ID-SOMGA: A Self Organising Migrating Genetic Algorithm-Based Solution for Intrusion Detection

Olusegun Folorunso
Department of Computer Science, University of Agriculture
Abeokuta, Ogun State, Nigeria
E-mail: folorunsolusegun@yahoo.com

Oluwatobi O. Akande
Department of Computer Science, University of Agriculture
Abeokuta, Ogun State, Nigeria
E-mail: akandetobi@gmail.com

Adewale O. Ogunde
Department of Mathematical Sciences, Redeemer's University (RUN)
Redemption City, Mowe, Ogun State, Nigeria
E-mail: adewaleogunde@yahoo.com

Olufunke R. Vincent
Department of Computer Science, University of Agriculture
Abeokuta, Ogun State, Nigeria
E-mail: rebecca.vincent@yahoo.com

Abstract

The study examined the detection of attacks against computer networks, which is becoming a harder problem to solve in the field of Network security. A problem with current intrusion detection systems is that they have many false positive and false negative events. Most of the existing Intrusion detection systems implemented depend on rule-based expert systems where new attacks are not detectable. In this study, optimization algorithms were added to intrusion detection system to make them more efficient. Self Organizing Migrating Genetic Algorithm (SOMGA) was integrated into intrusion detection system to obtain a more efficient intrusion detection system called ID-SOMGA. This study provides an equally efficient method to implement an intrusion detection system that returns very low false positives. Due to the complexities involved in security issues, and the implementation of the work, selected values of the network log was used to implement the system in order to reduce some of these complexities. The Self Organizing Migrating Genetic Algorithm – Intrusion Detection System was tested and values of the result were compared with that of an IDS with Genetic Algorithm Intrusion Detection System. In terms of detection rates, ID-SOMGA was found to be slower than an IDS with GA, the false positives in ID-SOMGA was lower than what obtains with genetic algorithm. Both schemes were able to identify new patterns almost in the same way. The ID-SOMGA system that was developed improved the security of systems in networked settings allowing for confidentiality, integrity and availability of system resources.

Keywords: Intrusion Detection, Computer Networks, Genetic Algorithms, Computer security

1. Introduction

An Intrusion Detection System (IDS) is a system for detecting intrusions in computer networks and reporting them accurately to the proper authority. Intrusion Detection Systems are important tools in the overall implementation of an organization's information security policy, as they reflect the organization by defining the rules and practices to provide security, handle intrusions, and recover from damage caused by security breaches (Bezroukov, 2003).

There are two generally accepted categories of intrusion detection techniques: misuse detection and anomaly

detection (Wei, 2004). 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 look for. Anomaly detection refers to techniques that define and characterize normal or acceptable behaviours of the system (e.g., CPU usage, job execution time, system calls). Behaviours that deviate from the expected normal behaviour are considered intrusions (Bezroukov, 2003; McHugh, 2001).

However, these systems often introduce significant computational overhead. Furthermore, many of them do not deal properly with so called 'rare' classes; the classes that have significantly smaller number of elements than the rest of the classes. This problem occurs mostly because of the tendency for generalization that most of these techniques exhibit. Intrusions can be considered rare classes since it is reasonable to assume that the amount of intrusive traffic is considerably smaller than the amount of normal traffic. Thus, we need a machine learning technique that is capable of dealing with this issue.

Genetic algorithm (GA) can be used to classify network connections. GA is robust, inherently parallel, and adaptable. Moreover, due to its inherent parallelism, it offers a possibility to implement the system using reconfigurable devices without the need of deploying a microprocessor. Here, we further investigate the hybridization of two algorithms; SOMA and GA into intrusion detection systems. Hence, this study provides an equally efficient method to implement an intrusion detection system that returns very low false positives.

2. Literature Review

The concept of intrusion detection dates back to the 1980s, and it has been defined to be the potential possibility of a deliberate unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable. Intrusion detection in order to identify attacks on computer systems has been a challenging problem in the domain of network security for quite some time. Software to detect network intrusions protects a computer network from unauthorized uses thereby preventing malicious activities. With growing problems in network security and the need to develop sophisticated and robust solutions, researchers across the world developed innovative methods to construct an IDS on a training and testing data set, popularly referred to as the KDD Cup 98 data set (Bancovic et. al., 2009).

The Intrusion detection system in a similar way complements the firewall security. Though, firewalls could filter incoming traffic from the Internet; however, there are ways to circumvent the firewall. There is no single security measure sufficient to independently protect information systems. Having a layered security architecture greatly reduces risk to system users. One invaluable layer is comprised of network intrusion detection systems (Selvakani and Rajesh, 2007).

