ABSTRACT

COMPUTER SCIENCE

BROWN, JR., DOUGLAS B.S./M.S., CLARK ATLANTA UNIVERSITY, 2005

VULNERABILITY ANALYSIS OF AIS-BASED INTRUSION DETECTION SYSTEMS USING GENETIC AND EVOLUTIONARY HACKERS

Major Advisor: Roy George, Ph.D.

Thesis dated May 2005

In this thesis, an overview of current intrusion detection methods, evolutionary computation, and immunity-based intrusion detection systems (IDSs) is presented. An application named Genetic Interactive Teams for Intrusion Detection Design and Analysis (GENERTIA) is introduced which uses genetic algorithm (GA)-based hackers known as a red team in order to find vulnerabilities, or holes, in an artificial immune system (AIS)-based IDS. GENERTIA also uses a GA-based blue team in order to repair the holes it finds. The performance of the GA-based hackers is tested and measured according to the number of distinct holes that it finds. The GA-based red team's behavior is then compared to that of 12 variations of the particle swarm optimization (PSO)-based red team named SW0, SW0+, SW1, SW2, SW3, SW4, CCSW0, CCSW0+, CCSW1, CCSW2, CCSW3, and CCSW4. Each variant of the PSO-based red team differs in terms of the way that it searches for holes in an IDS. Through this test, it is determined that none of the red teams based on PSO perform as well as the one based on a GA. However, two of the twelve PSO-based red teams, CCSW4 and SW0+, provide hole-finding capabilities closest to that of the GA. In addition to the ability of the different red teams to find holes in an AIS-based IDS, the search behaviors of the GA-based hackers,

PSO-based hackers that use a variable called a constriction coefficient, and PSO-based hackers that do not use the coefficient are compared. The results of this comparison show that it may be possible to implement a red team based on a hybrid "genetic swarm" that improves upon the performance of both the GA- and PSO-based red teams.

VULNERABILITY ANALYSIS OF AIS-BASED INTRUSION DETECTION SYSTEMS USING GENETIC AND EVOLUTIONARY HACKERS

A THESIS

SUBMITTED TO THE FACULTY OF CLARK ATLANTA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF MASTER OF SCIENCE

BY

DOUGLAS BROWN, JR.

DEPARTMENT OF COMPUTER SCIENCE

ATLANTA, GEORGIA

MAY 2005

ACKNOWLEDGEMENTS

I offer my gratitude to everyone that made the completion of this thesis possible. I also thank my advisors Dr. Gerry Dozier, Dr. Roy George, and Dr. John Hurley for providing me with the insight, support, and guidance that I needed in order to conduct my research and complete this thesis. My sincerest thanks also go to the faculty and staff of Clark Atlanta University's Computer and Information Systems department for providing me with the skills and knowledge necessary to fully understand the material used to write this thesis. I thank Dr. Melvin Webb of the PRISM-D Program, Dr. Isabella Jenkins, and the NSF Scholarship for Service program for providing me with funding so that I could complete my education. Finally, I thank my parents, Douglas and Frances Brown, my brother Benjamin Brown, the Smart family, and my closest friends Chandrea Dungy and Shelia Washington for providing me with the emotional support that I needed to complete both this thesis and my education at Clark Atlanta University.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

AIS	Artificial Immune System
AP	Access Point
AV	Anti-Virus
BIS	Biological Immune System
CC	Constriction Coefficient
CDIS	Computer Defense Immune System
EC	Evolutionary Computation
EP	Evolutionary Programming
ES	Evolutionary Strategies
FN	False Negative
FP	False Positive
GA	Genetic Algorithm
GBT	GENERTIA Blue Team
GENERTIA	Genetic Interactive Teams for Intrusion Detection Design and Analysis
GP	Genetic Programming
GRT	GENERTIA Red Team
HIDS	Host-Based Intrusion Detection System
IDS	Intrusion Detection System
JSDT	Java Shared Data Toolkit

LISYS	Lightweight Intrusion Detection System
MAC	Medium Access Control
MOM	Message Oriented Middleware
NC	Negative Characterization
NIDS	Network-Based Intrusion Detection System
PC	Positive Characterization
PSO	Particle Swarm Optimization
PT	Particle Termination
RB	Random Best
WEP	Wireless Encryption Protocol
WLAN	Wireless Local Area Network

CHAPTER 1

INTRODUCTION

With each passing year, the Internet has experienced an increase in popularity. This increase in popularity has had many positive consequences such as allowing large numbers of people to learn how to take advantage of the numerous services that the Internet provides and to gain interest in the Information Technology field. Despite these benefits, there have been several negative consequences as well. For instance, as the popularity of the Internet has increased, the frequency of attempts to hack into private networks and individual computers, exploits of vulnerabilities in software and operating systems, and other forms of malicious attacks have also increased.[1] Naturally, Internet users wish to protect their computers from attacks and seek some sort of computer mechanism to protect their computers and their data. Computer security systems can take numerous forms such as anti-virus (AV) systems[2] and firewalls.[3] Another popular form of computer security is an intrusion detection system (IDS).[4] This form of computer security is the focus of this thesis.

This thesis primarily focuses on an application, named GENERTIA, which attempts to find and repair vulnerabilities in an IDS which could fail to prevent a hacker's attacks. GENERTIA is based on a problem-solving method known as a genetic algorithm (GA). In this paper, the GENERTIA application is tested to determine

whether it can discover 300 vulnerabilities in an IDS. Also, 12 versions of GENERTIA, based on another algorithm named particle swarm optimization (PSO), are compared to the GA version in order to compare their performances. The results of these two experiments are discussed later in this thesis.

The following five chapters of this paper cover the following topics: IDSs, evolutionary computation (EC), immunity-based IDSs, PSO-based GENERTIA, and the conclusion of the thesis. Chapter two covers IDSs and gives a basic overview of current intrusion detection methods. Chapter three discusses EC, a family of problem-solving algorithms which includes GAs and PSO. Chapter four, on immunity-based IDSs, discusses four systems, including GENERTIA, that work to protect computer systems from unauthorized access and attack. This chapter contains the results of the first experiment with the GA-based GENERTIA mentioned in the previous paragraph. Chapter five, discusses PSO-based GENERTIA and the results of the second experiment with GENERTIA, which compares the GA version with the 12 PSO versions. The final chapter, the conclusion, gives a summary of the thesis.

CHAPTER 2

OVERVIEW OF INTRUSION DETECTION SYSTEMS (IDSs)

2.1. Introduction to IDSs

The concept of the modern IDS was originally developed by Dorothy Denning in 1987 as a method to detect security breaches, known as intrusion attempts, and to produce alerts.[1] An intrusion attempt occurs when a user (or users) tries to access computer resources that they are not authorized to use.[2] As its name suggests, an IDS is used to detect the presence of these attempts. In addition to the detection of intrusion attempts, IDSs can also be employed to perform a number of other network security tasks such as preventing people from exploiting vulnerabilities in software, preventing the disabling of resources by attackers, or even keeping network users from viewing inappropriate websites.[3]

IDSs can suffer from two types of errors: false negatives (FN) and false positives (FP).[4] A FN occurs when an intrusion attempt goes undetected by the IDS; a FP occurs when normal system activity is labeled as an intrusion attempt. Both of these occurrences can decrease the effectiveness of an IDS. A high FP rate can cause actual attacks to go unnoticed while an IDS with a high FN rate will not provide adequate protection.

2.2. Types of IDSs

The methods used by an IDS to detect malicious activity can be divided into two categories: anomaly detection and misuse detection.[5] An anomaly detection IDS[6] depends on a profile of normal network or computer behavior to detect attacks or intrusion attempts.[7] In order to generate this system profile, an anomaly-based IDS uses a training period to observe "normal" system activity. Computer or network activity is compared to the behavior profile in order to detect the presence of activity that deviates from what is considered normal; any deviating activity is labeled anomalous and may be a sign of malicious activity. A benefit of anomaly detection is that it is able to reveal attacks that have not previously been encountered. This is because anomaly-based IDSs look for any abnormal activity.[8] Anomaly-based IDSs suffer from high rates of FPs since benign system activity may deviate from what is considered to be normal behavior. Another weakness concerns the training period necessary to generate a profile of normal system behavior. If any attacks occur during this period, the IDS will consider this behavior to be normal and will not detect those attacks if they occur later.[9] The second method of intrusion detection, misuse detection,[10] uses a database of attack patterns, similar to the virus signatures used in AV systems, to detect malicious system activity.[11] Computer and network activity is compared to these attack patterns, and if any similarity between the two is found, an alarm is generated. A benefit of this type of IDS is that the attack signatures are able to be used in several different computing environments. However, while this method can accurately detect attacks whose patterns are stored in the pattern database, it is unable to detect new forms of attack.[12] Also, since this form of IDS

relies entirely on the database to detect attacks, the attack pattern database must be updated regularly.

IDSs can be further divided into categories according to how they are situated on a network: host-based or network-based.[13] A host-based IDS (HIDS)[14] provides security by residing on the computer that needs to be protected. In order to detect the presence of intrusion attempts, a HIDS monitors a computer's log files and incoming network traffic for any potentially harmful activity.[15] The benefits of a HIDS include the ability to monitor specific system activities such as attempts to access files and the fact that these types of IDSs do not require any specific hardware in order to be deployed. While these types of IDSs can be useful, they are susceptible to evasion, a process where attackers intrude into a system without generating an alarm.[16] This can be accomplished in many ways, one of which is avoiding the use of system calls. HIDSs are also prone to being disabled by an attacker.[17] A network-based IDS (NIDS)[18] provides protection by using intrusion detection software called a monitor or a sensor situated in one or a few locations on the network to observe all network activity.[19] These monitors analyze the traffic on the network in order to detect the presence of malicious activity. Since there are only a few monitors used in a NIDS, the administrator of a NIDS only has to manage a small number of sensors. Other benefits of using this type of IDS include a lower deployment cost due to the lower number of network monitors needed and the ability to detect attacks against multiple hosts.[20] Even though this method of intrusion detection may be more attractive for networks that have a large number of hosts to be monitored, these IDSs have some weaknesses as well. For instance, NIDSs suffer from the inability to provide as much information about attacks on specific computers as HIDSs can.[21]

2.2.1. Types of NIDSs

NIDSs can be further divided into three groups according to their overall structure: monolithic, hierarchical, and cooperative.

2.2.2. Monolithic NIDSs

A monolithic NIDS functions by having each sensor on the network send data regarding network traffic to a single server which analyzes the data in order to detect attacks.[22] While this method is the simplest and can work well for smaller networks, when the number of hosts on the system increases, the performance of the IDS will decrease due to the increased amount of data that the server would have to process.[23] Also, the single server presents a single point of failure; in order to disable the IDS, an attack would simply have to disable the server.

