Hybrid Intelligent Intrusion Detection System using Bayesian and Genetic Algorithm (BAGA): Comparitive Study

Y V Srinivasa Murthy
Research Scholar
Department of CSE
National Institute of Technology Karnataka

Kalaga Harish
4/4 B.Tech CSE
GITAM University

D K Vishal Varma
4/4 B.Tech CSE
GITAM University

Karri Sriram & B V S S Revanth
4/4 B.Tech CSE
GITAM University

ABSTRACT

Intrusion detection system (IDS) is one of the emerging techniques for information security. Security mechanisms for an information system should be designed to prevent unauthorized access of system resources and data. Many intelligent learning techniques of machine learning are applied to the large volumes of data for the construction of an efficient intrusion detection system (IDS). This paper presents an overview of intrusion detection system and a hybrid technique for intrusion detection based on Bayesian algorithm and Genetic algorithm. Bayesian algorithm classifies the dataset into various categories to identify the normal/ attacked packets where as genetic algorithm is used to generate a new data by applying mutation operation on the existing dataset to produce a new dataset. Thus this algorithm classifies $KDD99$ benchmark intrusion detection dataset to identify different types of attacks with high detection accuracy. The experimental result also shows that the accuracy of detecting attacks is fairly good.

Keywords:

Intrusion Detection System (IDS), Detection Accuracy, Bayesian classification, Genetic algorithms

1. INTRODUCTION

Information Security, intrusion detection is the act of detecting actions that attempt to compromise the confi-dentiality, integrity or availability of a resource. When Intrusion detection takes a preventive measure without direct human intervention, then it becomes an Intrusion-prevention system. Intrusion detection can be performed manually or automatically. Manual intrusion detection might take place by examining log files or other evidence for signs of intrusions, including network traffic. A system that performs automated intrusion detection is called an Intrusion Detection System (IDS). An IDS can be either host-based, if it monitors system calls or logs, or network-based if it monitors the flow of network packets. Modern IDSs are usually a combination of these two approaches. [9] Another important distinction is between systems that identify patterns of traffic or application data presumed to be malicious $(misusedetectionsystems)$, and systems that compare activities against a 'normal' baseline $(anomalydetectionsystems)$.

1.1 Intrusion Detection

Intrusion detection systems (IDS) are an essential part of the security infrastructure. They are used to detect, identify and stop intruders. The administrators can rely on them to find out successful attacks and prevent a future use of known exploits. IDS are also considered as a complementary solution to firewall technology as they recognize against the network that are missed by the firewall. Nevertheless, IDS are plagued with false positive alerts, letting security professionals to be overwhelmed by the analysis tasks.[8] Therefore, IDS employ several techniques in order to increase the detection probability of suspect threats while reducing the risk of false positives. While using pattern matching to detect intrusions, IDS users try to refine the attack signatures and limit the search to smaller traffic intervals. On the other hand, by using protocol analysis in the detection process, IDS rely on protocol specification in order to adequately analyze the traffic. So, they will be able to understand each field in the packet, and supervise the right execution of the protocols which leads to reduce the number of false positives. An Anomaly-Based Intrusion Detection System is a system for detecting computer intrusions and misuse by monitoring system activity and classifying it as either normal or anomalous. The classification is based on heuristics or rules, rather than patterns or signatures, and will detect any type of misuse that falls out of normal system operation. This is as opposed to signature based systems which can only detect attacks for which a signature has previously been created. In order to determine what attack traffic is, the system must be taught to recognize normal system activity. This can be accomplished in several ways, most often with artificial intelligence type techniques. Systems using neural networks have been used to great effect. Another method is to define what normal usage of the system comprises using a strict mathematical model, and flag any

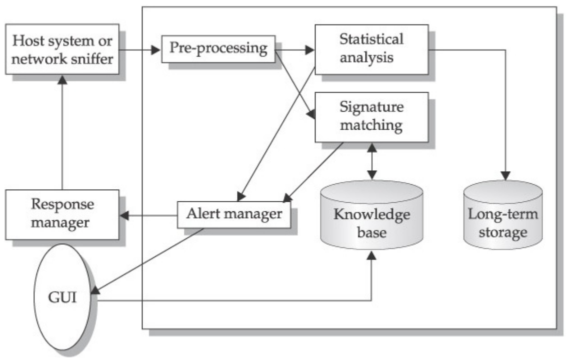

Fig. 1. Standard Intrusion Detection System

deviation from this as an attack. This is known as strict anomaly detection.[10]

