and six variants of particle swarm optimization [17]. Each of the seven GRTs evolved a population of 300 data triples. The fitness of a GRT data triple was simply the percentage of detectors that failed to detect it. For each cycle of the GA-based GRT, the worst fit data triple was replaced by an offspring if the offspring had a better fitness. An offspring was created by selecting two parents from the GRT population using binary tournament selection [10] and mating the parents using the BLX-0.5 crossover operator [8]. By using a steady-state GRT, the 300 best 'red' data triples will always remain in the population.

The swarm-based GRTs evolved a swarm of 300 particles where each particle contained: (a) a p-vector that recorded the best 'red' data triple that it has ever encountered, (b) a p-fitness value which represents the fitness of the p-vector, i.e. the percentage of the detectors of an AIS that failed to detect it, (c) an x-vector that recorded the current data triple that particle was visiting, (d) an x-fitness that recorded the fitness of the x-vector, and (e) a v-vector (velocity vector) which when added to the x-vector results in a new candidate 'red' data triple. The swarm-based GRTs were instances of the canonical PSO [4] where offspring were created as follows:

$$v_{id} = v_{id} + \eta_c \varphi_c (p_{id} - x_{id}) + \eta_s \varphi_s (p_{gd} - x_{id})$$
$$x_{id} = x_{id} + v_{id}$$

Where v_{id} represents the dth component of the ith particle's velocity vector, η_c and η_s represent the learning rates for the cognition and social components, φ_c and φ_s represent random numbers within [0..1] for the cognition and social components, and g represents the index of the particle with the best p-fitness in the neighborhood of particle i. Based on the suggestions of [4] the four swarm-based GRTs use asynchronous updating. Thus, each time a particle is updated the newly form x-vector is submitted to the AIS-based IDS where its x-fitness is assigned a value. The values of η_c and η_s were set to 2.3 and 1.8 according to [4].

The distinctions of the six swarms (denoted SW0, SW0+, SW1, SW2, SW3, and SW4) are based on: neighborhood (local, global), whether particles terminate their search when they discover a vulnerability, which we referred to as particle termination (PT), and whether the best particle, g, used for updating a particular particle is randomly selected or is the best particle with the lowest index (this only applies to those swarms that use a global neighborhood). This distinction is denoted, RB, for random best. The particles are arranged in a ring topology with a local neighborhood for a particle consisting of the particles adjacent to it. Table 1 shows the distinctions in terms of neighbohood, PT, and RB.

Table 1. The Distinctions of the Swarm-Based GRTs in Terms of Neighborhood, PT, and RB

Alg	Neighborhood	PT	RB
SW0	local	no	no
SW0+	local	yes	no
SW1	global	no	no
SW2	global	no	yes
SW3	global	yes	no
SW4	global	yes	yes

5 Training and Test Sets

Our training and test sets were obtained from the 1998 MIT Lincoln Lab data. The Lincoln Lab data represents 35 days of simulated network traffic for a Class B network. To obtain a host-based set, we extracted packets involving host 172.16.112.50 only. From this data, we converted each packet into data triples and filtered out all duplicates and data triples involving port 80. The ports were mapped into 70 distinct ports according to [11]. We then extracted the normal traffic to form the training set. Our final training set consisted of 112 self data triples. Our AISs were trained on approximately 80% of the training set, 89 self data triples. The other 23 self data triples were used to test for false positives (type I errors). Our test set consisted of all attacks launched during the 35 day period. This test set consisted of a total of 1604 data triples.

6 Experiment

Using the above training and test sets we conducted a comparison of the seven GRTs. Initially, an AIS was developed using a detector population size of 400.

The training of the AIS consisted of selecting a data triple from the training set (self set) and exposing the detector population to it. This process was repeated 400 times. After an AIS was trained, it was exposed to the test set. After the AIS was developed, each GRT, evolving a population of 300 malicious ('red') data triples, was allowed to interact with the AIS in order to discover holes in the system. A total of 5000 data triples were evaluated. This process was repeated ten times.

7 Results and Conclusions

Table 2 shows the average performance of the GRTs described above in terms of the number of total number of holes discovered, the number of duplicates, and the number of distinct holes. The ten AIS-based IDSs had an average detection rate of 0.747 with an average false positive rate of 0.4.

