http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 12 (11): 2167-2173, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.2167.2173

Improving Intrusion Detection Using Genetic Algorithm

[1]V. Moraveji Hashemi, [1]Z. Muda and [2]W.Yassin
[1]Faculty of Computer Science and Information Technology, University Putra Malaysia,
43400 UPM Serdang, Selangor Darul Ehsan, Malaysia
[2]Faculty of Information and Communication Technology, University Technical Malaysia Melaka,
76100 Durian Tunggal, Melaka, Malaysia

Abstract: Intrusion Detection System (IDS) is one of the key security components in today's networking environment. A great deal of attention has been recently paid to anomaly detection to accomplish intrusion detection. However, a major problem with this approach is maximizing detection rate and accuracy, as well as minimizing false alarm i.e., inability to correctly discover particular types of attacks. To overcome this problem, a genetic algorithm approach is proposed. Genetic Algorithm (GA) is most frequently employed as a robust technology based on machine learning for designing IDS. GAs are search algorithms which are based on the principles of natural selection and genetics. GA functions on a number of possible solutions using the principle of survival of the fittest with the aim to generate better approximations to solve a particular problem GA is facing. The validity of this approach is verified using Knowledge Discovery and Data Mining Cup 1999 (KDD Cup '99) dataset. The experimental results demonstrate that the proposed approach outperforms the existing techniques, with the detection rate of attack and false alarm rates of 95.7265 and 4.2735, respectively.

Key words: Genetic algorithm, intrusion detection system, false positive, false negative, detection rate, rule set

INTRODUCTION

IDS is a security component that can detect any intrusion against the computer system or network such as unauthorized access, misuses and any type of intrusions by hackers. This system first collects all traffic or behavior of the target network or computer, then learns and creates patterns and stores them in database as an instance and then checks and monitors all incoming traffic or behavior with training patterns and then generates alarm to announce dangers to the administrator. Figure 1 shows an organization of generalized IDS (Wu and Banzhaf, 2010).

Several machine-learning technologies have been used for designing IDS: Neural networks, Linear Genetic Programming (LGP), Support Vector Machines (SVM) such as (Lei and Zhou, 2012), Decision tree such as (Ali *et al*., 2009), Bayesian networks such as (Muda *et al*., 2011), Multivariate Adaptive Regression Splines (MARS), Fuzzy Inference Systems (FISs).

Generally intrusion detection systems are divided into two main categories: host-based IDS (HIDS) and network-based IDS (NIDS). Network-base intrusion detection system checks and controls all incoming and outgoing traffic at one network component. This happens by placing one capture tool (sensor) like one sniffer on

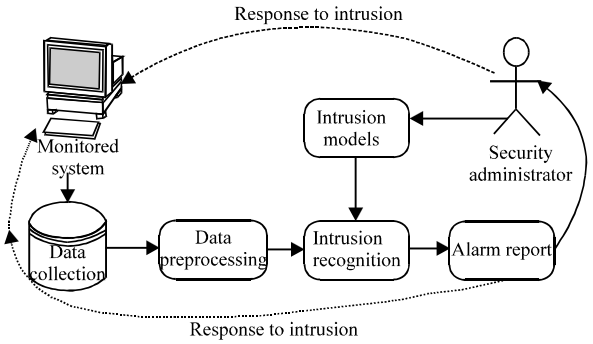

Fig. 1: Organization of a generalized intrusion detection system

the chock point of the segment. This sensor captures all network traffic in this segment and determines whether these traffic packets are attacks or normal. Unlike NIDS that check all network traffic, HIDS observes anything that occurs on each host individually. The HIDS have the ability to detect risk actions and normally operate by having access to log files or monitoring the host usage in real time (De Lima *et al*., 2008).

There are two general important terminologies in all intrusion detection system: False Positive, which is a

Corresponding Author: V. Moraveji Hashemi, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 UPM Serdang, Selangor Darul Ehsan, Malaysia

condition in which IDS announces warnings of attacks for something that is not actually an attack. If too many false positives take place, it makes the administrator less confident about the alarms and hence, there is a possibility that the administrator ignores the real attack. The second is False Negative, which is condition in which IDS does not manifest an alarm for a real attack, which means a real attack is taking place while no action is being taken. This will put the system in great dangers since the real attacks are entirely being unnoticed (Norouzian and Merati, 2011).

