A Mutual Construction for IDS Using GA

S. Selvakani Kandeeban
Professor and Head, MCA Department, Francis Xavier Engineering College, Tirunelveli, Tamilnadu, India
sselvakani@hotmail.com

R. S. Rajesh
Reader, Department of CSE, Manonmanium Sundaranar University,, Tirunelveli, Tamilnadu, India
rs_rajesh@yahoo.co.in

Abstract

A variety of intrusion prevention techniques, such as user authentication using passwords, avoidance of programming errors and information protection, has been used to protect computer systems. However information prevention alone is not sufficient to protect our systems as those systems become even more complex with the rapid growth and expansion of Internet technology and local network systems. Moreover, programming errors, firewall configuration errors and ambiguous or undefined security policies add to the system's complexity. An Intrusion Detection system (IDS) is therefore needed as another layer to protect computer systems. The IDS is one of the most important techniques of information dynamic security technology. It is defined as a process of monitoring the events occurring in a computer system or network and analyzing them to differentiate between normal activities of the system and behaviors that can be classified as suspicious or intrusive.

Current Intrusion Detection Systems have several known shortcomings, such as low accuracy (registering high False Positives and False Negatives); low real-time performance (processing a large amount of traffic in real time); limited scalability (storing a large number of user profiles and attack signatures); an inability to detect new attacks (recognizing new attacks when they are launched for the first time); and weak system reactive capabilities (efficiency of response). This makes the area of IDS an attractive research field. In recent years, researchers have investigated techniques such as artificial intelligence, autonomous agents and distributed systems for detecting intrusion in network environments.

In this work we have realized an Intrusion Detection System based on Genetic algorithm (GA) approach. For evolving and testing new rules for intrusion detection system the KDD99Cup training and testing dataset were used.

Keywords *Anomaly detection, Genetic Algorithm, KDD 99 Cup Set, False Positives.*

1. Introduction

Intrusion detection is one of the most challenging problems in network security Detection of attacks on a particular network is not an easy task. In this paper, we present a collaborative architecture for Intrusion detection system using Genetic algorithm.

IDS is a system that can be placed in a network to stop and detect network intrusions and anomalies. After the emergence of computer networks and especially the internet, the communication of data and the exchange of information have become quite easy and fast. The easy access and exchange of information over the network brings the question of how to

ensure the security of the information that is stored or exchanged against attacks by the intruders. There are two kinds of intruders; external intruders and internal intruders. External intruder are unauthorized users that enter the system and make disastrous changes in the system or use the network resources. On the other hand, an internal intruder is the one who is an authorized user of the system and wants to gain access to the network resources which he/she is not allowed to use.

ID techniques can be partitioned into two complementary approaches: misuse detection and anomaly detection. Misuse detection systems, such as [9], [11], model the known attacks and scan the system data for occurrences of these patterns. Anomaly detection systems, such as [2], [7], flag intrusions by observing significant deviations from typical or expected behavior of the system or users. However, these two kinds of IDS have their weaknesses. The former cannot detect novel attacks because their signatures are not yet available for pattern matching. The latter, in general for most such existing systems, have a high false alarm rate because it is difficult to generate practical normal behavior profiles for protected systems. The model is verified on the dataset taken from KDD99Cup which is a standard dataset used for intrusion detection.

Earlier studies have utilized a rule-based approach for intrusion detection, but had a difficulty in identifying new attack that had no previously describe patterns. Lately the emphasis is being shifted to learning by examples and data mining paradigms. The use of Artificial Intelligence (AI) for ID is widely considered as the only way to build efficient and adaptive intrusion detection systems. Recently many researches have been conducted on the application of neural networks for ID. Neural networks have been extensively used to identify both misuse and anomalous patterns. Several researchers proposed data mining techniques to identify key patterns that help in detecting intrusions. Distributed agent technology is being proposed by a few researchers to overcome the inherent limitations of the client-server paradigm and to detect intrusions in real time. However, these techniques have some significant shortages. The number of neurons affects the network's performance. Increasing the number of output nodes will increase the resolution of the map, but the computation time will dramatically increase. Moreover, feature selection, structure design and weight training are three key tasks for the application of neural network, and a number of researches on these three problems have been done. Traditional leaning algorithms of neural network take single detection task mostly they are difficult to solve the problem of Omission alarm, false alarm, and in distinguish ability of unknown attacks

2. Related Works

M A. Chittur [1] extended their idea by using GA for anomaly detection. Random numbers were generated using GA. Random numbers were generated using GA. A threshold value was established and any certainty value exceeding this threshold value was classified as a malicious attack. The experimental result showed that GA successfully generated an empirical behavior model from training data. The biggest limitation of this model was the difficulty of establishing the threshold value which might lead to detect novel or unknown attacks.