1.1.1 Missue-Based Detection. Misuse detection monitors for deviations in normal protocol. This method is useful to detect attempts by a user or application attempting to gain unauthorized access to a system. The misuse detection approach attempts to recognize attacks that follow intrusion patterns that have been recognized and reported by experts. Misuse Detection catches intrusions in terms of characteristics of known attacks or system vulnerabilities; any action that conforms to the pattern of a known attack or vulnerability is considered intrusive. These systems are vulnerable to intruders who use new patterns of behavior or who mask their illegal behavior to deceive the detection system [2]. This was implemented using the following approaches:

(1) Rule-Based Languages
  (a) RUSSEL
  (b) P − BEST
(2) State Transition Analysis Tool Kit
(3) Colored Petri Automata
(4) Automatically Build Misuse Detection Models
(5) Abstraction-based Intrusion Detection

1.1.2 Anomaly - Based Detection. Anomaly detection methods were developed to coun-ter this problem. With the anomaly detection approach, one represents patterns of normal behavior, with the assumption that an intrusion can be identified based on some deviation from this normal behavior; any action that significantly deviates from the normal behavior is considered intrusive and an intrusion alarm is produced. This was implemented using the following approaches:

(1) Statistical Methods
  (a) NIDES/STAT
  (b) Haystack
(2) Machine learning & Data Mining Techniques
  (a) Time − Based Inductive Machine
  (b) Instance based learning
  (c) Neural Networks
  (d) Audit Data Analysis and Mining
(3) Computer Immunological Approach
(4) Specification − Based Methods
(5) Information − Theoritic Measures

Table 1. Attack Classification

Denial of Service	Remote-to-Local	User to root	Probe
Smurf	Guess-Password	Imap	Ipsweep
Snmp	Warezmaster	Load module	Nmap
Back	ftp-write	Butter_overflow	Portsweep
Land	Multihop	Rootkit	Saint
Neptune	Phf	***	satan

Intrusion detection systems $(IDS's)$ are usually de-ployed along with other preventive security mechanisms, such as access control and authentication, as a second line of defence that protects information systems. There are several reasons that make intrusion detection a necessary part of the entire defence system. First, many traditional systems and applications were developed without security in mind .In other cases, systems and applications were developed to work in a different environment and may become vulnerable when deployed in the current environment. (For example, a system may be perfectly secure when it is isolated but become vulnerable when it is connected to the Internet). Intrusion detection provides a way to identify and thus allow responses to, attacks against these systems. Second, due to the limitations of information security and software engineering practice, computer systems and applications may have design flaws or bugs that could be used by an intruder to attack the systems or applications. As a result, certain preven-tive mechanisms (E.g., firewalls) may not be as effective as expected.
Intrusion detection complements these protective mechanisms to improve the system security. Moreover, even if the preventive security mechanisms can protect information systems successfully, it is still desirable to know what intrusions have happened or are happening, so that we can understand the security threats and risks and thus be better prepared for future attacks. In spite of their importance, IDS?s are not replacements for preventive security mechanisms, such as access control and authentication. Indeed, IDSs themselves cannot provide sufficient protection for information systems. As an extreme example, if an attacker erases all the data in an information system, detecting the attacks cannot reduce the damage at all. Thus, IDSs should be deployed along with other preventive security mechanisms as a part of a com-prehensive defence system.
Alternatively, IDSs may be classified into host-based IDSs, distributed IDSs, and network-based IDSs accord-ing to the sources of the audit information used by each IDS (Intrusion Detection System). Host-based IDSs get audit data from host audit trails and usually aim at de-tecting attack [4][10][7][6].

1.2 Input Data Description

KDD′99 has been the most widely used data set for the evaluation of anomaly detection methods. The data set is built based on the data captured in DARPA′98 evaluation program. The data set consists of approximately 4,900,000 single connection vectors each of which contains 41 features and is labeled as either normal or an attack. The attacks fall in one of the follow-ing four categories:
The datasets contain a total number of 24 training attack types, with an additional 14 types in the test data only.

(1) *Denial of Service Attack (DoS):* It is a type of attack in which an attacker denies legitimate users access to machines or makes computing resources too busy to handle requests.
(2) *Remote to Local (R2L):* R2L occurs when a user without an account has the ability to send packets to a machine gains local access as a user of that machine.

(3) *User to Root (U2R):* In U2R the attacker first accesses the system with a normal user account by sniffing passwords or social engineering and then gains root access to the system by exploiting some vulnerability.