There are two major types of intrusion detection systems (IDSs): misuse detection and anomaly based. Misuse detection systems are more commonly used and they identify intruders with known patterns. The signatures and patterns that are used to detect attacks consist of several fields of a network packet, including source address, destination address, source and destination ports or even some key words of the payload of a packet. These systems suffer from a shortcoming in the sense that only the attacks already existing in the attack database can be noticed. As a result, this model needs to be regularly updated, but it has a benefit of having very low false positive rate. Anomaly detection systems have the ability to recognize deviances from normal behavior as well as potential unknown or new attacks without having any prior knowledge of them. They display a higher rate of false alarms but they are capable of detecting unidentified attacks and look for deviations much faster (Bankovic *et al*., 2007).

Genetic Algorithm (GA) field is one of the up-coming fields in computer security, especially in Intrusion Detection Systems (IDS) (Gong *et al*., 2005; Chittur, 2001; Folino *et al*., 2005).

For the purpose of processing network data in real time and performing efficient intrusion detection, the most important piece of information should be extracted that can be employed for efficient detection of network attacks. Principal Component Analysis (PCA) which is also known as Karhunen Loe've transform, is used in order to extract the most relevant features of the data. The goal of PCA is to decrease the dimension of the data by making an effort to discern a few orthogonal linear combinations of the variables that have the largest variance. Although an effective machine learning algorithm has been devised in the intrusion detection fields and related work, improving the detection rate with low false alarm is still required. As compared to other approaches, the new GA approach brings about higher detection rates and lower false alarm in identifying anomaly-based network intrusions.

RELATED WORK

Lu and Traore (2004) developed a method to develop a set of classification rules by using Genetic Programming (GP) and past network data. In this method, using GP the practical implementation is more difficult due to the fact that the system required more data or time. Bridges and Vaughn (2000) implemented a method to identify both anomalies and network misuses by combining Genetic Algorithm's and Fuzzy data mining technologies. In this method, the salient network features were chosen and the best possible parameters of the fuzzy function were established by employing Genetic Algorithm. Xia *et al*. (2005) offered a method of identifying abnormal behaviors of network using Genetic Algorithm and information theory. The genetic algorithm complexity reduced by using mutual information. However, this methodology is only applicable to discrete features. Li (2004) succeeded in detecting network anomalous using Genetic Algorithm. Quantitative features inclusion may increase the detection rates, although there does not exist any implementation result. Crosbie and Spafford (1995) proposed a technique of identifying network anomalies using Genetic Programming (GP) and multiple agent technology. The training process continues for a long time once the agents are not well adjusted. The communication that is required to be conducted among small autonomous agents remains unresolved. Selvakani and Rajesh (2007) applied Genetic Algorithm to engender rules for training the IDS. In this method, the rules are created only for Smurf (DoS) and Warzemaster (R2L) attacks. The performance of this method of detection rate is low. The study demonstrated that using KDD Cup '99 dataset, the Intrusion Detection models proposed for R2L, U2R, Probe attacks takes low detection rates. For each category i.e., DoS, R2L, U2R and Probe, the present study deals with two types of attacks. The author used KDDCup dataset to detect the attacks.

GENETIC ALGORITHM OVERVIEW

Genetic Algorithms (GA) are search algorithms which function based on the principles of natural selection and genetics. GA develops a population of initial individuals to a population of high quality individuals, where each individual signifies a solution of the problem to be solved. Each individual is called chromosome and comprises predetermined number of genes (Polhlheim, 2006). The quality of each rule is measured by a fitness function as the quantitative representation of each rule's adaptation to a particular environment. The flow of GA is shown in

