Journal of Engineering and Applied Sciences 12 (6): 1523-1537, 2017
ISSN: 1816-949X

A New Distributed Learning Based Algorithm for Network Intrusion Detection System

Aryan Mohammadi Pasikhani and Elankovan A. Sundararajan
Faculty of Information Science and Technology, Centre for Software Technology and Management, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

Abstract: The significant increase in computer network usage and the huge amount of sensitive data being stored and transferred through them has escalated the attacks and invasions on these networks. Secure data communication over the internet and any other network is always under threat of intrusions and misuses. The system that monitors the events occurring in a computer system or a network and analyzes the events for signs of intrusion is known as an Intrusion Detection System (IDS). In information protection, the Intrusion Detection System (IDS) has become a crucial component in terms of computer and network security which monitors the network traffic to detect possible security threats. There are various approaches being utilized in intrusion detections but unfortunately any of the systems so far are not completely flawless and suffer from a number of drawbacks such as low accuracy to detect new types of intrusions and misclassification of normal and malicious traffic, in addition to long response time. It is necessary to develop an IDS that is accurate, adaptive and extensible to overcome these weaknesses. In this study, we proposed a learning-based method which improves IDS adaptability to new attacks and reduces false alarms. The method that has a distributed architecture to increase performance and scalability of the IDS and uses C4.5 decision trees with the feedback learning technique to adapt dynamic network behaviors. To evaluate the proposed method we used some well-known datasets in this context such as KDD Cup 99 and did several tests with approximately 97% detection accuracy on benchmarks. According to the promising results, the adaptable IDS approach is more accurate than traditional systems and it is more efficient against new complex network attacks.

Key words: Network-based, signature based, distributed, multi-agent, adaptive

INTRODUCTION

Over recent year, the internet has turned into a part of our daily life and evolved into a ubiquitous infrastructure. Secure data communication over the internet and any other network is always under threat of intrusions and misuses. Various security methods, like access control, encryption or firewalls have been provided to improve the security of networks. The contemporary network is used with a firewall which terminates any arriving traffic from breaking the security policy laid by the network administrator. However, these techniques have failed to fully protect against increasingly sophisticated intrusions as the firewall is not successful to discover the interior users who are bypassing such rules. As a result, Intrusion Detection Systems (IDSs) have become a crucial component of any network security infrastructure, detecting network attacks before they induce extensive damage which is not only sniffing the arriving traffic but also sniffing interior traffic. In order to find out what is Intrusion Detection (ID), we should declare the definition of intrusion. An intrusion can be defined as (Soni *et al*., 2015): "any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource". IDSs are systems that raise an alert by sniffing or watching the arriving packet traffic. IDS fundamentally serve as a last defender of the network against threat after firewall.

The objective of an IDS is to create a defense shield against malicious usages of computer systems by detecting a misuse or a breach of a security policy and notifying administrators to an ongoing intrusion. An IDS detects intrusions by watching a system or network and examining an audit stream collected from the system or network to search for signs of malicious behavior. If they are malicious to the system then IDS will detect it automatically. An IDS is considered to identify all kinds of malicious network traffic and computer usage that

Corresponding Author: Aryan Mohammadi Pasikhani, Faculty of Information Science and Technology, Centre for Software Technology and Management, Universiti Kebangsaan Malaysia, 43600 UKM Bangi, Selangor, Malaysia

traditional firewall could not detect them. Intrusion detection is dependent on the speculation that the activities of the intruder differ from lawful user behaviors in ways that could be qualified. Researchers always want to find an intrusion detection technology with better detection accuracy and less training time. Most Intrusion Detection Systems (IDS) can fall in to 2 categories: anomaly-based intrusion detection and misuse-based intrusion detection. Also depending on where these IDSs are located, we have 2 classes of host-based and network-based intrusion detection systems. The system that identifies patterns of traffic or application data presumed to be malicious known as misuse detection systems and methods that compare activities against a normal baseline called anomaly detection systems. The system is Network-based Intrusion Detection System (NIDS) if it monitors the flow of network packets and host-based IDS HIDS if it monitors system calls or logs. Network Intrusion Detection System (NIDS) resides on network and observes the malicious traffic passing through the network whereas Host Intrusion Detection System (HIDS) resides on the system and observes inbound and outbound traffic going or coming from/to the system; the example of the HIDS will be firewall (Desale *et al*., 2015). Existing HIDS approaches examine audit data provided by either an operating system or by a particular application such a web server (Rahmatian *et al*., 2012).

In this research, we have provided a method that provides considerable accuracy and efficiency with help of modular structure and intelligent agents that have employed C4.5 decision tree (Wang *et al*., 2009) structure in their detection layer and also with feedback system it could propose framework that is able to modify itself with changes in network data and with the help of distributed processing in detection modules it is able to detect attack much faster.

Literature review: In their method (Wang *et al*., 2009) they proposed an intrusion detection algorithm based on the C4.5 decision tree; C4.5 decision tree classification method is used to build a decision tree for intrusion detection then convert the decision tree into rules and save them into the knowledge base of intrusion detection system. These rules are used to judge whether the new network behavior is normal or abnormal. They used KDD Cup 99 dataset in their job and their experiments show that the detection accuracy rate of intrusion detection algorithm based on C4.5 decision tree is over 90% and the process of constructing rules is easy to understand. In this study (Kumar and Yadav, 2014) an artificial neural network based intrusion detection system is proposed which used gradient descent with momentum backpropagation algorithm for learning. Although, random patterns are selected for training but the proposed neural network is tested across complete testing data of KDD Cup 99 dataset. Their output is evaluated in terms of accuracy detection rate and false positive ratio. Based on their experiments result the accuracy of their method is 93%.

In their job (Gong *et al*., 2010), they discover that the research of neural network based data fusion IDS tries to combine the strong processability of neural network with the advantages of data fusion IDS such as low distorting and good information quality. They discover that the pruning of neural network can also provide better performance. Their proposed model could be used in large-scale and distributed system for intrusion detection. They used KDD dataset in their job and the experimental result shows that they have received 83.4 overall detection.

The purpose of this study (Lin *et al*., 2015) is to develop a new intrusion detection system that combines the idea of feature generation and visualization technology. They use a four-star graph to give an intuitive simulation of high dimension data classification for IDS. Finally, generation of new visualized numerical features decrease the dimensionality of the data 4-16 or 4 and increase the computation speed of the new IDS. Visualization is an intuitive way for feature selection and feature reduction. They used KDD data set over their job and the FASVFG-based classifier achieves a generalization accuracy of 94.3555% in validation experiment.

They (Shanmugavadivu and Nagarajan, 2011) have developed an anomaly based intrusion detection system in detecting the intrusion behavior within a network. A fuzzy decision-making module was designed to build the system more accurate for attack detection, using the fuzzy inference approach. An effective set of fuzzy rules for inference approach were identified automatically by making use of the fuzzy rule learning strategy which is more efficient for detecting intrusion in a computer network. At first, the definite rules were generated by mining the single length frequent items from attack data as well as normal data. Then, fuzzy rules were identified by fuzzifying the definite rules and these rules were given to fuzzy system which classify the test data. They 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 intrusions in computer networks. As they discussed by analyzing the result, the overall performance of the proposed system is improved and it achieves >90% accuracy.

In this study (Sahu and Jena, 2016) an MSVM classifier is used to detect and identify the attacks by type. Cross-validation and re-sampling methods are applied to improve the learning process to the datasets. The model can determine a particular known type of attack when the unknown instances need to be classified. Their accuracy over corrected KDD dataset was 91.445%.