(4) *Probing Attack:* It is a method of gathering information about a network of computers with an intention of circumventing its security controls.

1.3 Bayesian Classifier

We consider each data instance to be an n-dimensional vector of attribute values:

$$X = (x_1, x_2, x_3, ..., x_n) \tag{1}$$

In a Bayesian classifier which assigns each data instance to one of m classes $C_1, C_2, ..., C_m$ a data instance X is assigned to the class for which it has the highest posterior probability conditioned on X, i.e. the class which is most probable given the prior probabilities of the classes and the data X .That is to say, X is assigned to class C_i if and only if

$$P(C_i|X) > P(C_j|X) \forall j \ni 1 \leq j \leq m \tag{2}$$

According to Bayes Theorem

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)} \tag{3}$$

Since $P(X)$ is a normalizing factor which is equal for all classes, we need only maximize the numerator $P(X|C_i)P(C_i)$ in order to do the classification. We can estimate both the values we need, $P(X|C_i)$ and $P(C_i)$, from the data used to build the classifier.[11][4]

1.3.1 Estimating Class Prior Probabilities. In general, it can be very computationally expensive to compute the $P(X|C_i)$. If each component x_k of X can have one of r values, there are rn combinations to consider for each of the m classes.
In order to simplify the calculation, the assumption of class conditional independence is made, *i.e.* that for each class, the attributes are assumed to be independent. The classifier resulting from this assumption is known as the Naive Bayes classifier. The assumption allows us to write

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i) \tag{4}$$

i.e. the product of the probabilities of each of the values of the attributes of X for the given class C_i.

1.4 Bayesian Classification Algorithm

Bayesian is one of the simplest density estimation methods from which we can form one of the standard classification methods in machine learning.
Its fame is partly due to the following properties:

- Very easy to program and intuitive
- Fast to train and to use as a classifier
- Very easy to deal with missing attributes
- Very popular in fields such as computational linguistics/NLP

1.4.1 Steps Involved in Bayesian Algorithm

- In our improved Bayesian algorithm, we estimate the class conditional probabilities for each attribute value based on their frequencies in the training dataset.
- We calculate the sum of weights for each class from the training dataset. Initially all the training dataset have unit weight.
- Then we calculated the sum of weights for each attribute value with respect to same class.
- Finally, we calculate the class conditional probabilities for each attribute value from the training dataset by dividing the sum of weights for each attribute value by the sum of weights for each class.
- After calculating the class conditional probabilities for each attribute value from the training dataset, we classify the test dataset.
- If any test example is misclassified, we update the weights of training dataset. We compare each of test example with every of training examples and compute the similarity between them and then weights of training dataset are increased by a fixed small value multiplied by the corresponding similarity measure.
- If the test example is correctly classified, then the weights of training set will remain unchanged. After weights adjustment, the class conditional probabilities for attribute values are recalculated from the modified training dataset.
- If the new set of probabilities correctly classifies all the test examples, the algorithm terminates. Otherwise, the iteration continues until all the test examples are correctly classified or the target accuracy is achieved. At this stage the algorithm terminates, the class conditional probabilities are preserved for future classification of seen or unseen intrusions.[8]

1.5 Genetic Algorithm

Genetic algorithm is one of the components of evolutionary computation technique .A simple genetic algorithm may consist of a population generator and a selector, a fitness estimator and three genetic operators namely selection, mutation and crossover. The mutation operator inverts randomly chosen bits with a certain probability. The crossover operator combines parts of the species of two individuals, generates two new off springs, which are used to replace low fitness individuals in the population. After a certain number of generations, the search process will be terminated. A genetic algorithm (or GA for short) is a programming technique that mimics biological evolution as a problem-solving strategy. Given a specific problem to solve, the input to the GA is a set of potential solutions to that problem, encoded in some fashion, and a metric called a*fitness function*that allows each candidate to be quantitatively evaluated. These candidates may be solutions already known to work, with the aim of the GA being to improve them, but more often they are generated at random.
The GA then evaluates each candidate according to the fitness function. In a pool of randomly generated candidates, we choose promising candidates toward solving the problem. These promising candidates are kept and allowed to reproduce. Multiple copies are made of them, but the copies may not perfect; random changes are introduced during the copying process. These digital offspring then go on to the next generation, forming a new pool of candidate solutions, and are subjected to a second round of fitness evaluation. Those candidate solutions which were worsened, or made no better, by the changes to their code are again deleted; but again, purely by chance, the random variations introduced into the population may have improved some individuals, making them into

better, more complete or more efficient solutions to the problem at hand. Again these winning individuals are selected and copied over into the next generation with random changes, and the process repeats. The expectation is that the average fitness of the population will increase each round, and so by repeating this process for hundreds or thousands of rounds, very good solutions to the problem can be discovered. [3][5]

1.5.1 Methods ofChange. Once selection has chosen fit individuals, they must be randomly altered in hopes of improving their fitness for the next generation. There are two basic strategies to accomplish this, they are.

