An approach to implement a network intrusion detection system using genetic algorithms

An Approach to Implement a Network Intrusion Detection System using Genetic Algorithms

M. M. PILLAI, J. H.P. ELOFF AND H. S. VENTER
University of Pretoria

As the transmission of data over the internet increases, the need to protect connected systems also increases. Intrusion Detection Systems (IDSs) are the latest technology used for this purpose. Although the field of IDSs is still developing, the systems that do exist are still not complete, in the sense that they are not able to detect all types of intrusions. Some attacks which are detected by various tools available today cannot be detected by other products, depending on the types and methods that they are built on. Using a Genetic Algorithm (GA) is one of the methods that IDSs use to detect intrusions. They incorporate the concept of Darwin's theory and natural selection to detect intrusions. Not much research has been conducted in this area besides the Genetic Algorithm as an Alternative Tool for Security Audit Trails Analysis (GASSATA) tool; there are very few IDSs that are completely developed from using GAs. The focus of this paper is to introduce the application of GA, in order to improve the effectiveness of IDSs.

Categories and Subject Descriptors: D.4.6 [**Operating Systems**]: Security and Protection - *Invasive software (e.g., viruses, worms, Trojan horses)*; I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent agents*; *Multiagent systems*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection - *Invasive software (e.g., viruses, worms, Trojan horses)*
General Terms: Algorithms, Security
Additional Key Words and Phrases: Intrusion detection system (IDS), Evolution, Genetic algorithm (GA).

1. INTRODUCTION

The complexity, as well as the importance, of distributed computer systems and information resources is rapidly growing. Due to this, computers and computer networks are often exposed to computer crime. Many modern systems lack properly implemented security services; they contain a variety of vulnerabilities and, therefore, can be compromised easily. As network attacks have increased in number over the past few years, the efficiency of security systems such as firewalls have declined.

It is very important that the security mechanisms of a system are designed to prevent unauthorized access to system resources and data. However, building a complete secure system is impossible and the least that can be done is to detect the intrusion attempts so that action can be taken to repair the damage later.

Organizations are increasingly implementing various systems that monitor IT security breaches. Intrusion detection systems (IDSs) have gained a considerable amount of interest within this area [Dorosz et al., 2003]. The main task of an IDS is to detect an intrusion and, if necessary or possible, to undertake some measures eliminating the intrusions. Because most computer systems are vulnerable to attack, intrusion detection (ID) is a rapidly developing field.

Intrusion Detection Systems (IDSs) detect intrusions using specific methodologies that are specific to each of them. A method describes how an IDS analyzes data to detect possible intrusions, based on the analysis approaches. The analysis approaches are anomaly detection and misuse detection. There are many methods that are used. Examples of them include statistical approaches [Sundaram, 1996], protocol anomaly detection [Verwoerd et al, 2001], neural networks [Philippe, 2004], file checking [Verwoerd et al, 2001], expert systems [Sundaram, 1996], rule-based measures [Gordeev, 2004], and genetic algorithms (GAs) [Verwoerd et al, 2001].

The Genetic Algorithm (GA) field is one of the up-coming fields in computer security and especially in IDSs. GAs are based on genetics, especially on Darwin's theory: survival of the fittest [Fogel, 1998]. This states that the weaker members of a species tend to die away, leaving the stronger and fitter. The surviving members create offspring and ensure the continuing survival of the species. This concept together with the concept of natural selection, is used in information technology to enhance the performance of computers.