J. Gomez et al [3] proposed a linear representation scheme for evolving fuzzy rules using the concept of complete binary tree structure. GA is used to generate genetic operators for producing useful and minimal structure modification to the fuzzy expression tree represented by chromosomes. The biggest drawback of the proposed approach was that the training was time consuming. Liao and Vemuri used the K-nearest Vector Machine for profiling computer

programs. The KNN classifier was employed with an interesting analogy between classifying text documents and detecting intrusion using the sequences of system calls.

Wang et al. [10] used the evolutionary algorithm for discovering neural networks for intrusion detection. The connections of the network and its weights were encoded with binary bits and evolved simultaneously. Their detection system was evaluated with www log data and showed an accuracy rate of 95%. However, in their experiment, they used their own data set rather than a public bench mark dataset.

Hofmann et al [5] proposed the evolutionary learning of radial basis function networks for intrusion detection. They targeted a network based IDS. Their evolutionary algorithm performed two tasks simultaneously selecting the optimal feature set and learning the RBFN. The binary bits system was used to encode the 137 possible features of the network packet headers and three components of the RBFN, including the type of basis function, the number of hidden neurons, and the number of training epochs. In the experiments with the network audit data set, the RBFN optimized with the evolutionary algorithm outperformed the normal MLP and the normal RBFN.

Gonzalez et al. [4] proposed an intrusion detection technique based on evolutionary generated fuzzy rules. The conditional part of the fuzzy detection rules was encoded with binary bits and fitness was evaluated using two factors: the accuracy and the coverage of the rule. The performance was compared to the methods of different genetic algorithms and without the fuzziness of rules using two network audit datasets their own wireless dataset and the knowledge discovery and data mining cup 99 data set.

3. Genetic Algorithm

Genetic algorithm is an evolutionary optimization technique based on the principles of natural selection and genetics. The working of genetic algorithm was explained by John Holland [6]. It has been successfully employed to solve wide range of complex optimization problems where search space is too large. GA evolves a population of initial individuals to a population of high quality individuals, where each individual represents a solution of the problem to be solved. Each individual is called a chromosome, and is composed of a predetermined number of genes. The quality of fitness of each chromosome is measured by a fitness function that is based on the problem being solved. The algorithm starts with an initial population of randomly generated individuals. Then the population is evolved for a number of generations while gradually improving the qualities of the individuals in the sense of increasing the fitness value as the measure of quality. During each generation, three basic genetic operators are applied to each individual with certain probabilities, i.e. selection (rank selection, roulette wheel selection and tournament based selection), crossover and mutation.

4. 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.

4.1 Fitness Function

The fitness function is one of the most important parameters in genetic algorithm. To evolve input features and network structure simultaneously, the fitness, which is based on the detection accuracy rate, includes a penalty factor for the number of operative input nodes and

a penalty factor for the number of operative hidden nodes. The fitness function of the individual a is defined by

$$\text{Fitness}(a) = \text{drate}(a) \times \Psi(a) \times \varphi(a) \quad (1)$$

Where drate is the detection accuracy rate, Ψ is the penalty factor for the number of operative input nodes; φ is the penalty factor for the number of operative hidden nodes. The detection accuracy rate of the individual a is defined by

$$\text{drate}(a) = \text{correct}(a) / \text{sum} \quad (2)$$

Where correct is the number of accurate detections, sum is the total number of detections which include both normal instances and abnormal instances.

Once a mismatch happens, the penalty value is computed using the absolute difference. The penalty is defined by

$$\text{Penalty}(a) = 1 - \text{fitness}(a) \quad (3)$$

4.2 Crossover and Mutation Operator

The generated subnet of an input node is the set of its all output connections. The generated subnet of a hidden node is the set of itself, its all input and output connections. The generated subnet of an output node is the set of itself and its all input connections. The connection information includes whether or not the connection exists and the connection weight value. The node information includes bias and activation function. Obviously, the generated subnet of inoperative nodes is empty.