2.2.3. Hierarchical NIDSs

A hierarchical NIDS divides a network into sections and uses a separate sensor to monitor each section.[24] Sensors in this type of IDS are arranged in a hierarchy; local sensors send information regarding their sections of the network to another sensor higher up in the hierarchy. This method is similar to a monolithic NIDS because it uses a central server, except the central server only has to analyze the data transferred to it from the sensors at the level directly below the server to detect attacks. As a result, more hosts can be added to the system without significantly increasing the amount of data that the server must process, making this type of IDS more scalable.[25] While this type of IDS

eliminates the problem of scalability present in monolithic NIDS, it still presents a single point of failure at the highest level in the hierarchy.[26]

2.2.4. Cooperative NIDSs

A cooperative NIDS differs from the other forms of NIDSs in that it uses several host-based monitors that communicate with each other.[27] The functionality of the central server is distributed among the different individual monitors that make up the system. These individual monitors analyze traffic on each host and communicate with one another in order to determine whether or not an attack has occurred. Co-operative NIDSs eliminate the central server present in both monolithic and hierarchical NIDSs and, as a result, eliminate the single point of failure present in these approaches. While this method offers a major improvement over monolithic and hierarchical approaches, it can suffer from decreased performance due to the communication between the host-based monitors that make up the IDS.[28]

2.3. Wireless IDSs

The IEEE 802.11 wireless local area networks (WLANs) suffer from the same types of attacks that wired LANs do in addition to other types of attacks due to flaws in the 802.11 standard and the wireless nature of an 802.11 network.[29] The additional threat to WLANs makes IDSs developed for wired networks unable to provide adequate protection for a WLAN. As a result, wireless IDSs must be designed so that they can detect the presence of both "normal" wired network attacks and wireless network attacks. A wireless IDS must be able to handle occurrences that only affect WLANs such as

cracking the Wireless Encryption Protocol (WEP) key, spoofing Medium Access Control (MAC) addresses, and broadcasting deauthenticate frames.[30] The first attack, cracking WEP keys, involves a weakness in WEP, a method used to encrypt data transmitted across a WLAN.[31] This method of encryption uses an algorithm called RC4 which has a major weakness that allows the key used in order to encrypt data to be obtained by "sniffing," or collecting, enough encrypted data. Once the key has been obtained, the encrypted data can be deciphered. Tools such as AirSnort[32] and WEPCrack[33] can automate this type of attack, making it far easier to execute.[34] The second attack, spoofing MAC addresses, exploits the MAC address-based authentication that many types of WLANs use to authenticate users.[35] In order to gain access to the WLAN, an intruder simply has to change the MAC address of his or her network card to that of a user that is allowed to use the network. This process is known as spoofing. The third attack, broadcasting deauthenticate frames,[36] exploits the method that access points (APs)[37] use to disconnect users from a wireless network. An AP is a node on a wireless network that users can connect to in order to get wireless service. In an 802.11 network, an AP can disconnect a user by sending it a deauthenticate frame. An attacker can exploit this behavior by spoofing the address of an AP and sending a deauthenticate frame to the broadcast address, causing every computer connected to that AP to receive it and be disconnected from the WLAN.[38]

Examples of wireless IDSs that can protect WLANs against these attacks and others are AirDefense[39] and AirMagnet[40]. AirDefense provides protection against intruders by supplying users with a system of sensors that connect with a separate console that can be used to monitor intrusion attempts and possible misconfigurations of wireless

equipment. The AirMagnet system, on the other hand, is a software-based IDS that can be run on a computer, usually a laptop, and requires that a user walk around the area to be protected in order to detect attacks.

CHAPTER 3

OVERVIEW OF EVOLUTIONARY COMPUTATION (EC)

3.1. Introduction to EC

EC can be defined as using algorithms that mimic the process of evolution in order to solve problems.[1] These algorithms evolve a population comprised of possible problem solutions called individuals or candidate solutions until a solution for a problem is found. While there are different types of EC-based algorithms, they all generally follow the same process in order to solve problems.[2] First, a population is randomly generated, and each individual is given a fitness. The fitness of an individual determines the "quality" of the individual, often how close the individual is to being the optimal solution for a problem.[3] Afterwards, the population is used to create a new population where individuals are altered in order to allow the population to evolve to the point where an optimal solution is found. The process of determining fitnesses and generating and evolving new populations is repeated until a stopping condition is met.[4]

3.2. A Brief History of EC

Early work in EC was done by Bremermann, Friedberg, and others during the 1950s and 1960s.[5] Out of this early work grew new fields such as genetic

algorithms (GAs), evolutionary programming (EP), evolutionary strategies (ESs), and genetic programming (GP).[6] Although EP, ESs, and GP were both instrumental in the development of the field of EC, they are not detailed here as GAs are far more relevant to this application.

3.3. Introduction to Genetic Algorithms (GAs)

The origins of GAs can be found in the work of John H. Holland who worked in the area of EC during the 1960s.[7] With a GA, problems are solved through a process that mimics natural selection. Individuals in a population that are more "fit," or have a higher fitness, are more likely to pass on their "genes" to other generations. A GA is similar to other algorithms rooted in EC in that it evolves a population of candidate solutions in order to solve a problem or to find the optimum value of a function. However, GAs differ from other forms of EC in that individuals in a population are referred to as chromosomes. These chromosomes are made up of genes, and each gene has values known as alleles.[8] When chromosomes are represented as binary strings, each value in the string is an allele. Alleles in chromosomes can also be represented as a numerical value.[9] In this method, the value of each allele must stay within the range of valid values for the variable it represents. These two representations are known as binary-coded and real-coded, respectively. A pseudocode example of a GA can be found in Figure 1.

```
Procedure GA{
    t = 0;
    Initialize P(t);
    Evaluate P(t);
    While (Not Done)
    {
      Parents(t) = Select_Parents(P(t));
      Offspring(t) = Procreate(Parents(t));
      Evaluate(Offspring(t));
      P(t+1)= Select_Survivors(P(t),Offspring(t));
      t = t + 1;
    }
}
```

Figure 1: A pseudocode GA

3.3.1. Problem Solving with GA

The initial steps in a problem with a GA are similar to those found in many EC-based algorithms: a population is randomly generated, and the fitness of each member of the population is determined. After these two steps are completed, a new population is generated. In order to create this new population, a process known as reproduction or duplication occurs. In reproduction, an individual is picked, and a copy of it is added to the new population.[10] Several different methods can be used to determine which individuals are chosen for the new population.

Three popular methods of selection are proportionate selection, ordinal-based selection, and tournament selection.[11] In proportionate selection, individuals are given a probability that they will be selected based on their weight; the higher the weight, the more likely they are to be selected.[12] In ordinal-based selection, the assigned probability that an individual will be chosen is based on the individual's rank.[13] In order to determine

the rank, members of the population are ordered by their fitness; the higher an individual's fitness is, the higher its ranking will be. A higher rank means that there will be a higher probability that the individual will be selected. In tournament selection, a randomly selected group of individuals undergoes a "tournament" where the individual with the highest fitness "wins" and is copied into the new population.[14] This tournament process is repeated until the proper number of individuals has been copied.

After the parents are selected, the new population is altered, usually through processes known as crossover and mutation.[15] Crossover is similar to the exchanging of genes that occurs during the sexual reproduction of two organisms. In the crossover process, the population is divided into pairs known as parents, and a random number between 0 and 1, called the crossover rate, is generated. The crossover rate determines the probability that two individuals will undergo crossover.[16] When two parent individuals actually undergo crossover, they usually "swap" parts of their chromosomes in order to create two new individuals known as offspring. Otherwise, the parents are copied directly into the new population. An example of this swapping process can be seen in (a) in Figure 2. This method of crossover is known as two-point crossover since two crossover points are used. The pipe characters that delineate the bolded sections of (a) and (b) represent crossover points. These crossover points are randomly selected before crossover occurs and tell the GA which parts of the parents' chromosomes are to be swapped in order to create new offspring.[17] Other forms of crossover include single-point crossover, where only one crossover point is used, and uniform crossover, where the parent whose gene is to be used for each allele of the offspring's chromosome is

chosen randomly.[18] Examples of these two types of crossover can be seen in (b) and (c) in Figure 2. Other forms of crossover for real-coded GAs can be found in.[19]

Parent		Child
10|**0100**|11	->	10110111
11|**1101**|01	->	11010001

(a) Two-Point Crossover

Parent		Child
10|**010011**	->	10110101
11|**110101**	->	11010011

(b) Single-Point Crossover

Parent		Child
10010011	->	11010001
11110101	->	10110111

(c) Uniform Crossover

Figure 2: Different types of crossover: two-point, single-point, and uniform

After the reproduction phase, the new offspring may then undergo mutation, a process where alleles in each offspring's string are altered with a probability of p_{μ} which is called the mutation rate.[20] In binary-coded GAs, the most basic form of mutation can be performed by flipping bits in the offspring's string from 0 to 1 or vice versa. For real-coded GAs, mutation is performed by altering the value of an allele to a value within the range of possible values for that allele.[21] Additional mutation procedures for real-coded GAs can be found in the work of Herrara, Lozano, and Verdegay.[22]

After offspring are reproduced, undergo crossover, and are mutated, they may replace their parents. GAs that use this method of replacement are called Generational GAs.[23] The offspring may also replace the least fit or oldest member of the population.

This type of GA is called a Steady-State GA.[24] This process of selection, procreation, and replacement is repeated until an appropriate solution for the problem to be solved.

3.4. Introduction to Particle Swarm Optimization (PSO)

Another technique related to the field of EC is PSO[25] which applies sociological theory regarding how people or animals communicate and learn from one another to EC. Unlike other EC algorithms, in PSO, individuals are called particles, and the population is called a swarm. PSO uses the swarm of particles to solve, or find the optimum solution for, a problem. Also, unlike EC algorithms like GAs, particles are never replaced in the swarm. The name of this algorithm is derived from the fact that the particles in PSO behave similarly to "swarming" animals such as a flock of birds or a school of fish.[26]

3.4.1. Overview of Particles in PSO

A particle consists of 3 vectors: the x-vector, the p-vector, and the v-vector.[27] Each vector contains a number of elements equal to the number of dimensions in the problem space; this number of dimensions is equal to the number of variables in the problem to be solved. The x, v, and p vectors each serve a different purpose in a PSO. The x-vector stores the current position of the particle. Each element in this vector corresponds with a single dimension of the search space and stores a coordinate for the dimension that it represents. The p-vector, on the other hand, is used to store the particle's best recorded position. An evaluation function is used to determine the fitness of the particle's current position, and if the particle's current position has a higher fitness than that of all of the particle's previous positions, the position stored in the current x-

vector is stored in the p-vector. The third vector, the v-vector, is added to the x-vector in order to determine the particle's next position.

3.4.2. Problem Solving with PSO

The initial steps necessary to solve a problem with PSO are identical to those found in other EC algorithms. First, a swarm of particles must be generated. This initial swarm population is randomly generated, meaning that values of each particle's x- and v-vectors are initialized to random values.[28] Next, the fitness of each particle's x-vector is compared to that of its p-vector; if the x-vector's fitness is higher, the current x-vector becomes the new p-vector value. A particle's p-vector is then compared to that of its neighbors, i.e., the group of particles that are "close" to a particular particle.[29] The size and members of a particle's neighborhood depend on the swarm's topology. Two popular topologies are circle and star.[30] In a circle topology, a particle's neighborhood consists of its N closest neighbors, where N is less than the total number of particles in the swarm. In a star topology, a particle's neighborhood is every other particle in the swarm. When a particle with the best p-vector (the one with the highest fitness) in a neighborhood is found, it is labeled the lbest for a circle topology or the gbest for a star topology.[31] The neighborhood's best particle's p-vector is used to calculate the new v-vector for each particle in the neighborhood. In order to make a particle move, a new value for the v-vector is calculated and is added to the particle's current location. The formulas used to calculate the value of the v-vector can be found below.

$$v_{id} = v_{id} + (\varphi 1 \text{ rnd}() (p_{id} - x_{id})) + (\varphi 2 \text{ rnd}() (p_{gd} - x_{id})) \quad (1)$$
$$x_{id} = x_{id} + v_{id} \quad (2)$$

Formulas used to calculate the new v-vector value for a particle. The i represents the index of the particle whose v-vector is being updated, and g is the best performer in the neighborhood. The d represents the dimension of the v-vector whose value is being calculated. The ϕ_1 and ϕ_2 variables are called the cognition component and the social component.

The ϕ_1 and ϕ_2 variables present in the formulas, used to update a particle's v-vector, are called the cognition component and the social component, respectively.[32] In sociology, the social component represents the amount that a person's decisions are influenced by the decisions of those around him or her. With respect to the PSO, the social component determines how much the value of the particle's v-vector is affected by the neighborhood's best performer.[33] The cognition component represents the amount that a person's decisions are influenced by his or her own understanding and experiences. With respect to the PSO, the cognition component determines how much the value of the particle's v-vector is affected by the particle's own "experience." A particle's experience can be considered to be its best performance, determined by its p-vector value.[34] A particle swarm can allow a particle's previous best performance or the best performance of other particles in the swarm to have more of an impact on the movement of the particle. This is accomplished by altering the values of the social and cognition components.[35] To have the performance of the global best particle has more of an impact on the particle's new location, the social component could be made greater than the cognition component. This causes the v-vector value calculated in Equation 1 to be more affected by the global best particle's p-vector, which is multiplied by the social component, $\varphi 2$. To make the individual particle's own experience more influential, the

cognition component could be made greater than the social component. This causes the value calculated in Equation 1 to be more affected by the particle's own p-vector which is multiplied by the cognition component, $\varphi 1$.