(1) *Mutation:* By applying random changes to a single individual in the current generation to create a child.

(2) *Crossover:* By selecting vector entries, or genes, from a pair of individuals in the current generation and combines them to form a child.

1.5.2 Method Applied in this Paper. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation.

1.5.3 Initialization. Initially many individual solutions [3]are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions.

1.5.4 Selection. During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Fitness scaling converts the raw fitness scores that are returned by the fitness function to values in a range that is suitable for the selection function. The selection function uses the scaled fitness values to select the parents of the next generation. The selection function assigns a higher probability of selection to individuals with higher scaled values. [12]
Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions.

2. ARCHITECTURE

2.1 Components in the Architecture

• KDD Dataset: This is the initial dataset obtained from the MIT Lincoln labs.

• Pre-Processor: The pre-processor processes the large chunks of data and we obtain the minimized dataset with particular required no. of attributes.

• Bayesian Algorithm Processor: This processor applies the Bayesian Algorithm on the input dataset to classify it into various categories.

• Classified Dataset: Now we obtain the classified dataset for which we compute the false positive and false negative rates.

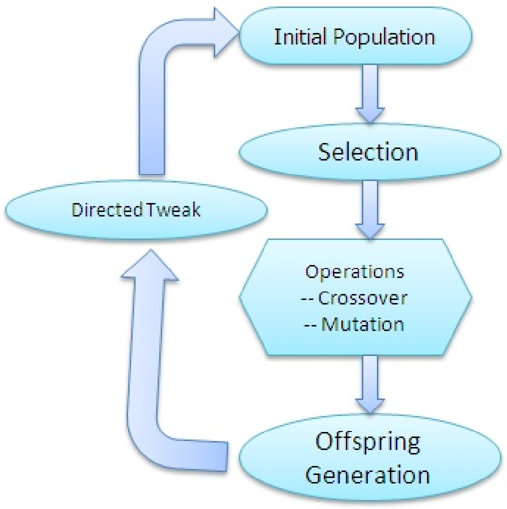

Fig. 2. Basic Genetic Algorithm Flow

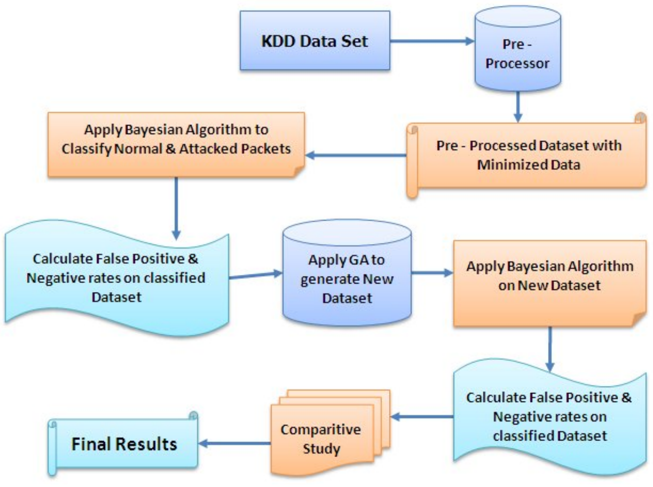
KDD Data Set
Pre - Processor
Pre - Processed Dataset with Minimized Data
Apply Bayesian Algorithm to Classify Normal & Attacked Packets
Calculate False Positive & Negative rates on classified Dataset
Apply GA to generate New Dataset
Apply Bayesian Algorithm on New Dataset
Calculate False Positive & Negative rates on classified Dataset
Comparitive Study
Final Results

Fig. 3. Architecture of the Proposed System

• Genetic Algorithm Processor: This takes the classified dataset as input and builds a new dataset using the Genetic Algorithm by observing the different variations in the dataset.

• New Dataset: After the implementation of Genetic Algorithm we have the new dataset for which we again compute the false positive and false negative rates.

