Using Genetic Algorithm for Network Intrusion Detection

Bahaa Mohsen Zbeel♦

1. Introduction

In recent years, Intrusion Detection System (IDS) has become one of the hottest research areas in Computer Security. It is an important detection technology and is used as a countermeasure to preserve data integrity and system availability during an intrusion.

When an intruder attempts to break into an information system or performs an action not legally allowed, we refer to this activity as an intrusion (Graham, 2002; see also Jones and Sielken, 2000).

Intruders can be divided into two groups, external and internal. The former refers to those who do not have authorized access to the system and who attack by using various penetration techniques.

The latter refers to those with access permission who wish to perform unauthorized activities. Intrusion techniques may include exploiting software bugs and system misconfigurations, password cracking, sniffing unsecured traffic, or exploiting the design flaw of specific rotocols(Graham, 2002).

An Intrusion Detection System is a system for detecting intrusions and reporting them accurately to the proper authority. Intrusion Detection Systems are usually specific to the operating system that they operate in and are an important tool in the overall implementation an organization's information security policy (Jones and Sielken, 2000), which reflects an organization's statement by defining the

♦ Babylon University / College of fine Art

rules and practices to provide security, handle intrusions, and recover from damage caused by security breaches.

There are two generally accepted categories of intrusion detection techniques: misuse detection and anomaly detection. Misuse detection refers to techniques that characterize known methods to penetrate a system. These\ penetrations are characterized as a 'pattern' or a 'signature' that the IDS looks for. The pattern/signature might be a static string or a set sequence of actions. System responses are based on identified penetrations. Anomaly detection refers to techniques that define and characterize normal or acceptable behaviors of the system (e.g., CPU usage, job execution time, system calls). Behaviors that deviate from the expected normal behavior are considered intrusions (Bezroukov, 2002; see also McHugh, 2001).

IDSs can also be divided into two groups depending on where they look for intrusive behavior: Network-based IDS (NIDS) and Host-based IDS. The former refers to systems that identify intrusions by monitoring traffic through network devices (e.g. Network Interface Card, NIC). A host-based IDS monitors file and process activities related to a software environment associated with a specific host. Some host-based IDSs also listen to network traffic to identify attacks against a host (Bezroukov, 2002; see also McHugh, 2001). There are other emerging techniques. One example is known as a blocking IDS, which combines a host-based IDS with the ability to modify firewall rules (Miller and Shaw, 1996). Another is called a Honeypot, which appears to be a 'target' to an intruder, but is specifically designed to trap an intruder in order to trace down the intruder's location and respond to attack (Bezroukov, 2002).

The Intelligent Intrusion Detection System (IIDS) is an ongoing project at the Center for Computer Security Research (CCSR) in Mississippi State University. The architecture combines a number of different approaches to the IDS problem, and includes different AI techniques to help identify intrusive behavior (Bridges

and Vaughn, 2001). It uses both anomaly detection and misuse detection techniques and is both a network-based and host-based system. Within the overall architecture of the IIDS, some open-source intrusion detection software tools are integrated for use as security sensors (Li, 2002), such as Bro (Paxson, 1998) and Snort (Roesch, 1999).

Genetic Algorithm (GA) has been used in different ways in IDSs. The Applied Research Laboratories of the University of Texas at Austin (Sinclair, Pierce, and Matzner 1999) uses different machine learning techniques, such as finite state machine, decision tree, and GA, to generate artificial intelligence rules for IDS. One network connection and its related behavior can be translated to represent a rule to judge whether or not a real-time connection is considered an intrusion. These rules can be modeled as chromosomes inside the population. The population evolves until the evaluation criteria are met. The generated rule set can be used as knowledge inside the IDS for judging whether the network connection and related behaviors are potential intrusions (Sinclair, Pierce, and Matzner 1999). The COAST Laboratory in Purdue University (Crosbie and Spafford, 1995) implemented an IDS using autonomous agents (security sensors) and applied AI techniques to evolve genetic algorithms. Agents are modeled as chromosomes and an internal evaluator is used inside every agent (Crosbie and Spafford, 1995).