In Table 2, one can see that the GA outperforms all of the swarms with respect to number of holes and distinct number of holes discovered. Of the six swarms, SW0+ has the next overall best performance. This shows that terminating the motion of a particle once it has discovered a hole improves the performance of swarm search. When a particle is terminated it allows other particles in the swarm to have an increased number of trials. One can see in Table 2 that the swarms that used PT found a greater number of holes than those that did not; however, these swarms also had the greatest number of duplicates as well. Our results also show that those swarms that used a local neighborhood outperformed those that used a global neighborhood. In this study, the use of RB did not seem to provide a performance improvement.

Table 2. Comparison of the Seven GRTs on the 10 AIS-Based IDSs with an Average Detection Rate of 0.747 and an Average False Positive Rate of 0.4

Alg.	Holes	Duplicates	Distinct
GA	300.0	4.9	295.1
SW0	271.7	8.4	263.3
SW0+	298.4	21.0	277.4
SW1	297.8	51.7	246.1
SW2	292.5	50.2	242.3
SW3	299.1	62.7	236.4
SW4	300.0	62.4	237.6

Figure 2 and 3 show the convergence rates in terms of the average fitness of the population and the best fitness within the population of the GA, SW0, and SW0+ GRTs. In Figure 2, the average fitness of the populations increases rapidly from 30% to 95% within 500 evaluations (of red packets). After this point, the algorithms slowly converge on an average fitness that is close to 100%.

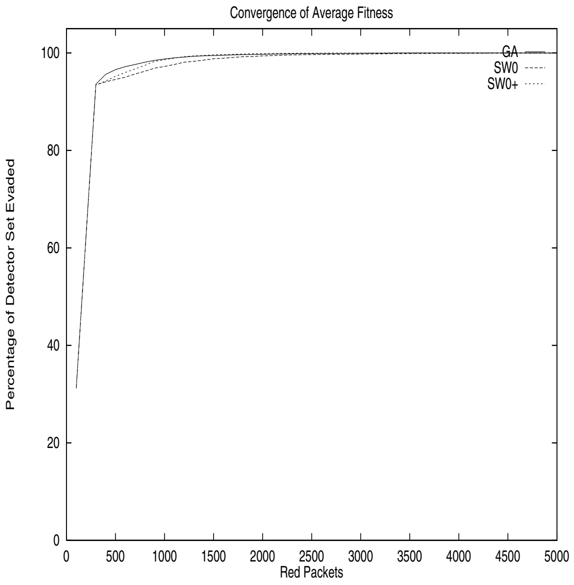

Fig. 2. Visualization of the Convergence Behavior in Terms of Average Fitness of the GA, SW0, and SW0+ GRTs Evolving a Population Size of 300 Data Triples

Figure 3 shows the convergence rates of the GA, SW0, and SW0+ GRTs with respect to the best fitness within a population. One can see that all of the algorithms start with an individual in the population that is capable of evading at least 98% of the 400 detectors. Each of the three algorithms finds their first hole in less than 1000 evaluations.

Figures 4-6 show a 3D visualization of the holes where the x-axis reprents the network, the y-axis represents the host and the z-axis represents the port. In this visualization we assume that the red packets are coming from a Class B network. In Figure 4-6, one can see that the GA and the swarms have a completely different search behavior. The GA tends to find solutions in clusters while the swarms seem to discover holes at the boundaries of the search space. This is interesting because based on our representation of a detector (constraint based intervals) the extreme values of source and host IP addresses and extreme values of port numbers are the hardest to cover. Based on the Figures 5 and 6 one can conclude that a better form of detector would one that uses wrap around intervals. This is a direction for future research. The results seen in Figures 4-6 suggest the possibility that a hybrid GA/PSO algorithm, one which we refer to as a genetic swarm, may be more effective than either GA or PSO alone. This is also a direction for future research.

