Future Prediction on SNIP using Fuzzy Logic

Sitender,Kananda*,Nikita Kumar*, Tushar Gupta*

Dept. of Information Technology,

Maharaja Surajmal Institute of Technology, GGSIPU, Delhi, India

Abstract: **IDS which are increasingly a key part of system defense are used to identify abnormal activities in a computer system. Early systems employed the usage of firewalls, and to take reactive measures, NIDS was developed. Various data-mining and machine learning techniques have been used in the literature. In the proposed system, we have designed a system using fuzzy logic and the famous Mamdani Model- for effectively identifying and inferring the intrusion activities within a network. The proposed fuzzy logic-based system will be able to detect an intrusion, specifically a probe attack of the networks, since the rule base contains a better set of rules. The experiments and evaluations of the proposed intrusion detection system are performed with the KDD Cup 99 intrusion detection dataset. The experimental results clearly show that the proposed system achieved higher precision in identifying whether the records are normal or probe attacks.**

Keywords: ***IDS, Fuzzy Logic, Probe Attack, KDD Cup 99, Mamdani Model***

I. INTRODUCTION

In the early nineties preventive measures to defend networks such as firewalls were developed. It is only over the last few years that Intrusion Detection Systems have evolved at a large scale. In computing, a firewall is a network security system that controls the incoming and outgoing network traffic based on an applied rule set. A firewall establishes a barrier between a trusted, secure internal network and another network (e.g., the Internet) that is assumed not to be secure and trusted. If firewalls are security guards, intrusion detection systems are security cameras. The need to be aware of when an attack has taken place crossing the firewall, and the need to take reactive measures, obliged researching and industry to develop Intrusion Detection Systems. An IDS monitors traffic and spots patterns of activity, alerting you if it concludes that your network is under attack .These systems are still evolving and there are still a lot of problems for them to become a really automatic tool. IDSes are classified in many different ways, including active and passive, network-based and host-based, and knowledge-based and behavior-based:

A host-based IDS requires small programs (or agents) to be installed on individual systems to be monitored. The agents monitor the operating system and write data to log files and/or trigger alarms. A host-based IDS can only monitor the individual host systems on which the agents are installed; it doesn't monitor the entire network. A network-based IDS usually consists of a network appliance (or sensor) with a Network Interface Card (NIC) operating in promiscuous mode and a separate management interface. The IDS is placed along a network segment or boundary and monitors all traffic on that segment. A knowledge-based (or signature-based) IDS references a database of previous attack profiles and known system vulnerabilities to identify active intrusion attempts. Knowledge-based IDS is currently more common than behavior-based IDS. A behavior-based (or statistical anomaly–based) IDS references a baseline or learned pattern of normal system activity to identify active intrusion attempts. Deviations from this baseline or pattern cause an alarm to be triggered. A passive IDS is a system that's configured only to monitor and analyze network traffic activity and alert an operator to potential vulnerabilities and attacks. It isn't capable of performing any protective or corrective functions on its own. An active IDS(now more commonly known as an Intrusion Prevention System –IPS) is a system that is configured to automatically block suspected attacks and progress without any intervention required by an operator.