Their (Koc *et al*., 2012) model is a multinomial classifier that is used to classify network events as normal or attack events. The model is based on a new data mining method called Hidden Naive Bayes (HNB). Their framework includes a feature selection model based on the three filter methods: Correlation based (CFS), Consistency-based (CONS) and interact feature selection methods. These approaches are leading filter-based. Based on their experimental result, they have used KDD dataset over their job and they received 0.9372 accuracy.

This (Gumus *et al*., 2014) study focuses on dynamic anomaly-based IDS and they have used online Naive Bayes classifier in their job. The classifier starts with a small number of training examples of normal and bad classes; then as it classifies the rest of the samples one at a time, it continuously updates the mean and the standard deviations of the features (IDS variables). They have used KDD dataset over their job and based on their experimental results they have received 0.93 accuracy for online Naive Bayes and 0.96 for online K-NN.

MATERIALS AND METHODS

Proposed framework: In this part, we will go through the techniques that we have already employed in our framework to overcome those issues and we explain each of them individually in details. Our framework has a modular constructor and each part of it has specific duties and tasks. In the following, each of the components of the proposed method is discussed in detail; Fig. 1 has shown an overview of our proposed model architecture. As it has demonstrated in Fig. 2 there are 2 phases in our method that are training phase and testing phase. The profiles are built in the training phase and used to detect intrusions in the testing phase. In the first step, the algorithm starts training period. In this period, the decision trees on each of detection modules should be trained with training instances. In here each of detection modules is a decision tree that its structure is different than another one due to it has prepared by different proportion of training data. After the training phase, the decisions of these detection agents (decision trees) about incoming packet will be used in the election among detection modules in the testing phase. In the testing phase we will examine and test our method with different data than we have used in training period.

Capturing network traffic: Generally, for detecting the probable intrusion, the IDS systems are placed in the path of the stream network traffic and by analyzing the captured network packets the probability of intrusion will be reviewed. In the proposed method also algorithm inputs contain packets of network traffic and every time a copy of captured packet will be given to the algorithm for analyzing. Each packet contains 2 main parts that are header and data. The header contains the information for transportation and packet control and data includes transferring information. As regards a packet contains a negligible piece of transferring data on the network and also it could be encrypted and analyze it individually could not facilitate detecting intrusion, the initial concentration of provided method is in the header of the packet which contains several features. The value of each element based on the type of it could be an integer, decimal and also a string. These features will be analyzed by devices and system on the network and based on them the packet will be routed and controlled. For example, "protocol-type" feature specifies the protocol type that the packet should be transferred based on it and it could be TCP, UDP or ICMP. Studying and analyzing the features of each packet would give us valuable information about the probability of intrusion. For example, "service" feature that specifies service type of the packet is one of the effectual factors in specifying attacks. For instance, the attacks that consider destroying web servers will target HTTP service and the mail service attacks are seeking to target SMTP. Of course, this feature is just one of the features to detect attacks and combination of them with the rest of the features could resulting in a model that can distinguish attack packet from normal packet. In Table 1, each of the features with explanation has shown in details.

Distribution module: In the proposed method, the first part is capturing network traffic for distribution module that receives a copy of passing packet on the network. This module controls the distribution of packet among detection modules. Since, the quantity of detection module is selected based on hardware and processor resources, so it could be different quantity and generally, there are N detection modules in the system that distribution module sends a copy of current packet on the network to each of them. In order to increase the speed of the system, part of the pre-processing of the packets such as extracting features will be done by distributed module and result of it will be sent to next component. With this