An Intrusion Detection System has a database of attack signatures. The attack signatures are patterns of different types of previously detected attacks. If the sensors detect any malicious activity, it matches the malicious packet against the attack signature database. In case it finds a match, the sensor reports the malicious activity to the management console. The sensor can take different actions based on how they are configured. For example, the sensor can reset the TCP connection, modify the access control list on the gateway router or the firewall or send an email notification to the administrator for appropriate action.

There are broadly two types of Intrusion Detection systems. These are hosts based Intrusion Detection System and network based Intrusion Detection System. A Host based Intrusion Detection system has only host based sensors and a network based Intrusion detection system has network-based sensor. Intrusion detection systems provide the following advantages among other to the security infrastructure of an organization: lower cost of ownership, easier to deploy, detect network based attacks: retaining evidence, real time detection and quick responses, and detection of failed attacks.

A field of research in intrusion detection has focused on the ability of the IDS to detect intrusion attempts, using statistical and algorithm based approaches, and discern between what is merely anomalous (unknown to the system) and not a risk, and what is potentially harmful to the system and should be prevented. Since precious time is used in detecting an attack, these systems will need to adopt some autonomous response capability, using not only risk and response categorization but also a response escalation algorithm, similar to biological and immune response systems. Most of these systems also spend time learning about the systems they are protecting and establishing a baseline, before they and are able to function as intended. Since much of this data is available from system vendors, greater cooperation among vendors will obviate much of the need for this learning process and improve intrusion detection systems.

2.1 IDS design models

Price has categorized intrusion detection systems, based on their detection models, into the following:

- Misuse detection model detects intrusions by looking for activity that corresponds to known intrusion techniques (signatures) or system vulnerabilities.
- Anomaly detection model detects intrusions by looking for activity that is different from a user's or systems normal behaviour.

The following observations need to be noted:

- Both anomaly detection and misuse detection will depend on a learning process or the establishment of a baseline: in the anomaly detection case, logging 'normal' activity, in the misuse detection model, logging or cataloguing intrusion signatures and system vulnerabilities. In both cases, therefore, there will be a period of time when the intrusion detection system does not have the baseline data it needs to compare the new activity against, and will therefore be unable to provide information about the attack or respond to the attack.
- The misuse detection model can further be characterized as backward-looking or reactive, relying on historical data of past attack signatures and system vulnerabilities; the anomaly detection model can be characterized as forward looking, detecting heretofore unknown system behaviour.

2.2 Intrusion detection model

Any set of actions that attempt to compromise the integrity, confidentiality, or availability of resource is termed an intrusion. An intruder is the individual or group of individuals who initiates the action in the intrusion. Intrusion detection systems are based on the belief that an intrusion will be reflected by a change in the normal patterns of resource usage. As such, intrusion detection systems have been developed to monitor specific types of activities and announce anomalies in the behaviour observed. The anomalies announced by an intrusion detection system serve as an indication that an intrusion may be in progress.

2.3 Self Organising Migrating Genetic Algorithm (SOMA)

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 et. al, 1999; 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).

SOMGA is a hybridization of self organizing migrating algorithm and the simple binary genetic algorithm (Kusum and Dipti, 2009). SOMA is not a so well known algorithm which surfaced around the year 2000. The main feature of that motivated incorporating SOMA into GA is the fact that it works with both high and low population size and has more exploration capabilities. The hybridizing of SOMA with GA is expected to increase the reliability, efficiency and robustness of the search/optimization algorithm.

3. Methodology

Intrusion detection systems can of their own accord deal with the issue of detecting security breaches but it has been observed that there issues of false positives, having to have known the attack signatures or nature before hand has led to the inclusion of a learning algorithm as the self organising migrating genetic algorithm SOMGA.

3.1 Application of Self Organizing Migrating Genetic Algorithm (SOMGA) to Intrusion Detection System– ID-SOMGA

SOMGA like Genetic algorithms can be used to evolve simple rules for network traffic. These rules can then be 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:

if { condition } then { act }

The condition 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 SOMGA 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 SOMGA is a pre-classified data set that differentiates normal network connections from anomalous ones. The data set is manually classified based on experts' knowledge. It is used for the fitness evaluation during the execution of SOMGA. By starting SOMGA 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 SOMGA 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. Considering the KDDCUP '98 dataset, we have about 41 fields that the rules would be based on. Thus 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}*

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 behaviour 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 behaviour, 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 above, some wild cards (the '*' character and the '?' character) are used to represent an appropriate range of specific values. 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 SOMGA 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 SOMGA

There are many parameters to consider for the application of SOMGA. Each of these parameters heavily influences the effectiveness of the algorithm.