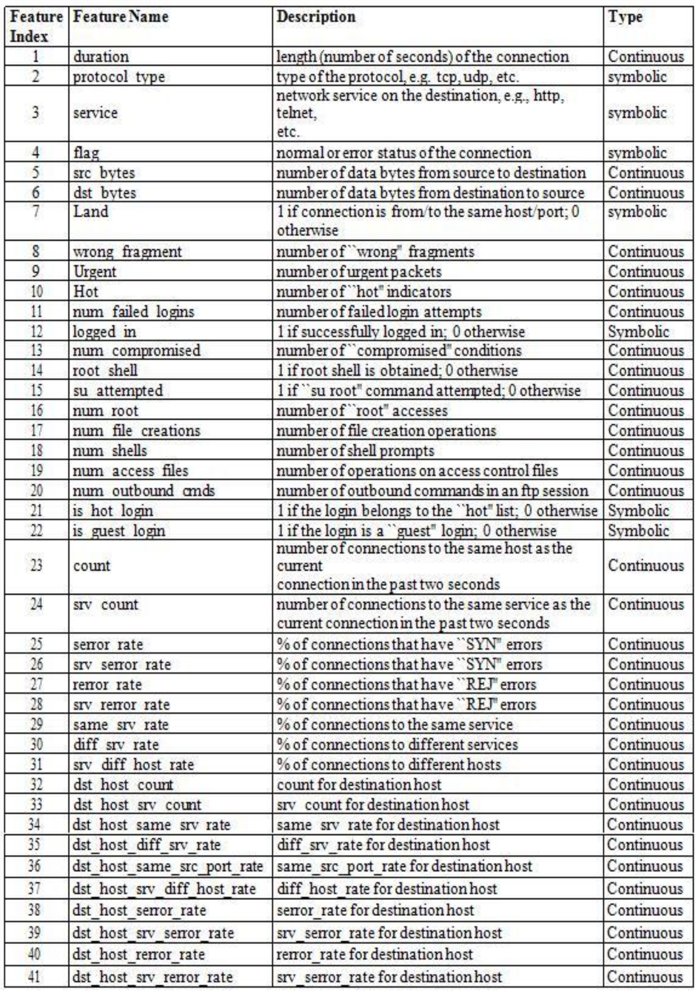
II. KDD CUP 99 DATASET

2.1 Intrusion Detection Dataset

As the human population grew in number, so did the data about them. In 1998, DARPA in concert with Lincoln Laboratory at MIT launched the DARPA 1998 dataset for evaluating IDS . The DARPA 1998 dataset contains seven weeks of training and also two weeks of testing data. In total, there are 38 attacks in training data as well as in testing data. The refined version of DARPA dataset which contains only network data (i.e. Tcp-dump data) is termed as KDD dataset. KDD training dataset consists of relatively 4,900,000 single connection vectors where each single connection vectors consists of 41 features and is marked as either normal or an attack, with exactly one particular attack type.

2.2 Features of the KDD Cup 99 Dataset

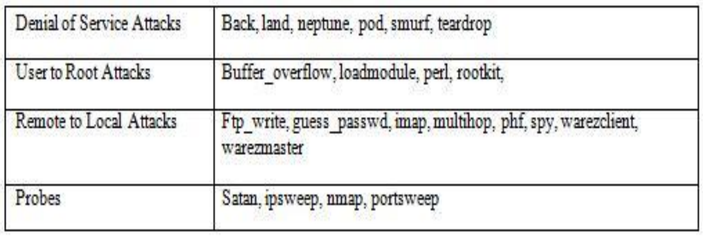
Table 1: A complete list of features given in KDD cup-99 data set [6]



Feature Index	Feature Name	Description	Type
1	duration	length (number of seconds) of the connection	Continuous
2	protocol type	type of the protocol, e.g. tcp, udp, etc.	symbolic
3	service	network service on the destination, e.g., http, telnet, etc.	symbolic
4	flag	normal or error status of the connection	symbolic
5	src bytes	number of data bytes from source to destination	Continuous
6	dst bytes	number of data bytes from destination to source	Continuous
7	Land	1 if connection is from/to the same host/port; 0 otherwise	symbolic
8	wrong fragment	number of ``wrong" fragments	Continuous
9	Urgent	number of urgent packets	Continuous
10	Hot	number of ``hot" indicators	Continuous
11	num failed logins	number of failed login attempts	Continuous
12	logged in	1 if successfully logged in; 0 otherwise	Symbolic
13	num compromised	number of ``compromised" conditions	Continuous
14	root shell	1 if root shell is obtained; 0 otherwise	Continuous
15	su attempted	1 if ``su root" command attempted; 0 otherwise	Continuous
16	num root	number of ``root" accesses	Continuous
17	num file creations	number of file creation operations	Continuous
18	num shells	number of shell prompts	Continuous
19	num access files	number of operations on access control files	Continuous
20	num outbound cmds	number of outbound commands in an ftp session	Continuous
21	is hot login	1 if the login belongs to the ``hot" list; 0 otherwise	Symbolic
22	is guest login	1 if the login is a ``guest" login; 0 otherwise	Symbolic
23	count	number of connections to the same host as the current connection in the past two seconds	Continuous
24	srv count	number of connections to the same service as the current connection in the past two seconds	Continuous
25	serror rate	% of connections that have ``SYN" errors	Continuous
26	srv serror rate	% of connections that have ``SYN" errors	Continuous
27	rerror rate	% of connections that have ``REJ" errors	Continuous
28	srv rerror rate	% of connections that have ``REJ" errors	Continuous
29	same srv rate	% of connections to the same service	Continuous
30	diff srv rate	% of connections to different services	Continuous
31	srv diff host rate	% of connections to different hosts	Continuous
32	dst host count	count for destination host	Continuous
33	dst host srv count	srv count for destination host	Continuous
34	dst host same srv rate	same srv rate for destination host	Continuous
35	dst_host_diff_srv_rate	diff_srv_rate for destination host	Continuous
36	dst_host_same_src_port_rate	same_src_port_rate for destination host	Continuous
37	dst_host_srv_diff_host_rate	diff_host_rate for destination host	Continuous
38	dst_host_serror_rate	serror_rate for destination host	Continuous
39	dst_host_srv_serror_rate	srv_serror_rate for destination host	Continuous
40	dst_host_rerror_rate	rerror_rate for destination host	Continuous
41	dst_host_srv_rerror_rate	srv_serror_rate for destination host	Continuous