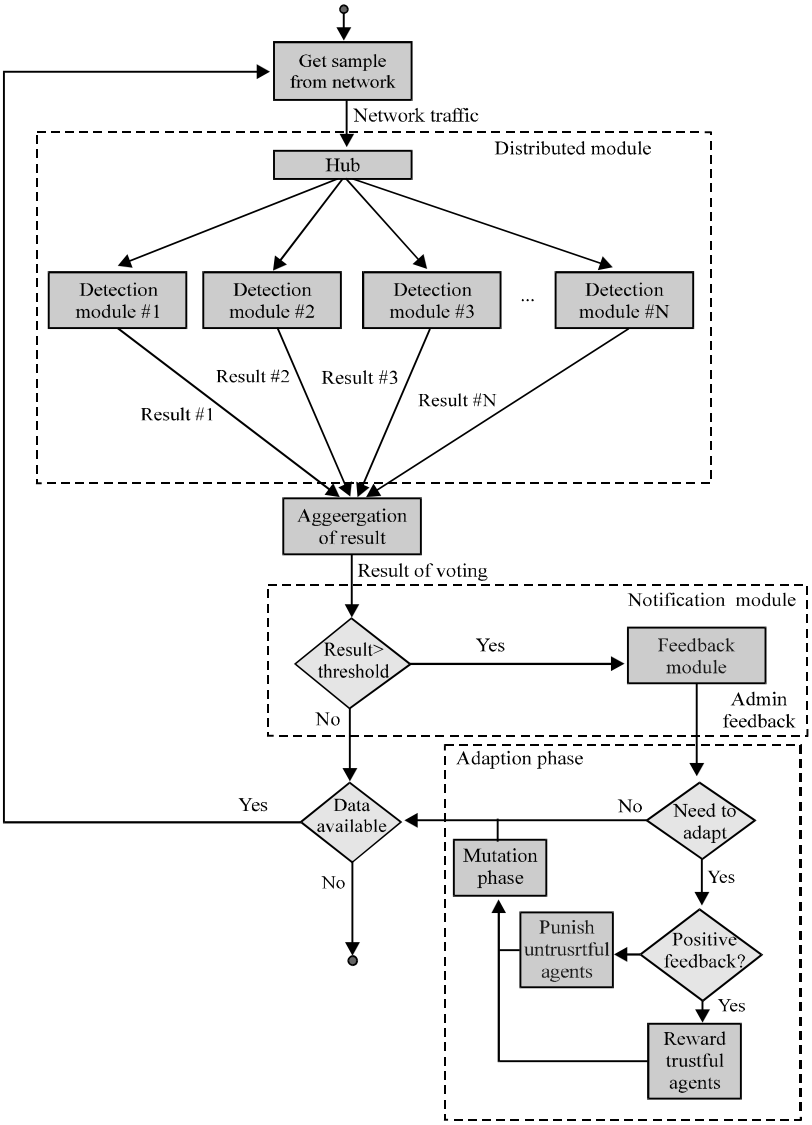

Fig. 1: Overview of our framework

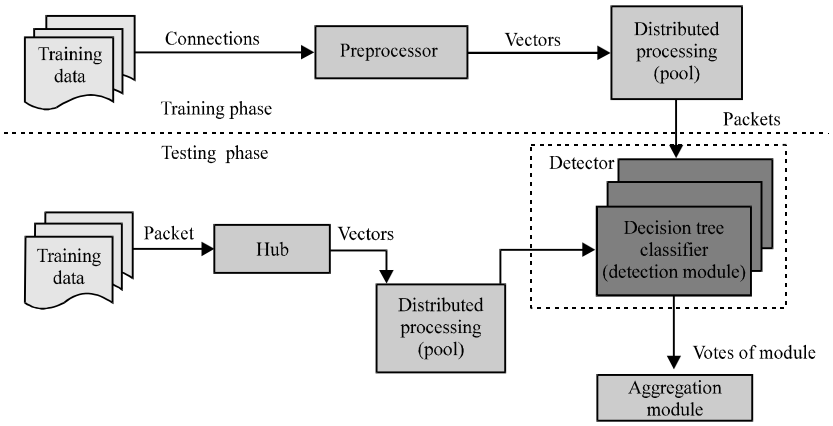

Fig. 2: Training and testing phase

Table 1: KDD dataset features with description and sample value

Feature name (category)	Description	Type	Sample values
Basic features			
Duration	# of seconds of the connection	Integer	0<x<58329
Protocol_type	Type of the protocol	String	Tcp, udp, icmp
Service	Network service on the destination	String	Http, smtp, domain_u, auth,Finger,telnet, etc
Flag	Normal or error status of the connection	String	SF, S2, S1, S3, OTH, REJ,RSTO, S0,RSTR, RSTOS0, SH
Src_bytes	# of data bytes from source to destination	Integer	0<x<1379963888
Dst_bytes	# of data bytes from destination to source	Integer	0<x<1309937401
Land	1: connection is from or to the same host or port; 0: otherwise	Integer	0 or 1
Wrong_fragment	# of "wrong" fragments	Integer	0 or 1 or 2
Urgent	# of urgent packets	Integer	0<x<14
Content features			
Hot	# of "hot" indicators	Integer	0<x<77
num_failed_logins	# of failed login attempts	Integer	0<x<5
Logged-in	1: successfully logged in; 0: otherwise	Integer	0 or 1
Num_compromised	# of "compromised" conditions	Integer	0<x<7479
Root_shell	1: root shell is obtained; 0: otherwise	Integer	0 or 1
Su_attempted	1: "su root" command attempted; 0: otherwise	Integer	0 or 1 or 2
Num_root	# of "root" accesses	Integer	0<x<7468
Num_fIle_crations	# of file creation operations	Integer	0<x< 43
Num_shells	# of shell prompts	Integer	0 or 1 or 2
Num_access_files	# of operations on access control files	Integer	0<x<9
Num_outbound_cmds	# of outbound commands in an ftp session	Integer	0 or 1
Is_host_login	1: the login belongs to the "hot" list; 0: otherwise	Integer	0 or 1
Is_guest_login	1: the login is a "guest" login; 0: otherwise	Integer	0 or 1
Traffic features			
Count	# of connections to the same host	Integer	0<x<511
Srv_count	# of connections to the same service	Integer	0<x<511
Serror_rate	% of connections with SYN errors to the same host	Float	0<x<1
Srv_serror_rate	% of connections with SYN errors to the same service	Float	0<x<1
Rerr or_rate	% of connections with REJ errors to the same host	Float	0<x<1
Srv_rerr or_rate	% of connections with REJ errors to the same service	Float	0<x<1
Same_srv_rate	% of connections to the same service	Float	0<x<1
Diff_srv_rate	% of connections to different services	Float	0<x<1
Srv_diff_host_rate	% of connections to different hosts	Float	0<x<1
Dst_host_count	# of connections to the same host	Integer	0<x<255
Dst_host_srv_count	# of connections to the same service and to the same destination host	Integer	0<x<255
Dst_host_same_srv_rate	% of connections to the same service and to the same destination host	Float	0<x<1
Dst_host_diff_srv_rate	% of connections to different services and to the same destination host	Float	0<x<1
Dst_host_same_src_port_rate	% of connections from the same source port and to the same destination host	Float	0<x< 1
Dst_host_srv_diff_hostrate	% of connections to different hosts and to the same destination host	Float	0<x< 1
Dst_host_serror_rate	% of connections with SYN errors to the same host	Float	0<x< 1
Dst_host_srv_serror_rate	% of connections with SYN errors to the same service and to the same destination host	Float	0<x< 1
Dst _host_rerror_rate	% of connections with REJ errors to the same host	Float	0<x< 1
Dst_host_srv_rerror_rate	% of connections with REJ errors to the same service and to the same destination host	Float	0<x< 1
Class			
Label	"Normal" or a specific attack type	String	Normal, smurf, imap, nmap, etc.

situation, there is no need to pre-process and extract data inside of the packet in each detection module. The detection modules will be run on different threads in order to increase the speed of the system. So incoming data will be sent to various detection modules that each of these detection modules is an individual unit to decide about it.

Detection modules: These modules analyze the packets that have already received on network traffic based on their own individual structure in distributed way and hand over the result of it to aggregation module. Each of the modules in their own structure has a C4.5 decision tree that has trained with a different bunch of labeled packets in the training phase of the method. The result of the analyzing each module is directly related to the structure of its own tree and structure of each tree could be different based on the data which has already used to train them. The main idea of this part of proposed model

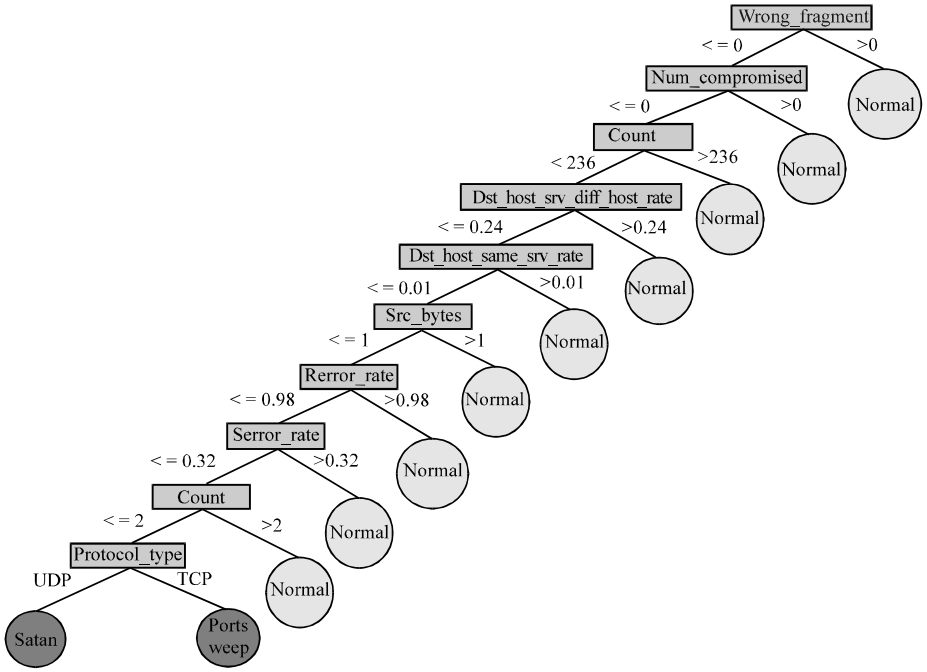

Fig. 3: Structure of tree based on given data

is similar to evolutionary algorithms; it means that in the proposed method the detection modules will decide about one instance based on their own structures and detect it as an abnormal or normal packet. These decisions will be aggregated in next component and in the next detection turn, it will increase the efficacy of the detection module that offered better performance and has a correct recognition and decreases the effectiveness of those that have incorrect recognition or even replace weak detection module with those that are more efficient. This module after few turns helps the system to keep the detection modules which are more suitable for the current situation of network and play more effective in detecting attacks and also they decrease wrong detections of the algorithm. As long as each of modules has trained with a different part of data so they have different tree structure than each other and also at the end they work together as a unit because of election among all of the modules. As discussed before, the structure of each detection modules is based on the C4.5 decision tree (Wang *et al*., 2009) and because of that in the next section we are going to talk about this algorithm and also explain how it works.

Structure of C4.5 in detection modules: We have implemented the C4.5 decision tree algorithm (Wang *et al*., 2009) in our detection module in order to categorize the samples. The input records of training have given and the analogous gained ratio for every one of the element has computed. The continuous and discrete elements have recognized from the records of input. After that, the tree has created based on the gained ratio (Paul *et al*., 2016).