In the approaches described above, the IDS can be viewed as a rule-based system (RBS) and GA can be viewed as a tool to help generate knowledge for the RBS. These approaches have some disadvantages. In order to detect intrusive behaviors for a local network, network connections should be used to define normal and anomalous behaviors. Sometimes an attack can be as simple as scanning for available ports in a server or a password-guessing scheme. But typically they are complex and are generated by automated tools that are freely available from the Internet. An example can be a Trojan horse or a backdoor that

can run for a period of time, or can be initiated from different locations. In order to detect such intrusions, both temporal and spatial information of network traffic should be included in the rule set. The current GA applications do not address these issues extensively. This paper shows how network connection information can be modeled as chromosomes and how the parameters in genetic algorithm can be defined in this respect. Some examples are used to show the implementation.

The rest of the paper is organized as follows. Section 2 provides a brief introduction to genetic algorithm. Section 3 describes the detailed implementation of applying genetic algorithm to intrusion detection. Section 4 discusses the architecture for the proposed implementation. Section 5 presents the conclusion and future work.

2. Introduction to Genetic Algorithm

Genetic algorithm is a family of computational models based on principles of evolution and natural selection. These algorithms convert the problem in a specific domain into a model by using a chromosome-like data structure and evolve the chromosomes using selection, recombination, and mutation operators. The range of the applications that can make use of genetic algorithm is quite broad (Sinclair, Pierce, and Matzner 1999; see also Whitley, 1994). In computer security applications, it is mainly used for finding optimal solutions to a specific problem.

The process of a genetic algorithm usually begins with a randomly selected population of chromosomes. These chromosomes are representations of the problem to be solved. According to the attributes of the problem, different positions of each chromosome are encoded as bits, characters, or numbers. These positions are sometimes referred to as genes and are changed randomly within a range during evolution. The set of chromosomes during a stage of evolution are called a population. An evaluation function is used to calculate the

"goodness" of each chromosome. During evaluation, two basic operators, crossover and mutation, are used to simulate the natural reproduction and mutation of species. The selection of chromosomes for survival and combination is biased towards the fittest chromosomes.

Figure 1 shows the structure of a simple genetic algorithm. It starts with a randomly generated population, evolves through selection, recombination (crossover), and mutation. Finally, the best individual (chromosome) is picked out as the final result once the optimization criterion is met (Pohlheim, 2001).

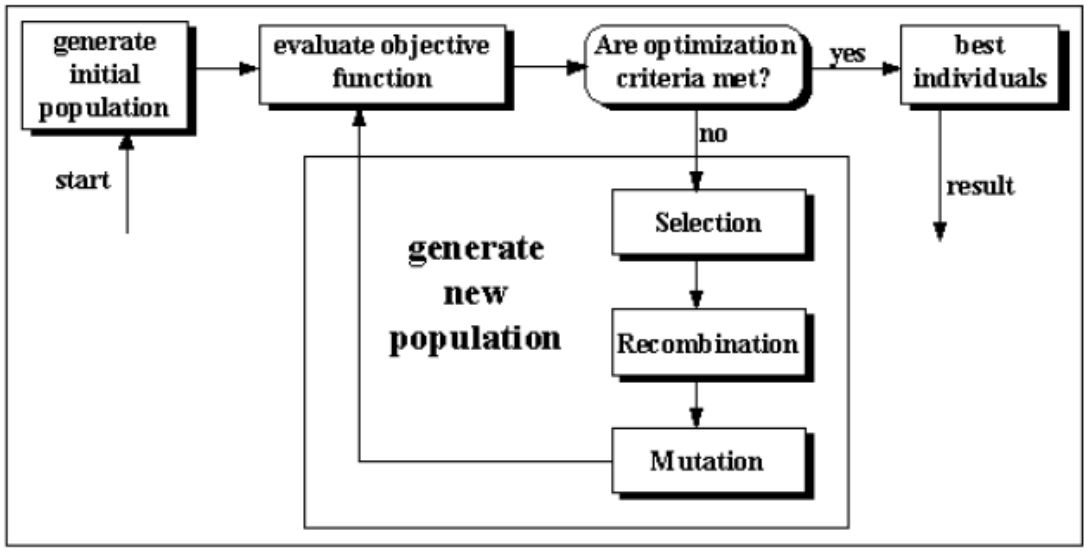

Figure 1. Structure of a simple genetic algorithm (Pohlheim, 2001)

A genetic algorithm is quite straightforward in general, but it could be complex in most cases. For example,during the crossover operation, there could be one-point crossover or even multiple point crossovers. There are also parallel implementations of genetic algorithms. Sometimes series of parameters (for example, mutation rate, crossover rate, population size, chromosome size, number of evolutions or generations, and how the selection is done) needs to be considered with specific selection process. The final goal is to search the solution space in a relatively short period of time (Pohlheim, 2001).