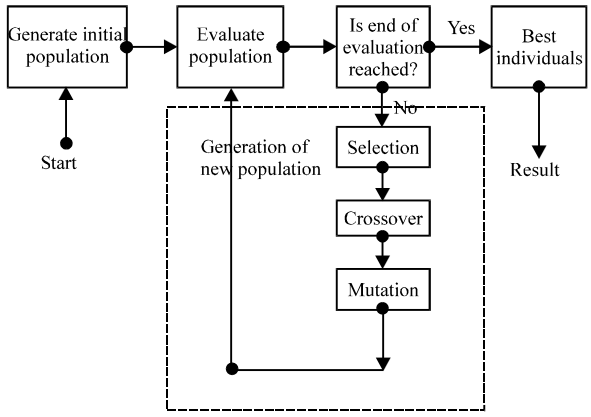

Fig. 2: Genetic algorithm flow

Fig. 2. The procedure starts from an initial population of randomly generated individuals. The population is then 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 sequentially applied to each individual with certain probabilities, i.e., selection, crossover and mutation.

PROPOSED APPROACH

The proposed approach includes two stages. In the first one, that is the training stage, a set of rules for detecting intruders is generated using network audit data offline. In the second stage, the best rules, the rules with the highest fitness values are used for intrusion detection in the real-time environment. KDD Cup '99 dataset is used to verify the validity of this approach.

As some of the network characteristics have higher possibilities to be involved in network intrusions, PCA approach is used to identify these characteristics. The PCA algorithm was implemented in MATLAB and deployed over the training dataset in order to define the features that are most frequently involved in a machinery of an attack. According to the results, three features out of forty-one are selected to describe each connection of KDD Cup '99 dataset. The purpose was to select the smallest possible number of the features while maintaining high detection rate of intrusions. As such, detection could be performed as a real-time one. Table 1 represents the features selected as well as their explications. 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 three bytes.

Table 1: Selected network features

Name of feature	Explication	No. of genes
Duration	Length (number of seconds) of the connection	1
Src_bytes	Number of data bytes from source to destination	1
dst_host_srv_serror_rate	Percentage of connections that have "SYN" errors	1

Every rule for intrusion detection is simple if-then clause. Features from Table 1 are connected using an and function thus forming the conditional part of a rule. The result of every rule is the confirmation of an intrusion. For example, one rule could be:

- If (duration = "1" and src_bytes= "0" and dst_host_srv_serror_rate= "0") then intrusion

To determine a fitness value of each rule, the following fitness function is deployed:

$$\text{Fitness} = a/A - b/B \text{ Fitness function} \quad (1)$$

where, a is the number of correctly detected attacks, A is the whole number of attacks in the training dataset, b is the number of normal connections that are falsely identified as attacks, i.e., false-positives and B is the total number of normal connections in the training dataset. Scale of fitness values is [-1, 1], where -1 and 1 represent the lowest and highest values, respectively. High detection rate and low rate of false-positives contribute to a high fitness value. On the other hand, low detection rate and high rate of false-positives bring about a low fitness value.

The algorithm for generating new rules is performed as follows. The first step is initialization of an initial population during which 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 identified and the network audit data is being loaded. Next, the initial population is being evolved for a number of generations. In every generation, the quality of every rule, i.e. fitness value, is calculated according to the fitness function, then a number of rules with the highest fitness values are selected and the genetic operators (crossover and mutation) are finally performed with a certain probability. The output of the algorithm generates rules for intrusion detection (Bankovic *et al*., 2007).

IMPLEMENTATION, EXPERIMENTS AND RESULT

Implementation: The system proposed here is implemented in MATLAB

(TRAIN)

1. Initial population
   1.1 Define 200 rules
   1.2 Calculate fitness value for each rule by Fitness = a/A – b/B
2. For i=1 to 1000
   2.1 Do Selection touruament2 for selecting parent1 and parent2
   2.2 Do Crossover single point for creating child1 and child2
   2.3 Do Mutation with rate of 0.01
   2.4 Reinsertion (parent1, parent2, child1, child2) to the population

(TEST)

3. Input Test data
4. Calculate Detection rate, False Alarm and Accuracy

In training phase:

First the population is initialized, 200 rules are defined as the population. Each rule represents a single individual from the population and determines whether each individual is belonging to an attack or normal connection. Then fitness value is calculated for each rule based on defined formula (a/A – b/B).