2.3 Attributes of the various types of Attacks

Table 2: Various types of attacks described in four major categories [6]



Denial of Service Attacks	Back, land, neptune, pod, smurf, teardrop
User to Root Attacks	Buffer_overflow, loadmodule, perl, rootkit,
Remote to Local Attacks	Ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster
Probes	Satan, ipsweep, nmap, portsweep

1.Denial of Service Attack (DOS): In this category the attacker makes some computing or memory resources too busy or too full to handle legitimate request, or deny legitimate users access to machine. DOS contains the attacks: 'neptune', 'back', 'smurf', 'pod', 'land', and 'teardrop'.

2. Users to Root Attack (U2R): In this category the attacker starts out with access to a normal user account on the system and is able to exploit some vulnerability to obtain root access to the system. U2R contains the attacks: 'buffer_overflow', 'loadmodule', 'rootkit' and 'perl'

3.Remote to Local Attack (R2L): In this category the attacker sends packets to machine over a network but who does not have an account on that machine and exploits some vulnerability to gain local access as a user of that machine. R2L contains the attacks: 'warezclient', 'multihop', 'ftp_write', 'imap', 'guess_passwd', 'warezmaster', 'spy' and 'phf.

4. Probing Attack (PROBE): In this category the attacker attempt to gather information about network of computers for the apparent purpose of circumventing its security. PROBE contains the attacks: 'portsweep', 'satan', 'nmap', and 'ipsweep'.

III. PROPOSED SOLUTION

Probe-response attacks are a new threat for collaborative intrusion detection systems. Site scanning/probing is the initial phase of any attack on Web applications. During this phase, the attacker gathers information about the structure of the Web application (pages, parameters, etc.) and the supporting infrastructure (operating system, databases, etc.). Target Web sites are scanned for known vulnerabilities in infrastructure software (such as IIS) as well as unknown vulnerabilities in the custom code developed for the specific target application. Site scanning/probing is the main technique attackers use to gather as much information as possible about a Web application and the supporting infrastructure. A standard site scan is composed of several steps. First, the attacker detects the operating system installed on the server. This can be done using automatic tools like nmap, by identifying the Web Server type in the HTTP Request (for example, IIS runs on Windows-based sites) or by guessing according to file extensions. Identifying which Web server runs on the target machine is very useful to the attacker. Knowing the specific type of Web server (and by extension its default configuration), an attacker may try to exploit known vulnerabilities, access sample files, and try default user accounts. After the attacker analyzes the infrastructure, the entire application can be scanned. Application scanning provides a map of the entire site including: all pages, parameters used by dynamic pages, cookies used by the site and transactions flow. This information leads the attacker to an understanding of the application's authentication, authorization, logic, and transactional mechanisms. This body of information provides the basis of a strategy to attack the target site.

IV. IMPLEMENTATION

The focus has shifted on fuzzy rule learning for effective intrusion detection using data mining techniques. We have developed a fuzzy rule based system in detecting the PROBE attack. The fuzzy rules generated from the proposed strategy can be able to provide better classification rate in detecting the intrusion behavior.The different steps involved in the proposed system for anomaly-based intrusion detection are described in the figure:

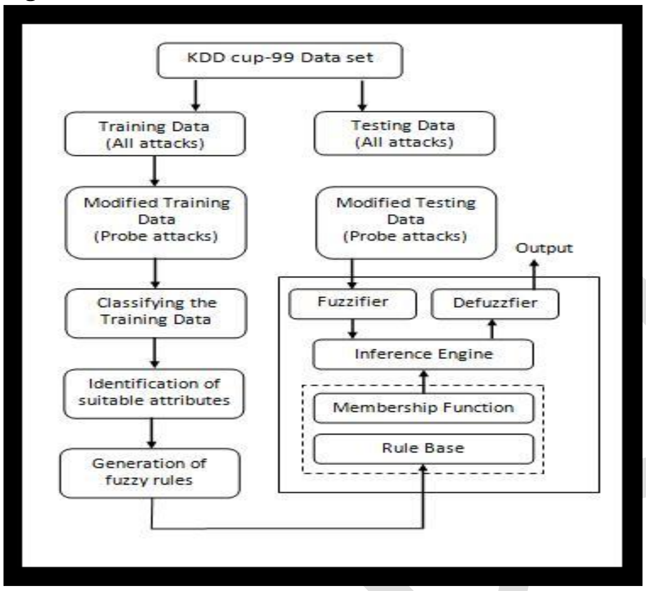

KDD cup-99 Data set
Training Data (All attacks)
Testing Data (All attacks)
Modified Training Data (Probe attacks)
Modified Testing Data (Probe attacks)
Output
Classifying the Training Data
Fuzzifier
Defuzzfier
Inference Engine
Identification of suitable attributes
Membership Function
Rule Base
Generation of fuzzy rules