In the next the every one of specific path in the tree, the rules have created. The decision tree is the yield of this module. The structure of the tree that has had 2 intrusion records on its leaves has shown in Fig. 3.

Algorithm illustrate its DT structure:

```
wrong_fragment <= 0
| num_compromised <= 0
| | count <= 236
| | | dst_host_srv_diff_host_rate <= 0.24
| | | | dst_host_same_srv_rate <= 0.01
| | | | | src_bytes <= 1
| | | | | | rerror_rate <= 0.98
| | | | | | | serror_rate <= 0.32
| | | | | | | | count <= 2
| | | | | | | | | protocol_type = tcp: portsweep
| | | | | | | | | protocol_type = udp: satan
```

The rules of this two kind of intrusion is as following: IF "wrong_fragment" <= 0 AND "num_compromised" <= 0 AND count <= 2 AND "dst_host_srv_diff_host_rate" <= 0.24 AND "dst_host_same_srv_rate" <= 0.01 AND s"rc_bytes" <= 1 AND "rerror_rate" <= 0.98 AND "serror_rate" <= 0.32 AND "protocol_type" = udp THEN attack = "satan".

The rule for the "portsweep" intrusion: IF "wrong_fragment" < = 0 and "num_compromised" < = 0 and count < = 2 and "dst_host_srv_ diff_host_rate" < = 0.24 and "dst_host_same_srv_rate" < = 0.01 and "src_bytes" < = 1 and "rerror_rate" <= 0.98 and "serror_rate" < = 0.32 AND "protocol_type" = tcp THEN attack = "portsweep"

This two example of intrusions rules which we have shown in above, cover "satan" intrusion and "portsweep"

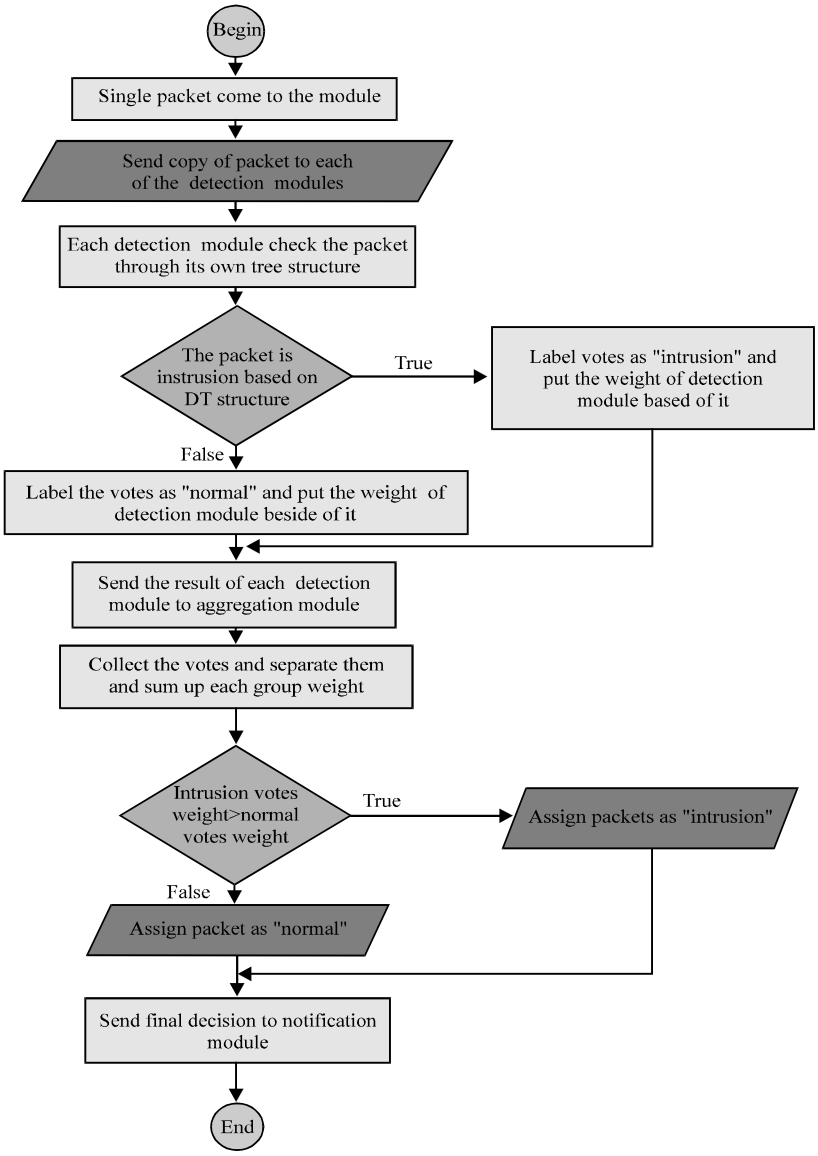

Fig. 4: Distribution module

one. The rest of intrusions have covered in the same way by the further rules that have created through different part of training data by implementing C4.5 decision tree for each of our detection modules.

Aggregation module: After labeling the instance that has analyzed by detection modules, the result of all module will be aggregated in the aggregation module. The purpose of this module is to give a final decision based on the result that it has received from the previous phase. The weighted voting method has employed to aggregate the results in this module. This method that is utilized ordinary weighted voting will assign specific weight to each vote based on voter features and after that the agent that has the biggest weight will be selected as a final answer. The default weight of each detection modules is assigned as 1/N that N in here represent the number of detection modules and in future, it will be changed depends on each of modules mistakes. If the decision of each module recognized as the correct answer, the weight of it would increase into maximum 1 or if it is incorrect it will reduce into minimum 0. The zero number means that it will not affect the result. In the following, we have explained sample scenario. Figure 4 has shown flowchart of distribution module.

Based on Table 2, 5 modules will give their votes about the same packet. Meanwhile, modules number 1 and 4 have considered it as an intrusion but rest of modules