3. Genetic Algorithm Applied to Intrusion Detection

Applying genetic algorithm to intrusion detection seems to be a promising area. We discuss the motivation and implementation details in this section.

3.1 Overview

Genetic algorithms can be used to evolve simple rules for network traffic (Sinclair, Pierce, and Matzner 1999). These rules are used to differentiate normal network connections from anomalous connections. These anomalous connections refer to events with probability of intrusions. The rules stored in the rule base are usually in the following form (Sinclair, Pierce, and Matzner 1999):

if { condition } then { act }

For the problems we presented above, the condition usually refers to a match between current network

connection and the rules in IDS, such as source and destination IP addresses and port numbers (used in TCP/IP network protocols), duration of the connection, protocol used, etc., indicating the probability of an intrusion. The act field usually refers to an action defined by the security policies within an organization, such as reporting an alert to the system administrator, stopping the connection, logging a message into system audit files, or all of the above. For example, a rule can be defined as:

if {the connection has following information: source IP address 124.12.5.18; destination IP address:

130.18.206.55; destination port number: 21; connection time: 10.1 seconds }

then {stop the connection}

This rule can be explained as follows: if there exists a network connection request with the source IP address 124.12.5.18, destination IP address 130.18.206.55, destination port number 21, and connection time 10.1 seconds, then stop this connection establishment. This is because the IP address

124.12.5.18 is recognized by the IDS as one of the blacklisted IP addresses; therefore, any service request initiated from it is rejected.

The final goal of applying GA is to generate rules that match only the anomalous connections. These rules are tested on historical connections and are used to filter new connections to find suspicious network traffic.

In this implementation, the network traffic used for GA is a pre-classified data set that differentiates normal network connections from anomalous ones. This data set is gathered using network sniffers (a program used to record network traffic without doing something harmful) such as Tcpdump (http://www.tcpdump.com) or Snort (http://www.snort.com). The data set is manually classified based on experts' knowledge. It is used for the fitness evaluation during the execution of GA. By starting GA with only a small set of randomly generated rules, we can generate a larger data set that contains rules for IDS. These rules are "good enough" solutions for GA and can be used for filtering new network traffic.

3.2 Data Representation

In order to fully exploit the suspicious level, we need to examine all fields related with a specific network connection. For simplicity, we only consider some obvious attributes for each connection. The definition of rules (for TCP/IP protocols) is shown in Table 1.

The corresponding rule for the "Example Value" attribute in Table 1 could be translated as:

if {the connection has following information: source IP address 209.11.??.??; destination IP address: 130.18.176+?.??; source port number: 42335; destination port number: 80; connection time: 482 seconds; the connection is stopped by the originator; the protocol used is TCP; the originator sent 7320 bytes of data; and the responder sent 38891 bytes of data }

then {stop the connection}

Table 1. Rule definition for connection and range of values of each field

Attribute	Range of Values	Example Values	Descriptions
Source IP address	0.0.0.0~255.255.255.255	d1.0b.**.** (209.11.??.??)	A subnet with IP address 209.11.0.0 to 209.11.255.255
Destination IP address	0.0.0.0~255.255.255.255	82.12.b*.** (130.18.176+?.??)	A subnet with IP address 130.18.176.0 to 130.18.255.255
Source Port Number	0~65535	42335	Source port number of the connection
Destination Port Number	0~65535	00080	Destination port number, indicates this is a http service
Duration	0~99999999	00000482	Duration of the connection is 482 seconds
State	1~20	11	The connection is terminated by the originator, for internal use
Protocol	1~9	2	The protocol for this connection is TCP
Number of Bytes Sent by Originator	0~9999999999	0000007320	The originator sends 7320 bytes of data
Number of Bytes sent by Responder	0~9999999999	0000038891	The responders sends 38891 bytes of data

We can convert the above example into the chromosome form, as described in Figure 3.