Figure 1: Overall steps ofthe proposed intrusion detection system

4.1 Fuzzy Logic

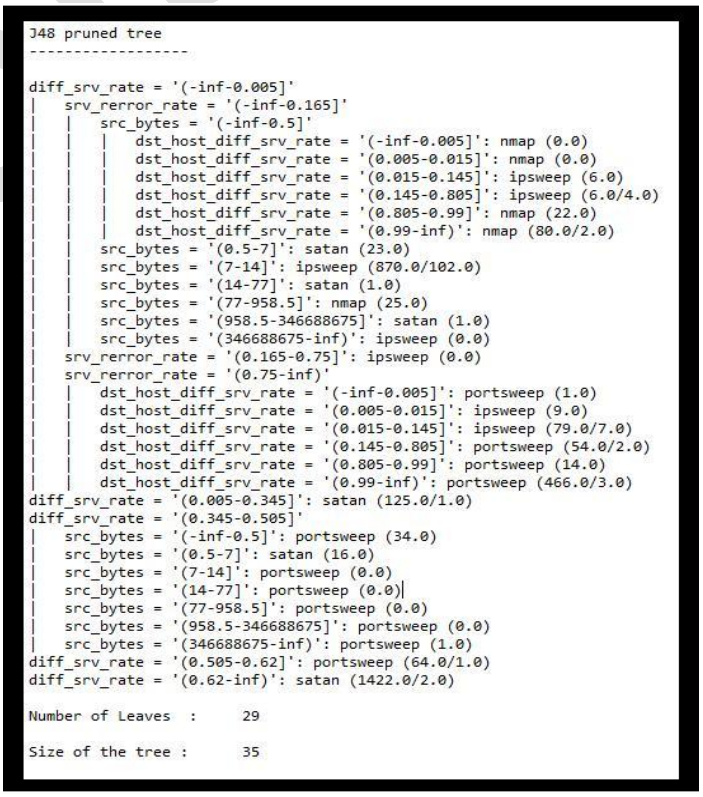
Fuzzy logic is a form of many-valued logic that deals with approximate, rather than fixed and exact reasoning. Compared to traditional binary logic (where variables may take on true or false values), fuzzy logic variables may have a truth value that ranges in degree between 0 and 1. Fuzzy logic has been extended to handle the concept of partial truth, where the truth value may range between completely true and completely false. Furthermore, when linguistic variables are used, these degrees may be managed by specific functions.

4.2 Mamdani Model

Mamdani method is widely accepted for capturing expert knowledge. It allows us to describe the expertise in more intuitive, more human-like manner. Mamdani-type FIS entails a substantial computational burden as compared to its Sugeno counterpart. The most fundamental difference between Mamdani-type FIS and Sugeno-type FIS is the way the crisp output is generated from the fuzzy inputs. While Mamdani-type FIS uses the technique of defuzzification of a fuzzy output, Sugeno-type FIS uses weighted average to compute the crisp output. The expressive power and interpretability of Mamdani output is lost in the Sugeno FIS since the consequents of the rules are not fuzzy. Due to the interpretable and intuitive nature of the rule base, Mamdani-type FIS is widely used in particular for decision support application. Other differences are that Mamdani FIS has output membership functions whereas Sugeno FIS has no output membership functions. Mamdani FIS is less flexible in system design in comparison to Sugeno FIS as latter can be integrated with ANFIS tool to optimize the outputs.