The goal of ID is to monitor network activities automatically, and to detect malicious attacks [Bace, 2000]. Once an intrusion is detected, administrators are required to take correct action. The methods proposed to detect attacks extract features from network data, and detect intrusions. The disadvantage of these methods is that they have to be manually revised for every type of intrusion invented. The GA approach uses trained network data for this task. When unknown

Author Addresses:
M.M. Pillai, Information and Computer Security (ICSA) Research Group, Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa; s23183358@tuks.up.ac.za.
J.H.P. Eloff, Information and Computer Security (ICSA) Research Group, Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa; eloff@cs.up.ac.za.
H. S. Venter, Information and Computer Security (ICSA) Research Group, Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa; hventer@cs.up.ac.za.

types of attacks need to be detected, these techniques have the advantage to automatically retrain detection models on input data. Generally a large volume of network data is encountered for this purpose. Existing IDSs also rely heavily on human analysts to differentiate normal and abnormal network connections. The use of GAs in IDSs helps to avoid this tremendous task of human analysts by generating rules for ID. GAs also provides multiple solutions to a problem, for example here it will be providing multiple rules for an anomalous connection. Hence it will be able to detect many intrusions. Besides this, there are very few tools in the market that use the concept of a GA, and research is undergoing in this field.

The focus of this paper is to provide an approach to implement a network intrusion detection system (NIDS) using GAs, such that it automatically generate rules, and generates new rules for a specific connection.

Since real network data is used for this research, the proposed design provided aims to be efficient and properly structured to be realistically implemented, up to date.

The rest of the paper is organized as follows. Section 2 provides a background for IDSs and GAs. Section 3 discusses related work in this area. Section 4 describes the methodology used in this research to implement a NIDS, followed by conclusion in Section 5.

2. BACKGROUND

Intruders tend to find new ways to compromise systems each day. As more intrusions occur, the weaknesses of existing technologies like firewalls are exposed. Since it is impossible to build a complete secure system, IDSs are used to detect the intrusions that occur. This is why IDSs are gaining acceptance in every organization. To understand what an IDS is, first one should know what intrusion and intruders are.

Sundaram [Sundaram, 1996] defines intrusion as the unauthorized attempt to access information, manipulate information, or render a system unreliable or unusable. To detect intrusions and to prevent them, one has to be aware of how an intruder can cause intrusions. The primary ways an intruder can get into the system is through primary intrusion, system intrusion and remote intrusion [Graham, 2000]. ID is the process of monitoring the events occurring in a computer system or network, and analyzing them for intrusions [Bace et al, 2004]. The prevention of intrusions should be done through effective IDSs. An IDS is a software or hardware product that automates this monitoring and analysis process [Gordeev, 2004].

The types of IDSs can be described in terms of three fundamental functional components. They are the information source, analysis and response [Bace et al, 2004]. The information source of the system mainly depends on where the IDSs are being placed, hence it is also known as the monitoring locations of the IDS. The information sources are mainly of three types: network-based IDSs, host-based IDSs and application-based IDSs [Bace, 2000]. Since the focus of this research is on network-based IDS, the other two types will not be considered here. Network-based IDSs detect attacks by capturing and analyzing network packets. They search for attack signatures within the packets. Signatures might be based on actual packet contents, and are checked by comparing bits to known patterns of attack. If the bits are matched to known patterns of attack, then an intrusion is triggered. Once the information sources have monitored network traffic, the next step is to analyze the events to detect the intrusion. As stated before, the two main techniques or approaches used to analyze events to detect attacks are misuse detection and anomaly detection [Sundaram, 1996; Bace et al, 2004]. Response is the set of actions that the system takes once it detects intrusions. Some of the responses involve reporting results and findings to a pre-specified location, while others are more actively automated responses. Commercial IDSs support both active and passive responses, and sometimes a combination of the two [Bace et al, 2004].

IDSs can be viewed as the second layer of protection against unauthorized access to networked information systems because despite the best access control systems, intruders are still able to enter computer networks. IDSs expand the security provided by the access control systems by providing system administrators with a warning of the intrusion. They also provide the system administrators with necessary information about the intrusions. This assists the system administrators in controlling the intrusions that has occurred, in order to avoid them in the future or to minimize the damage that may occur due to an intrusion. Although IDSs can be designed to verify the proper operation of access control systems by looking for the attacks that get past the access control systems, IDSs are more useful when they can detect intrusions that use methods that are different from those used by the access control systems. For this purpose, they must use more general and more powerful methods than simple database look-ups of known attack scenarios.