(d, 1, 0, b, -1, -1, -1, -1, 8, 2, 1, 2, b, -1, -1, -1, 4, 2, 3, 3,
5, 0, 0, 0, 8, 0, 0, 0, 0, 0, 0, 4, 8, 2, 1, 1, 2, 0, 0, 0,
0, 0, 0, 7, 3, 2, 0, 0, 0, 0, 0, 0, 3, 8, 8, 9, 1)

Figure 3. Chromosome structure for example in Table 1

Altogether there are fifty-seven genes in each chromosome. For simplicity, we use hexadecimal representations for the IP addresses. The rule can be explained as follows: if a network connection with source IP address 209.11.??.?? (209.11.0.0 ~ 209.11.255.255), destination IP address 130.18.176.?? (130.18.176.0 ~ 130.18.255.255), source port number 42335, destination port number 80, duration time 482 seconds, ends with state 11 (the connection terminated by the originator), uses protocol type 2 (TCP), and the originator sends 7320 bytes of data, the responders sends 38891 bytes of data, then this is a

suspicious behavior and can be identified as a potential intrusion. The actual validity of this rule will be examined by matching the historical data set comprised of connections marked as either anomalous or normal. If the rule is able to find an anomalous behavior, a bonus will be given to the current chromosome. If the rule matches a normal connection, a penalty will be applied to the chromosome. Clearly no single rule can be used to separate all anomalous connections from normal connections. The population needs evolving to find the optimal rule set.

In the example shown in Table 1, some wild cards (the '*' character and the '?' character) are used and the corresponding genes within the chromosome are shown as –1. These wild cards are used to represent an appropriate range of specific values (Crosbie and Spafford, 1995). It is useful when representing a network block (a range of IP addresses or port numbers) in a rule. Once the spatial information is included in the rules, the capability of the IDS can be greatly improved as an intrusion may initiate from many different locations. The inclusion of the duration time of a network connection in the chromosome ensures incorporation of temporal information for network connections. The maximum value of duration time is 99999999 seconds, which is more than a year. This is helpful for identifying intrusions because complex intrusions may span hours, days, or even months.

The genetic algorithm starts with a population that has randomly selected rules. The population can evolve by using the crossover and mutations operators. Due to the effectiveness of the evaluation function, the succeeding populations are biased toward rules that match intrusive connections. Ultimately as the algorithm stops, rules are selected and added into the IDS rule base.

3.3 Parameters in Genetic Algorithm

There are many parameters to consider for the application of GA. Each of these parameters heavily influences the effectiveness of the genetic algorithm. We will discuss the methodology and related parameters in the following section.

Evaluation function

The evaluation function is one of the most important parameters in genetic algorithm. The proposed implementation differs from the scheme used by (Crosbie and Spafford, 1995) in that the definition on calculations of outcome and fitness is different. The following steps are used to calculate the evaluation function. First the overall outcome is calculated based on whether a field of the connection matches the pre classified data set, and then multiply the weight of that field. The Matched value is set to either 1 or 0.

$$Outcome = \sum_{i=1}^{57} Matched * Weight_i$$

The order of weight values in the function is shown in Figure 4. These orders are categorized according to different fields in the connection record as reported by network sniffers. Therefore, all genes representing destination IP address field have the same weight. The actual values can be finely tuned at execution time.

Thebasic idea behind this order is the importance of different fields in TCP/IP packets. This scheme is straightforward and intuitive. Destination IP address is the target of an intrusion while the source IP address is the originator of the intrusion. These are the most important pieces of information needed to capture an intrusion. Destination port number indicates to applications that the target system is running (for example, FTP service usually runs on port 21). Some IP addresses are more probable targets for intrusions—for example, IP addresses for military domains. Domain-specific information is less important compared with the source IP addresses. Other parameters like duration, bytes sent by the originator, bytes sent by the receiver, and state are usually less important than

the above fields but are still useful. The protocol and source port number fields are commonly dispensable and are used for identifying some specific intrusions.