4.3 Rule Generation

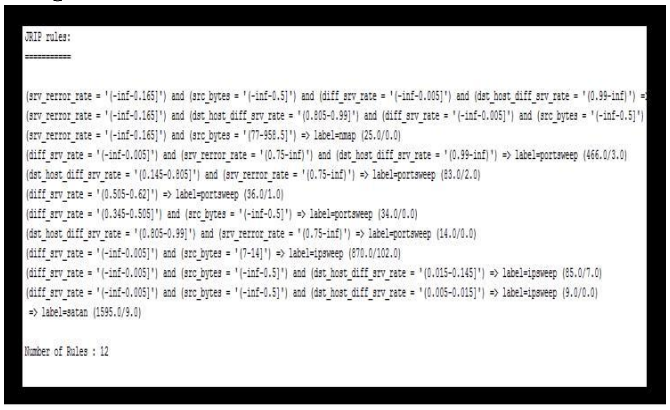
For the generation of rule been put to use. Through which, the following rules were generated.
J48: It is an open source algorithm in Weka data mining tool. A decision tree can be generated from the input data by C4.5 proram. It is an algorithm used to generate a decisiontreeandisanextensionofQuinlan'searlierID3 Algorithm. The decision trees generated by this can be used for classification and so referred to as statistical classifier



```
J48 pruned tree
------------------

diff_srv_rate = '(-inf-0.005]'
|   srv_rerror_rate = '(-inf-0.165]'
|   |   src_bytes = '(-inf-0.5]'
|   |   |   dst_host_diff_srv_rate = '(-inf-0.005]': nmap (0.0)
|   |   |   dst_host_diff_srv_rate = '(0.005-0.015]': nmap (0.0)
|   |   |   dst_host_diff_srv_rate = '(0.015-0.145]': ipsweep (6.0)
|   |   |   dst_host_diff_srv_rate = '(0.145-0.805]': ipsweep (6.0/4.0)
|   |   |   dst_host_diff_srv_rate = '(0.805-0.99]': nmap (22.0)
|   |   |   dst_host_diff_srv_rate = '(0.99-inf)': nmap (80.0/2.0)
|   |   src_bytes = '(0.5-7]': satan (23.0)
|   |   src_bytes = '(7-14]': ipsweep (870.0/102.0)
|   |   src_bytes = '(14-77]': satan (1.0)
|   |   src_bytes = '(77-958.5]': nmap (25.0)
|   |   src_bytes = '(958.5-346688675]': satan (1.0)
|   |   src_bytes = '(346688675-inf)': ipsweep (0.0)
|   srv_rerror_rate = '(0.165-0.75]': ipsweep (0.0)
|   srv_rerror_rate = '(0.75-inf)'
|   |   dst_host_diff_srv_rate = '(-inf-0.005]': portsweep (1.0)
|   |   dst_host_diff_srv_rate = '(0.005-0.015]': ipsweep (9.0)
|   |   dst_host_diff_srv_rate = '(0.015-0.145]': ipsweep (79.0/7.0)
|   |   dst_host_diff_srv_rate = '(0.145-0.805]': portsweep (54.0/2.0)
|   |   dst_host_diff_srv_rate = '(0.805-0.99]': portsweep (14.0)
|   |   dst_host_diff_srv_rate = '(0.99-inf)': portsweep (466.0/3.0)
diff_srv_rate = '(0.005-0.345]': satan (125.0/1.0)
diff_srv_rate = '(0.345-0.505]'
|   src_bytes = '(-inf-0.5]': portsweep (34.0)
|   src_bytes = '(0.5-7]': satan (16.0)
|   src_bytes = '(7-14]': portsweep (0.0)
|   src_bytes = '(14-77]': portsweep (0.0)|
|   src_bytes = '(77-958.5]': portsweep (0.0)
|   src_bytes = '(958.5-346688675]': portsweep (0.0)
|   src_bytes = '(346688675-inf)': portsweep (1.0)
diff_srv_rate = '(0.505-0.62]': portsweep (64.0/1.0)
diff_srv_rate = '(0.62-inf)': satan (1422.0/2.0)

Number of Leaves  :     29

Size of the tree :      35
```

Figure 2: Rules generated using J48 classification

JRip: This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER). It is based in association rules with reduced error pruning (REP),a very common and effective technique found in decision tree algorithms.