The employed mutation operator, which can prevent the premature convergence in evolution, includes five operations: node deletion, connection deletion, connection addition, node addition and weight adjustment. The number of output nodes is invariable, so the deletion and addition of node are limited for input and hidden nodes.

The node deletion means making a node inoperative and deleting its generated subnet. The node addition is that an inoperative node is turned operative and assigned connection weights with the other operative nodes. The mutation operator has the same influence as the crossover, thus the necessary step following mutation operation is the same as the crossover.

There are also other parameters that need to be considered. These parameters should be adjusted according to the application environment of the system and the organization's security policy.

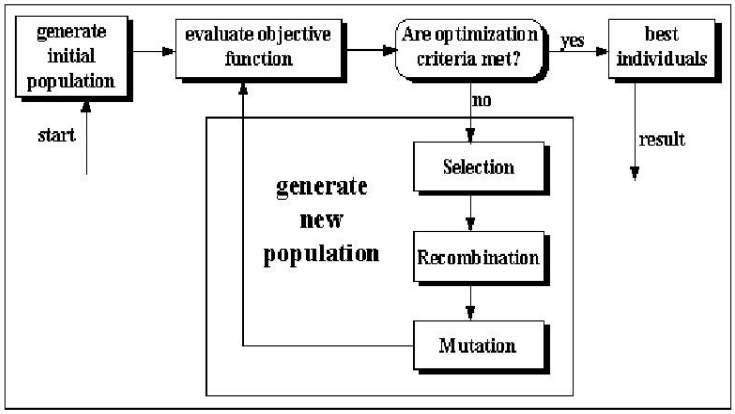

Figure 1: Genetic Algorithm Flow

5. Kddcup99 Dataset Description

The KDDCUP99 data set [8] is having 41 features and one target class feature. The dataset contains six million records. The raw training data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records. A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol. Each connection is labeled as either normal, or as an attack, with exactly one specific attack type. Each connection record consists of about 100 bytes. Attacks fall into four main categories:

• DOS: denial-of-service, e.g. SYN flood;

• R2L: unauthorized access from a remote machine, e.g. guessing password;

• U2R: unauthorized access to local super user (root) privileges, e.g., various ``buffer overflow'' attacks;

• Probing: surveillance and other probing, e.g., port scanning.

Table 1: Attacks Present in DARPA 1999 Data Set

Attack Class	Attack Type
Probe	portsweep, ipsweep, lsdomain, ntinfoscan, mscan, illegal-sniffer, queso, satan
DoS	apache2, smurf, neptune, dosnuke, land, pod, back, teardrop, tcpreset, syslogd, crashiis, arppoison, mailbomb, selfping, processtable, udpstorm, warezclient
R2L	dict, netcat, sendmail,imap, ncftp, xlock, xsnoop, sshtrojan, framespoof, ppmacro, guest, netbus, snmpget, ftpwrite, httptunnel, phf, named
U2R	sechole, xterm, eject, ntfsdos, nukepw, secret, perl, ps, yaga, fdformat, ppmacro ffbconfig, casesen, loadmodule, sqlattack

6. Experiments and Results

The proposed approach contains two stages. In the first one, the training stage, a set of rules for detecting intruders is generated using network audit data offline. In the second stage, the best rules, i.e. the rules with the highest fitness values, are used for intrusion detection in the real-time environment. As some of the network characteristics have higher possibilities to be involved in network intrusions, we have deployed Correlation approach to identify these characteristics. The correlation algorithm was implemented in MATLAB and deployed over the training dataset in order to define the features that participate most frequently in a machinery of an attack. According to the obtained results, we have selected nine features out of forty one used to describe each connection of KDD99Cup dataset as mentioned in the Figure 2. The objective was to select the smallest possible number of the features while maintaining high detection rate of intrusions. In such a way detection could be performed as a real-time one. Every feature represents one gene of the chromosome. As one byte is being used to represent every feature, i.e. every gene, a chromosome that represents each individual is composed of nine bytes.