Secondly, a loop generated for 1000 times as for to 2.1, 2.2, 2.3 and 2.4 processes simultaneously. In 2.1, process for Selection Tournament2 began with select two parents as parent1 and parent2. The pseudo code for touruament selection is as follows:

```
fnnc tournament_selection(pop, k):
 best = null
 for i=1 to k
 ind = pop[random(1, N)]
 if (best == null) or fitness(ind) > fitness(best)
 best = ind
 return best
```

(k refer to the number of tournaments where k = 2)

2.2 Started crossover single point process for creating child 1 and child2
2.3 Proceed with mutation process at rate of 0.01(it means that among 100 individuals one of them is mutated)
2.4 Finally, reinsert parent1, parent2, child1 and child2 to the population due to population size will not be changed.

In testing phase:

Firstly, the test data is entered. Secondly the Detection rate, False Alarm and Accuracy are calculated using the following formulas:

Detection rate = TN/Total Attacks *100
Percentage of false detect attack = FP/Total Attacks *100
Accuracy = (TP+TN)/ (TP+TN+FP+FN)

Dataset description: One of the biggest challenges in network-based intrusion detection is the extensive amount of data gathered from the network. Therefore, before feeding the data to a machine learning algorithm, raw network traffic should be summarized into higher-level events such as connection records. Each higher-level event is described with a set of features. Selecting good features is an essential task and necessitates extensive domain knowledge. KDD Cup '99 intrusion detection datasets, which are based on DARPA 98 dataset, provide labelled data for researchers working in the field of intrusion detection and are the only labelled datasets publicly available. Numerous researchers have used the datasets in KDD Cup '99 intrusion detection competition to investigate the utilization of machine learning for intrusion detection and reported detection rates up to 91% with false positive rates less than 1%. To substantiate the performance of machine learning based detectors, KDD Cup '99 training dataset was used (Kayacik *et al*., 2010).

Training and testing the rules for intrusion detection: For the purpose of this work, two subsets of KDD Cup '99 dataset for training and testing are derived. Each connection has the corresponding marking that states whether it is a normal connection or attack. The subset used for training contained 100000 connections includes normal connections and attacks. The testing subset contained 137 attacks and 839 normal connections.

GA parameters deployed for training the rules: The system was trained using the fitness function defined in formula 1 with the following parameters of genetic algorithm: 1000 generations, 200 initial rules, "one-point" crossover, "tournament 2" selection and the mutation rate of 0.01. When the process of training was finished, 200 rules were used for the classification of the intrusions and the normal connections in the testing dataset.

Evaluation measurement: An efficient IDS necessitates high degree of accuracy and detection rate and low false alarm rate. In general, the performance of an IDS is assessed in terms of accuracy, detection rate and false alarm rate as in the following formula:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

$$\text{No. of all errors} = FN+FP$$

$$\text{Percentage of true detect attack (detection rate)} = \frac{TN}{\text{Total attacks}} \times 100$$

$$\text{Percentage of true detect normal} = \frac{TP}{\text{Total normal}} \times 100$$

$$\text{Percentage of undetected attack} = \frac{FN}{\text{Total attacks}} \times 100$$

$$\text{Percentage of false detect attack} = \frac{FP}{\text{Total attacks}} \times 100$$

where, FN is false negative, FP is false positive, TN is true negative, TP is true positive.

RESULTS AND DISCUSSION

This part of study evaluates number of data for training phase based on minimal overhead of the system. In this phase four different numbers of KDD Cup '99 dataset are selected with 1000, 4000, 80000 and 100000

Table 2: Comparison between our approach and Bankovic's approach

Measurement	Proposed approach	Bankovic's approach
Detection rate	95.6204	92.74
False alarm	4.3796	7.26

Table 3: Last result based on training and testing dataset

Experiment	Detection rate	False alarm	Accuracy
1	95.6204	4.3796	0.9939
2	95.7265	4.2735	0.9898

connections, respectively. The purpose of any IDS is to increase a detection rate and decrease an overhead and a process time. The system must select specific number of attacks to get best result of detection rate and minimal overhead.