3.3.1 Fitness function

The fitness function is one of the most important parameters in SOMGA. 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.

$$Outcome = \sum_{i=1}^{6} Matched * Weight_i \quad \ldots\ldots(1)$$

The order of weight values in the function is shown in the figure below. 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. The basic idea behind this order is the importance of different fields in TCP/IP packets. This scheme is straight forward 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. This is shown in figure 2.

The absolute difference between the outcome of the chromosome and the actual suspicious level is then computed using the following equations. 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.

$$\Delta = |outcome - suspicious\_level| \ldots\ldots(2)$$

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.

$$penalty = \left(\frac{\Delta * ranking}{100}\right) \ldots\ldots(3)$$

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

$$fitness = 1 - penalty \quad \ldots\ldots(4)$$

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

3.3.2 Generating New Population

Traditional genetic algorithms have been used to identify and converge populations of candidate hypotheses to a single global optimum. This same attribute can be extended to the SOMGA. For this problem, a set of rules is needed as a basis for the IDS. As stated 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 mean that several good rules are more effective than a single best rule (Sinclair et. al., 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.

Thus, we need to find local maxima (a set of "good-enough" solutions) as opposed to the global maximum (the best solution). Once the initial population (of chromosomes) is evaluated, the algorithm experiments with new

generations and iteratively refines the initial outcomes so that those that are most fit are more probable to be ranked higher as results. The objective is to produce new generation of chromosomes to evaluate.

3.3.3 Crossover

In essence, the crossover operation creates new chromosomes that share optimistic characteristics of the parent chromosomes while at the same time lowering the negative attributes in a child chromosome (Marakas, 2003). Figure 3 provides an example of a crossover of chromosomes from the parents to their offspring.

Although this step is typical in most genetic algorithms, in the case of this project's the crossover operation may not be beneficial. While a Source or Destination IP may be bound by upper and lower IP settings, a crossover of the IP octet values would probabilistically not be advantageous. For example, the crossover of the parental values of 209.103.51.134 and 101.1.25.193 could result in child IP addresses of 209.103.25.193 and 101.1.51.134. However, the probability that this offspring will be potential suspicious Source or Destination IP addresses is low.

3.3.4 Mutation

This phase randomly alters a gene's value to create a different one (Marakas, 2003). Figure 4 details how a gene's (or allele's) value is changed thereby creating a new chromosome. Concerning the applicability of this step with the network intrusion chromosome, as was the case in the crossover step above the probability of useful outcomes is minimal.

3.3.5 Sorting

The last step in the process of generating a new population is sorting based on already defined criteria. The new population is arranged in order so as to eliminate repeating chromosomes while fitness of the generated population is carried out. The sorting for simplicity sake makes use of the calculated fitness value they are arrange in decreasing order, starting from the best one of the new population.

4. Implementation

The Java Development Kit – jdk 1.6u12, Netbeans IDE and Java Genetic Algorithm Package (jgap_3.4.4) were the tools used for implementation. Four java classes were used to implement the intrusion detection system. The classes are: IDS.java, IPv4Chromosomes.java, IDSFitnessFunction.java, Progress.java. The IDS.java class is the main entry point of the program. It contains several methods of which the first it loads makes a call to the Progress.java class to load the essentials of the IDS system. The code consists of a function class in the IDSFitnessFuntion.java class which contains the problem domain and fitness code, a value class that holds the IPv4 chromosome attributes in the IPv4Cromosomes.java, and the class containing the main method for configuration and the fitness function call in the IDS.java class.

The fitness function works with the JGAP framework by creating the six genes and then compares them with a suspicious range of values for those genes. If they match, the outcome or returned value is set to a maximum amount for the gene. If not, then a lower value is returned. It should be noted that the "suspicious" values as well as the "weight" values were hard coded for test and demonstration purposes. The UML diagram for the system is shown in figure 5.

Finally, the JGAP framework takes each gene value and returns "a more optimal" result for the genes. Running the SOMGA produced the following output (in Figure 10.) from the input of Source IP = 2098411163 (which is an IPv4 address of 125.19.54.155)| Destination IP = 1828782356 (which is an IPv4 address of 109.1.1.20)| Destination Port = 8184 | Protocol = 5 | Originator Bytes = 10500 | Responder Bytes = 2500000 (see table 2).

A rule set is created from the computation which is then compared to a set of known intrusive patterns. If there happens to be one or more match, it is reported as a suspected intrusion and connection from the source IP address is terminated. Some snapshots of the workings of the system are shown in figures 6 to 9.