• Comparative Study and Final Results: We show a comparative study of the false positive and false negative rates of the classified dataset and the newly obtained dataset and show the final results.[7]

[3]

3. IMPLEMENTATION & EXPERIMENTAL RESULTS

3.1 Flow of Steps

• The Dataset from the MIT Lincoln labs is collected and refined to obtain distinct data packets. This is done using java programming [13]

Table 3. Individual Cond. Probabilities: **Normal**

$P(A_1) = 0.6714$	$P(A_2) = 0.6429$	$P(A_3) = 0.0714$
$P(A_4) = 0.8571$	$P(A_5) = 0.0000$	$P(A_6) = 0.0428$

Table 4. Minimized Dataset after Attribute Selection

0.0	1.0	3.0	1.0	54540.0	3.0
0.0	1.0	3.0	1.0	54540.0	3.0
0.0	1.0	3.0	1.0	54540.0	3.0
0.0	1.0	3.0	1.0	54540.0	3.0
0.0	1.0	3.0	1.0	54540.0	4.0

Table 5. Minimized Dataset after Attribute Selection

0.6714	0.6428	0.0714	0.8571	0	0.0428
0.6714	0.6428	0.0714	0.8571	0	0.0428
0.6714	0.6428	0.0714	0.8571	0	0.0428
0.6714	0.6428	0.0714	0.8571	0	0.0428
0.6714	0.6428	0.0714	0.8571	0	0.0428

- A set of data is selected to train the process and the algorithm. Then we have another set called the test set with which the Bayesian Algorithm is implemented.
- Thus after classifying the dataset into attack and normal packets using Bayesian Algorithm, we calculate the false positive, false negative rates and the detection accuracy.
- We then apply Genetic algorithm to obtain a new generation of dataset by selecting the required attributes from the already existing dataset. Here, we use mutation as the reproduction operator.
- Now we use this dataset to again implement the Bayesian Algorithm and classify the dataset into attack and normal packets. Also we calculate the false positive, false negative rates and the detection accuracy.
- Finally we show a comparative analysis of the false positive, false negative rates and the detection accuracy.

3.2 Implementation Process

The data packet being used is of the following type:
0, udp, private, SF, 105, 146, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 255, 254, 1, 0.01, 0, 0, 0, 0, 0, 0
We select 6 attributes out of the 41 attributes from the above data packet. The final set of data used for the processing is:
Example format of 6 attributes:
0, udp, private, SF, 105, 1
In the training set we have the following probabilities for normal and attact packets
$P(normal) = 70/307 \quad P(attack) = 237/307$
Now the individual conditional probabilities for each attribute being normal are shown in Table. 3
Now we find the probability of the packet to be an attack and a normal packt.
These probabilities obtained are:
$P(normal) = 0 \quad P(attack) = 0.0001$
So, from the above results, we say that the packet is an ATTACK.
The minimized dataset after attribute selection is shown in Table. 4
Now the Conditional probabilities for Normal and Attack packets are shown in Table. 5 & 6 respectively.
Now as we observe the dataset is classified as each an attack by the algorithm.
The overall probabilities for each data packet are shown in Table. 7

Table 6. Minimized Dataset after Attribute Selection

0.8016	0.6962	0.0843	0.7046	0.0843	0.0548
0.8016	0.6962	0.0843	0.7046	0.0843	0.0548
0.8016	0.6962	0.0843	0.7046	0.0843	0.0548
0.8016	0.6962	0.0843	0.7046	0.0843	0.0548
0.8016	0.6962	0.0843	0.7046	0.0843	0.0548

Table 7. Minimized Dataset after Attribute Selection

Normal	0	0	0	0	0
Attack	0.0001	0.0001	0.001	0.0001	0.0001

Table 8. Minimized Dataset after Attribute Selection

Attack/ Normal	**Instances Detected**	**Instances Taken**	**Detection Accuracy**
Back	99	99	100
Buffer_overflow	25	25	100
ftp_write	5	5	100
Guess_Password	96	100	96
Imap	0	0	0
Ipsweep	148	150	98.6
Land	8	9	89
Load_module	2	2	100
Neptune	104	100	100
Nmap	79	84	94
Portsweep	97	102	95
Rootkit	6	13	46
Smurf	102	100	100
Snmpguess	92	104	88
Teardrop	8	12	67
Normal	105	100	98

3.3 Experimental Results

Detection Accuracy is computed with the implemantation of the algorithm on the each attack as shown in Table. 8
The information provided in Table. 8 is represented in graphical format shown in figure. 4 which describe the number of instances taken, detected with accuracy information for each attack.
Figure. 6 shows the false positive rate obtained when the input dataset is classified using Bayesian Algorithm. As we observe, there are 5 false positives in classifying the dataset. Figure. 5 shows the false negative rate obtained when the input dataset is classified using Bayesian Algorithm. As we observe, there are 10 false negatives in classifying the dataset. Figure. 7 shows the false negative rate obtained when the newly generated input dataset is classified using Bayesian Algorithm. As we observe, a number of attacker profiles may be observed as normal users by making certain variations in the profile.