The process of updating the particle's vectors to allow it to move about the problem space is repeated until an optimum solution is found or until some other condition is met.

3.4.3. Improving PSO Performance Through Velocity Control

The velocity of a particle can be controlled in order to increase or decrease the precision of the swarm's search capability or to keep its velocity from becoming too large.[36] The velocity can be controlled in a variety of ways. One method is the use of a Vmax, a maximum value for the magnitude of a particle's velocity.[37] This variable creates a range from –Vmax to Vmax that the particle's velocity must remain within. This is the most basic form of particle velocity control. Another method is the use of an inertia weight, represented as ω.[38] The inertia weight is used to change how much the old v-vector value affects the new v-vector value. This value is multiplied by a particle's old v-vector value when the new v-vector value is calculated.[39] Equations 3 and 4 show the inertia weight version of v-vector formula. A third method of velocity control is the constriction coefficient, represented as K.[40] The use of this variable has also been shown to improve swarm performance.[41] Equations 5 and 6 show an updated version of the v-vector formula which uses the constriction coefficient, and Equations 7 and 8 show the formula used to calculate the constriction coefficient.

$$v_i = (\omega\, v_{id}) + (\varphi 1\ \text{rnd}()\ (p_{id} - x_{id})) + (\varphi 2\ \text{rnd}()\ (p_{gd} - x_{id})) \quad (3)$$
$$x_i = x_i + v_i \quad (4)$$

The v-vector formula with an inertia weight of ω

$$v_i = K\,(v_{id} + (\varphi 1\ \text{rnd}()\ (p_{id} - x_{id})) + (\varphi 2\ \text{rnd}()\ (p_{gd} - x_{id}))) \quad (5)$$
$$x_{id} = x_{id} + v_{id} \quad (6)$$

The v-vector formula with a constriction coefficient of K

$$K = 2/|\,2 - \varphi - \text{sqrt}(\varphi^2 - 4\varphi)\,| \quad (7)$$
$$\varphi = \varphi 1 + \varphi 2,\ \varphi > 4 \quad (8)$$

The formula used to calculate K

3.5. Artificial Immune Systems (AISs)

Hofmeyr and Forrest[42] propose a network IDS called Lightweight Intrusion Detection System (LISYS) which helps to detect network-based attacks on broadcast LANs. This IDS was based on an artificial immune system (AIS) named ARTIS. The purpose of ARTIS was to develop a system that exhibits the properties of a biological immune system (BIS) such as robustness, adaptivity, and autonomy that make it effective in fighting infection in the body. This goal is accomplished through the use of components that replicate the behavior of the special cells called lymphocytes that are present in a BIS.[43] Lymphocytes are a type of white blood cell and respond to the presence of harmful elements in the body.[44]

3.5.1. An Overview of the BIS

One of the basic functions of a BIS is to detect the presence of foreign, potentially harmful elements, known as pathogens, in the body.[45] In order to accomplish this, the BIS divides elements that it observes into two sets: self, or cells that belong in

the body, and non-self, foreign cells that do not belong in the body and may be harmful.[46] To detect non-self elements in the body, cells called lymphocytes are used.[47] These cells are covered with special proteins called receptors. Receptors allow the cell to bind itself to parts of the pathogen called epitope. Whether or not a lymphocyte and a pathogen will bind depends on the affinity of the receptor and the epitope; the affinity is determined by the similarity of the shape and electrical charge of both the epitope and the receptor.[48]

The immune system protects the body by generating a set of lymphocytes from the stem cells present in bone marrow.[49] In the marrow, lymphocytes are given a randomly generated set of receptors. These cells then undergo a training process in the thymus, a gland located in the throat, where the lymphocytes are exposed to self cells; if a lymphocyte binds itself to a self cell, it is destroyed.[50] This process is called negative selection. After the training process is completed, ideally, a set of lymphocytes that will only bind with non-self elements in the body will be created. This set can then be used to detect the presence of pathogens in the body. The BIS is also able to "memorize" previously observed pathogens in order to increase its speed and efficiency.[51] This is accomplished through the use of memory cells. When a lymphocyte has been bound to a pathogen, the lymphocyte becomes what is known as a memory cell. These memory cells memorize previously encountered pathogens in order to increase the speed of the BIS's response to potentially harmful agents in the body and to allow the BIS to detect pathogens with similar structures to previously encountered ones.[52]

3.5.2. AIS Design

AISs are designed so that they have components and behaviors that are analogous to those in a BIS.[53] The lymphocytes that are used to detect the presence of non-self elements in a BIS are represented by detectors. The receptors that cover the surface of lymphocytes and the epitope regions of a pathogen are both represented by fixed-length binary strings.[54] In a computer system, a pathogen is considered to be an attack, a virus, or any other potentially harmful network traffic or computer activity. In order to replicate the negative selection process by which lymphocytes are trained to only bind with non-self cells, an AIS first exposes newly created, or immature, randomly generated detectors to a training set of self strings for a time period T.[55] This time period is known as the tolerization period, and any detectors that match a self string during this period are eliminated from the set. This process creates a set of mature detectors that will only bind to non-self elements. These mature detectors are then released into the AIS's distributed environment which is represented as a graph.[56] The graph consists of a set of vertices and a set of edges that connect those vertices. Each vertex in a graph has its own set of detectors, and each edge can be used by a detector in order to travel from one point to another. If one of these mature detectors happens to match the number of non-self elements required by the activation threshold (τ), the number of non-self elements a mature detector must match in order to become activated, a signal that a potentially harmful non-self element has been found is produced.[57] Since in the body, the lymphocytes are able to bind with several similar kinds of pathogens, the AIS uses approximate matching rather than exact matching so that detectors can be bound to more than one type of non-self element.[58] This functionality increases the system's ability to

find non-self elements. ARTIS also implements the memory-based detection of the immune system. A memory detector is created when more than one mature detector is matched by a non-self string. The AIS compares the extent to which each of the matching detectors matches the non-self string, and the closest matches are promoted to memory detector status.[59] Copies of these memory detectors are distributed to nearby nodes, improving each of the neighboring nodes' ability to match and respond to that particular non-self string and others like it.

CHAPTER 4

IMMUNITY-BASED INTRUSION DETECTION SYSTEMS

4.1. Introduction to Immunity-Based IDSs

Much research has been done in the area of immunity-based IDSs.[1] As a result of this research, several IDSs have been developed that apply the characteristics of a BIS to the field of intrusion detection.[2] Out of these many IDSs, the ones that are most relevant to this thesis are those developed by Hofmeyr and Forrest,[3] Harmer et al., [4] and Dasgupta.[5]

4.2. Lightweight Intrusion Detection System (LISYS)

The focus of Hofmeyr and Forrest[6] is LISYS, an application of ARTIS aimed at providing network intrusion detection capabilities. LISYS uses data structures called datapath triples in order to represent both self and non-self traffic.[7] These triples consist of the source and destination IP addresses of two communicating computers and the TCP port used by the two machines. In order for this datapath information to be used by LISYS, it is converted into a 49-bit string used to uniquely identify each connection across the network. In LISYS, self traffic is considered to be any normally occurring connections and non-self is considered to be any abnormal connections.[8] In LISYS, the distributed

environment of ARTIS is the network to be monitored. The network is represented as a fully connected graph which consists of a set of vertices, which represent computers, and connective edges, which represent connections between the computers. Since LISYS is being applied to a broadcast LAN, it can be assumed that every computer is connected to every other computer on the network, hence the fully connected graph. At each vertex in the graph, there exists a set of detectors that the computer can use in order to detect non-self activity.

The LISYS offers several improvements over the forms of network intrusion detection that exist today.[9] Many current NIDSs, such as Snort, use signature-based detection in order to detect attacks.[10] Signature-based systems search packets for strings called attack signatures that are associated with a specific type of attack.[11] If an attack signature is found in a packet, an alarm is generated. This approach to intrusion detection suffers from problems that decrease its effectiveness.[12] For instance, the use of attack signatures requires a human operator to ensure that the signature database is always up to date and to manually add new signatures. Because of this property, NIDSs cannot automatically adapt to new forms of attack or to attacks with signatures similar to those of already existing ones.[13] Another problem is that signature based NIDSs are susceptible to high FP rates.[14] This high false alarm rate is caused by the presence of benign packets on the network which happen to contain a piece of data that matches an attack signature. This weakness also causes signature-based NIDSs to be susceptible to attacks that intentionally generate high numbers of FPs in order to conceal the alarms produced by actual attacks.[15] Also, NIDSs often employ intrusion detection sensors at several locations in the network.[16] These sensors collect data and send them to a central

server so that they can be analyzed and attacks can be detected. The centralized server introduces a single point of failure into the NIDSs, allowing the system to be easily disabled if the server is attacked.[17] Since the intrusion detection sensors will most likely be similar, it may be possible to disable the entire system with a single type of attack that takes advantages of a vulnerability in the NIDS software.[18] A third issue with this method is that if the number of sensors becomes large, performance will suffer due to the large amount of data that the server must process. [19]

The AIS that the LISYS is based on allows the system to avoid these shortcomings of current NIDSs.[20] The lack of required communication between the components of LISYS allows the system to be less vulnerable to attack. If a node or several nodes are disabled, the system should still be able to perform sufficiently.[21] Also, since components of the system do not communicate with a central server or with each other, the system is more scalable.[22] More nodes can be added without increasing the amount of data that has to be analyzed in order to detect anomalous activity and to protect the network. Also, the approximate signature matching capability of the LISYS allows the system to detect the presence of attacks with similar signatures. This ability mimics the BIS's ability to detect the presence of pathogens that have chemical and molecular structures that are similar to others.[23] The LISYS is also able to "teach" itself about new attacks and changes in the behavior of the system without needing a human operator to constantly update a signature database.[24] This is accomplished through the occasional death of mature detectors and their replacement with new, immature detectors.[25] The constant introduction of new trained detectors allows the detector set to cope with changing self and non-self sets. Another major benefit of the LISYS is its accuracy.

Whereas current intrusion detection systems, such as those researched by Lippman et al. in,[26] suffer from high FP rates, the LISYS uses methods such as activation thresholds and tolerization which decrease the number of FPs generated.[27]

4.3. Computer Defense Immune System (CDIS)

Harmer et al.[28] proposes an agent-based IDS called the Computer Defense Immune System (CDIS) based on a BIS which focuses primarily on the detection and elimination of computer viruses. This system aims to overcome the shortcomings that plague current AV systems. A major problem with current AV systems is that viruses are being produced at a faster rate than researchers can learn how to eradicate them.[29] When a new virus is discovered, virus experts analyze the virus's code, behavior, and the signature it contains in order to update virus scanners to handle the new threat. The time-consuming nature of this process, along with the rapid increase in the rate at which viruses are being developed and deployed, threatens to make the problem of eradicating viruses impossible to handle.[30] Harmer et al.[31] aims to solve this problem by basing their application on a BIS so that it can exhibit properties such as adaptivity, autonomy, and selective response.

4.3.1. CDIS System Hierarchy

The CDIS is divided into a three-layer hierarchy which consists of a system layer, a network layer, and a local layer. The system is broken up in this manner in order to decrease the amount of resources needed to run the system by splitting up the system's tasks.[32] Each of these layers has different functions that, when combined,

provide an effective method of identifying and eliminating viruses. The system level determines what problems are occurring throughout the entire system, stores memory detectors, and provides information on handling viruses that it receives from the network layer to all of the computers in the system. The next layer, the network layer, works to protect a local group of computers. It determines what viruses have infected the computers in the group, reports to the system layer, and gives vaccinations for viruses to computers in its group.[33] The lowest layer, the local layer, operates in each individual computer. Each computer has a set of detectors which detects the presence of virus infection and allows alarms to be produced. The size of this detector set is limited in order to decrease the computational cost associated with generating the set.[34] Since information about protection from a particular virus is always shared throughout the system, this limited detector set does not decrease the effectiveness of the system.