4.1 Evaluation of the System

The IDS developed was evaluated on three specific criteria which are very important issues with intrusion detection systems. They are: Speed, Accuracy, Adaptability, Size of data. It was observed that the comparisons between an ordinary intrusion detection system, and intrusion detection system with genetic algorithm and the ID-SOMGA gave results that further confirmed that the inclusion of learning algorithms in the intrusion detection helps in the automation of the system to a high level. For a larger part of the test, the outcomes of the systems that had the optimization algorithms was better than that of the ordinary system with both IDS with GA and IDS with SOMGA have similar figure. It was however observed that the system with SOMGA could operate better on low population sizes as well as high population sizes while the GA system performed best when it have

a high population size to work on.

A comparison was made between ordinary IDS, one with GA and the system developed. The results obtained from the test are given in the tables 3, 4, 5, and 6. It is pertinent to note that there are certain issues that should be considered while testing and evaluating intrusion detection system. They include: Coverage of know attacks, Probability of false alarms, Probability of detection, Ability to detect attacks not known before, False alarm rate, Ease of use, Ease of maintenance. The experimental set up is shown in table 3. To consider the detection rate, we experiment using Genetic algorithm and SOMGA as against time. The result is shown in table 4.

Detection rate is the time taken to scan through and discover certain the first intrusive pattern. SOMGA spent more time in the computation and this can be explained away by the extra overhead incurred through the addition of a sorting process. Figure 10 shows the plot of the detection rate as against time. Though, it was expected that the detection rate would increase with time but after the second time set, but decrease of the detection rates were observed.

As observed from the chart shown above, it was found out that the operations with GA seems to be faster than when operation were carried out using SOMGA. The time lag for SOMGA is explained off by the fact that the sorting of the new newly generated population adds an overhead. This is observed in figure 10. Table 5 shows the false alarm rate against the performance of GA and SOMGA.

From the results, the false alarm rate given by the intrusion detection system that has genetic algorithm is much more than that of the newly tested system. It was also observed that subsequent values while testing the application for false alarms were reducing. This actually explains the fact that the system was learning the new intrusive patterns. The next experiment conducted on the system was the ability to identify attack. This was done by using some trials time T, to get new intrusive patterns. The result is shown in table 6. It is a very good feature for the IDS to be able to identify new intrusive patterns. With the introduction of new intrusive patterns, both systems performed equally. They had similar results in the identification process. Table 7 gives a summary of the comparison of the system using both GA and SOMGA.

5. Conclusions and Future Works

With the proliferation of computer networks, intrusions and security threats are inevitable most especially in competitive environments and hence the need to develop more sophisticated ways of improving security is an ongoing and a rather challenging task. In achieving this, the development of a system such as ID-SOMGA that would improve performance is a step in the right direction. As with any security product designed to protect information systems and the data they process, there are limitations. If the intrusion detection system lacks rules clever enough to detect traffic of interest, the system will neither send alerts nor drop packets appropriately. Thus, keeping signatures updated and maintaining other rules intended to find exactly what you want is an ongoing endeavor. The study focused on the procedure for incorporating a new algorithm SOMGA into an intrusion detection system as a way of getting considerable improvement over using basic genetic algorithms. The Self Organizing Migrating Genetic Algorithm – Intrusion Detection System was tested and values of the result were compared with that of an IDS with Genetic Algorithm Intrusion Detection System. In terms of detection rates, ID-SOMGA was found to be slower than an IDS with GA, the false positives in ID-SOMGA was lower than what obtains with genetic algorithm. Both schemes were able to identify new patterns almost in the same way. The ID-SOMGA system that was developed improved the security of systems in networked settings allowing for confidentiality, integrity and availability of system resources. As a future work, other learning algorithms suitable for optimization could also be looked into as a way of achieving a secured environment for distributed computing. As there would always be the need to share resources, the issue of intrusion is unavoidable and as such the security of such systems becomes an important issue.

References

Bankovic Zorana, José M. Moya, Álvaro Araujo, Slobodan Bojanic and Octavio Nieto-Taladriz. (2006). Improving Network Security Using Genetic Algorithm Approach", *Computers & Electrical Engineering,* Vol.33, Issue 5-6, pp. 438-451.

Bezroukov, Nikolai. (2003). "Intrusion Detection (General Issues)." Softpanorama: Open Source Software Educational Society. Nikolai Bezroukov. URL: http://www.softpanorama.org/Security/intrusion_detection.shtml.

Kusum Deep and Dipti. (2009). Reliability Optimization of Complex Systems through C-SOMGA, *Journal of Information and Computing Science* Vol. 4, No. 3, pp. 163-172

Marakas, G.M. (2003). *Modern Data Warehousing, Mining, and Visualization:* Core Concepts. Upper Saddle River, NJ: Pearson Education.

McHugh, John. (2001). "Intrusion and Intrusion Detection." Technical Report. CERT Coordination Center, Software Engineering Institute, Carnegie Mellon University.