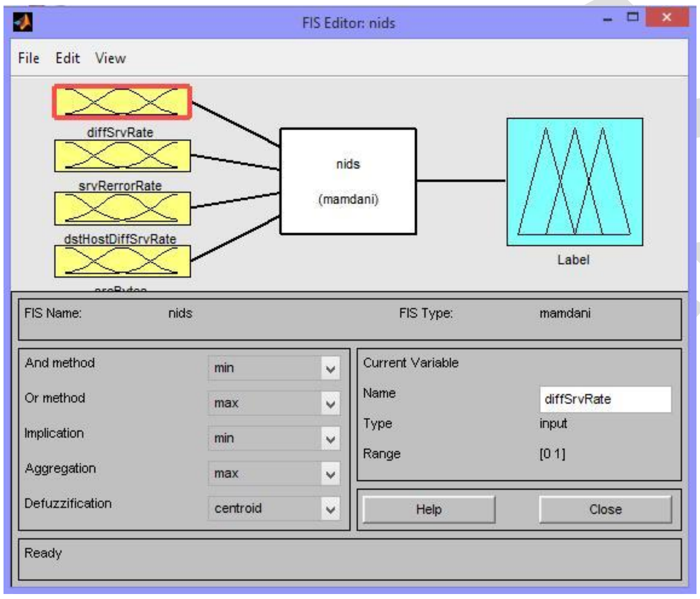
```
JRIP rules:
===========

(srv_rerror_rate = '(-inf-0.165]') and (src_bytes = '(-inf-0.5]') and (diff_srv_rate = '(-inf-0.005]') and (dst_host_diff_srv_rate = '(0.99-inf)') =
(srv_rerror_rate = '(-inf-0.165]') and (dst_host_diff_srv_rate = '(0.805-0.99]') and (diff_srv_rate = '(-inf-0.005]') and (src_bytes = '(-inf-0.5]')
(srv_rerror_rate = '(-inf-0.165]') and (src_bytes = '(77-958.5]') => label=nmap (25.0/0.0)
(diff_srv_rate = '(-inf-0.005]') and (srv_rerror_rate = '(0.75-inf)') and (dst_host_diff_srv_rate = '(0.99-inf)') => label=portsweep (466.0/3.0)
(dst_host_diff_srv_rate = '(0.145-0.805]') and (srv_rerror_rate = '(0.75-inf)') => label=portsweep (83.0/2.0)
(diff_srv_rate = '(0.505-0.62]') => label=portsweep (36.0/1.0)
(diff_srv_rate = '(0.345-0.505]') and (src_bytes = '(-inf-0.5]') => label=portsweep (34.0/0.0)
(dst_host_diff_srv_rate = '(0.805-0.99]') and (srv_rerror_rate = '(0.75-inf)') => label=portsweep (14.0/0.0)
(diff_srv_rate = '(-inf-0.005]') and (src_bytes = '(7-14]') => label=ipsweep (870.0/102.0)
(diff_srv_rate = '(-inf-0.005]') and (src_bytes = '(-inf-0.5]') and (dst_host_diff_srv_rate = '(0.015-0.145]') => label=ipsweep (85.0/7.0)
(diff_srv_rate = '(-inf-0.005]') and (src_bytes = '(-inf-0.5]') and (dst_host_diff_srv_rate = '(0.005-0.015]') => label=ipsweep (9.0/0.0)
 => label=satan (1595.0/9.0)

Number of Rules : 12
```

Figure 3: Rules generated using JRip classification

4.4 Creating FIS model using MATLAB FIS editor

To open a new FIS file with the FIS editor, type the following into MATLAB:

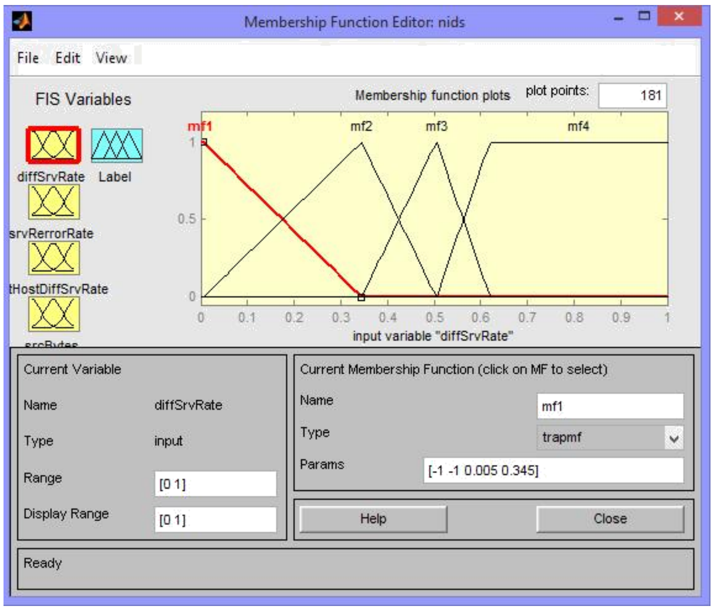
```
>>fuzzy
```



FIS Name: nids
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: diffSrvRate
Type: input
Range: [0 1]
Ready

Figure 4: FIS editor

Above shown is the Fuzzy Inference System editor. With the FIS editor the input and output membership functions can be defined, the rule base is defined, and the fuzzy operators are set.The default system has one input and one output and uses the Mamdani inference and aggregation method. This editor also illustrates the three aspects of a fuzzy contoller; fuzzification, inference, and defuzzification. To begin, let us define the membership functions. Double click on the yellow input box and you will be brought to the membership function editor