4.3.2. Agents of CDIS

As mentioned earlier, the CDIS is an agent-based system. The functionality of the CDIS is divided among seven types of agents: antibodies, detectors, monitors, helpers, classifiers, cleaners, and controllers. These agents provide functions similar to the different entities of a biological immune system. Their functions are summarized below and are described more in depth in Harmer et al.'s work:[35]

- Antibody – Generates detector strings that will only match non-self strings, strings that may represent code associated with a virus
- Detector – Uses several antibodies in order to determine if a string is associated with a virus

- Monitor – Communicates with controller agents and produces alarms
- Helper – Validates and tells user about alarms
- Classifier – Determines what type of virus has infected the system and how it should be fixed
- Cleaner – Eradicates viruses and repairs damage done
- Controller – Coordinate the activity of the entire system

Each of these agents can interact in a variety of ways with other agents in order to use different services to complete tasks. For instance, the generation of an alarm requires communication between a detector and monitor.[36] In order to allow the different agents in the system to collaborate and interact to replicate the functionality of an immune system, CDIS uses a combination of Message Oriented Middleware (MOM) and the Java Shared Data Toolkit (JSDT). The MOM used in this system is called AgentMOM. AgentMOM is a communications framework geared towards the development of multiagent systems and supports broadcast, multicast, and secure communication between agents.[37] JSDT is software that allows for the development of Java applications that can interact with one another.[38]

4.3.3. Benefits of CDIS

The system proposed by Harmer et al.[39] provides several improvements over current AV systems. First, the organization of the system into a hierarchy allows different tasks to be delegated to different layers of the application. This allows the cost of executing the CDIS to be spread across all computers using the system. Another benefit is the application's ability to disseminate information related to the eradication of

viruses to all computers in the system and to be able to detect unknown viruses through constant introduction of new detectors.[40] This capability removes the need for virus experts to rush to determine how to eliminate newly developed viruses and for computer users to constantly update virus scanners with new signature databases. Because of these benefits, this method of virus detection and elimination is a vast improvement over current AV systems.

4.4. Negative Characterization (NC)

Dasgupta[41] proposes an anomaly detection system based on an AIS that can detect the level of abnormality of traffic on the network. His approach differs from those used in other AISs[42] in that it does not divide activity on networks and computer systems into "self" and "non-self." Dasgupta chooses this approach because the behavior of a system cannot simply be divided into "good" and "bad" categories.[43] While the behavior of a computer system or network may stray from what is considered normal, this deviation may be caused by benign occurrences such as the installation of new software or letting another person borrow a computer.[44] This harmless activity may trigger alarms in some AIS-based security systems. In order to correct this shortcoming, Dasgupta proposes a system that would divide traffic data into multiple categories based on the traffic's level of deviation from what is considered normal. In order to accomplish this, a method called negative characterization (NC) is used which extends the negative selection process found in the works of others.[45] In NC, a variability parameter is used which determines how much a sample is allowed to deviate from the self space, the group of strings which represents normal behavior without being labeled as non-self.[46] The

parameter is used as an input to a GA along with sets of data that represent the normal behavior of the system in order to produce a set of rules that can effectively detect malicious activity. In order to detect varying levels of abnormality, the NC method uses different values for the variability parameter.[47]

4.4.1. NC vs. Positive Characterization (PC)

Dasgupta[48] compares the performance of NC to that of a method called positive characterization (PC). PC stores the entire collection of strings that represents normal behavior and tests incoming samples to determine whether they belong to the self set or the non-self set. While this is an effective method of detecting the presence of non-self elements, it suffers from the fact that a large number of samples must be stored.[49] Both of these methods were tested against the Lincoln Laboratory's 1999 intrusion detection data set.[50] This data set contains five weeks of network traffic which includes simulated network attacks. In Dasgupta's paper,[51] only the first two weeks of data are used, and only five attacks are considered: back, 2 portsweep attempts, satan, and neptune. These attacks are explained in detail in the DARPA Intrusion Detection Evaluation.[52] During the test, Dasgupta discovered that while PC outperforms NC, it is only by a small margin. Since NC requires far less resources than PC and produces similar results, NC is a more preferable algorithm.

4.5. Overview of GENERTIA

The Genetic Interactive Teams for Intrusion Detection Design and Analysis (GENERTIA)[53] system is designed to improve the performance of AIS-based IDSs[54] by

detecting vulnerabilities and repairing them. These vulnerabilities are referred to as holes and represent attacks that the AIS detector set cannot detect.[55] The GENERTIA system consists of two components based on GAs:[56] the GENERTIA red team (GRT), which is used to detect holes, and the GENERTIA blue team (GBT), which is used to patch them.[57] These two subsystems work together in order to correct holes in an AIS-based IDS. The GRT detects holes in the IDS by first using its GA to generate a group of packets, called "red" packets, which represent possible attacks against a host or network. These packets are then exposed to the AIS's detector set to see if the set can detect the malicious packet; if an attack packet goes completely undetected, it is registered as a vulnerability. Afterwards, the GRT collects information related to the detected vulnerabilities and passes it to the GBT which produces patches for these vulnerabilities. This process should decrease the FN rate of an AIS-based IDS.

4.5.1. Testing GENERTIA

In order to test GENERTIA's ability to detect vulnerabilities, an experiment was conducted. In this experiment, network traffic is represented by datapath triples.[58] A datapath triple consists of the four octets of the destination IP address, the port being used by the remote host, and a direction flag which determines whether the packet it represents is part of inbound or outbound traffic. The AIS uses constraint-based detectors[59] that consist of six value ranges in order to detect malicious data triples.[60] The first four ranges represent the set of IP addresses that the detector can detect; each range represents one octet of the IP address. The fifth value range represents the range of ports that the detector can detect. The sixth value range can have the value of 0 or 1; 0

represents incoming traffic, and 1 represents outgoing traffic. In order to determine whether or not a packet and a detector match, a matching threshold[61] is used. The value of the matching threshold determines how many fields in a malicious data triple must be in the detector's corresponding value ranges in order to be considered a match. For instance, if the matching threshold is 4, 4 of the datapath triple's fields must fall into the ranges of the detector in order to be considered a match. In this experiment, the matching threshold is set to 3.

(192.168.0.16, 68, 0)

Figure 3: An example of a datapath triple. 192.168.0.16 is the destination host, 68 is the port being used by the remote host, and 0 is the direction flag.

In order to train and test the AIS, the 1998 MIT Lincoln Lab Intrusion Detection data set[62] was used. Only traffic involving one host (172.16.112.50) was used and all of the packets were converted into the datapath triple format. 112 packets that represented normal network traffic were used to train the AIS's detector set, and 1604 packets that represent the attacks were used to test the AIS. In order to test the AIS, the detectors were exposed to a packet from the training set in order to generate a set of mature detectors. Next, the AIS was tested with the testing set. Finally, the GRT was used to generate 300 red packets in order to detect 300 vulnerabilities in the AIS within 5000 iterations of the system. This experiment was executed 10 times each with 100, 200, 400, and 800 detectors. In this experiment, the AISs with detector set sizes of 100, 200, and 400 averaged 300 vulnerabilities while the AIS with the detector set size of 800 averaged

292.6 vulnerabilities. It was also observed that all 4 GRTs were able to detect at least one vulnerability within 1000 iterations of GENERTIA. While these performance rates are impressive, it may be possible to improve on GENERTIA's ability to detect vulnerabilities through the use of other EC-based methods such as PSO.

CHAPTER 5

PARTICLE SWARM-BASED GENETIC RED TEAMS (GRTs)

5.1. Overview of Particle Swarm-Based GRTs

As was mentioned in the previous chapter, although GA-based GRTs are able to find 300 holes in an AIS-based IDS, it may be possible to improve GENERTIA's performance through the use of red teams based on PSO. The goal of the PSO-based GRT is the same as that of the GA-based GRT: to evolve a population that will detect 300 holes in an AIS-based IDS.[1] In order to accomplish this, the PSO-based GRT generates particles that consist of an x-vector, which records the current red packet represented by the particle, a p-vector, which records the "best" red packet that this particle has represented, and a v-vector, which is used to generate a new red packet. Both the x- and p-vectors have associated fitnesses which determine what percentage of the AIS-based IDS a red packet was able to evade.

5.2. Comparison of GRTs

The original GA-based GRT's performance was compared to that of 12 different variants of PSO-based GRTs. The twelve swarm-based GRTs are named SW0, SW0+, SW1, SW2, SW3, SW4, CCSW0, CCSW0+, CCSW1, CCSW2, CCSW3, and CCSW4. The PSO-based GRTs differ in terms of the use of a constriction coefficient (CC),

neighborhood size, the use of particle termination (PT),[2] and the use of a random best particle (RB).[3] PT determines whether or not a particle will stop searching for vulnerabilities once one has been encountered, and RB determines whether or not the best particle used to update each particle's v-vector is randomly selected or not. Table 1 shows the features of each of the twelve swarms.

Alg	CC	Neighborhood	PT	RB
SW0	no	local	no	no
SW0+	no	local	yes	no
SW1	no	global	no	no
SW2	no	global	no	yes
SW3	no	global	yes	no
SW4	no	global	yes	yes
ccSW0	yes	local	no	no
ccSW0+	yes	local	yes	no
ccSW1	yes	global	no	no
ccSW2	yes	global	no	yes
ccSW3	yes	global	yes	no
ccSW4	yes	global	yes	yes

Table 1: The features of each of the 12 swarm-based GRTs

The GA and the 12 swarms were compared using the same experiment used to test GENERTIA described in the previous chapter. The only difference is that rather than test detector sets of 100, 200, 400, and 800, only detector sets of 400 were used. The results of the experiment can be seen in Table 2.

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
ccSW0	129.7	1.0	128.7
ccSW0+	275.0	3.0	272.0
ccSW1	272.8	27.8	245.0
ccSW2	284.6	50.5	234.1
ccSW3	296.1	25.4	270.7
ccSW4	297.1	16.7	280.4

Table 2: Results of the comparison of the GA-based and the 12 PSO-based GRTs. The holes column shows the average (over 10 trials) number of holes found by each GRT, the duplicates column shows how many duplicate holes each GRT found, and the distinct holes column shows how many of the holes found by each GRT were unique.

After running each GRT 10 times, it was determined that the GA still had the best overall performance, measured by the average number of distinct holes found over the 10 trials. Among the swarms, CCSW4 had the best performance overall, followed by SW0+. CCSW4 and CCSW0+ had the best performance out of the swarms that used CC, and SW0 and SW0+ had the best performance of those that did not use the coefficient. With respect to the neighborhood size, when no CC was used, the swarms that had a local neighborhood outperformed those that used a global one. When CC was used, the swarms that used global neighborhood outperformed those that used a local neighborhood. With respect to PT, when no CC was used, the PT swarms all outperformed the non-PT swarms in terms of total holes found, but in terms of distinct holes, only SW0+ outperformed the non-PT swarms. When CC was used, the PT swarms outperformed the non-PT swarms both in terms of total and distinct holes found. With

respect to RB, both with and without CC, the performance of the RB swarms and non-RB swarms differed by a few holes. These results of this experiment show that the use of PT can effectively improve the ability of a PSO based GRT, especially when used in conjunction with CC.[4] In Figures 4-6, 3D graphs of the holes found by the GA, SW0+, and CCSW4 can be found. These three graphs display the different search behaviors of the different GRTs. Upon reviewing the graphs, one can see that the GA tends to find several small clusters of vulnerabilities, SW0+ tends to find vulnerabilities along the boundaries of the search space, and CCSW4 tends to find vulnerabilities in a single large cluster. These results suggest that these different search capabilities can be combined to create a highly efficient "genetic swarm"-based GRT.