Pohlheim, H. (2001). Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms. Genetic and Evolutionary Algorithm Toolbox. Retrieved 10-11- 2009 from: http://www.geatbx.com/docu/algindex.html.

Selvakani S. and Rajesh R.S. (2007). Genetic Algorithm for framing rules for Intrusion Detection, *International Journal of Computer Science and Network Security,* VOL.7 No.11, November 2007.

Sinclair, C. et al. (1999). *An Application of Machine Learning to Network Intrusion Detection.* Retrieved 10-11-2009 from: http://www.acsac.org/1999/papers/fri-b-1030-sinclair.pdf.

Wei Li. (2004). Using Genetic Algorithm for network intrusion detection (2004) In Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference at http://www.security.cse.msstate.edu/docs/Publication

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

Table 1. Data Representation Table (from Bankovic, 2006)

Attributes	Range of values	Example Values	Description
Source IP Address	0.0.0.0 ~255.255.255.255	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	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 connection is 482
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 byte sent by originator	0~9999999999	0000007320	The originator sent 7320 bytes of data
Number of bytes sent by Responder	0~9999999999	000038891	The responder sent 38891 bytes of data

Table 2. Table showing details of a network connection

Source IP	125.19.54.155
Destination IP	109.1.11.20
Destination Port	8184
Protocol	5
Originator Bytes	10500
Responder Byte	2500000

Table 3. Experimental setup

Network records	50
Intrusions	8
Time	In seconds
New intrusive patterns	10

Table 4. Attack detection rate table

Time	Using GA	Using SOMGA
T_1	0.40006	0.60061
T_2	0.67098	0.73800
T_3	0.52004	0.55002
T_4	0.53036	0.62954
T_5	0.67846	0.76540

Table 5. False alarm rate table

Alarms	**Using GA**	**Using SOMGA**
A_1	0.35	0.31
A_2	0.305	0.30
A_3	0.32	0.29
A_4	0.31	0..29
A_5	0.30	0.285

Table 6. New attack identification table

Trials	**Using GA**	**Using SOMGA**
T_1	0.8	0.7
T_2	0.8	0.9
T_3	0.9	0.9
Average	0.83	0.83

Table 7. Evaluation Summary

	IDS with GA	**IDS with SOMGA**
Detection rate	Better	Good
False alarms rate	Good	Better
Ability to identify attacks	Good	Good
Adaptability/learning	Good	Good