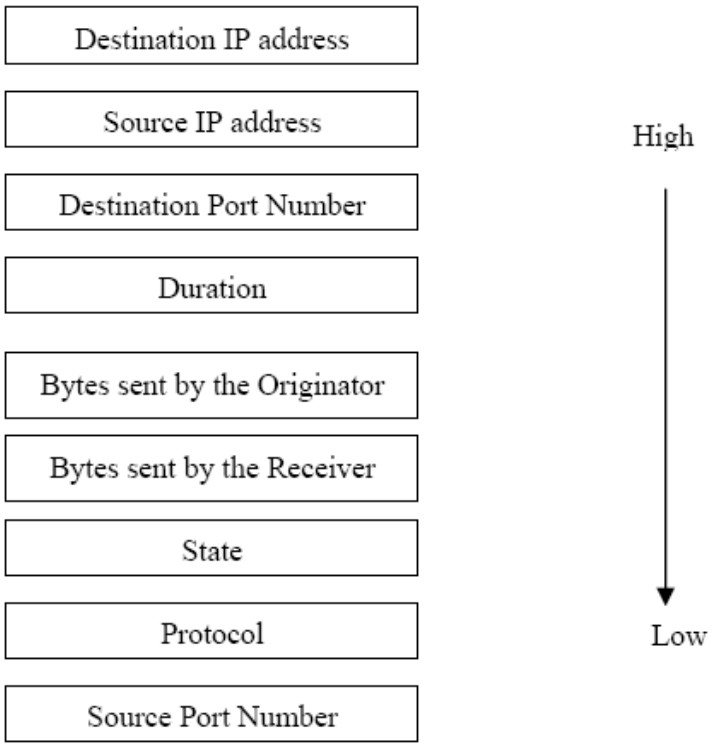

Figure 4. Order of weights for fields in the evaluation function

The absolute difference between the outcome of the chromosome and the actual suspicious level is then computed using the following equation. The suspicious_level is a threshold that indicates the extent to which two network connections are considered a "match." The actual value of suspicious_level reflects observations from historical data.

$$\Delta = | \, outcome - suspicious\_level \, |$$

Once a mismatch happens, the penalty value is computed using the absolute difference. The ranking in the equation indicates whether or not an intrusion is easy to identify.

$$penalty = (\frac{\Delta * ranking}{100})$$

The fitness of a chromosome is computed using the above penalty:

fitness = 1- penalty

Obviously, the range of the fitness value is between 0 and 1. By defining evaluation, we have incorporated both temporal and spatial information needed for identification of network intrusions.

Crossover and Mutation

Traditional genetic algorithms have been used to identify and converge populations of candidate hypotheses to a single global optimum. For this problem, a set of rules is needed as a basis for the IDS. As mentioned earlier, there is no way to clearly identity whether a network connection is normal or anomalous just using one rule. Multiple rules are needed to identify unrelated anomalies, which means that several good rules are more effective than a single best rule (Sinclair, Pierce, and Matzner 1999). Another reason for finding multiple rules is that because there are so many network connection possibilities, a small set of rules will be far from enough.

Using the genetic algorithm, we need to find local maxima (a set of "good-enough" solutions) as opposed to the global maximum (the best solution) (Sinclair, Pierce, and Matzner 1999). The niching techniques can be used to find multiple local maxima (Miller and Shaw, 1996; see also Sinclair, Pierce, and Matzner 1999). It is based on the analogy to nature in that within each environment, there are different subspaces (niches) that can support different types of life. In a similar manner, genetic algorithm can maintain the diversity of each population in a multimodal domain, which refers to domains requiring the identification of multiple optima. Two basic methods, crowding and sharing can be used for niching. The crowding method uses the most similar member for replacement to slow down the population to converge towards a single point in the following generations. The sharing method reduces the fitness of individuals, that have highly similar members and forces individuals to evolve to other local

maxima that may be less populated. The similarity metrics used in these techniques can be phenotype to genotype similarity such as Hamming distance between bit representations, or phenotype similarity such as the relation between two network connections in this problem. The latter one is more fitful for finding rules used in IDS. The disadvantage of this approach is that it requires more domain-specific knowledge (Miller and Shaw, 1996; see also Sinclair, Pierce, and Matzner 1999).

The mutation operation should be meaningful during evolution. For example, each segment of the P address should not exceed 255 (decimal representation). Mutations should be done following the requirements specified in able 1. These limitations can be enforced by defining proper mutation rules.

Other parameters

There are also other parameters that need to be considered, such as mutation rate, crossover rate, number of

populations, and number of generations. These parameters should be adjusted according to the application

environment of the system and the organization's security policy.