4. COMPARITIVE STUDY

Now, we compare the detection accuracies and false alarm rates of the algorithms. We prove by this that the detection accuracy has increased and the false alarm rate has decreased. We also compare the intrusion detection performance among Neural Network (NN), Support Vector Machines (SVM), Naive Bayesian Classifier, and our Bayesian algorithm implemented with Genetic Algorithm (GABA) on KDD'99 dataset. The comparative results are summarized in the following table. [4]
As we may observe the detection accuracies of various algorithms have been compared. The proposed system of implementing

Table 2. Selected Feature Description of KDD-CUP'99 DATASET [1]

Feature name	Description	Type
duration	Length of the connection in number of Seconds	Continuous
protocol_type	Type of the protocol, e.g. TCP, UDP, etc.	Discrete
service	Network service on the destination, e.g., http, telnet, etc.	Discrete
src_bytes	Number of data bytes from source to destination	Continuous
dst_bytes	Number of data bytes from destination to source	Continuous
count	Number of connections to the same host as the current connection in the past two seconds	Continuous

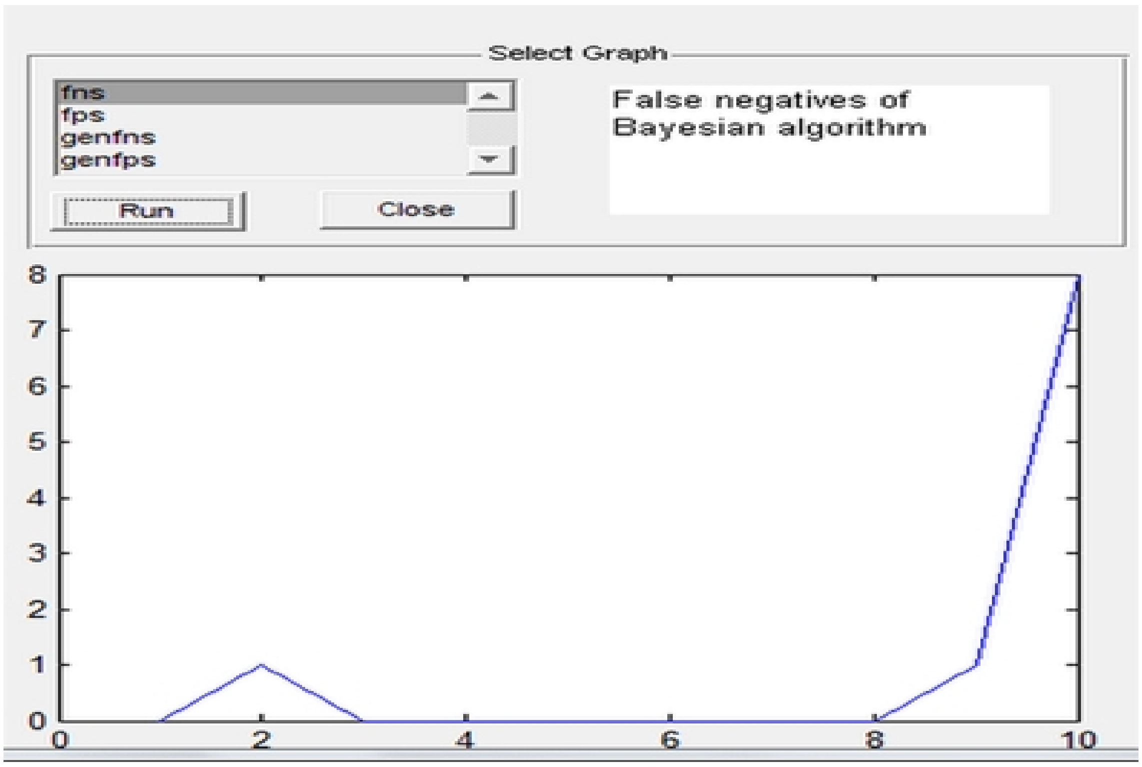

Fig. 7. False Negative Rate after Genetic Algorithm approach

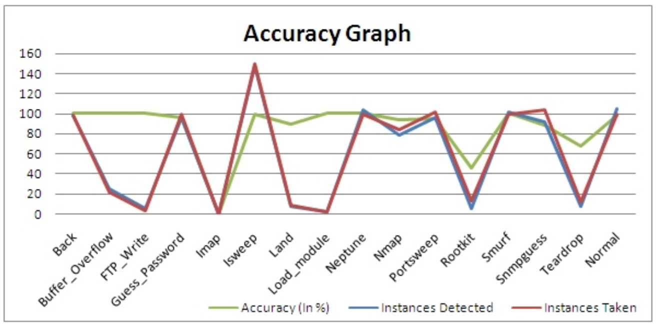

Fig. 4. Detection Accuracy Graph

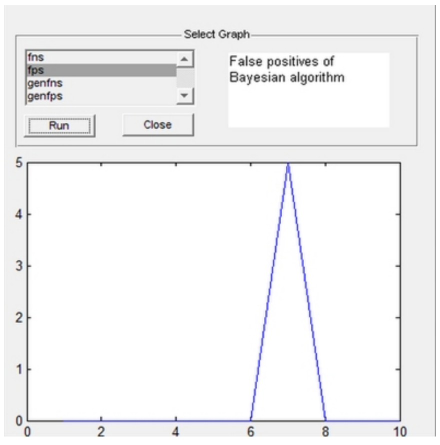

Fig. 5. False Positive Rate