One such method that will be used in this research is GA. A GA is essentially a type of search algorithm which is used to solve a wide variety of problems. The goal of a GA is to create optimal solutions to problems. Potential solutions are encoded as a sequence of bits, characters or numbers. This unit of encoding is called a gene, and the encoding sequence is known as a chromosome. The GA begins with a set of these chromosomes and an evaluation function that measures the fitness of each chromosome. It uses reproduction, such as crossover and mutation to create new solutions, which are then evaluated. GAs are defined as a computational concept inspired by the mechanics of natural evolution, including survival of the fittest, reproduction, and mutation [Engelbrecht, 2002]. In the standard GA an initial population of individuals is generated at random or heuristically. In every generation the individuals in the current population are evaluated according to some predefined quality criterion, referred to as the fitness [Whitley, 1993]. Fitness is determined by the fitness function. The fitness function takes a string and assigns a relative fitness value to the string. Based on their fitness, strings are selected as parents using selection operators [Engelbrecht, 2002]. To form a new generation or child, the strings are put together, and they reproduce through operators, crossover and mutation [Whitley, 1993]. The GA comes to a halt when the determined fitness value is met, or when variation of individuals from one generation to the next reaches a prespecified level of stability. The iteration loop of a basic genetic algorithm is given in figure 1.

In figure 1, firstly, an initial population of strings is created. The process then iteratively selects individuals according to the fitness. Based on the fitness value of each string, strings which comply with the fitness value are combined to make a new generation that may be able to solve the problem. Initially the process selects individuals referred to as ‘parents’. The fit individuals of the new generation then become parents. If a solution is found, then the loop terminates, otherwise the loop starts from the individuals selected from the new generation, and continues until the termination criteria are met.

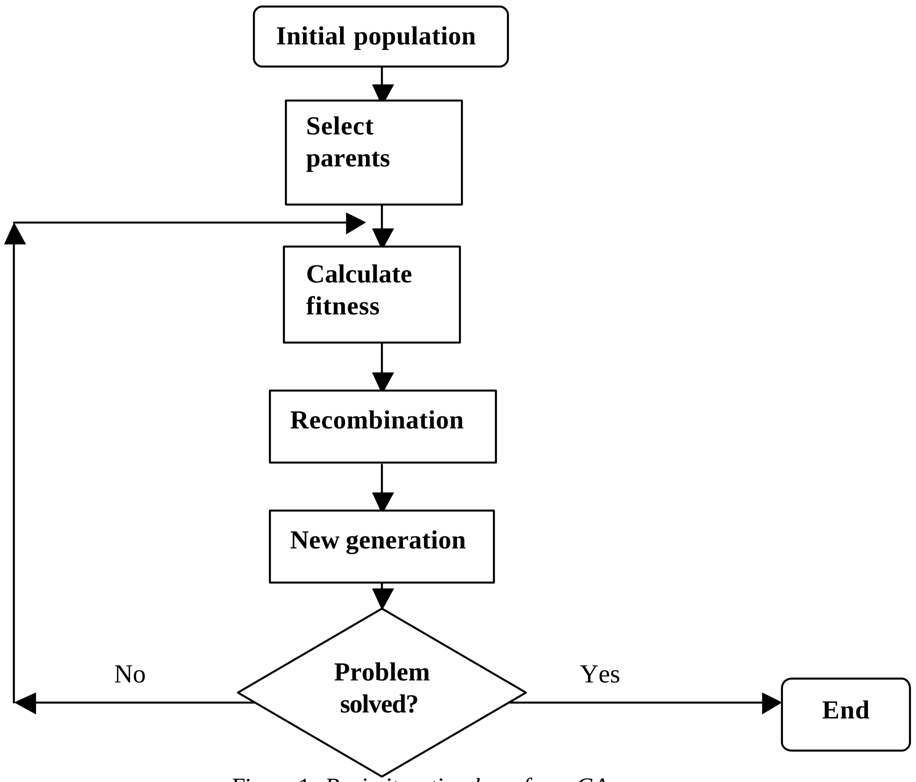

Figure 1*: Basic iteration loop for a GA*

The next section explains some related work conducted in the field of IDSs using GAs.