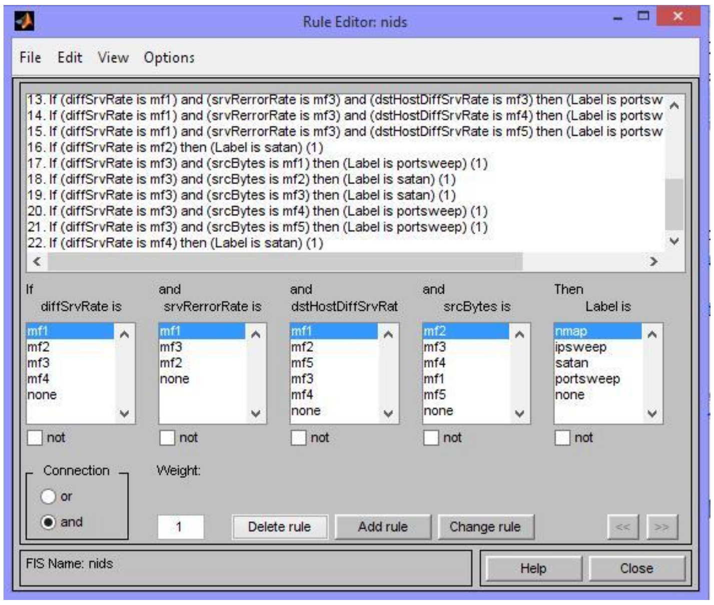


Current Variable
Name: diffSrvRate
Type: input
Range: [0 1]
Display Range: [0 1]
Current Membership Function (click on MF to select)
Name: mf1
Type: trapmf
Params: [-1 -1 0.005 0.345]
Ready

Figure 5: Membership Function editor

Above shown is the membership function editor. To manipulate the membership functions simply click on them and drag them. Click on the line itself to move the function and click on the boxes at the corners and drag to change the shape. If you know the shape already, you can also select the function and enter the shape in the Params box. The name can also be changed using the Name box. The range can be redefined as well.

Next step is to define the rules. To do this double click on the white box between the input and the output membership functions and you will be brought to the Rule editor.



```
13. If (diffSrvRate is mf1) and (srvRerrorRate is mf3) and (dstHostDiffSrvRate is mf3) then (Label is portsw
14. If (diffSrvRate is mf1) and (srvRerrorRate is mf3) and (dstHostDiffSrvRate is mf4) then (Label is portsw
15. If (diffSrvRate is mf1) and (srvRerrorRate is mf3) and (dstHostDiffSrvRate is mf5) then (Label is portsw
16. If (diffSrvRate is mf2) then (Label is satan) (1)
17. If (diffSrvRate is mf3) and (srcBytes is mf1) then (Label is portsweep) (1)
18. If (diffSrvRate is mf3) and (srcBytes is mf2) then (Label is satan) (1)
19. If (diffSrvRate is mf3) and (srcBytes is mf3) then (Label is satan) (1)
20. If (diffSrvRate is mf3) and (srcBytes is mf4) then (Label is portsweep) (1)
21. If (diffSrvRate is mf3) and (srcBytes is mf5) then (Label is portsweep) (1)
22. If (diffSrvRate is mf4) then (Label is satan) (1)
```

If diffSrvRate is	and srvRerrorRate is	and dstHostDiffSrvRat	and srcBytes is	Then Label is
mf1	mf1	mf1	mf2	nmap
mf2	mf3	mf2	mf3	ipsweep
mf3	mf2	mf5	mf4	satan
mf4	none	mf3	mf1	portsweep
none		mf4	mf5	none
		none	none	
not	not	not	not	not

Connection: or [✓ and]
Weight: 1
FIS Name: nids

Figure 6: Rule editor

4.5 Results Obtained

After applying the JRIP classification algorithm using WEKA on probing KDD Cup training dataset, quiet interesting results were discovered in the variation of frequency of attacks.

The training dataset has a maximum of Probes attacks respectively and the normal records accounted to 25% of the total records. Probe had the maximum frequency with 45% of all the attacks.

The result shown in fig 10 in probing attack with four major categories Satan, Nmap, Ipsweep, Portsweep are active. The attacker uses the probe attack to target protocols. The hackers exploit the ICMP protocol majorly using the Dos attacks, followed by PROBE.