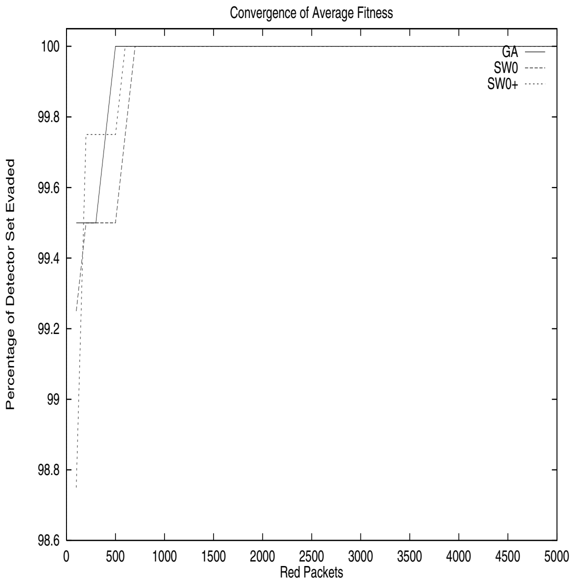

Fig. 3. Visualization of the Convergence Behavior in Terms of Average Fitness of the GA, SW0, and SW0+ GRTs Evolving a Population Size of 300 Data Triples