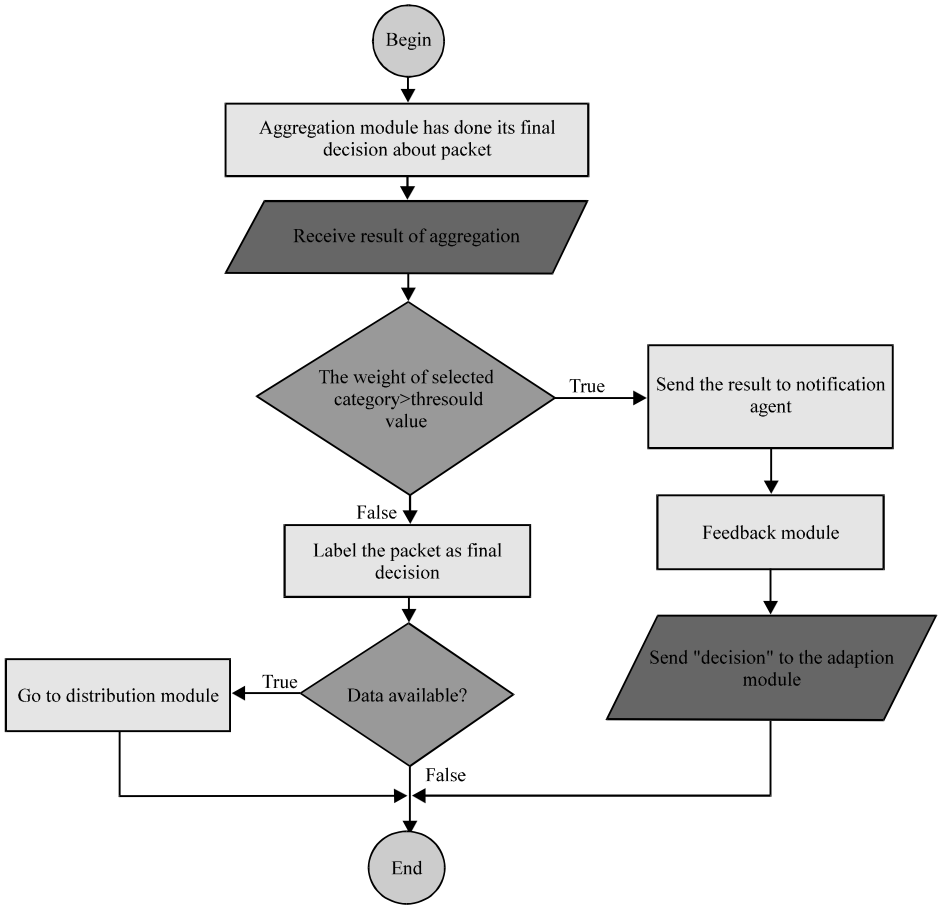

Fig. 5: Notification phase

Table 2: Weight table of detection modules

	Module				
Variables	1	2	3	4	5
Weight	0.3	0.15	0.1	0.55	0.25
Label	Yes	No	No	Yes	No

consider it as a normal one. Based on performances of these modules which had different performance about packet before so their weight is different than each other as it is showed in Table 2. Based on the values on the table, the summarization of the vote for label "yes" is 0.85 and for label "No" is 0.50 so based on mentioned results, aggregation module will consider the packet as an intrusion.

The duty of every IDS is to notify upper layer about detected intrusion that it could be a human operator (Administrator) or other systems. In the proposed method after aggregation of votes and receiving a final decision, if the packet has detected as intrusion, upper layer will be notified. Figure 5 has shown flowchart of notification module. In order to increase the flexibility of proposed system in this module, we have designed control parameter that enables the system to inform upper level based on the probability of attack. For example, the packet that has detected by most of the detection module as the intrusion is more probable to be an attack than the one that most of the modules has considered it as normal. As result of that with changing control parameter we could control the sensitivity of the system in confronting with detecting threats. The default amount of this module have assigned based on the testing dataset and it could be modified by administrator practically. Assigning this parameter appropriately will effect on reducing FAR and also increase the accuracy of our method. After notifying upper layer this module receives a feedback and send it to adaption module in order to improve the performance of the system.

Adaptation module: The primary purpose of this module is to add the ability of learning and adaption to our model to be able to modify itself by changes in network traffic patterns and threats. As we discussed in previous sections, detection modules depend on their own structure of decision tree may hand over the different decision about the same packet. It is evident that some of this judgment could be wrong for example a normal packet

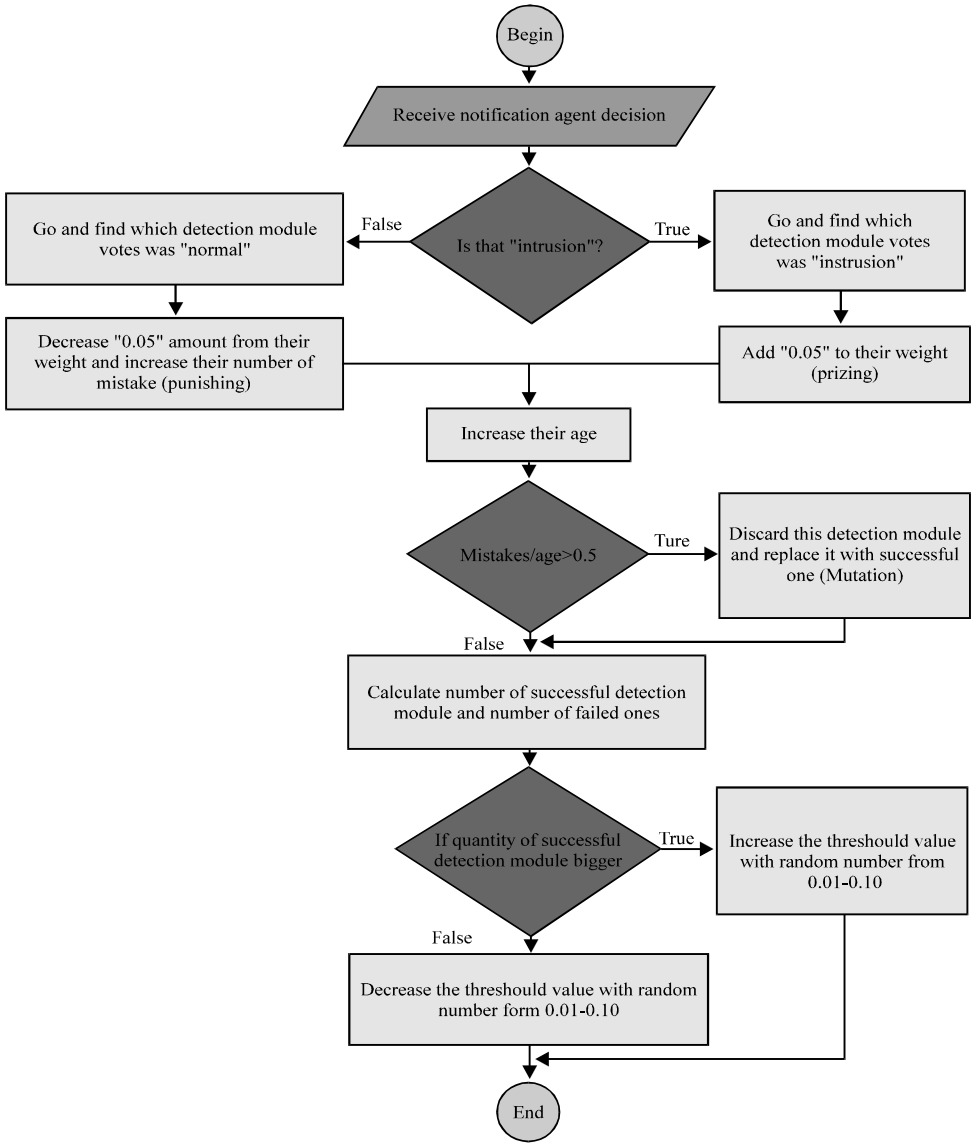

Fig. 6: Adaption phase

would be considered as an attack or a malicious packet could be seen as normal. So in order to improve system performance, it is necessary to modify the modules that make the wrong decision. In this module, three solutions (Malialis and Kudenko, 2015) have provided to correct and change the behavior of detection modules that are based on reward and punishment method and GA mutation (Pal and Parashar, 2014) method. In next, each of these solutions has explained in details. Figure 6 has shown the flowchart of adaption phase of our method. In reward solution those modules that were successful to detect the purpose of the packet will be prized. In here the term prize means that their weight will be increased in order to make them more efficient in confronting with new packets. This increase will be as specific number and their maximum number is 1. If a module correctly recognizes a packet its weight is going to increase. For example we have a model that its weight is 0.7 after it detects an intrusion correctly its weight is going to increase to 0.8.

In mutation solution, due to detection modules may have weak performance, it is necessary to replace and modify weak modules with better ones in order to increase the performance of the system. This process is same to GA mutation (Pal and Parashar, 2014) phase. In order to do it a new successful decision tree that has constructed

base on the latest packet on the network will be replaced by the prior version of modules that are not efficient and all of their weight will be assigned as default. If the new structure shows better performance it receives reward and its weight will be increased but if it is not it would be punished or even replaced with new structure again.