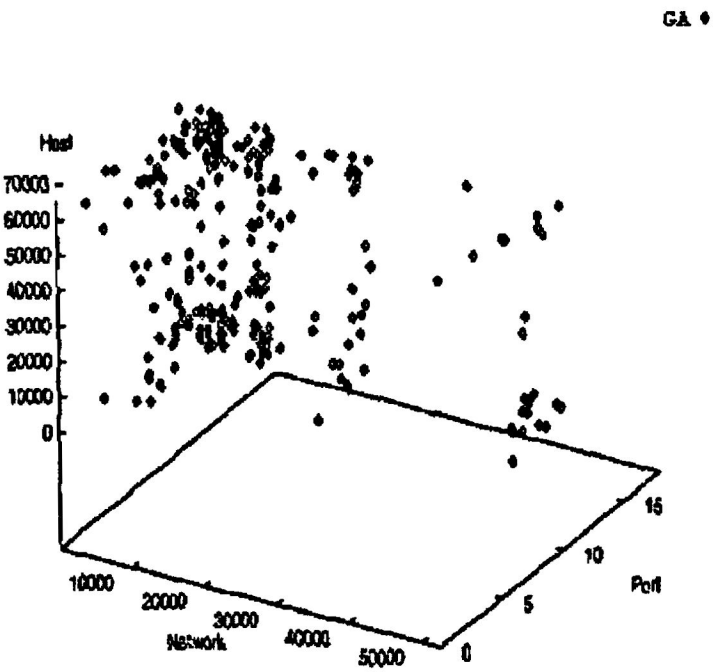

Figure 4: A graph of the holes discovered by GA

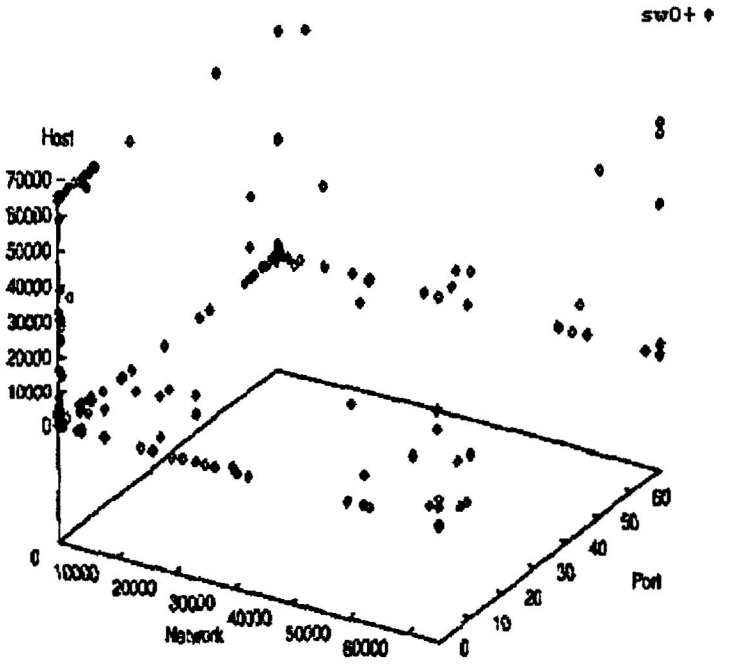

Figure 5: A graph of the holes discovered by SW0+

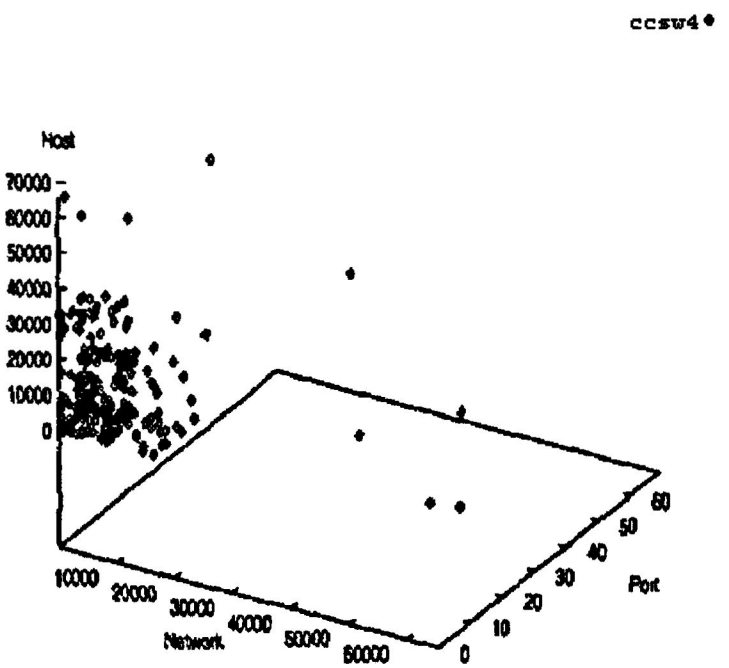

Figure 6: A graph of the holes discovered by CCSW4

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this thesis, an overview of current intrusion detection methods, EC, and immunity-based IDSs was given. In the overview of immunity-based IDSs, an application known as GENERTIA was introduced. GENERTIA consists of a GA-based red team which finds vulnerabilities in an AIS-based IDS and a GA-based blue team which repairs the vulnerabilities found by the red team. The GA-based red team was tested in order to determine how many holes it could find and how fast it could find them. Afterwards, the GA red team's behavior was compared to that of 12 different PSO-based hackers. It was determined that although the GA found more unique holes than all 12 of the swarm-based red teams, the CCSW4 and SW0+ swarms were able to provide a performance closest to that of the GA. By studying the results of the comparison of the GA and the 12 swarms, it was concluded that the use of CC and PT can greatly improve a PSO-based red team's ability to find vulnerabilities in an AIS-based IDS.

6.1. Future Work

There are several different possible research projects that could be based on this thesis. During this experiment, the search behaviors of the GA, swarms that use CC, and swarms that do not use CC were compared. It was hypothesized that the different search

behaviors of the different types of red teams may be able to be combined in order to create a more effective "genetic swarm." A possible future research project could be to implement the genetic swarm-based red team. In addition to the development of a genetic swarm-based red team, the blue team mentioned earlier could be implemented in order to patch any holes found by the red team and tested in order to measure its performance. Also, a version of GENERTIA could be developed which could be applied to a wireless LAN.

ENDNOTES

Chapter 1

1. Jeffery O. Kephart, "A Biologically Inspired Immune System for Computers," in *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems Held in Boston, Massachusetts 6-8 July 1994*, eds. R. Brooks and P. Maes (Boston: MIT Press, 1994), 130-139; Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks* 31, no. 23-24 (December 1999): 2435-2463.

2. *Norton AntiVirus 2005* (Cupertino: Symantec, 2004) [Online]; accessed 28 November 2004, available at http://www.symantec.com/nav/nav_9xnt.

3. *Zone Alarm* (San Francisco: Zone Labs, 2004) [Online]; accessed 8 November 2004, available at http://www.zonelabs.com.

4. Vern Paxson, "Bro"; *Snort* (Columbia: Sourcefire, 2004) [Online]; accessed 30 October 2004, available at http://www.snort.org; *SPECTER Intrusion Detection System* (Bern: NETSEC, 2004) [Online]; accessed 30 October 2004, available at http://www.specter.com; Giovanni Vigna and Richard A. Kemmerer, "NetSTAT: A Network-Based Intrusion Detection System," *Journal of Computer Security* 7, no. 1 (1999): 37-71.

Chapter 2

1. Ioanna Stamouli, "Real-time Intrusion Detection for Ad hoc Networks" (M.S. thesis, Trinity College Dublin, 2003).

2. Nicholas J. Puketza et al., "A Methodology for Testing Intrusion Detection Systems," *IEEE Transaction on Software Engineering* 22, no. 10 (October 1996): 719-729.

3. Paul K. Harmer et al., "An Artificial Immune System Architecture for Computer Security Applications," *IEEE Transactions on Evolutionary Computation* 6, no. 3 (June 2002): 252-280.

4. Luis J. Gonzalez, *Current Approaches to Detecting Intrusions* (2002) [Online]; accessed 28 November 2004, available at http://citeseer.ist.psu.edu/gonzalez02current.html.

5. Mark Crosbie and Gene Spafford, *Defending a Computer System Using Autonomous Agents* (Indianapolis: Purdue University, 1996), Technical Report No. 95-022.

6. Teresa F. Lunt, *Detecting Intruders in Computer Systems* (1993); [Online] accessed 28 November 2004, available at http://citeseer.ist.psu.edu/lunt93detecting.html; Terran Lane and Carla E. Brodley, "Sequence Matching and Learning in Anomaly Detection for Computer Security," in *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management Held in Providence, Rhode Island 28 July 1997* (Menlo Park: AAAI Press, 1997), 43-49.

7. Andrew Watkins, "An Immunological Approach to Intrusion Detection," in *Proceedings of the 12th Annual Canadian Information Technology Security Symposium Held in Ottawa, Canada 19-23 June 2000* (2000), 447-454.

8. Uwe Aickelin, Julie Greensmith, and Jamie Twycross, "Immune System Approaches to Intrusion Detection - A Review," in *Artificial Immune Systems, Third International Conference Held in Catania, Italy 13-16 September 2004*, eds. G. Nicosia et al. (New York: Springer, 2004), 316-329.

9. Anup K. Ghosh, Aaron Schwartzbard, and Michael Shatz, "Learning Program Behavior Profiles for Intrusion Detection," in *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring Held in Santa Clara, California 9-12 April 1999* (Berkeley: USENIX Association, 1999), 51-62.

10. Koral Ilgun et al., "State Transition Analysis: A Rule-Based Intrusion Detection Approach," *IEEE Transactions on Software Engineering* 21, no. 3 (March 1995): 181-199; Sandeep Kumar and Eugene H. Spafford, *A Software Architecture to Support Misuse Intrusion Detection* (Indianapolis: Purdue University,1995), Technical Report CSD-TR-95-009.

11. Kumar and Spafford, *A Software Architecture to Support Misuse Intrusion Detection.*

12. Yongguang Zuang and Wenke Lee, "Intrusion Detection in Wireless Ad-Hoc Networks," in *Proceedings of the Sixth Annual Conference on Mobile Computing and Networking Held in Boston, Massachusetts 6-11 August 2000* (2000), 275-283.

13. David Wagner and Paolo Soto, "Mimicry Attacks on Host-Based Intrusion Detection Systems," in *Proceedings of the 9th ACM Conference On Computer And Communication Security Held in Washington, DC 18-22 November 2002*, ed. Vijayalakshmi Atluri (New York: ACM, 2002), 255-264.

14. Stephanie Forrest et al., "Self-Nonself Discrimination in a Computer," in *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy Held in Oakland, California 16-18 May 1994* (Los Alamitos: IEEE Computer Society Press, 1994), 202-212; Stephanie Forrest et al., "A Sense of Self for Unix Processes," in *Proceedings of the 1996 IEEE Symposium on Security and Privacy Held in Oakland, California 6-8 May 1996* (Los Alamitos: IEEE Computer Society Press, 1996), 120-128; Steven A. Hofmeyr et al., "Intrusion Detection Using Sequences of System Calls," *Journal of Computer Security* 6, no. 3 (1998): 151-180.

15. Thomas Daniels and Eugene Spafford, "Identification of Host Audit Data to Detect Attacks on Low-level IP Vulnerabilities," *Journal of Computer Security* 7, no. 1 (1999): 3-35.

16. Wagner and Soto, "Mimicry Attacks."

17. Diego Zamboni, "Doing Intrusion Detection Using Embedded Sensors -- Thesis Proposal" (M.S. thesis, Purdue University, 2000).

18. Paxson, "Bro"; *Snort*; Vigna and Kemmerer, "NetSTAT."

19. Jungwon Kim and Peter Bentley, "An Artificial Immune Model for Network Intrusion Detection," in *7th European Conference on Intelligent Techniques and Soft Computing Held in Aachen, Germany 13-16 September 1999* (Aachen: Verlang Mainz, 1999).

20. Aickelin, Greensmith, and Twycross, "Immune System Approaches to Intrusion Detection."

21. Ibid.

22. Crosbie and Spafford, *Defending a Computer System.*

23. Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection."

24. Rajeev Gopalakrishna and Eugene H. Spafford, *A Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents* (Indianapolis: Purdue University, 2001), CERIAS TR 2001-44.

25. Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection."

26. Gopalakrishna and Spafford, *A Framework for Distributed Intrusion Detection.*

27. Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection."

28. Gopalakrishna and Spafford, *A Framework for Distributed Intrusion Detection.*

29. Jamil Farshchi, *Wireless Intrusion Detection Systems* (SecurityFocus, 2003) [Online]; accessed 28 November 2004, availableat http://www.securityfocus.com/infocus/1742.

30. Joshua Lackey, Andrew Roths, and James Goddard, *Wireless Intrusion Detection* (Armonk: IBM, 2003) [Online]; accessed 28 November 2004, available at http://www-1.ibm.com/services/us/bcrs/pdf/wp_wireless-intrusion-detection.pdf.