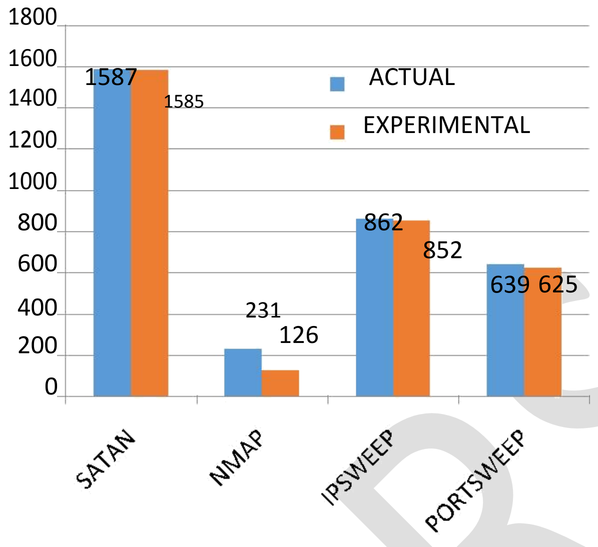
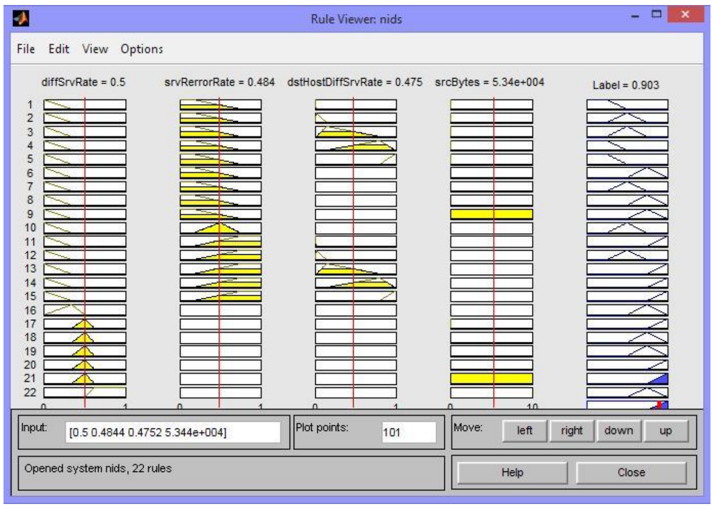


Fig.7 Statistical view of Category wise PROBE attacks

V. CONCLUSIONS

We have developed a Fuzzy Inference System, detecting the probe attack and its sub-types from the other attacks within a network. An effective set of fuzzy rules for inference approach were identified automatically by making use of the fuzzy rule learning strategy, which are more effective for detecting intrusion in a computer network. At first, the definite rules were generated then, fuzzy rules were identified by manually and these rules were given to fuzzy system, which classify the test data. We have used KDD cup 99 dataset for evaluating the performance of the proposed system and experimentation results showed that the proposed method is effective in detecting various PROBE attacks in computer networks.



diffSrvRate = 0.5 srvRerrorRate = 0.484 dstHostDiffSrvRate = 0.475 srcBytes = 5.34e+004 Label = 0.903

Input: [0.5 0.4844 0.4752 5.344e+004] Plot points: 101

Opened system nids, 22 rules

Figure 8: Rule viewer

REFERENCES

[1]. R. ShanmugavadivuDr.N.Nagarajan, “Network Intrusion Detection System
[2]. http://www.cs.waikato.ac.nz/~ml/weka/
[3]. http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html
[4]. http://www.sigkdd.org/kddcup/index.php?section=1999&method=data
[5]. Yao, J. T., S.L. Zhao, Fuzzy Intrusion Detection”Data Mining, Intrusion Detection, Information Assurance, And Data Networks Security, SPIE, Vol. 5812, pp. 23-30, Orlando, Florida, USA, 2005.
[6]. MahbodTavallaee, EbrahimBagheri, Wei Lu and Ali A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set", in Proceedings of the Second IEEE international conference on Computational intelligence for security and defense applications, pp. 53-58, Ottawa, Canada, 2009
[7]. M. SanieeAbadeh, J. Ha detection using a fuzzy genetics-based learning al Journal of Network and Computer Applications, vol.30, no.1, pp. 414–428, 200
[8]. Arman Tajbakhsh, Mohammad Rahmati, AbdolrezaMirzaei, "Intrusion detection using fuzzy association rules", Applied Soft Computing, Vol: 9, No: 2, pp: 462-469, 2009
[9]. K. Labib and V. Rao-of- V Service And Network Probe Attacks Using Principal Component Analysis”.
[10]. Shilpalakhina, Sini Joseph, Bhupendraverma, Feature Reduction using Principal Component Analysis for Effective Anomaly–Based Intrusion Detection on NSL-KDD, International Journal of Engineering Science and Technology, Vol. 2(6), 2003, pp.1790-1799.