RESULTS AND DISCUSSION

Evaluation and experimental results: In the previous section we discussed the framework of proposed method in this research. In this part we are going to evaluate our proposed method over KDD Cup 99 dataset and also analyze our results according to existing IDS evaluation methods. First we have described the dataset that has used for our evaluation, then we talked about evaluation measures that have been used by researchers in order to evaluate their approaches. Finally, we have talked about our experimental results and the performance of the proposed method and also we have compared our job with two other algorithms in the end.

Dataset: Since 1999, the KDD Cup 99 dataset has been the most widely used dataset for the evaluation Intrusion detection methods and systems. The KDD'99 dataset is a subset of the DARPA 1998 dataset which is the most popular data set used to evaluate IDSs. This dataset was prepared by Stolfo *et al.* and is built on the data captured in the DARPA98 IDS evaluation program. Stolfo and Lee processed the tcpdump data of the 1998 DARPA dataset and made it available for the KDD'99 classifier-learning contest. Through the processing, the binary tcpdump data is transformed to connections that contain some context information for each network session. Despite some drawbacks, the KDD'99 dataset is still the most widely used benchmark data set for evaluating machine learning-based IDSs. It can be used for testing machine learning algorithms without further time-consuming preprocessing. Moreover, the data set contains 39 different types of attacks which makes it a comprehensive source for IDS evaluation. The KDD'99 data contains 2 sets: the training set and the testing set. Each set consists of a number of records called connections. A connection is a sequence of packets in a time frame when data flows to and from a source IP address to a target IP address under some well defined protocol. In the TCP protocol a connection has multiple packets. For the UDP protocol, each connectionless packet is treated as a connection. Each connection is labeled as either normal or a specific attack type. There are 4,898,431 connections in the training set and 494,021 in the testing set. In our job we have used KDD 10% dataset as test dataset that contains 97278 normal instances and 396743 intrusion instances.

Features: In the KDD99 dataset, each connection has 42 features (including its class label) that contain information about the session. The features can be divided into 4 categories: basic, content, traffic and class. The basic features contain the essential characteristics about a connection record. The content features are constructed from the payload of traffic packets and contain host-related information such as the number of login failures. The traffic features contain statistical information such as the number of connections to the same host within a two-second time window. The class feature indicates if the connection is normal or intrusive; it is used for training and evaluation. A description of each of the 42 features is listed in Table 1. These 42 input attributes have either discrete or continuous values and divided into three groups. The first group of attributes is the basic features of network connection which include the duration, prototype, service, number of bytes from source IP addresses or from destination IP addresses and some ages in TCP connections. The second group of attributes in KDD99 is composed of the content features of network connections and the third group is composed of the statistical features that are computed either by a time window or a window of certain kind of connections. KDD'99 data include three independent sets: the whole KDD training data, 10% KDD training data and KDD correct data. Each record represents a network connection described by 41 features and a label specifying the status of this record as either normal or one of 39 specific attack types. In KDD99 dataset these four attacks (DoS, U2R, R2L and probe) are divided into 22 different attacks that tabulated in Table 3.

Evaluation of IDS: An evaluation of a method or a system in terms of accuracy or quality is a snapshot in time. As time passes, new vulnerabilities may evolve and current evaluations may become irrelevant. In this section we discuss various measures used to evaluate network intrusion detection methods and systems.

Accuracy: Accuracy (Lippmann *et al*., 2000) is a metric that measures how correctly an IDS works, measuring the percentage of detection and failure as well as the number of false alarms that the system produces. If a system has 70% accuracy, it means that it correctly

Table 3: Different type of attacks in KDD99 dataset

4 main intrusion class	Type of features
Denial of Service (DOS)	Teardrop, smurt, pod,neptune, land, back
Probing	Satan, portsweep, nmap, ipsweep
User to Root (U2R)	Rootkit, loadmodule, perl,overflow, buffer
Remote to User (R2L)	Warezmaster, warezclient, spy,phf, multihop, imap,guess passwd, ftp write

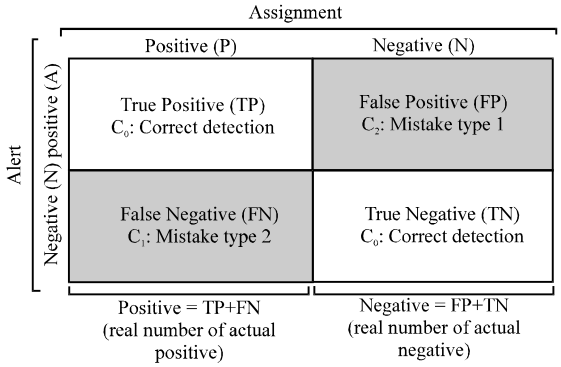

Fig. 7: Confusion matrix

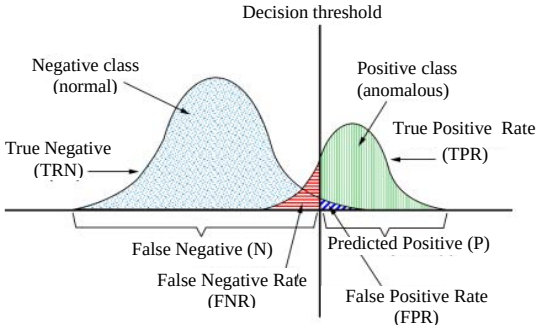

Fig. 8: Illustration of confusion matrix in term of related evaluation measurre

classifies 70 instances out of 100 to their actual classes. While there is a big diversity of attacks in intrusion detection the main focus is that the system be able to detect an attack correctly. From real life experience, one can easily conclude that the actual percentage of abnormal data is much smaller than that of the normal. Consequently, intrusions are harder to detect than normal traffic, resulting in excessive false alarms as the biggest problem facing IDSs. The following Eq. 1 is the accuracy measure:

$$\text{Accuracy} = \frac{\text{True Positive (TP)} + \text{True Negative (TN)}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \quad (1)$$

Sensitivity and specificity (Wang, 2008): These 2 measures attempt to measure the accuracy of classification for a 2-class problem. When an IDS classifies data, its decision can be either right or wrong. It assumes true for right and false for wrong, respectively.

If S is a detector and Dt is the set of test instances there are four possible outcomes described using the confusion matrix given in Fig. 7. When an anomalous test instance (p) is predicted as Anomalous (A) by the detector S, it is counted as True Positive (TP); if it is predicted as Normal (N), it is counted as False Negative (FN). On the other hand, if a Normal (n) test instance is predicted as Normal (N) it is known as True Negative (TN) while it is a False Positive (FP) if it is predicted as Anomalous (A).

The True Positive Rate (TPR) is the proportion of anomalous instances classified correctly over the total number of anomalous instances present in the test data. TPR is also known as Fig. 8 showes of confusion matrix in terms of related evaluation Measures sensitivity. The False Positive Rate (FPR) is the proportion of normal instances incorrectly classified as anomalous over the total number of normal instances contained in the test data as Eq. 2 has shown:

$$\text{False PositiveRate(FPR)} = \frac{\text{False Positive (FP)}}{\text{Negative (N)}} = \frac{\text{False Positive (FP)}}{\text{FP} + \text{TN}} \quad (2)$$

The True Negative Rate (TNR) is also called specificity that has shown in Eq. 3. The False Negative Rate (FNR) has shown in Eq. 4:

$$\text{True Negative Rate (TNR)} = \frac{\text{True Negative (TN)}}{\text{Negative (N)}} = \frac{\text{True Negative (TN)}}{\text{FP} + \text{TN}} = 1 - \text{False PositiveRate (FPR)} \quad (3)$$

$$\text{FalseNegativeRate (FNR)} = \frac{\text{FalseNegative (FN)}}{\text{Positive}} = \frac{\text{FalseNegative (FN)}}{\text{TP} + \text{FN}} = 1 - \text{TruePositiveRate (TPR)} \quad (4)$$

Sensitivity is also known as the hit rate. Between sensitivity and specificity, sensitivity is set at high priority when the system is to be protected at all cost and specificity gets more priority when efficiency is of major concern. Consequently, the aim of an IDS is to produce as many TPs and TNs as possible while trying to reduce numbers of both FPs and FNs. The majority of evaluation criteria use these variables and the relations among them to model the accuracy of the IDSs. Misclassification rate: This measure attempts to estimate the probability of disagreement between the true and predicted cases by dividing the sum of FN and FP by the total number of pairs observed, (TP+FP+FN+TN). In other words, misclassification rate is defined as (FN+FP)/ (TP+FP+ FN+TN).