4. System Architecture

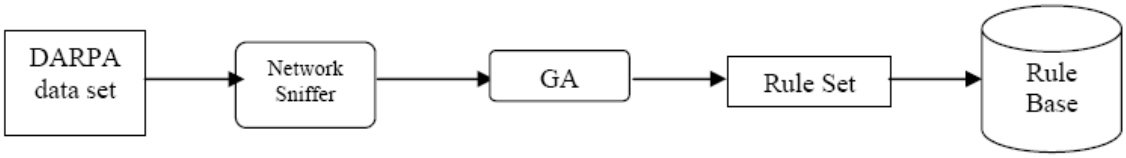

Figure 5. Architecture of applying GA into intrusion detection

Figure 5 shows the structure of this implementation. We need to collect enough historical data that includes both normal and anomalous network connections. The MIT Lincoln Laboratory (http://www.ll.mit.edu/) data set for testing IDSs, which is represented in the Tcpdump binary format, is a good choice. This is the first art inside the system architecture. This data set is analyzed by the network sniffers and results are fed into GA for fitness valuation. Then the GA is

executed and the rule set is generated. These rules are stored in a database to be used by the IDS.

5. Conclusion and Future Work

In this paper, we discussed a methodology of applying genetic algorithm into network intrusion detection techniques. A brief overview of Intrusion Detection System (IDS), genetic algorithm, and related detection techniques are discussed. The system architecture is also introduced. Factors affecting the GA are addressed in detail. This implementation of genetic algorithm is unique as it considers both temporal and spatial information of network connections during the encoding of the problem; therefore, it should be more helpful for identification of network anomalous behaviors.

Future work includes creating a standard test data set for the genetic algorithm proposed in this paper and applying it to a test environment. Detailed specification of parameters to consider for genetic algorithm should be determined during the experiments. Combining knowledge from different security sensors into a standard rule base is another promising area in this work.

REFERENCES

Bezroukov, Nikolai. 19 July 2003. “Intrusion Detection (general issues).” Softpanorama: Open Source Software Educational Society. Nikolai Bezroukov. URL:http://www.softpanorama.org/Security /intrusion_detection.shtml (30Oct. 2008).

Bridges, Susan, and Rayford B. Vaughn. 2007. “Intrusion Detection Via Fuzzy Data Mining.” In Proceedings of 12th Annual Canadian Information Technology Security Symposium, pp. 109-122. Ottawa, Canada.

Crosbie, Mark, and Gene Spafford. 2006. “Applying Genetic Programming to Intrusion Detection.” In Proceedings of 1995 AAAI Fall Symposium on Genetic Programming,pp.1-8.Cambridge,Massachusetts.URL: http://citeseer.nj.nec.com/crosbie95applying.html (30 Oct. 2003).

Graham, Robert. Mar. 21, 2009. "FAQ: Network Intrusion Detection Systems." RobertGraham.com Homepage. Robert Graham. URL: http://www.robertgraham.com/pubs/network-intrusion-detection.html (30 Oct. 2003).

Jones, Anita. K. and Robert. S. Sielken. 2008. "Computer System Intrusion Detection: A Survey." Technical Report. Department of Computer Science, University of Virginia, Charlottesville, Virginia.

Li, Wei. 2006. "The integration of security sensors into the Intelligent Intrusion Detection System (IIDS) in a cluster environment." Master's Project Report. Department of Computer Science, Mississippi State University.

McHugh, John, 2004. "Intrusion and Intrusion Detection." Technical Report. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University.

Miller, Brad. L. and Michael J. Shaw. 2005. "Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization." In Proceedings of IEEE International Conf. on Evolutionary Computation, pp. 786-791. Nagoya University, Japan.

Paxson, Vern. 2005. "Bro: A System for Detecting Network Intruders in Real-time." In Proceedings of 7th USENIX Security Symposium, pp. 31-51. San Antonio, Texas.Pohlheim, Hartmut. 30 Oct. 2004. "Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms." Genetic and Evolutionary Algorithm Toolbox. Hartmut Pohlheim. URL: http://www.geatbx.com /docu/algindex.html.

Roesch, Martin. Nov. 7-12, 2002. "Snort - Lightweight Intrusion Detection for Networks." In Proceedings of 13th Systems Administration Conf. (LISA '99), pp. 229-238.Seattle, Washington.

Sinclair, Chris, Lyn Pierce, and Sara Matzner. 2006. "An Application of Machine Learning to Network Intrusion Detection." In Proceedings of 2006

Annual Computer Security Applications Conf(ACSAC), pp. 371-377. Phoenix, Arizona. URL: http://www.acsac.org/1999/papers/fri-b-1030- sinclair.pdf (30 Oct. 2006).

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