Bayesian Algorithm with Genetic Algorithm has increased detection accuracy in the case of a U2R and R2L attack types where as a lesser detection accuracy in the case of DOS and PROBE attack types.

Table 9. Variations observed in developing New Dataset

Sl.No	Attack/Normal	A_1	A_2	A_3	A_4	A_5	A_6	Variations Observed
1.	Back	0	1	3	1	54540	2-5	$A_5! = 54540; 1 < A_6 < 60$
2.	Buffer_overflow	0-2225	1	5	1	1000-3000	1	$A_5 < 1000$
3.	ftp_write	20/26	1	4/12	1	74/90	1	$A5 > 200; A3! = 12$
4.	Guess_Password	4/5; 0	1	6/2	1	25-32; 120-130	1	$A_1! = 0; 35 < A_5 < 100$
5.	Imap	91	1	7	4	1352	1	$A_3! = 7;$
6.	Ipsweep	0	3	8	1	8/18	1	$A_5 = 30/1 - 4$
7.	Land	0	1	9	2	0	1	$A_4 = 1; 5 < A_5 < 12$
8.	Load_module	75-85	1	5	1	277	1	$A_1 = 0; A_5! = 277$
9.	Neptune	0	1	1	2/3	0	30-300	$A_4! = 2; A_6 = 45/56$
10.	Nmap	0	1	1	5	0	1	$A_4! = 5$
11.	Portsweep	0	1	1	3/6	0	1-3	$A_6 = 45/56;; A_1 = 3; 30 < A_6 < 165$
12.	Rootkit	15-55	1	4	1	40-200	1	$A_1 = 0; A_5 > 200$
13.	Smurf	0	3	11	1	1032; 508; 520	100-511	$A_5 = 30; 1 < A_6 < 3$
14.	Snmpguess	0	2	1	1	40-50	2-5	$A_5 = 105; 1 < A_6 < 3$
15.	Teardrop	0	2	1	1	28	10-100	$A_5 > 200$

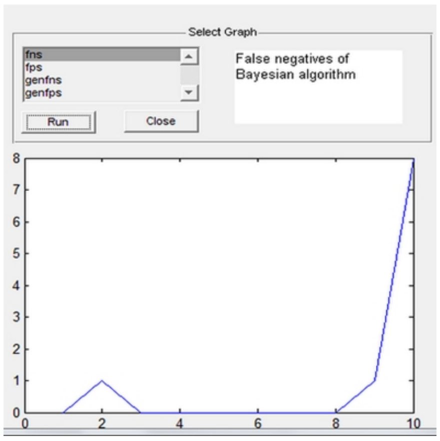

Fig. 6. False Negative Rate

Table 10. Acronym Description for Table. 9

Feature	Represenation
A1	duration
A2	protocol_type
A3	service
A4	src_bytes
A5	dst_bytes
A6	flag

cse.nitk.ac.in/researchscholars/y-v-srinivasa-murthy

Table 11. Comparison of accuracies of various algorithms

Attack Type in KDD'99	NN (%)	SVM (%)	NB (%)	GABA (%)
DOS	91.82	99.43	99.67	95.4
PROBE	95.27	98.53	99.39	96.8
R2L	94.43	96.44	99.01	98
U2R	49.05	64.07	64.45	82