Confusion matrix (Ghorbani *et al*., 2009): The confusion matrix is a ranking method that can be applied to any kind

of classification problem. The size of this matrix depends on the number of distinct classes to be detected. The aim is to compare the actual class labels against the predicted ones as shown in Fig. 7. The diagonal represents correct classification. The confusion matrix for intrusion detection is defined as a 2-by-2 matrix, since there are only 2 classes known as intrusion and normal. Thus, the TNs and TPs that represent the correctly predicted cases lie on the matrix diagonal while the FNs and FPs are on the right and left sides. As a side effect of creating the confusion matrix, all four values are displayed in a way that the relation between them can be easily understood. The costs associated with the classification are shown in the Fig. 7. As is evident, correct Classification (C_0) has no cost while incorrect classification (C_2 and C_1) brings with it some cost. The cost elements c2 and c1 can be same or different depending on the cost of misclassification. In IDS scenario a FN is far more damaging than a FP, so c2> c1.

Precision recall and f-measure (Wang *et al.*, 2010): As it has shown in Eq. 5 precision is a measure of how a system identifies attacks or normal. A flagging is accurate if the identified instance indeed comes from a malicious user, which is referred to as true positive. The final quantity of interest is recall, a measure of how many instances are identified correctly as it has shown in Eq. 6. Precision and recall are often inversely proportional to each other and there is normally a trade-off between these two ratios. An algorithm that produces low precision and low recall is most likely defective with conceptual errors in the underlying theory. The types of attacks that are not identified can indicate which areas of the algorithm need more attention. Exposing these flaws and establishing the causes assist future improvement:

$$\text{Precision} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Positive (FP)}} \quad (5)$$

$$\text{True Positive Rate (TPR)} = \text{Recall} = \frac{\text{TruePositive (TP)}}{\text{Positive}} = \frac{\text{True Positive (TP)}}{\text{TP} + \text{FN}} \quad (6)$$

The F-measure mixes the properties of the previous 2 measures as the harmonic mean of precision and recall (Ghorbani *et al.*, 2009). If we want to use only one accuracy metric as an evaluation criterion, F-measure is the most preferable. Note that when precision and recall both reach 100%, the F-measure is the maximum, i.e., 1 meaning that the classifier has zero present false alarms and detects 100% of the attacks. Thus a good classifier is expected to obtain F-measure as high as possible. The F-measure has shown in Eq. 7.

$$\text{F-measure} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} \quad (7)$$

Evaluation of the proposed IDs: In this study, we evaluate our method over different measures that we have discussed before. Due to IDSs process stream data on the network we have simulated KDD dataset as stream network traffic to evaluate our job. The Training data in our method contains 5 million records that we have shuffled them randomly in order to change the records order and prevent from affecting on the evaluation result. After preparing data, the records respectively has given to the algorithm. To be able to simulate different scenario we have designed function that it could simulate traffic when it has provided by particular input number. As a result when we change the value of this service we could simulate different network traffic data. As long as in the proposed algorithm, receiving feedback (Paul *et al.*, 2016) has considered as a method to evaluate algorithm and also it is necessary for modifying detection modules of the algorithm, simulating feedback scenarios is so necessary to evaluate our algorithm. Because of that, we have designed feedback structure that it could give feedback to algorithm about reported instances from 0-100%. In this structure, zero amount means that the system will not receive any feedback from upper level agent about its performance and it will behave based on its own structure only. The 100% amount of feedback means that the system will receive feedback per each recognition and this feedback let the system to improve its performance based on received feedback. Another amount that would be assigned from 0-100 will demonstrate the different percentage of feedback. If the value of feedback is something among 0-100, the recognitions that have assigned by feedback will be chosen randomly based on random function in order to control feedback value through simulation.

Experimental results: In the case of evaluating our proposed model we have done 6 different test over it which each of them has considered to evaluate particular parameter inside of our method or compare our proposed method with another job. The results of evaluation of these parts have calculated based on the average of ten times execution in order to examine our proposed method more precisely. As it has shown in Table 4 and Fig. 9 when we increase the number of detection modules it will effect on the accuracy of our algorithm and it is because of the fact that we are increasing the number of the classifier in detection modules. Of course, the relation of

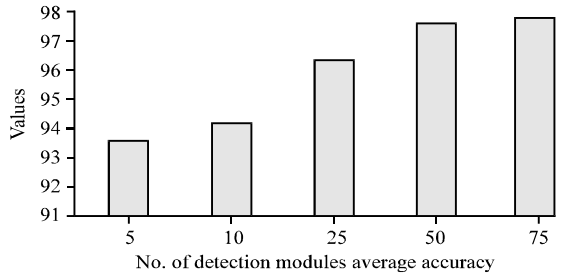

Fig. 8: Accuracy of detection modules

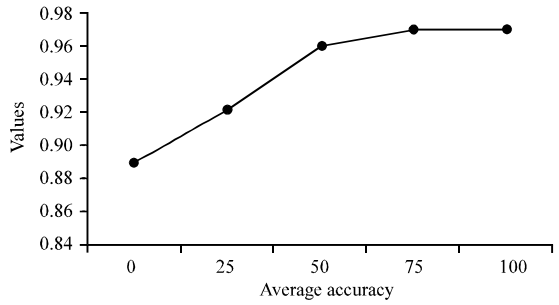

Fig. 10: Accuracy of prosposed methods based on the amount of received feedback

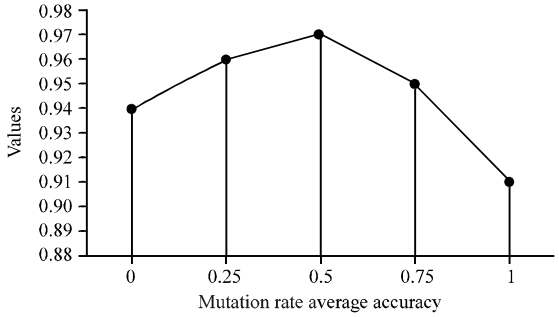

Fig. 11: Impact of mutation on the accuracy of proposed method

increasing accuracy is not completely linear by growing number of detection modules; it means that using a various number of detection modules will not give us one hundred percent accuracy. By the way based on the promising results of this evaluation we can say by increasing number of detection modules we could receive better performance and accuracy.