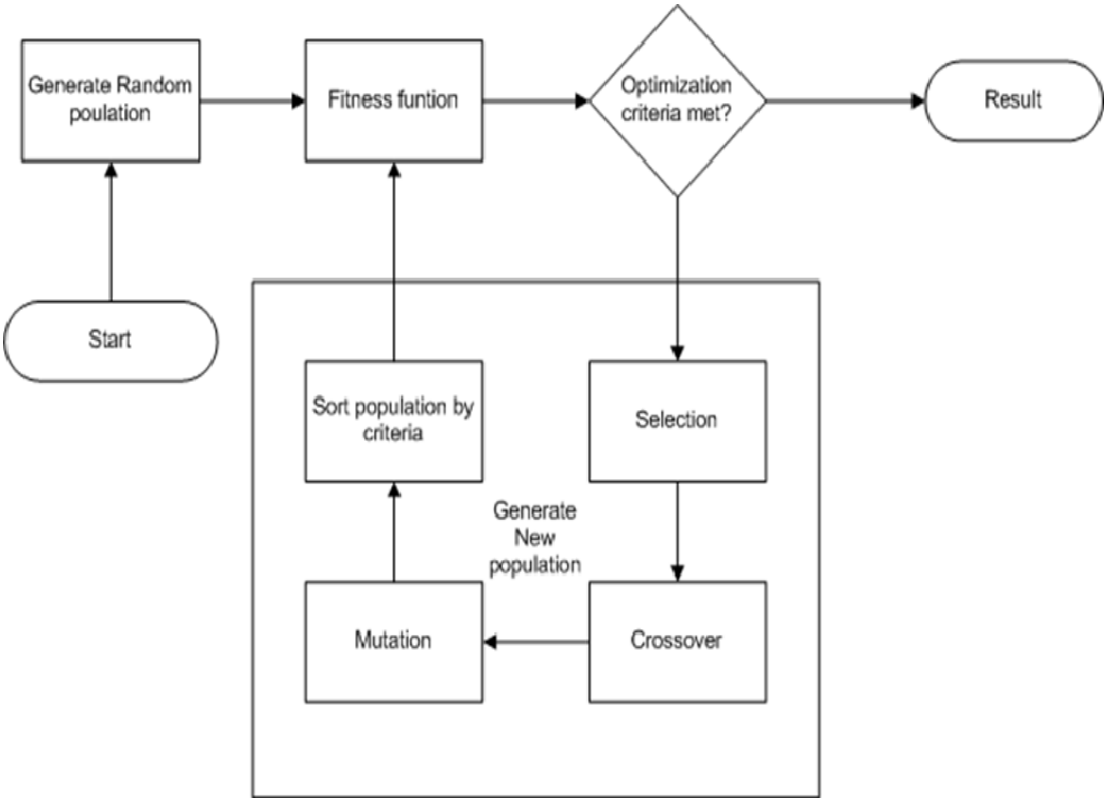

Figure 1. A model diagram for SOMGA

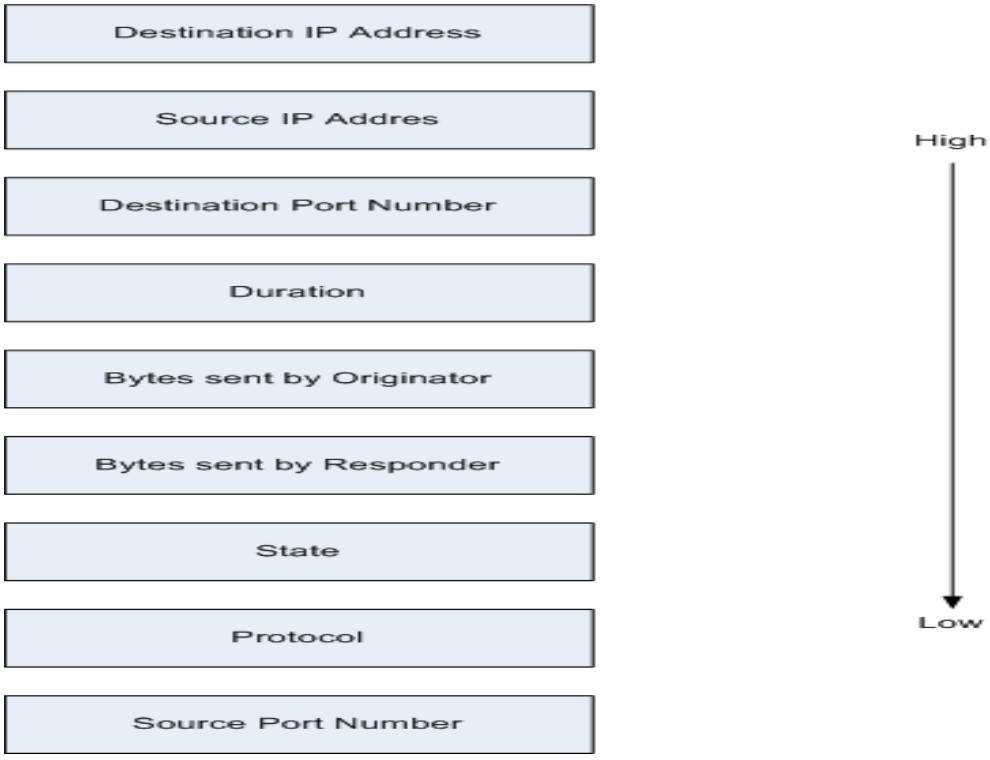

Figure 2. Weight order for network audit log (Adapted from Polheim, 2001)