31. Scott Fluhrer et al., "Weakness in the Key Scheduling Algorithm of RC4," *Lecture Notes in Computer Science* 2259 (2001): 1-24.

32. *AirSnort* (2004) [Online]; accessed 30 November 2004, available at http://airsnort.shmoo.com.

33. *WEPCrack* (2004) [Online]; accessed 30 October 2004, available at http://wepcrack.sourceforge.net.

34. Yu-Xi Lim et al., "Wireless Intrusion Detection and Response," in *Proceedings of the 2003 IEEE Workshop on Information Assurance Held in West Point, New York 18-20 June 2003* (Piscataway: IEEE, 2003), 68-75.

35. Lackey, Roths, and Goddard, *Wireless Intrusion Detection.*

36. IEEE Computer Society LAN MAN Standards Committee, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications* (New York: IEEE, 1999), IEEE Std 802.11-1999.

37. Ibid.

38. *Wireless Intrusion Protection* (Sunnyvale: Aruba, 2004) [Online]; accessed 30 November 2004, available at http://www.arubanetworks.com/pdf/techbrief-IDS.pdf.

39. *AirDefense* (Alpharetta: AirDefense, 2004) [Online]; accessed 30 November 2004, available at http://www.airdefense.net.

40. *AirMagnet* (Sunnyvale: AirMagnet, 2004) [Online]; accessed 30 November 2004, available at http://www.airmagnet.com.

Chapter 3

1. Gerry Dozier et al., "An Introduction to Evolutionary Computation" in *Intelligent Control Systems Using Soft Computing Methodologies*, eds. A. Zilouchian and M. Jamshidi (Boca Raton: CRC Press, 2001), 365-380.

2. James Kennedy and Russell C. Eberhart, *Swarm Intelligence* (San Francisco: Morgan Kaufmann Publishers, 2001).

3. William M. Spears et al., "An Overview of Evolutionary Computation," in *Lecture Notes in Computer Science* 667 (1993): 442-459.

4. Kennedy and Eberhart, *Swarm Intelligence*.

5. Hans Schwefel, "On the Evolution of Evolutionary Computation," *Computational Intelligence Imitating Life*, eds. R. J. Marks II and C. J. Robinson (New York: IEEE, 1994), 116-124.

6. Thomas Bäck, Ulrich Hammel, and Hans-Paul Schwefel, "Evolutionary Computation: Comments on the History and Current State," *IEEE Transactions on Evolutionary Computation* 1, no. 1 (April 1997): 3-17.

7. Kennedy and Eberhart, *Swarm Intelligence*.

8. Samir Mahfoud, "Niching Methods for Genetic Algorithms" (Ph. D. diss., University of Illinois at Urbana-Champaign, 1995).

9. Dozier et al., "Evolutionary Computation."

10. Darrell Whitley, "A Genetic Algorithm Tutorial," *Statistics and Computing* 4 (1994): 65-85.

11. Tobias Blickle and Lothar Thiele, *A Comparison of Selection Schemes Used in Genetic Algorithms* (Zurich: Swiss Federal Institute of Technology, 1995), TIK-Report No. 11.

12. Brad L. Miller and David E. Goldberg, "Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise," *Evolutionary Computation* 4, no. 2 (Summer 1996): 113-131.

13. Ibid.

14. Brad L. Miller and David E. Goldberg, "Genetic Algorithms, Tournament Selection, and the Effects of Noise," *Complex Systems* 9, no. 3 (June 1995): 193-212.

15. M. Tomassini, "A Survey of Genetic Algorithms," *Annual Reviews of Computational Physics* 3 (October 1995): 87-118.

16. Kennedy and Eberhart, *Swarm Intelligence*.

17. William M. Spears and Kenneth A. De Jong, "An Analysis of Multi-Point Crossover," *Foundations of Genetic Algorithms*, ed. G. J. E. Rawlins (San Mateo, CA: Morgan Kaufmann Publishers, 1991), 301-315.

18. Shumeet Baluja and Rich Caruana, "Removing the Genetics from the Standard Genetic Algorithm," in *Proceedings of ML-95, Twelfth International Conference on Machine Learning Held in Lake Tahoe, California 9-12 July 1995*, eds. A. Prieditis and S. Russel (San Mateo: Morgan Kaufmann Publishers, 1995), 38-46.

19. F. Herrara, M. Lozano, and J. L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review* 12, no. 4 (August 1998): 265-319.

20. Dozier et al., "Evolutionary Computation"; Whitley, "Genetic Algorithm Tutorial."

21. Herrara, Lozano, and Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators."

22. Ibid.

23. Frank Vavak and Terence C. Fogarty, "Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments," in *International Conference on Evolutionary Computation Held in Nagoya, Japan 20-22 May 1996* (Piscataway: IEEE, 1996), 192-195.

24. Ibid.

25. James Kennedy and Russell Eberhart, "Particle Swarm Optimization," in *Proceedings of the 4th IEEE International Conference on Neural Networks Held in Perth, Australia 27 November - 1 December 1995* (Piscataway: IEEE, 1995), 1942-1948.

26. Ibid.

27. Thomas Beielstein, Konstantinos E. Parsopoulos, and Michael N. Vrahatis, *Tuning PSO Parameters Through Sensitivity Analysis* (Dortmund: University of Dortmund, 2002), CI 124/02.

28. Gerhard Venter and Jaroslaw Sobieszczanski-Sobieski, "Particle Swarm Optimization," *AIAA Journal* 41, no. 8 (August 2003): 1583-1589.

29. Ender Ozcan and Chilukuri K. Mohan, "Particle Swarm Optimization: Surfing the Waves," in *Proceedings of the IEEE Congress on Evolutionary Computation Held in Washington, DC 6-9 July 1999*, eds. Peter J. Angeline et al. (Piscataway: IEEE, 1999), 1939-1944.

30. Thiemo Krink, Jakob S. Vesterstrøm, and Jacques Riget, "Particle Swarm Optimisation with Spatial Particle Extension," in *Proceedings of the IEEE Congress on Evolutionary Computation Held in Honolulu, Hawaii 12-17 May 2002*, ed. Xin Yao (Piscataway: IEEE, 2002), 1474-1479.

31. Anthony Carlisle and Gerry Dozier, "An Off the Shelf PSO," in *Proceedings of the Workshop on Particle Swarm Optimization Held in Indianapolis, Indiana 6-7 April 2001* (Indianapolis: Purdue School of Engineering and Technology, 2001), 1-6.

32. Beielstein, Parsopoulos, and Vrahatis, *Tuning PSO Parameters.*

33. Carlisle and Dozier, "Off the Shelf PSO."

34. Ibid.

35. Kennedy and Eberhart, *Swarm Intelligence.*

36. Carlisle and Dozier, "Off the Shelf PSO."

37. Kennedy and Eberhart, *Swarm Intelligence.*

38. Yuhui Shi and Russell C. Eberhart, "Parameter Selection in Particle Swarm Optimization," in *The Seventh Annual Conference on Evolutionary Programming Held in San Diego, California 25-27 March 1998*, eds. V. William Porto et al. (New York: Springer, 1998), 591-600.

39. Kennedy and Eberhart, *Swarm Intelligence.*

40. Carlisle and Dozier, "Off the Shelf PSO."

41. R. C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization," in *Proceedings of the Congress on Evolutionary Computation 2000 Held in San Diego, California 16-19 July 2000* (Piscataway: IEEE, 2000), 84-88.

42. Steven A. Hofmeyr and S. Forrest, "Architecture for an Artificial Immune System," *Evolutionary Computation* 8, no. 4 (Winter 2000): 443-473.

43. Ibid.

44. C. Starr and B. McMillan, *Human Biology* (Stamford: Thomson Learning, 2004).

45. Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji, "Computer Immunology," *Communications of the ACM* 40, no. 10 (December 1997), 88-96.

46. Olfa Nasaroui, Dipankar Dasgupta, and Fabio Gonzalez, "The Promise and Challenges of Artificial Immune System Based Web Usage Mining: Preliminary Results," in *Proceedings of the Workshop on Web Analytics, Second SIAM Conference on Data Mining Held in Arlington, Virginia 11-13 April 2002* (2002), 29-39.

47. Starr and McMillan, *Human Biology.*

48. Hofmeyr and Forrest, "Architecture for an Artificial Immune System"; Anil Somayaji, Steven Hofmeyr, and Stephanie Forrest, "Principles of a Computer Immune System," in *Proceedings of the Second New Security Paradigms Workshop Held in Langdale, United Kingdom 23-26 September 1997* (New York: ACM, 1997), 75-82.

49. Starr and McMillan, *Human Biology.*

50. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

51. Dipankar Dasgupta and Nii Attoh-Okine, "Immunity-Based Systems: A Survey," in *The Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Held in Orlando, Florida 12-15 October 1997* (Piscataway: IEEE, 1997) 363-374.

52. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

53. Ibid.

54. Steven A. Hofmeyr, "An Immunological Model of Distributed Detection and its Application to Computer Security" (Ph. D. diss., University of New Mexico, 1999).

55. Jungwon Kim and Peter Bentley, "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection," in *Proceedings of the 2001 Congress on Evolutionary Computation Held in Seoul, Korea 27-30 May 2001* (Piscataway: IEEE, 2001), 1244-1252.

56. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

57. Hofmeyr, "An Immunological Model of Distributed Detection."

58. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

59. Ibid.

Chapter 4

1. Aickelin, Greensmith, and Twycross, "Immune System Approaches to Intrusion Detection"; Justin Balthrop, Stephanie Forrest, and Matthew R. Glickman, "Revisiting LISYS: Parameters and Normal Behavior," in *Proceedings of the 2002 Congress on Evolutionary Computing Held in Honolulu, Hawaii 12-17 May 2002,* ed. Xin Yao (Honolulu: IEEE, 2002), 1045-1050; Dasgupta and Attoh-Okine, "Immunity-Based Systems"; Dipankar Dasgupta and Fabio Gonzalez, "An Immunity-Based Technique to Characterize Intrusions in Computer Networks," *IEEE Transactions on Evolutionary Computation* 6, no. 3 (June 2002): 281-291; Leandro Nunes de Castro and Fernando Jose von Zuben, *Artificial Immune Systems: Part II - A Survey of Applications* (Sao Paulo: State University of Campinas, 2000), Technical Report DCA-RT 02/00; Forrest et al., "Self-Nonself Discrimination"; Forrest, Hofmeyr, and Somayaji, "Computer Immunology"; Harmer et al., "An Artificial Immune System Architecture"; Hofmeyr, "An Immunological Model of Distributed Detection"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System"; Kephart, "A Biologically Inspired Immune System"; Jungwon Kim, "An Artificial Immune System for Network Intrusion Detection," in *Graduate Student Workshop, Genetic and Evolutionary Computation Conference Held in Orlando, Florida 13-17 July 1999*, eds. Wolfgang Banzhaf et al. (San Francisco: Morgan Kaufmann Publishers, 1999), 369-370; Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection"; Kim and Bentley, "Towards an Artificial Immune System for Network Intrusion Detection"; Jean-Yves Le Boudec and Slavisa Sarafijanovic, *An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks* (Lausanne: EFPL, 2003), Technical Report IC/2003/59; Toru Ohira, *Immune Pattern Recognition System* (Tokyo: Sony Computer Science Laboratory, 1995) [Online]; accessed 28 November 2004, available at http://citeseer.ist.psu.edu/149378.html; Slavisa Sarafijanovic and Jean-Yves Le Boudec, *An Artificial Immune System Approach with Secondary Response for Misbehavior Detection in Mobile Ad-Hoc Networks* (Lausanne: EPFL, 2003), Technical Report IC/2003/65; Somayaji, Hofmeyr, and Forrest, "Principles of a Computer Immune System"; Watkins, "An Immunological Approach to Intrusion Detection."

2. Balthrop, Forrest, and Glickman, "Revisiting LISYS"; Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions"; Harmer et al., "An Artificial Immune System Architecture"; Hofmeyr, "An Immunological Model of Distributed Detection"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System"; Kephart, "A Biologically Inspired Immune System"; Le Boudec and Sarafijanovic, *An Artificial Immune System Approach to Misbehavior Detection*; Sarafijanovic and Le Boudec, *An Artificial Immune System Approach with Secondary Response for Misbehavior Detection.*

3. Balthrop, Forrest, and Glickman, "Revisiting LISYS"; Hofmeyr, "An Immunological Model of Distributed Detection"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

4. Harmer et al., "An Artificial Immune System Architecture."

5. Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions."

6. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

7. Balthrop, Forrest, and Glickman, "Revisiting LISYS."

8. Ibid.

9. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

10. Martin Roesch, *Snort - Lightweight Intrusion Detection for Networks* (Columbia: Sourcefire, 1998) [Online]; accessed 30 October 2004, available at http://www.snort.org/docs/lisapaper.txt.

11. Samuel Patton, William Yurick, and David Doss, "An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT," in *Fourth International Symposium on Recent Advances in Intrusion Detection Held in Davis, California 10-12 October 2001* (New York: Springer, 2001), 1-8.

12. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

13. Watkins, "An Immunological Approach to Intrusion Detection."

14. Harmer et al., "An Artificial Immune System Architecture."

15. Patton, Yurick, and Doss, "An Achilles' Heel in Signature-Based IDS."

16. Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection."

17. Ibid.

18. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

19. Kim and Bentley, "An Artificial Immune Model for Network Intrusion Detection."

20. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

21. Ibid.

22. Ibid.

23. Somayaji, Hofmeyr, and Forrest, "Principles of a Computer Immune System";

24. Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

25. Kim and Bentley, "Towards an Artificial Immune System for Network Intrusion Detection."

26. Richard P. Lippmann et al., "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation," in *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition Held in Hilton Head, South Carolina 25-27 January 2000* (Los Alamitos: IEEE, 2000), 12-26.

27. Hofmeyr, "An Immunological Model of Distributed Detection."

28. Harmer et al., "An Artificial Immune System Architecture."

29. Kephart, "A Biologically Inspired Immune System."

30. Ibid.

31. Harmer et al., "An Artificial Immune System Architecture."

32. Ibid.

33. Ibid.

34. Ibid.

35. Ibid.

36. Ibid.

37. Chairoj Mekprasertvit, "Applying Broadcasting/Multicasting/Secured Communication To AgentMOM In Multiagent-Systems" (M.S. thesis, Kansas State University, 2004).

38. *Java Shared Data Toolkit* (Santa Clara: Sun Microsystems, 2004) [Online]; accessed 28 November 2004, available at http://java.sun.com/products/java-media/jsdt/index.jsp.

39. Harmer et al., "An Artificial Immune System Architecture."

40. Ibid.

41. Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions."

42. Harmer et al., "An Artificial Immune System Architecture"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System"; Kephart, "A Biologically Inspired Immune System"; Le Boudec and Sarafijanovic, *An Artificial Immune System Approach to Misbehavior Detection*; Sarafijanovic and Le Boudec, *An Artificial Immune System Approach with Secondary Response for Misbehavior Detection.*

43. Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions."

44. Lane and Brodley, "Sequence Matching and Learning."

45. Forrest et al., "Self-Nonself Discrimination"; Harmer et al., "An Artificial Immune System Architecture"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System"; Kephart, "A Biologically Inspired Immune System"; Le Boudec and Sarafijanovic, *An Artificial Immune System Approach to Misbehavior Detection*; Sarafijanovic and Le Boudec, *An Artificial Immune System Approach with Secondary Response for Misbehavior Detection.*

46. Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions."

47. Ibid.

48. Ibid.

49. Ibid.

50. *DARPA Intrusion Detection Evaluation* (Boston: Lincoln Laboratory, 1999) [Online]; accessed 5 December 2004, available at http://www.ll.mit.edu/IST/ideval/index.html.

51. Dasgupta and Gonzalez, "An Immunity-Based Technique to Characterize Intrusions."

52. *DARPA Intrusion Detection Evaluation.*

53. Gerry Dozier, "IDS Vulnerability Analysis Using Genertia Red Teams," in *Proceedings of the International Conference on Security and Management Held in Las Vegas, Nevada 23-26 June 2003*, eds. H.R. Arabnia and Y. Mun, (Las Vegas: CSREA Press, 2003), 171-176.

54. Balthrop, Forrest, and Glickman, "Revisiting LISYS"; Dasgupta and Attoh-Okine, "Immunity-Based Systems"; Hofmeyr and Forrest, "Architecture for an Artificial Immune System."

55. Steven A. Hofmeyr and Stephanie Forrest, "Immunity By Design: An Artificial Immune System," in *Proceedings of the Genetic and Evolutionary Computation Conference Held in Orlando, Florida 13-17 July 1999*, eds. Wolfgang Banzhaf et al. (San Francisco: Morgan Kaufmann Publishers, 1999), 1289-1296.

56. Kennedy and Eberhart, *Swarm Intelligence*.

57. Dozier, "IDS Vulnerability Analysis."

58. Ibid.

59. Haiyu Hou, Jun Zhu, and Gerry Dozier, "Artificial Immunity Using Constraint Based Detectors," in *Proceedings of the 5th Biannual World Automation Congress Held in Orlando, Florida 9-13 June 2002* (Albuquerque: World Automation Congress, 2002), 239-244.

60. Dozier, "IDS Vulnerability Analysis."

61. Hou, Zhu, and Dozier, "Artificial Immunity."

62. *DARPA Intrusion Detection Evaluation* (Boston: Lincoln Laboratory, 1998) [Online]; accessed 5 December 2004, available at http://www.ll.mit.edu/IST/ideval/index.html.

Chapter 5

1. Gerry Dozier et al., "Vulnerability Analysis of AIS-Based Intrusion Detection Systems via Genetic and Particle Swarm Red Teams," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation Held in Portland, Oregon 20-23 June 2004* (Piscataway: IEEE, 2004), 111-116.

2. Ibid.

3. Ibid.

4. Ibid.

BIBLIOGRAPHY

Aickelin, Uwe, Julie Greensmith, and Jamie Twycross. "Immune System Approaches to Intrusion Detection - A Review." In *Artificial Immune Systems, Third International Conference Held in Catania, Italy 13-16 September 2004*, edited by G. Nicosia et al., 316-329. New York: Springer, 2004.

AirDefense. Alpharetta: AirDefense, 2004. Online. Accessed 30 November 2004. Available at http://www.airdefense.net.

AirMagnet. Sunnyvale: AirMagnet, 2004. Online. Accessed 30 November 2004. Available at http://www.airmagnet.com.

AirSnort. 2004. Online. Accessed 30 November 2004. Available at http://airsnort.shmoo.com.

Bäck, Thomas, Ulrich Hammel, and Hans-Paul Schwefel. "Evolutionary Computation: Comments on the History and Current State." *IEEE Transactions on Evolutionary Computation* 1, no. 1 (April 1997): 3-17.

Balthrop, Justin, Stephanie Forrest, and Matthew R. Glickman. "Revisiting LISYS: Parameters and Normal Behavior." In *Proceedings of the 2002 Congress on Evolutionary Computing Held in Honolulu, Hawaii 12-17 May 2002,* edited by Xin Yao, 1045-1050. Piscataway: IEEE, 2002.

Baluja, Shumeet and Rich Caruana. "Removing the Genetics from the Standard Genetic Algorithm." In *Proceedings of ML-95, Twelfth International Conference on Machine Learning Held in Lake Tahoe, California 9-12 July 1995*, edited by A. Prieditis and S. Russel, 38-46. San Mateo: Morgan Kaufmann Publishers, 1995.

Beielstein, Thomas, Konstantinos E. Parsopoulos, and Michael N. Vrahatis. *Tuning PSO Parameters Through Sensitivity Analysis*. Dortmund: University of Dortmund, 2002. CI 124/02.

Blickle, Tobias and Lothar Thiele. *A Comparison of Selection Schemes Used in Genetic Algorithms*. Zurich: Swiss Federal Institute of Technology, 1995. TIK-Report No. 11.

Carlisle, Anthony and Gerry Dozier. "An Off the Shelf PSO." In *Proceedings of the Workshop on Particle Swarm Optimization Held in Indianapolis, Indiana 6-7 April 2001*, 1-6. Indianapolis: Purdue School of Engineering and Technology, 2001.

Crosbie, Mark and Gene Spafford. *Defending a Computer System Using Autonomous Agents*. Indianapolis: Purdue University, 1996. Technical Report No. 95-022.

T. Daniels and E. Spafford. "Identification of Host Audit Data to Detect Attacks on Low-level IP Vulnerabilities." *Journal of Computer Security* 7, no. 1 (1999): 3-35.

DARPA Intrusion Detection Evaluation. Boston: Lincoln Laboratory, 1998. Online. Accessed 5 December 2004. Available at http://www.ll.mit.edu/IST/ideval/index.html.

DARPA Intrusion Detection Evaluation. Boston: Lincoln Laboratory, 1999. Online. Accessed 5 December 2004. Available at http://www.ll.mit.edu/IST/ideval/index.html.

Dasgupta, Dipankar and Nii Attoh-Okine. "Immunity-Based Systems: A Survey." In *The Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics Held in Orlando, Florida 12-15 October 1997*, 363-374. Piscataway: IEEE, 1997.

Dasgupta, Dipankar and Fabio Gonzalez. "An Immunity-Based Technique to Characterize Intrusions in Computer Networks." *IEEE Transactions on Evolutionary Computation* 6, no. 3 (June 2002): 281-291.

de Castro, Leandro Nunes and Fernando Jose von Zuben. *Artificial Immune Systems: Part II - A Survey of Applications*. Sao Paulo: State University of Campinas, 2000. Technical Report DCA-RT 02/00.

Dozier, Gerry. "IDS Vulnerability Analysis Using Genertia Red Teams." In *Proceedings of the International Conference on Security and Management Held in Las Vegas, Nevada 23-26 June 2003*, edited by H.R. Arabnia and Y. Mun, 171-176. Las Vegas: CSREA Press, 2003.

Dozier, Gerry et al. "Vulnerability Analysis of AIS-Based Intrusion Detection Systems via Genetic and Particle Swarm Red Teams." In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation Held in Portland, Oregon 20-23 June 2004*, 111-116. Piscataway: IEEE, 2004.

Dozier, Gerry et al. "An Introduction to Evolutionary Computation." In *Intelligent Control Systems Using Soft Computing Methodologies*, edited by A. Zilouchian and M. Jamshidi, 365-380. Boca Raton: CRC Press, 2001.

Eberhart, R. C. and Y. Shi. "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization." In *Proceedings of the Congress on Evolutionary Computation 2000 Held in San Diego, California 16-19 July 2000*, 84-88. Piscataway: IEEE, 2000.

Farshchi, Jamil. *Wireless Intrusion Detection Systems*. SecurityFocus, 2003. Online. Accessed 28 November 2004. Available at http://www.securityfocus.com/infocus/1742.

Fluhrer, Scott et al. "Weakness in the Key Scheduling Algorithm of RC4." *Lecture Notes in Computer Science* 2259 (2001): 1-24.

Forrest, Stephanie et al. "Self-Nonself Discrimination in a Computer." In *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy Held in Oakland, California 16-18 May 1994*, 202-212. Los Alamitos: IEEE Computer Society Press, 1994.

Forrest, Stephanie, Steven A. Hofmeyr, and Anil Somayaji. "Computer Immunology." *Communications of the ACM* 40, no. 10 (December 1997), 88-96.

Forrest, Stephanie et al. "A Sense of Self for Unix Processes." In *Proceedings of the 1996 IEEE Symposium on Security and Privacy Held in Oakland, California 6-8 May 1996*, 120-128. Los Alamitos: IEEE Computer Society Press, 1996.

Ghosh, Anup K., Aaron Schwartzbard, and Michael Shatz, "Learning Program Behavior Profiles for Intrusion Detection." In *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring Held in Santa Clara, California 9-12 April 1999*, 51-62. Berkeley: USENIX Association, 1999.

Gonzalez, Luis J. *Current Approaches to Detecting Intrusions*. 2002. Online. Accessed 28 November 2004. Available at http://citeseer.ist.psu.edu/gonzalez02current.html.

Gopalakrishna, Rajeev and Eugene H. Spafford. *A Framework for Distributed Intrusion Detection using Interest Driven Cooperating Agents*. Indianapolis: Purdue University, 2001. CERIAS TR 2001-44.