3. RELATED WORK

The concept of incorporating GAs into NIDS, however, is only developing. Some work already conducted in this area is provided in this section.

One IDS tool that uses GAs to detect intrusions, and is available to the public is the Genetic Algorithm as an Alternative Tool for security Audit Trails Analysis (GASSATA). GASSATA finds among all possible sets of known attacks, the subset of attacks that are the most likely to have occurred in a set of audit data. Since there can be many possible attack types, and finding the optimal subset is very expensive to compute. GAs are used to search efficiently. The population to be evolved consists of vectors with a bit set for each attack that is comprised in the data set. Crossover and mutation converge the population to the most probable attacks [Verwoerd et al, 2001].

A second tool that is implemented and undergoing more advanced enhancements is the Network Exploitation Detection Analyst Assistant (NEDAA). The Applied Research Laboratories of the University of Texas at Austin has developed the NEDAA [Sinclair et al, 2004], which uses different machine learning techniques, such as a finite state machine, a decision tree, and a GA, to generate artificial intelligence (AI) 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 or not. 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 behaviors are potential intrusions.

Another IDS that does not generate rules but uses the concept of GAs has been proposed by Crosbie and Spafford [Crosbie et al, 1995]. The COAST laboratory in Purdue University implemented an IDS using autonomous agents and applied AI techniques to evolve GAs. Agents are modeled as chromosomes and an internal evaluator is used inside every agent [Crosbie et al, 1995].

A host-based IDS that uses GAs has also been proposed by Balajinath and Raghavan [Balajinath et al, 2000]. Balajinath and Raghavan proposed an algorithm for anomaly IDS called Genetic Algorithm Based Intrusion Detector (GBID). The GBID is based on learning the individual user behavior. Alphabets mapped from user commands are used as an input stream to the GBID model. The current user behavior can be predicted by GA based on the past observed user behavior [Balajinath et al, 2000].

The approach described in this paper differs from the approaches that are mentioned above. First of all, it is not a host-based system that monitors a single host as proposed by Crosbie et al and Balajinath et al. Unlike the NEDAA it uses only the GA to detect intrusions, making it simpler, rather than using different machine learning techniques. It differs from the GASSATA by refining rules from the rule set rather than using log files to detect intrusions. The major feature of this research is that it is much simpler than the other methods discussed above. The proposed design classifies intrusions based on network connections. The current IDSs use rules, regardless of whether they are good or bad rules for detecting intrusions. The current rules in the rule set may also produce a lot of false alarms. The GA in the proposed design evaluates the rules and discards bad rules, while generating more rules to reduce the false alarm rate and to increase the intrusion detection rate.

The next section describes the proposed design and methodology to implement a NIDS using a GA for this research.

4. NETWORK INTRUSION DETECTION SYSTEM (NIDS) USING A GA

As explained above, the concept of a GA is used in this paper to build a new NIDS. The proposed design for this system is given in figure 2 that follows. A more detailed discussion of these components follows.