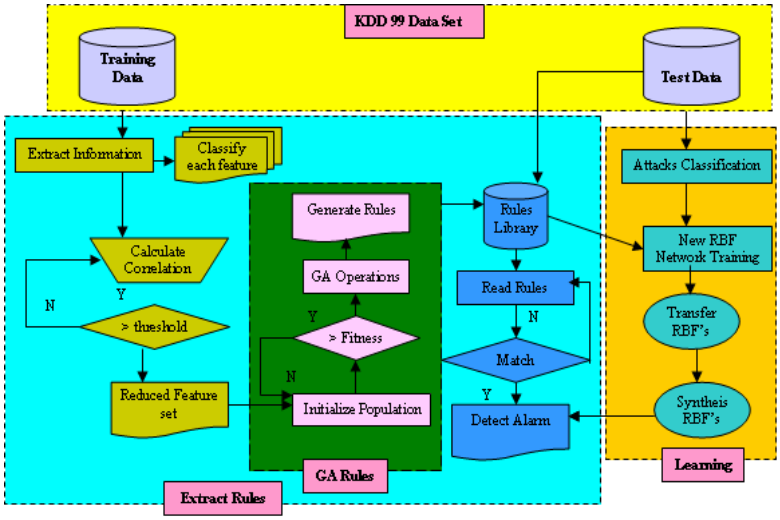

Figure 2: System Flow

We have also exploited the possibility of GA to detect the exact type of an attack. Detecting the type of each intrusion is not very important for intrusion detection, but it is important for forensics in order to recover from an attack. In this case a rule can be presented as:

if (duration = "1" **and** src_bytes ="0" **and** dst_host_srv_serror_rate = "50") **then** (attack_name = "portsweep");
if protocol_type = tcp **then**
if service = http **then** back
Else if service= private **then** neptune
else if service =finger | telnet **then**
if count=1 **then** land
if count >=1**&&** <=302 **then** Neptune
if protocol_type = tcp **then**
if service=http **then**
if logged_in=1 **then**
if dst_host_count =255 **then**
if dst_host_same=0 **then** back
Default: normal

The algorithm for generating new rules is performed as follows. The first step is initialization of an initial population when each gene is given a random value. Then the parameters of genetic algorithm (crossover and mutation rate, size of population, end of evolution of rules) are specified and the network audit data is being loaded. After that the initial population is being evolved for a number of generations. In every generation the quality of every rule, i.e. fitness value, is being calculated according to the fitness function, then a number of rules with the highest fitness values are being selected and at the end the genetic operators (crossover and mutation) are performed with a certain probability. The outputs of the algorithm are rules for intrusion detection.

For the purpose of this work, two subsets of KDD99Cup datasets for training and testing are derived. Each connection has the corresponding marking that states whether it is a normal connection or a certain type of an attack. The subset used for training contained 137 attacks and 839 normal connections. The testing subset contained 234 attacks and 743 normal

connections. The most of the connections selected are normal, which is generally the case in real-world networks. The subsets contained three types of network attacks: portsweep, smurf and Neptune. Portsweep is a kind of an attack that sweeps through many ports to determine which services are supported on a single host. Smurf is a denial-of-service attack that sends a stream of ICMP ‘‘ECHO’’ to the broadcast address of many subnets, resulting in a large continuous stream of ‘‘ECHO’’ replies that floods the victim. Neptune (or SYN-flood)is a denial-of-service attack where attacking system continues sending IP-spoofed packets requesting new connections faster than the victim system can close pending connections, i.e. they will expire. In some cases, the system may exhaust memory, crash or be rendered otherwise inoperative.

In the terms of the features used to describe a particular type of an attack, portsweep attacks experiment greater time range, i.e. duration feature usually has a value higher that ‘0’, considering that it takes some period of time to perform its attack, while the other two in most of the cases have ‘0’ value. They also exhibit wider range of dst_host_srv_serror_rate than the other two attacks. Neptune attack exhibits great value of the dst_host_srv_serror_rate, usually around 100%, considering that the attacker sends a stream of SYN packets to a port on the target machine and dst_host_srv_serror_rate provides the information of the connections that have SYN errors, while duration and src_bytes features have ‘0’ value in most of the cases. Smurf attack, on the other side, has a high value of the src_bytes feature, few times bigger than the number of src_bytes in the normal connection, while the values of dst_host_srv_serror_rate and duration remain ‘0’.

The training dataset contained 74 Neptune, 24 smurf and 39 portsweep attacks. The testing dataset contained 87 Neptune, 107 smurf and 40 portsweep attacks.

7. Conclusions

In this work we have deployed genetic algorithm approach to intrusion detection. Software implementation of the proposed approach is presented. Genetic algorithm was used to obtain classification rules for intrusion detection while correlation technique was used to identify the most important features of network connections.