A number of research studies have been carried out to compare the performance of the proposed approach with the previous approach (Bankovic *et al*., 2007) using training and testing datasets. The approach was trained using the fitness function defined in Fitness Function 1 with the following parameters of GA: 1000 generations, 200 initial rules, "single point" crossover, "Tournament 2" selection with mutation rate of 0.01 while (Bankovic *et al*., 2007) approach used 1000 generations, 500 initial rules, "one point" crossover, "Roulette wheel" selection with mutation rate of 0.01.

Table 2 shows the comparison between the proposed approach and (Bankovic *et al*., 2007) approach against testing dataset1. In this table detection rate and false alarm are calculated with the proposed algorithm and compared with (Bankovic *et al*., 2007) approach. The proposed approach detects better in term of detection rate and false alarm. The approach performed with 95.62% as a detection rate and 4.37% as false alarm which is more accurate compared than Bankovic's approach that obtained 92.74, 7.26% as detection rate and false alarm.

Table 3 shows the last result based on training and testing dataset. Experiments are performed with two different test datasets. In the first experiment, testing dataset1 contains 137 attacks and 840 normal connections that totally include 977 connections and in the second experiment the testing dataset2 contains 234 attacks and 744 normal connections that totally include 978 connections. In both experiments, the proposed approach manages to get high detection rate, accuracy with above 95.5% and reasonable false alarm at below 4.5%.

Table 4 shows Detection rates (%) in different experiments of the system trained w ith Fitness Function 1. In this table different experiments are performed to calculate detection rate with different training and testing dataset. In the first experiment, Training dataset consists of 1000 connections and testing dataset1 contains 977 connections while testing dataset2 contains 978 connections. In the second experiment

Table 4: Detection rates (%) in different experiments of the system trained with Fitness Function 1

Type of connection	Training dataset	Testing dataset1	Testing dataset2	Detection rate using test dataset1	Detection rate using test dataset2
Normal and attack	1000	977	978	95.6204	95.7265
	4000	977	978	95.6204	95.7265
	80000	977	978	95.6204	95.7265
	100000	977	978	95.6204	95.7265

Table 5: False alarm and number of errors with testing dataset1

Type of connection	Training dataset	Testing dataset1	False alarm	No. of errors
Normal and attack	1000	977	4.3796	6
	4000	977	4.3796	6
	80000	977	4.3796	6
	100000	977	4.3796	6

Table 6: False alarm and number of errors with testing dataset2

Type of connection	Training dataset	Testing dataset2	False alarm	No. of errors
Normal and attack	1000	978	4.2735	10
	4000	978	4.2735	10
	80000	978	4.2735	10
	100000	978	4.2735	10

Training dataset consists of 4000 connections and testing dataset1 contains 977 connections while testing dataset2 contains 978 connections. In the third experiment Training dataset consists of 80000 connections and testing dataset1 contains 977 connections while testing dataset2 contains 978 connections. Finally, in the last experiment Training dataset consists of 100000 connections and testing dataset1 contains 977 connections while testing dataset2 contains 978 connections. The proposed approach manages to maintain the detection rate and still capable in identifying normal and attack connection although number of connection added in every experiment for training dataset.

Table 5 shows percentage of undetected attack and Number of Errors with Testing Dataset1. In this table different experiments are performed to calculate false alarm and number of errors with different training datasets and testing dataset1. It can be noticed that proposed approach achieve the low false alarm and maintain the number of errors on testing dataset eventhough the normal and attack connection in training dataset increased at each experiments.

Table 6 shows percentage of undetected attack and Number of Errors with Testing Dataset2. In this table different experiments are performed to calculate false alarm and number of errors with different training datasets and testing dataset2. Once again, the proposed approach maintain the number of errors and achieve the resonable false alarm on testing dataset eventhough the normal and attack connection in training dataset increased at each experiments.

CONCLUSION

In this study a genetic algorithm approach is deployed to intrusion detection. Software implementation of the proposed approach is presented. Genetic algorithm was employed to generate classification rules for intrusion detection while principal component analysis was used to identify the key features of network connections. GA-approach, with the appropriate and simple representation of the rules and effective fitness functions that can be applied, is easy to carry out and maintain. Moreover, the system is flexible enough to be used in different application environments, if the proper attack taxonomy and the proper training dataset exist. The classification of attacks is not important in intrusion detection, given the fact that the goal of intrusion detection is detecting attacks in real time so they could be retained before bringing about any damage.