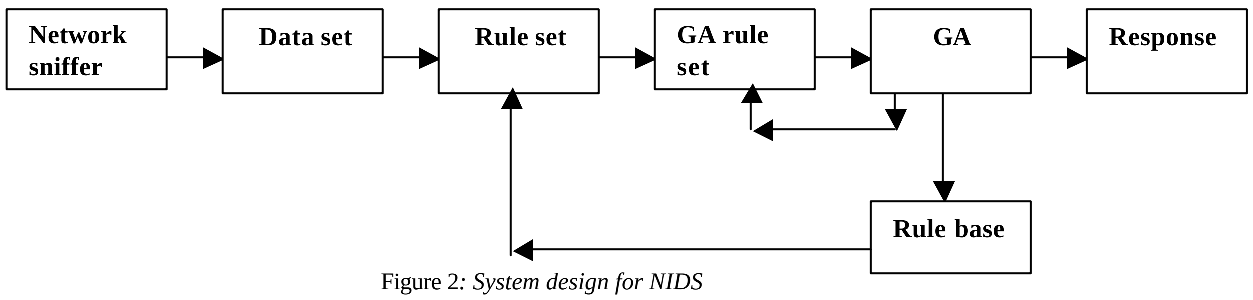

Figure 2*: System design for NIDS*

In figure 2, the network traffic used for the GA is a pre-classified data set that differentiates normal network connections from anomalous ones. This pre-classified data set is manually created by analyzing the data captured by the network sniffer. The network sniffer is a program used to record network traffic without doing something harmful to the network traffic. WinDump, the Windows version of TcpDump, is used for this purpose. TcpDump and WinDump are freeware products which can be downloaded from the WinDump website [Windump, 2004].

Based on the traffic from WinDump, the data set is created. The data set includes the necessary information to generate rules. This information includes the source IP address, the destination IP address, the source port, the destination port, the protocol used, and finally a field indicating whether the specific connection indicates an intrusion or not. The data set will include both normal and anomalous network connections. A connection refers to an entry in the dataset. If the connection is an intrusion, then it will be indicated by the value *true*, and if it is not an intrusion, it will be indicated by the value *false*. These network connections in the dataset are, as stated before, manually created. This is the initial phase of developing the NIDS using the GA. The dataset created initially is a simple table, as shown in table 1 below, used to generate rules for the rule set prior to training the GA. Once the GA is trained with the rules, more network connections can be added to the dataset. This means that the dataset will have to be updated by administrators to add a new connection or to discard a connection. Once the initial dataset is created, the next action is to create the rule set.

Source IP	Destination IP	Source Port	Destination Port	Protocol	Intrusion
150.165.13.1	130.179.16.43	0025	0080	IP	TRUE
150.165.14.5	130.179.16.66	0077	0021	IP	FALSE
150.165.13.4	130.179.17.11	0034	0080	IP	TRUE

Table 1*: Example of a dataset*

By analyzing the dataset, rules will be generated in the rule set. These rules will be in the form of an 'if then' format as follows.

if {condition} then {act}

The condition in the format above refers to the attributes in the rule set that forms a network connection in the dataset, as shown in table 1, such as source and destination IP addresses, source and destination port numbers, protocol used, and a field indicating the possibility of an intrusion. Note that the condition will result in a *'true'* or *'false'*. The act field in the 'if-then' format above will refer to an action once the condition is true, such as reporting an alert to the system administrator. For example, a rule in the rule set can be defined as follows:

if {the connection has the following information: source IP address 150.165.13.1; destination IP address: 130.179.16.43; source port number: 25; destination port number: 80; protocol used: IP} then {detect whether the connection is an intrusion or not}

This rule will detect an intrusion because the source IP address 150.165.13.1 is recognized by the IDS as, for example, a blacklisted address. Hence any service requested from this address is rejected.

Since the GA has to use such rules to detect intrusions, such rules in the rule set will be codified to the GA format in the GA rule set. Each rule will be represented in the form of a chromosome in the GA. This is carried out by extracting certain characteristics of the attributes in the rule set into a GA format. Network intrusion detection systems (NIDSs) monitor specific parts of a network to avoid the large volume of traffic, and to function properly. Hence the NIDS using the GA will monitor a local area network (LAN). In a LAN, the IP addresses will contain the same network ID (i.e. 150.165) but the workstation ID (i.e. 13.1) in a class-B network will be different. Hence only the workstation ID will be used to represent the IP address in the GA format. Since the subnet mask extends the workstation ID to a value of 255, a * is included to represent a wildcard for the possible range of values. For the specific rule, which results in the rule set as mentioned in the example above, the corresponding GA format represented in the GA rule set is as follows.