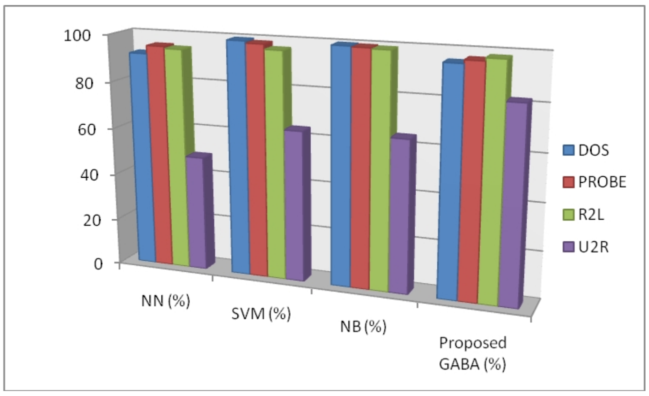

Fig. 8. Comparision of Accuracies

5. CONCLUSION

Information Security plays an important role in Hi-tech computing world. Even though firewall is used to provide security between two different networks, it fails to care about the intranet security *(security within a single network)*. In order to overcome the problem a model called Intrusion Detection System is used. The process of monitoring the events occurring in a computer system or network and analyzing them for sign of intrusions is known as intrusion detection system (IDS).In this paper we applied mathematical Bayesian Classifier Algorithm technique on Dataset to identify the normal and attacked packets. After that to enhance the algorithm we applied Genetic Algorithm which generates new Dataset using Mutation concept. Again we applied Bayesian algorithm to detect the attacked packets. We prove that, applying the combination of Bayesian algorithm and Genetic Algorithm further improves the detection rate to a greater extent. Here, we present an Intrusion Detection System which uses the Bayesian Algorithm to detect and classify the attacks from normal users. We compute the false alarm rate in implementing this algorithm on an input dataset. Since a user profile does not remain the same at all the time, we implement the Genetic Algorithm to observe the different variations possible in a user?s profile. In this way we generate a new dataset, classify this new dataset using the Bayesian algorithm and computed the false alarm rate.

6. REFERENCES

[1] KDD CUP - 99 task description. `https://kdd.ics.uci.edu/databases/kddcup99/task.html`.

[2] James Cannady. Artificial neural networks for misuse detection. In *National information systems security conference*, pages 368–81.

[3] Mark Crosbie and Gene Spafford. Applying genetic programming to intrusion detection. In *Working Notes for the AAAI Symposium on Genetic Programming*, pages 1–8. MIT, Cambridge, MA, USA: AAAI, 1995.

[4] Dewan Md Farid, Mohammad Zahidur Rahman, and Chowdhury Mofizur Rahman. Adaptive intrusion detection based on boosting and naïve bayesian classifier. *International Journal of Computer Applications*, 24(3):12–19, 2011.

[5] Thorsten Joachims. Making large scale svm learning practical. 1999.

[6] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM transactions on Information and system Security*, 3(4):262–294, 2000.

[7] Srinivas Mukkamala and Andrew Sungand Ajith Abraham. Cyber security challenges: designing efficient intrusion detection systems and antivirus tools. *Vemuri, V. Rao, Enhancing Computer Security with Smart Technology.(Auerbach, 2006)*, pages 125–163, 2005.

[8] Srinivas Mukkamala, Andrew H Sung, and Ajith Abraham. Intrusion detection using ensemble of soft computing paradigms. In *Intelligent Systems Design and Applications*, pages 239–248. Springer, 2003.

[9] Srinivas Mukkamala, Andrew H Sung, and Ajith Abraham. Intrusion detection using an ensemble of intelligent paradigms. *Journal of network and computer applications*, 28(2):167–182, 2005.

[10] Sandhya Peddabachigari, Ajith Abraham, Crina Grosan, and Johnson Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of network and computer applications*, 30.

[11] J Ross Quinlan. Decision trees and decision-making. *Systems, Man and Cybernetics, IEEE Transactions on*, 20(2):339–346, 1990.

[12] John Ross Quinlan. *C4. 5: programs for machine learning*, volume 1. Morgan kaufmann, 1993.

[13] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali-A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications 2009*, 2009.