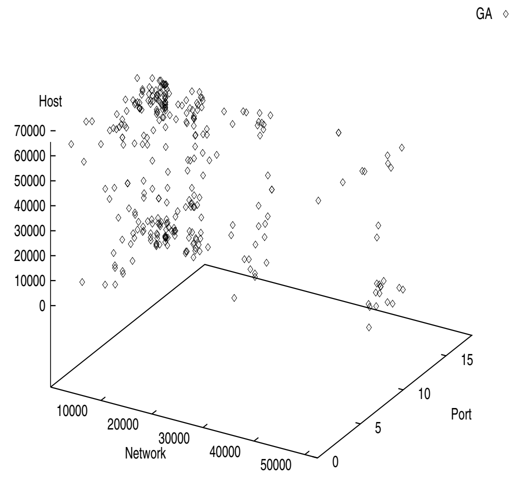

Fig. 4. Visualization of the Holes Evolved by the GA-Based GRT

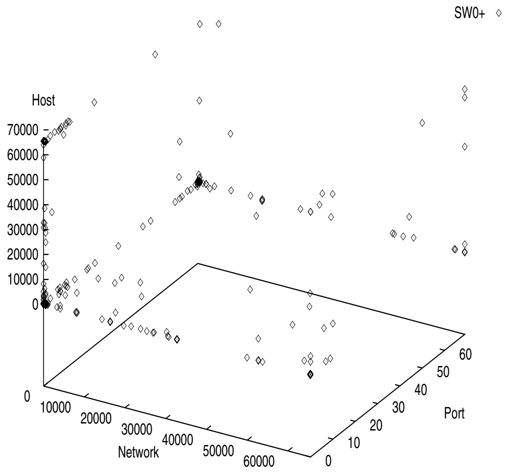

Fig. 5. Visualization of the Holes Evolved by the SW0+-Based GRT

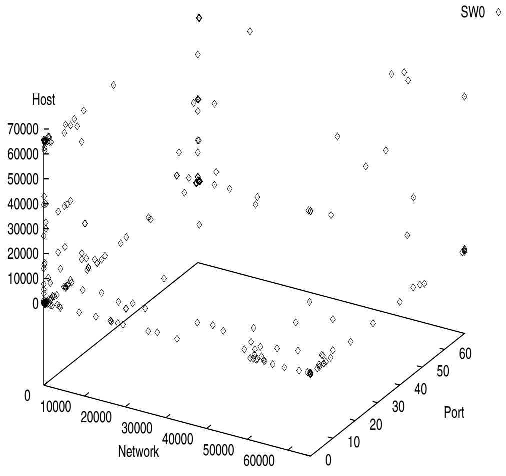

Fig. 6. Visualization of the Holes Evolved by the SW0-Based GRT

References

1. Balthrop, J., Forrest, S. and Glickman, M. (2002). “Revisiting LISYS: Parameters and Normal Behavior”, *Proceedings of the 2002 Congress on Evolutionary Computation (CEC’02)*, IEEE Press.
2. Bloedorn, E., Christiansen, A. D., Hill, W., Skorupka C., Talbot, L. M., and Tivel, J. (2001). “Data Mining for Network Intrusion Detection: How to Get Started”, available at: www.mitre.org/support/papers/archive01.shtml, The MITRE Corporation, August 2001.
3. Cannady, J. (1998). “Artificial Neural Networks for Misuse Detection”, *Proceedings of the 1998 National Information Systems Security Conference (NISSC’98)*, pp. 443-456, October 5-8 1998, Arlington, VA.
4. Carlisle, A. and Dozier, G. (2001). “An Off-The-Shelf PSO”, Proceedings of the 2001 Workshop on Particle Swarm Optimization, pp. 1-6, Indianapolis, IN.
5. Davis, L. (1991). *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.
6. Dasgupta, D. (1999). “An Overview of Artificial Immune Systems and Their Applications”, *Artificial Immune Systems and Their Applications*, D. Dasgupta, ed., pp. 3-21, Springer.
7. Dasgupta, D. and Gonzalez, F. (2002)., “An Immunity-Based Technique to Characterize Intrusions in Computer Networks”, *IEEE Transactions on Evolutionary Computation*, Vol. 6, No. 3, pp. 281-291. IEEE Press.
8. Eshelman, L. and Schaffer, J. D. (1992). “Real-Coded Genetic Algorithms and Interval-Schemata”, Proceedings of the 1992 Foundations of Genetic Algorithms (FOGA-2), L. Darrell Whitley, ed., pp. 187-202, Morgan Kaufmann.
9. Fogel, D. B. (2000). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd Edition, IEEE Press.
10. Goldberg, D. E. and Deb, K. (1989). “A Comparative Analysis of Selection Schemes Used in Genetic Algorithms”, *Foundations of Genetic Algorithms (1989)*, Gregory J. E. Rawlins, ed., pp. 69-93, Morgan Kaufmann Publishers.
11. Hofmeyr, S. (1999). *An Immunological Model of Distributed Detection and Its Application to Computer Security*, Ph.D. Dissertation, Department of Computer Science, The University of New Mexico.
12. Hofmeyr, S., and Forrest, S. (1999). “Immunity by Design: An Artificial Immune System”, The Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO-1999), pp. 1289-1296, Morgan-Kaufmann, San Francisco.
13. Hofmeyr, S., and Forrest, S. (1999). “Architecture for an Artificial Immune System”, Evolutionary Computation, 7(1):45-68.
14. Honeynet Project (2002). *Know Your Enemy: Revealing the Security Tools, Tactics, and Motives of the Blackhat Community*, Addison-Wesley.
15. Hou, H. (2002). *Artificial Immunity for Computer Network with Constraint-Based Detector*, Masters Thesis, Department of Computer Science and Software Engineering, Auburn University.
16. Hou, H., Zhu, J., and Dozier, G. (2002). “Artificial Immunity Using Constraint-Based Detectors”, Proceedings of the 2002 World Automation Congress (WAC’02), vol. 13, pp. 239-244, TSI Press.
17. Kennedy, J. and Eberhart, R. (2001). *Swarm Intelligence*, Morgan Kaufmann.
18. Marchette, D. J. (1999). “A Statistical Method for Profiling Network Traffic”, *Proceedings of the USENIX Workshop on Intrusion Detection and Network*, pp. 119-128.

19. Marchette, D. J. (2001). *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint*, Springer.
20. Mitchell, T.M. (1997). *Machine Learning*, McGraw-Hill.
21. Northcutt, S. and Novak, J. (2001). *Network Intrusion Detection: An Analyst's Handbook*, 2nd edition, New Riders.
22. Northcutt, S., Cooper, M., Fearnow M., and Frederick, K. (2001). *Intrusion Signatures and Analysis*, New Riders.
23. Somayaji, A., Hofmeyr, S., and Forrest, S. (1997). "Principles of a Computer Immune System", The 1997 New Security Paradigm Workshop, pp. 75-82.
24. Stewart, A. J. (1999). "Distributed Metastasis : A Computer Network Penetration Methodology", available at: *citeseer.nj.nec.com/387640.html*.
25. Yokoo, M. (2001). *Distributed Constraint Satisfaction*, Springer-Verlag.
26. Zhu, J. (2002). *Use of an Immune Model to Improve Intrusion Detection on Dynamic Broadcast Local Area Networks*, Masters Thesis, Department of Computer Science & Software Engineering, Auburn University.