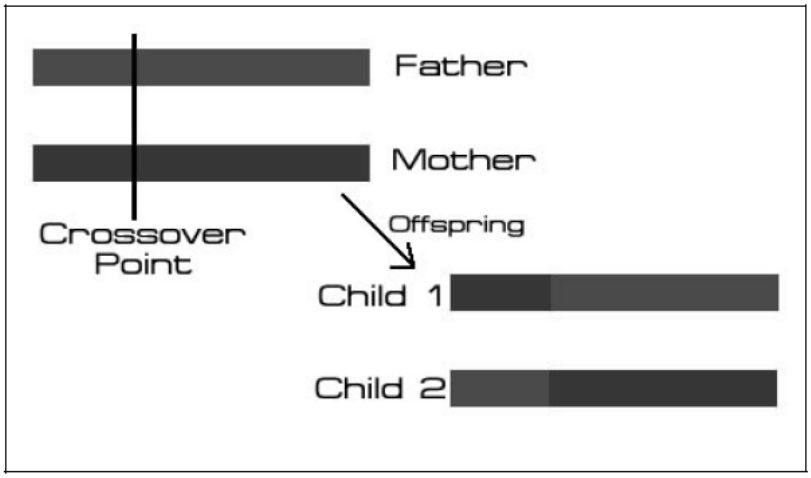

Figure 3. Crossover diagram

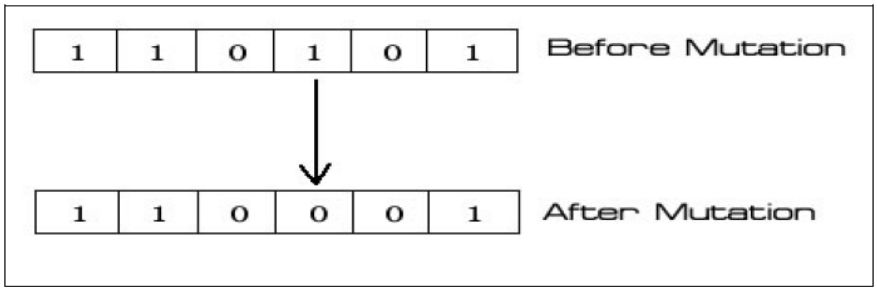

Figure 4. Mutation

Figure 5. UML Class Diagram

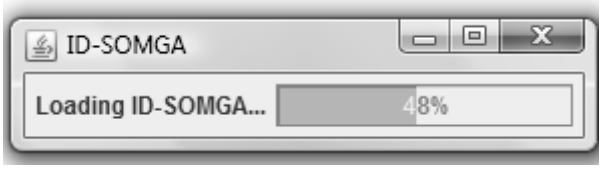

Figure 6. ID-SOMGA Starting up

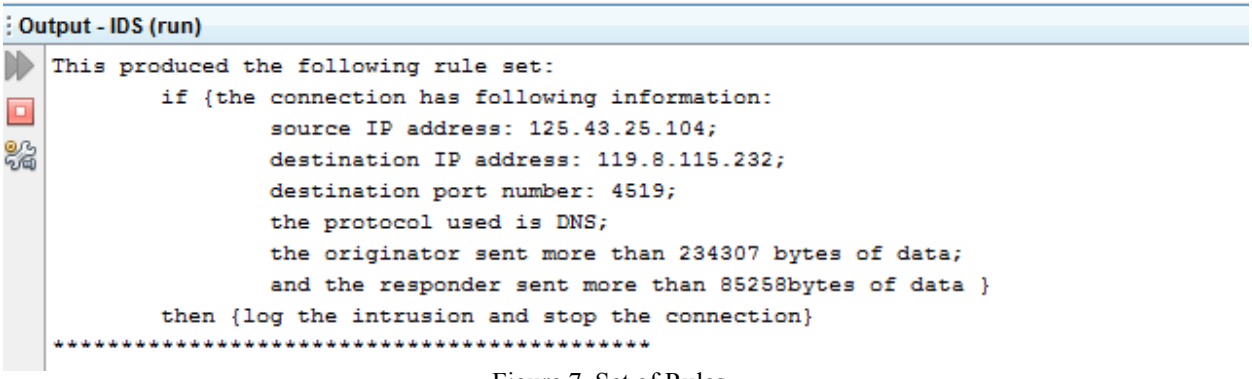