131*16*43**25**80it*

The first six characters represent the workstation ID of the source IP address (150.165.13.1), e.g. *13**1. The next six characters represent the workstation ID of the destination IP address (130.179.16.43), e.g. *16*43 followed by the source port number (0025), e.g. **25 and destination port number (0080), e.g. **80. The IP protocol is represented by 'i'. If the protocol identified was a Simple Network Management Protocol (SNMP), for example, it would have been represented by 's'. If the connection is an intrusion then it is represented by 't', otherwise represented by 'f'. In the example above, the rule indicates that the connection is an intrusion.

As stated before the GA uses the rules in the GA rule set which are encoded as chromosomes to detect anomalous connections. The first part of the GA will act as a search algorithm. In the initial stage, only the search algorithm will be executed. This is to help the rules acquire values which are to be later used in the fitness function, when the complete GA is executed. Initially the search algorithm will match the rules with any anomalous connections that occur on the network to detect an intrusion. Each rule will carry values for the intrusions that they have detected, and a value for a false alarm that the rule produces. The initial values for the rule will be initialized to zero. The rules will acquire these values when the search algorithm is executed. Once the rules have acquired the values, then the complete GA, which includes the fitness function and mutation, is executed.

The second part of the GA is the fitness function. The fitness function 'F' determines whether a rule is 'good' i.e. it detects intrusions, or whether the rule is 'bad', i.e. it does not detect intrusions. 'F' is calculated for each rule. It will depend on the following equation [Chittur, 2001]. In the initial stage, this equation will be used to determine the fitness function, but future work will test and improve the equation to make the GA more effective in selecting fit individuals.

$$F = a / A - b / B$$

In the fitness function, 'a' contains the value that the specific rule carries for the number of correctly detected intrusions. 'b' contains the value that the specific rule carries for the number of false alarms. 'A' is calculated by adding the value of the correctly detected intrusions from all the rules. 'B' is the total number of normal connections in the dataset. A normal connection is not an intrusion, and is indicated by the value *false*. When an intrusion occurs, it is notified by the response mechanism. The response mechanism is a popup window indicating the rule, and a message notifying that an intrusion has occurred. When an intrusion does not occur, but the response mechanism confirms it as an intrusion, then it is considered as a false alarm. When a rule pops up indicating an intrusion, but the connection actually has not taken place, then it is a false alarm. The network sniffer provides the information of connections on the network. Hence, when an intrusion is indicated, the network sniffer will be executed to determine whether it is an intrusion or a false alarm.

The fitness function will assign a fitness value for each rule. The fitness value will be a predetermined value. However the precise fitness value, which will qualify the rule to be selected, is currently unknown, since it can only be determined from how the rules perform in detecting intrusions. The rules that have acquired the required fitness value will be selected to form a new generation to produce new rules. For example, consider that the predetermined fitness value for 'F' is 1. As mentioned before 'F' will be calculated for each rule by the GA. When a rule acquires the predetermined value of 1, then the rule is selected to undergo reproduction to produce a new generation of rules. After a successive number of generations, if the rule still has not gained the value 1, and is therefore unfit, the rule will be discarded.

Normally reproduction in a GA is carried out by the crossover of chromosomes. In this case, as stated before, the chromosomes are rules. Crossover here indicates a crossover between the rules. One approach would be to cross the condition and act-parts of two rules, which will either produce the same rule, or change a normal connection into an anomalous one or vice versa. For example, consider the crossing over of the two rules as shown below. Here the condition-parts of the parent rules are *13**1*16*43**25**80i and *14**5*16*66**77**21i respectively. The 't' and 'f' are the act-parts in the rules respectively.