As in real word types of intrusions are changing rapidly and becoming increasingly complex, an intrusion detection system should be adaptive in order to be able to cope withthe evolution of the threat-space. As our system can upload and update new data y evolve new rules for detecting new intrusions, it is adaptive and representation of the rules and effective fitness functions that can be applied, is easy to implement and maintain. Moreover, our system is flexible enough to be used in different application environments, provided the proper attack taxonomy and the proper training dataset.

We have demonstrated that GA-approach can be used either to classify network connections as either normal or intrusive or further classify attacks by their type. Classification of attacks is not important in intrusion detection, considering that the goal of intrusion detection is detecting attacks in real time so they could be retained before creating any damage. On the contrary, classification of the attacks is very important in network forensics because a good recovery after the damage made by an attack can be done by knowing the exact type of an attack and its mechanism.

High attack detection rate and low false-positive rate demonstrate advantages of applying this technique to intrusion detection without using any complementary technique typically used with other soft-computing techniques. Our system is using only three features of the network connections maintaining high detection rates, so it can perform intrusion detection process fast and could be applied to high speed networks.

References

[1] Chittur.A, "Model Generation for an Intrusion Detection System using Genetic Algorithms", High school Honors Thesis, http://www1.cs.columbia.edu/ids/publications/gaids-thesis01.pdf, accessed in 2006.

[2] E. Michailidis, S.K. Katsikas, and E. Georgopoulos, "Intrusion Detection Using Evolutionary Neural Networks", Proceedings of Panhellenic Conference on Informatics, pp. 8-12, Aug. 2008.

[3] Gomez.J, Dasgupta.D, Nasaroui.D and Gonzalez.F, "Complete expression Trees for evolving Fuzzy classifiers system with Genetic Algorithms and Applications to Network Intrusion Detection". In proceedings of NAFIPS-FLINT Joint Conference, New Orleans, LA, PP.469-474, June 2002**.**

[4] Gonzalez.F, Gomez.J, Kaniganti.M, and Dasgupta.D, "An evolutionary approach to generate fuzzy anomaly signatures", In proceedings 4th Annual IEEE Information Assurance workshop, West point, NY, PP.251-259, June 2003.

[5] Hofmann.A and Sick.B, "Evolutionary optimization of radial basis function networks for Intrusion Detection", In proceedings of Int.Joint.Conf. Neural Networks, Portland, OR, Vol 1, PP.415-420, July 2003.

[6] 1. H. Holland, "Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence", The MIT Press, 1992.

[7] Joao B.D. Cabrera, B. Bavichandran, R.K. Mehra, "Statistical Traffic Modeling for Network Intrusion Detection", Proceedings of 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication systems, Aug. 2000, pp. 466-473

[8] KDD cup 1999 data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[9] S.C. Lee, D.V. Heinbuch, "Training a Neural-Network Based Intrusion Detector to Recognize Novel Attacks", IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 31, no. 4, pp. 294-299, Jul. 2001.

[10] Wang.L, Yu.G, Wang.G and Wang D, "Method of evolutionary neural network based intrusion detection", In proceeding of Int.Conf.Info-tech and Info-net, Beijing, China, Vol 5, PP.13-18, Oct 2001.

[11] W. Lee, S. J. Stolfo, K. Mok, "A Data Mining Framework for Building Intrusion Detection Models", Proceedings of 1999 IEEE Symposium of Security and Privacy, pp. 120-132.

Authors

Selvakani Kandeeban received the MCA degree from Manonmanium Sundaranar University and M.Phil degree from Madurai Kamaraj University. Presently she is working as an Professor & Head, MCA Dept in Francis Xavier Engineering College, Tirunelveli. Previously she was with Jaya College of Engineering and Technology as an Assistant Professor, MCA Department. She has presented 4 papers in National Conference and 1 paper in international conference. She has published 2 paper in National journal and 8 papers in International Journal. She is currently pursuing her PhD degree in Network Security.

Dr. R. S Rajesh received his B.E and M.E degrees in Electronics and Communication Engineering from Madurai Kamaraj University, Madurai, India in the year 1988 and 1989 respectively, and completed his Ph.D in Computer Science and Engineering from Manonmaniam Sundaranar University in the year 2004. In September 1992 he joined in Manonmaniam Sundaranar University where he is currently working as Reader in the Computer Science and Engineering Department.