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

REFERENCES

Ali, S.A., N. Sulaiman, A. Mustapha and N. Mustapha, 2009. K-means clustering to improve the accuracy of decision tree response classification. Inform. Technol. J., 8: 1256-1262.

Bankovic, Z., D. Stepanovic, S. Bojanic and O. Nieto-Taladriz, 2007. Improving network security using genetic algorithm approach. Comput. Electr. Eng., 33: 438-451.

Bridges, S.M. and R.B. Vaughn, 2000. Fuzzy data mining and genetic algorithms applied to intrusion detection. Proceedings of the 12th Annual Symposium on Canadian Information Technology Security, June 19-23, 2000, Ottawa, Canada, pp: 109-122.

Chittur, A., 2001. Model generation for an intrusion detection system using genetic algorithms. http://www.hacktory.cs.columbia.edu/sites/default/files/gaids-thesis01.pdf.

Crosbie, M. and G. Spafford, 1995. Applying genetic programming to intrusion detection. Proceeding of the AAAI Fall Symposium on Genetic Programming, November 10-12, 1995, Cambridge, UK., pp: 1-8.

De Lima, I.V.M., J.A. Degaspari and J.B.M. Sobral, 2008. Intrusion detection through artificial neural networks. Proceedings of the Symposium on Network Operations and Management, April 7-11, 2008, Salvador, Bahia, Brazil, pp: 867-870.

Folino, G., C. Pizzuti and G. Spezzano, 2005. GP ensemble for distributed intrusion detection systems. Proceedings of the 3rd International Conference on Advances in Pattern Recognition, August 22-25, 2005, Bath, UK., pp: 54-62.

Gong, R.H., M. Zulkernine and P. Abolmaesumi, 2005. A software implementation of a genetic algorithm based approach to network intrusion detection. Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks, May 23-25, 2005, Towson, Maryland, USA., pp: 246-253.

Kayacik, H.G., A.N. Zincir-Heywood and M.I. Heywood, 2010. Selecting features for intrusion detection: A feature relevance analysis on KDD 99 intrusion detection datasets. Faculty of Computer Science, Dalhousie University, Halifax, Nova Scotia, Canada.

Lei, X. and P. Zhou, 2012. An intrusion detection model based on GS-SVM Classifier. Inform. Technol. J., 11: 794-798.

Li, W., 2004. A genetic algorithm approach to network intrusion detection. SANS Institute, USA.

Lu, W. and I. Traore, 2004. Detecting new forms of network intrusion using genetic programming. Comput. Intell., 20: 475-494.

Muda, Z., W. Yassin, M.N. Sulaiman and N.I. Udzir, 2011. A K-means and naive bayes learning approach for better intrusion detection. Inform. Technol. J., 10: 648-655.

Norouzian, M.R. and S. Merati, 2011. Classifying attacks in a network intrusion detection system based on artificial neural networks. Proceedings of the13th International Conference on Advanced Communication Technology, February 13-16, 2011, Seoul, Korea, pp: 868-873.

Polhlheim, H., 2006. GEATbx: Genetic and evolutionary algorithm toolbox for use with Matlab. Version 3.80, December, 2006. http://www.geatbx.com/docu/index.html.

Selvakani, S. and R.S. Rajesh, 2007. Genetic algorithm for framing rules for intrusion detection. Int. J. Comput. Sci. Network Secur., 7: 285-290.

Wu, S.X. and W. Banzhaf, 2010. The use of computational intelligence in intrusion detection systems: A review. Appl. oft Comput., 10: 1-35.

Xia, T., G. Qu, S. Hariri and M. Yousif, 2005. An efficient network intrusion detection method based on information theory and genetic algorithm. Proceedings of the 24th IEEE International Conference on Performance, Computing and Communications, April 7-9, 2005, Phoenix, Arizona, USA., pp: 11-17.