Harmer, Paul K. et al. "An Artificial Immune System Architecture for Computer Security Applications." *IEEE Transactions on Evolutionary Computation* 6, no. 3 (June 2002): 252-280.

Herrara, F., M. Lozano, and J. L. Verdegay. "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis." *Artificial Intelligence Review* 12, no. 4 (August 1998): 265-319.

Hofmeyr, Steven A. "An Immunological Model of Distributed Detection and its Application to Computer Security." Ph. D. diss., University of New Mexico, 1999).

Hofmeyr, Steven A. and S. Forrest. "Architecture for an Artificial Immune System." *Evolutionary Computation* 8, no. 4 (Winter 2000): 443-473.

Hofmeyr, Steven A. and Stephanie Forrest. "Immunity By Design: An Artificial Immune System." In *Proceedings of the Genetic and Evolutionary Computation Conference Held in Orlando, Florida 13-17 July 1999*, edited by Wolfgang Banzhaf et al., 1289-1296. San Francisco: Morgan Kaufmann Publishers, 1999.

Hofmeyr, Steven A. et al. "Intrusion Detection Using Sequences of System Calls." *Journal of Computer Security* 6, no. 3 (1998): 151-180.

Hou, Haiyu, Jun Zhu, and Gerry Dozier. "Artificial Immunity Using Constraint Based Detectors," in *Proceedings of the 5th Biannual World Automation Congress Held in Orlando, Florida 9-13 June 2002*, 239-244. Albuquerque: World Automation Congress, 2002.

IEEE Computer Society LAN MAN Standards Committee. *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. New York: IEEE, 1999. IEEE Std 802.11-1999.

Ilgun, Koral et al. "State Transition Analysis: A Rule-Based Intrusion Detection Approach." *IEEE Transactions on Software Engineering* 21, no. 3 (March 1995): 181-199.

Java Shared Data Toolkit. Santa Clara: Sun Microsystems, 2004. Online. Accessed 28 November 2004. Available at http://java.sun.com/products/java-media/jsdt/index.jsp.

Kennedy, James and Russell Eberhart. "Particle Swarm Optimization." In *Proceedings of the 4th IEEE International Conference on Neural Networks Held in Perth, Australia 27 November - 1 December 1995*, 1942-1948. Piscataway: IEEE, 1995.

Kennedy, James and Russell C. Eberhart. *Swarm Intelligence*. San Francisco: Morgan Kaufmann Publishers, 2001.

Kephart, Jeffery O. "A Biologically Inspired Immune System for Computers." In *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems Held in Boston, Massachusetts 6-8 July 1994*, edited by R. Brooks and P. Maes, 130-139. Boston: MIT Press, 1994.

Kim, Jungwon. "An Artificial Immune System for Network Intrusion Detection." In *Graduate Student Workshop, Genetic and Evolutionary Computation Conference*

Held in Orlando, Florida 13-17 July 1999, edited by Wolfgang Banzhaf et al., 369-370. San Francisco: Morgan Kaufmann Publishers, 1999.

Kim, Jungwon and Peter Bentley. "An Artificial Immune Model for Network Intrusion Detection," in *7th European Conference on Intelligent Techniques and Soft Computing Held in Aachen, Germany 13-16 September 1999*. Aachen: Verlang Mainz, 1999.

Kim, Jungwon and Peter Bentley. "Towards an Artificial Immune System for Network Intrusion Detection: An Investigation of Dynamic Clonal Selection." In *Proceedings of the 2001 Congress on Evolutionary Computation Held in Seoul, Korea 27-30 May 2001*, 1244-1252. Piscataway: IEEE, 2001.

Krink, Thiemo, Jakob S. Vesterstrøm, and Jacques Riget. "Particle Swarm Optimisation with Spatial Particle Extension." In *Proceedings of the IEEE Congress on Evolutionary Computation Held in Honolulu, Hawaii 12-17 May 2002*, edited by Xin Yao, 1474-1479. Piscataway: IEEE, 2002.

Kumar, Sandeep and Eugene H. Spafford. *A Software Architecture to Support Misuse Intrusion Detection.* Indianapolis: Purdue University, 1995. Technical Report CSD-TR-95-009.

Lackey, Joshua, Andrew Roths, and James Goddard. *Wireless Intrusion Detection.* Armonk: IBM, 2003. Online. Accessed 28 November 2004. Available at http://www-1.ibm.com/services/us/bcrs/pdf/wp_wireless-intrusion-detection.pdf.

Lane, Terran and Carla E. Brodley. "Sequence Matching and Learning in Anomaly Detection for Computer Security." In *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management Held in Providence, Rhode Island 28 July 1997*, 43-49. Menlo Park: AAAI Press, 1997.

Le Boudec, Jean-Yves and Slavisa Sarafijanovic. *An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks.* Lausanne: EFPL, 2003. Technical Report IC/2003/59.

Lim, Yu-Xi et al. "Wireless Intrusion Detection and Response." In *Proceedings of the 2003 IEEE Workshop on Information Assurance Held in West Point, New York 18-20 June 2003*, 68-75. Piscataway: IEEE, 2003.

Lippmann, Richard P. et al. "Evaluating Intrusion Detection Systems: The 1998 DARPA Off-line Intrusion Detection Evaluation." In *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition Held in Hilton Head, South Carolina 25-27 January 2000*, 12-16. Los Alamitos: IEEE, 2000.

Lunt, Teresa F. *Detecting Intruders in Computer Systems.* 1993. Online. Accessed 28 November 2004. Available at http://citeseer.ist.psu.edu/lunt93detecting.html.

Mahfoud, Samir. "Niching Methods for Genetic Algorithms." Ph. D. diss., University of Illinois at Urbana-Champaign, 1995.

Mekprasertvit, Chairoj. "Applying Broadcasting/Multicasting/Secured Communication To AgentMOM In Multiagent-Systems." M.S. thesis, Kansas State University, 2004.

Miller, Brad L. and David E. Goldberg. "Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise." *Evolutionary Computation* 4, no. 2 (Summer 1996): 113-131.

Miller, Brad L. and David E. Goldberg. "Genetic Algorithms, Tournament Selection, and the Effects of Noise." *Complex Systems* 9, no. 3 (June 1995): 193-212.

Nasaroui, Olfa, Dipankar Dasgupta, and Fabio Gonzalez. "The Promise and Challenges of Artificial Immune System Based Web Usage Mining: Preliminary Results." in *Proceedings of the Workshop on Web Analytics, Second SIAM Conference on Data Mining Held in Arlington, Virginia 11-13 April 2002*, 29-39. 2002.

Norton AntiVirus 2005. Cupertino: Symantec, 2004. Online. Accessed 28 November 2004. Available at http://www.symantec.com/nav/nav_9xnt.

Ohira, Toru. *Immune Pattern Recognition System*. Tokyo: Sony Computer Science Laboratory, 1995. Online. Accessed 28 November 2004. Available at http://citeseer.ist.psu.edu/149378.html.

Ozcan, Ender and Chilukuri K. Mohan. "Particle Swarm Optimization: Surfing the Waves." In *Proceedings of the IEEE Congress on Evolutionary Computation Held in Washington, DC 6-9 July 1999*, edited by Peter J. Angeline et al, 1939-1944. Piscataway: IEEE, 1999.

Patton, Samuel, William Yurick, and David Doss. "An Achilles' Heel in Signature-Based IDS: Squealing False Positives in SNORT." In *Fourth International Symposium on Recent Advances in Intrusion Detection Held in Davis, California 10-12 October 2001*, 1-8. New York: Springer, 2001.

Paxson, Vern. "Bro: A System for Detecting Network Intruders in Real-Time." *Computer Networks* 31, no. 23-24 (December 1999): 2435-2463.

Puketza, Nicholas J. et al. "A Methodology for Testing Intrusion Detection Systems." *IEEE Transaction on Software Engineering* 22, no. 10 (October 1996): 719-729.

Roesch, Martin. *Snort - Lightweight Intrusion Detection for Networks*. Columbia: Sourcefire, 1998. Online. Accessed 30 October 2004. Available at http://www.snort.org/docs/lisapaper.txt.

Sarafijanovic, Slavisa and Jean-Yves Le Boudec. *An Artificial Immune System Approach with Secondary Response for Misbehavior Detection in Mobile Ad-Hoc Networks.* Lausanne: EPFL, 2003. Technical Report IC/2003/65.

Schwefel, Hans. "On the Evolution of Evolutionary Computation." *Computational Intelligence Imitating Life*, edited by R. J. Marks II and C. J. Robinson, 116-124. New York: IEEE, 1994.

Shi, Yuhui and Russell C. Eberhart. "Parameter Selection in Particle Swarm Optimization." In *The Seventh Annual Conference on Evolutionary Programming Held in San Diego, California 25-27 March 1998*, edited by V. William Porto et al., 591-600. New York: Springer, 1998.

Snort. Columbia: Sourcefire, 2004. Online. Accessed 30 October 2004. Available at http://www.snort.org.

Somayaji, Anil, Steven Hofmeyr, and Stephanie Forrest. "Principles of a Computer Immune System." In *Proceedings of the Second New Security Paradigms Workshop Held in Langdale, United Kingdom 23-26 September 1997*, 75-82. New York: ACM, 1997.

Spears, William M. and Kenneth A. De Jong. "An Analysis of Multi-Point Crossover." *Foundations of Genetic Algorithms*, edited by G. J. E. Rawlins, 301-315. San Mateo, CA: Morgan Kaufmann Publishers, 1991.

Spears, William M. et al. "An Overview of Evolutionary Computation." In *Lecture Notes in Computer Science* 667 (1993): 442-459.

SPECTER Intrusion Detection System. Bern: NETSEC, 2004. Online. Accessed 30 October 2004. Available at http://www.specter.com.

Stamouli, Ioanna. "Real-time Intrusion Detection for Ad hoc Networks." M.S. thesis, Trinity College Dublin, 2003.

Starr, C. and B. McMillan. *Human Biology*. Stamford: Thomson Learning, 2004.

Tomassini, M. "A Survey of Genetic Algorithms." *Annual Reviews of Computational Physics* 3 (October 1995): 87-118.

Vavak, Frank and Terence C. Fogarty. "Comparison of Steady State and Generational Genetic Algorithms for Use in Nonstationary Environments." In *International Conference on Evolutionary Computation Held in Nagoya, Japan 20-22 May 1996*, 192-195. Piscataway: IEEE, 1996.

Venter, Gerhard and Jaroslaw Sobieszczanski-Sobieski. "Particle Swarm Optimization." *AIAA Journal* 41, no. 8 (August 2003): 1583-1589.

Vigna, Giovanni and Richard A. Kemmerer. "NetSTAT: A Network-Based Intrusion Detection System." *Journal of Computer Security* 7, no. 1 (1999): 37-71.

Wagner, David and Paolo Soto. "Mimicry Attacks on Host-Based Intrusion Detection Systems." In *Proceedings of the 9th ACM Conference On Computer And Communication Security Held in Washington, DC 18-22 November 2002*, edited by Vijayalakshmi Atluri, 255-264. New York: ACM, 2002.

Watkins, Andrew. "An Immunological Approach to Intrusion Detection." In *Proceedings of the 12th Annual Canadian Information Technology Security Symposium Held in Ottawa, Canada 19-23 June 2000*, 447-454. 2000.

WEPCrack. 2004. Online. Accessed 30 October 2004. Available at http://wepcrack.sourceforge.net.

Whitley, Darrell. "A Genetic Algorithm Tutorial." *Statistics and Computing* 4 (1994): 65-85.

Wireless Intrusion Protection. Sunnyvale: Aruba, 2004. Online. Accessed 30 November 2004. Available at http://www.arubanetworks.com/pdf/techbrief-IDS.pdf.

Zamboni, Diego. "Doing Intrusion Detection Using Embedded Sensors -- Thesis Proposal." M.S. thesis, Purdue University, 2000.

Zone Alarm. San Francisco: Zone Labs, 2004. Online. Accessed 8 November 2004. Available at http://www.zonelabs.com.

Zuang, Yongguang and Wenke Lee. "Intrusion Detection in Wireless Ad-Hoc Networks." In *Proceedings of the Sixth Annual Conference on Mobile Computing and Networking Held in Boston, Massachusetts 6-11 August 2000*, 275-283. 2000.