*Parent rule1: *13**1*16*43**25**80i t*

*Parent rule2:*14**5*16*66**77**21i f*

After crossover

*Child1: *13**1*16*43**25**80i* $\underline{f}$

*Child2: *14**5*16*66**77**21i* $\underline{t}$

The new rules formed after crossing over are not 'good' rules. The underlined part of the new rules formed, shows where crossover has taken place. The first parent rule originally indicated an intrusion. The child rule produced, indicates that it is not an intrusion, making the original connection of the specific rule normal, although it is an intrusion. The same applies to the second parent rule. If the act part of the second parent rule was 't,' then the child rules produced would have been exactly the same as the parent rules.

Another approach would be to cross parts of the condition in the 'if-then' rule. For example, perform a crossover from the port numbers up to the protocol field. This will produce an entirely new connection, which is not indicated in the dataset, and hence will not detect any intrusions. Consider the example that follows.

*Parent rule1: *13**1*16*43* $\underline{\text{**25**80}}$ *it*

*Parent rule2:*14**5*16*66* $\underline{\text{**77**21}}$ *if*

After crossover

*Child1: *13**1*16*43* $\underline{\text{**77**21}}$ *if*

*Child2: *14**5*16*66* $\underline{\text{**25**80}}$ *it*

The underlined part of the new rules formed, show where crossover has taken place. The new rules formed are a new connection from the respective source IP address to destination IP address using a new source port number and a new destination port number. This connection is not mentioned in the dataset, and it might be a connection which is not blacklisted

Up to now crossover does not function properly, hence reproduction will be performed through mutation. It is indicated by Sinclair, Pierce and Matzner [Sinclair et al, 2004] that in cases where a crossover or mutation operator cannot be imposed on the space of solutions, a classic genetic algorithm (with encoding to chromosomes) is rather used. However, mutation, but not crossover, is used here to produce new rules, since crossover will not function properly for this purpose as proved above.

Mutation is performed by altering the pattern of a chromosome to produce a new chromosome. Here, *Inorder mutation* [Engelbrecht, 2002] is performed, where two bit positions are randomly selected and only bits between these positions are mutated. Hence, parts of the rules will be altered to produce a new one. An example of this operation is shown below.

The original rule: *13**1*16*43**2580it

After mutation: *13**1*16*43**0080it

The underlined part in the rules indicates where mutation occurs. The original rule is mutated by altering the port numbers to zero, which indicates that the new rule should detect any intrusions in the network connection from the workstation ID of the source IP address 13.01 to the workstation ID of the destination IP address 16.43, using the IP protocol *regardless of the source ports that it uses*. This increases the effectiveness of detecting more legitimate intrusions for the specific connection.

The new rules that are formed will be supplied to the rule set in the 'if-then' format. The GA will supply the codified form of the rule to the GA rule set. The advantage of producing these new rules is that, when an intrusion is detected, it will match all the rules available for that specific connection. When a source IP address is blacklisted for creating connection establishments to the destination IP address, through the source port number using the specific protocol, initially a single rule, will be triggered to indicate an intrusion. Since the source IP address, is blacklisted for using the specific protocol, any connection establishments from that source IP address to the destination IP address is therefore an intrusion. The NIDS using the GA explained in this paper therefore produces new rules for each connection, through mutation, in such a manner that all connection establishments from the source IP address to the destination IP address, using the specific protocol, is considered as an intrusion regardless of source ports it uses. This increases the effectiveness of detecting intrusions for that specific connection.

The GA thus continues to detect intrusions and produce new rules, storing the 'good' rules in the rule base and discarding the 'bad' ones. Once in a while, the dataset will have to be updated for new connections, and hence the rule set will also be updated, making the human factor important, since a human's input is still required to the dataset.

The rule set initially will contain the new generation of rules that the GA produces. Once the GA has been trained, i.e. when it has supplied a 'good' set of rules for the corresponding dataset to the rule base, the rules in the rule base will be supplied to the rule set. The rule set can then start using these rules to detect intrusions. In this way intrusions can be detected effectively.