Figure 7. Set of Rules

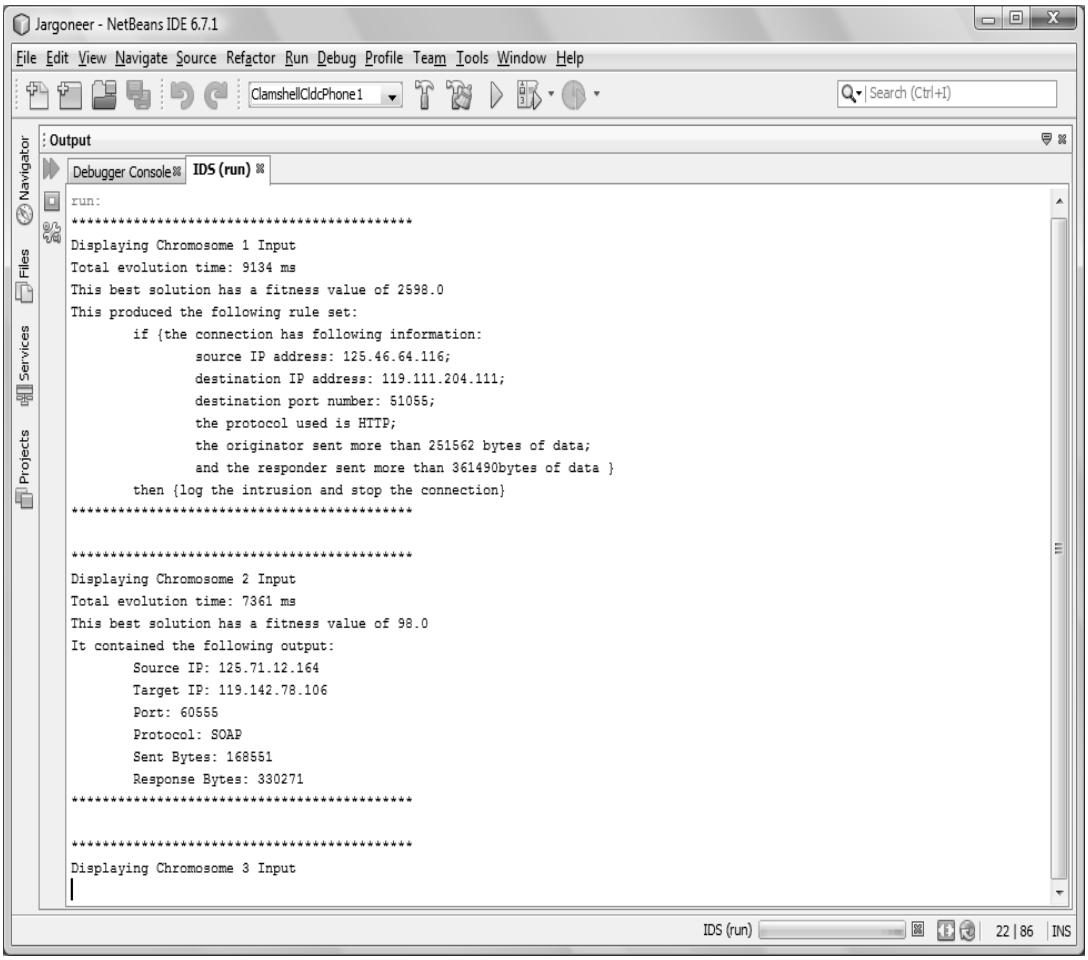

Figure 8. IDS checks through network log records

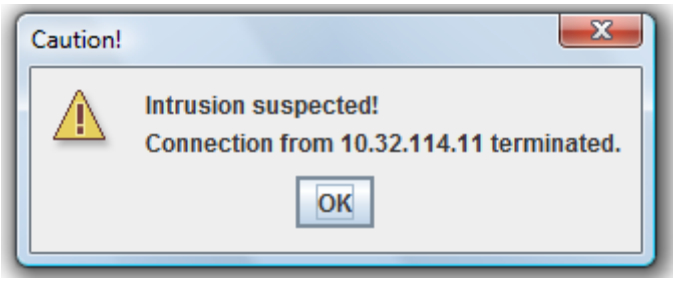

Figure 9. A report of an Intrusion

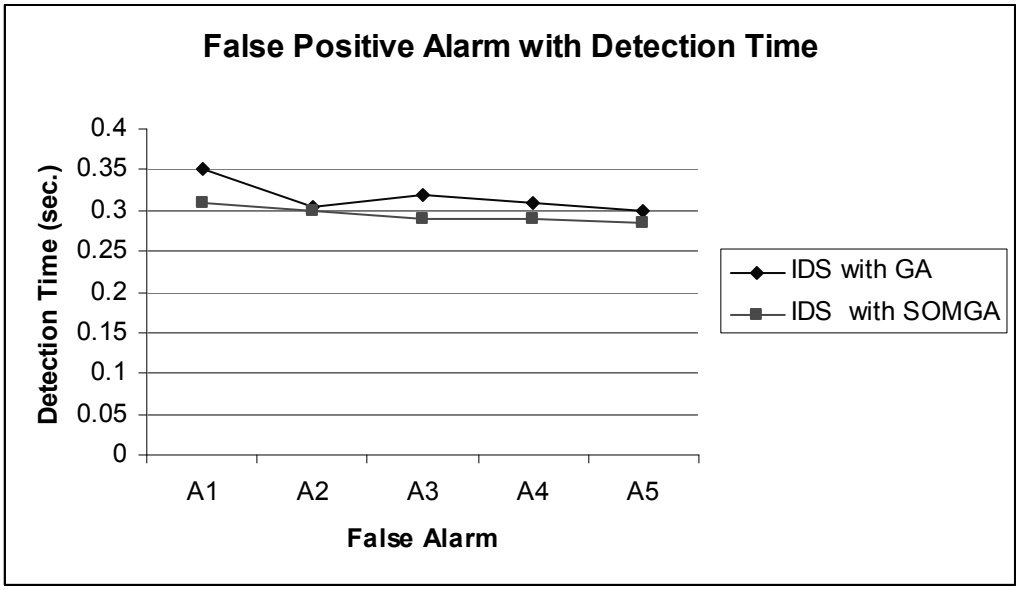

Figure 10. False Positives Alarms