The NIDS explained in this paper uses the dataset to classify the anomalous connections. This dataset serves as the basis for the NIDS to detect intrusions. There are, in fact, many different ways in which the data set can be constructed [Bace, 2000]. For example, the dataset could be represented in the form of a table, a record set, a flat-file structure, etc. In future work, a generalized form in which the dataset should be represented from various representations of datasets, still needs to be researched in order for a GA as discussed in this paper, to be compatible with any kind of IDS.

5. CONCLUSION

This paper described an approach to implement a NIDS using GA. The network sniffer traffic was analyzed to create a dataset. The next step was to automatically generate the rule set. The skeletal structure of the GA has also been developed, but will need the input from the rule set to function properly.

No results of the detection rate are available yet, but will be produced in the near future. However, the NIDS using the GA is a promising approach to improve the effectiveness of detecting intrusions by producing new rules so that multiple rules are used to detect a specific connection.

Future work will attempt to use other techniques such as the niching technique [Mahfoud, 1995] to produce multiple rules. Future work will also attempt to improve the current method for the detection of false alarms. The NIDS will also try to use evolutionary programming [Engelbrecht, 2002], which is similar to GAs, to determine which is more efficient in detecting intrusions.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Research Foundation under Grant number 2054024. Any opinion, findings and conclusions or recommendations expressed in this material are those of the authors and therefore the NRF does not accept any liability thereto.

REFERENCES

BACE, R AND MELL, P., 2004, NIST Special Publication on Intrusion Detection Systems, http://www.nist.gov, Retrieved -23-02-2004.
BACE, R. 2002. Intrusion Detection, Macmillan Technical Publishing
BALAJINATH, B., RAGHAVAN, S.V. 2000, Intrusion Detection through learning behaviour model;http://www.elsevier.com, 13-11-2000.
CHITTUR, A. 2001, Model Generation for an Intrusion Detection System Using Genetic Algorithms, http:// www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf, 27-11-2001.
CROSBIE,M., AND SPAFFORD, G. 1995. Applying Genetic Programming to Intrusion Detection;http://www.citeseer.nj.nec.com, 9-05-1995.
KAZIENKO, P., AND DOROSZ, P. 2003, Intrusion Detection Systems (IDS) Part I - (network intrusions; attack symptoms; IDS tasks; and IDS architecture), http://www.windowsecurity.com, Apr 07, 2003.
ENGELBRECHT, A.P. 2002, Computational Intelligence; John Wiley & Sons Ltd; 2002.
FOGEL, D.B. 1998, Evolutionary Computation; Second Edition, IEEE Press, 1998.
GORDEEV, M. 2004. Intrusion Detection Techniques and Approaches, http://www.ict.tuwein.ac.a, Retrieved 12-03-2004.
GRAHAM, R. 2000. FAQ: Network Intrusion Detection Systems, http://www.robertgraham.com, 21-04-2000.
MAHFOUD, S.W. 1995. Niching Methods for Genetic Algorithms, Home Page: http:// www.citeseer.nj.nec.com, 00-03-1995.
JEAN-PHILIPPE. 2004. Application of Neural Networks to Intrusion Detection, http://www.sans.org, Retrieved -23-02-2004.
SINCLAIR, C., PIERCE, L., AND MATZNER; S. 2004. An application of machine learning to Network Intrusion Detection, http:// www.citeseer.nj.nec.com, Retrieved- 9-04-2004.
SUNDARAM, A. 2001. An Introduction to Intrusion Detection, http://www.acm.org, Retrieved -23-1-2001.
VERWOERD T., AND HUNT, R. 2001. Intrusion Detection Techniques and Approaches, http://www. Elsevier.com, 11-12-2001.
WHITELY, D. 1993. A Genetic Algorithm Tutorial", Technical Report CS-93-103, http://www.citeseer nj.nec.com, 10-11-1993.
WINDUMO. 2004: http://www.windump.polito.it, Retrieved- 3-04-2004.