In this evaluation, our proposed method have examined through different scenarios of feedback (Paul *et al*., 2016) chance. As it has shown in Table 5 and Fig. 10, increasing the feedback chance will increase the accuracy of our method. This is due to it is more probable for our algorithm to modify and improve the performance of the detection modules. Receiving more feedback will let

Table 4: Accuracy of detection modules

No. of detection modules	Averageaccuracy
5	93.6
10	94.2
25	96.4
50	97.7
75	97.9

Table 5: Accuracy of proposed method based on amount of received feedback

Feedback chance (%)	Averageaccuracy
0	0.89
25	0.92
50	0.97
75	0.96
100	0.97

Table 6: Impact of mutation on the accuracy of proposed method

Mutation rate	Average accuracy
0.00	0.94
0.25	0.96
0.50	0.97
0.75	0.95
1.00	0.91

Table 7: Impact of amount of instances on respond time of proposed method

No. of samples	Total runtime (sec)
100	3.52
1,000	3.66
10,000	5.54
100,000	17.80
1,000,000	136.05

the algorithm to replace the structure of those detection modules which are not useful in detecting and recognizing packets properly or reduce their weight.

The mutation method in adaption module provides a better structure for detection modules with random changes in the structure of some modules. As it has shown in Table 6 and Fig. 11, increasing the rate of mutation method will not help to receive better accuracy all of the time. It is because when lots of mutation happen in the algorithm, it is probable to wrongly replace the structure of efficient and beneficial detection modules with those that are weaker and inefficient and cause lower accuracy rate. As result of that Mutation rate should be assign as the appropriate amount which hands over a good accuracy.

The speed of detection is so important in IDS because of that we have examined the relation between the speed of executing algorithm and amount of incoming instances. As it has shown in Table 7 and Fig. 12 when we increase the number of instances the runtime will raise that it is because of the training of decision trees in detection module and modify them when it is necessary. In another word if we consider training time of detection modules as t this amount will be separated from the number of instances that will be examined by algorithm after training and diagnosis time of each new sample is linear.

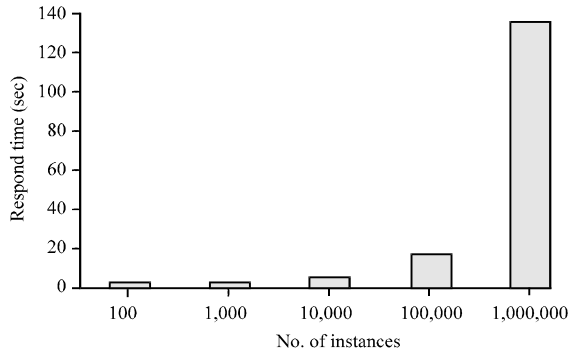

Fig. 12: Impact of amount of instances on respond time of porposed method

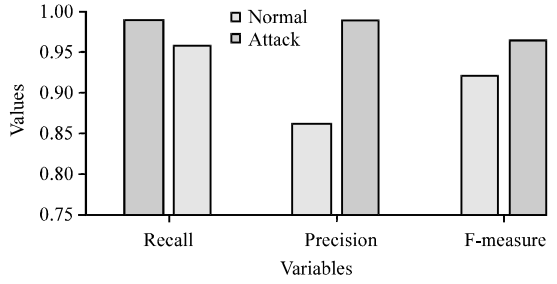

Fig. 13: F-measure amount for noraml category and instrusion category

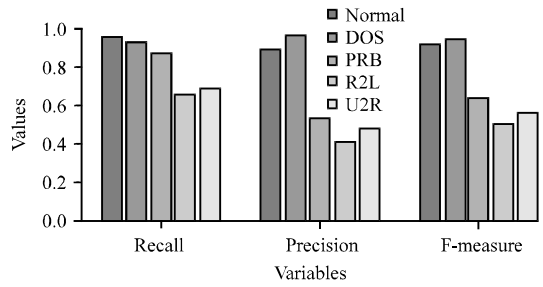

Fig. 14: F-measure, recall, precision for each type of attack

In the first step we consider 10% KDD dataset that contains 97278 normal instances and 396743 intrusion instances as a test dataset. In below we have shown confusion matrix of proposed algorithm in Table 8. As it has demonstrated, the quantity of normal instances that have recognized wrongly as an intrusion (false alarm) is 893 instance or 0.09%. Based on Table 8, the F-measure amount for normal category and intrusion category of data is in Table 8 and Fig. 13. In order to evaluate the performance of proposed method in detecting the various type of attacks, we have put instances of 10% KDD dataset in to 5 main categories. Table 9 has shown confusion matrix result of this evaluation. Based on the

Table 8: Confusion matrix of proposed algorithm

Variables	Normal	Attack	Recall	Precision	F-measure
Normal	96,385	893	0.99	0.86	0.92
Attack	15,097	381,646	0.96	0.99	0.97

Table 9: Confusion matrix in various type of attack

Variables	Normal	DOS	PRB	R2L	U2R	Recall	Precision	F-measure
Normal	95,983	1,058	149	68	20	0.98	0.91	0.94
DOS	9,037	378,652	2845	908	16	0.96	0.99	0.97
PRB	116	240	3,658	93	0	0.89	0.54	0.67
R2L	77	202	63	784	0	0.69	0.42	0.52
U2R	3	8	0	4	37	0.71	0.50	0.58

Table 10: Comparison of proposed method with two other algorithm

Variables	Proposed method	Online Naive Bayes	Online K-NN
Accuracy	0.97	0.93	0.96
Total runtime (sec)	7.09	768.28	365.42×105

results that we have received, the amount of F-measure could be calculated for each class as it has shown in Table 9 and Fig. 14. We have compared our proposed algorithm with the algorithms that has proposed by Gumus *et al.* (2014). In their method, they have employed online Naive Bayes and online K-NN algorithms with using KDD 99 datasets. In Table 10, the best accuracy of their algorithms have compared with our proposed algorithm. The respond time of algorithms has calculated based on 20 thousand instances and their accuracy after training.

As it has shown in Table 10, learning time of proposed method is much faster than other 2 algorithms and it is due to distributed structure of detection modules in our job which are a group of classifiers that are working distributed and each of them has trained by a different portion of data. This point will decrease training time considerably. In addition, we can notice that accuracy of proposed method is much better than 2 other algorithms. Based on promising results, our method has better respond time in comparison with online K-NN algorithm and it has better accuracy as compared to online Naive Bayes algorithm.

CONCLUSION

In this study, the proposed model of an adaptive network based IDS which has a modular structure and its detection modules are based on C4.5 decision tree is explained. The experimental result shows that our proposed method has better accuracy and respond time in comparison with two other algorithms. We believe that this improvement is due to the fact that our approach has a modular structure and its detection modules work distributed and also adaption module enables our proposed method to change and modify itself to cope with the new intrusion. In future work, we try to work for online capturing of packets from the network and will use that data as a test dataset which is tested against this method.

ACKNOWLEDGEMENT

This research was supported partially by the High Impact Fund of National University of Malaysia (Malaysia; Grant code: DIP-2014-037)

REFERENCES

Desale, K.S., C.N. Kumathekar and A.P. Chavan, 2015. Efficient intrusion detection system using stream data mining classification technique. Proceedings of the 2015 International Conference on Computing Communication Control and Automation (ICCUBEA), February 26-27, 2015, IEEE, New York, USA., ISBN:978-1-4799-6892-3, pp: 469-473.

Ghorbani, A.A., W. Lu and M. Tavallaee, 2009. Network Intrusion Detection and Prevention: Concepts and Techniques. Vol. 47, Springer, Berlin, Germany, ISBN:978-0-387-88770-8, Pages: 211.

Gong, W., W. Fu and L. Cai, 2010. A neural network based intrusion detection data fusion model. Proceedings of the 2010 Third International Joint Conference on Computational Science and Optimization (CSO), Vol. 2, May 28-31, 2010, IEEE, Huangshan, China, ISBN:978-1-4244-6812-6, pp: 410-414.

Gumus, F., C.O. Sakar, Z. Erdem and O. Kursun, 2014. Online Naive Bayes classification for network intrusion detection. Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), August 17-20, 2014, IEEE, Beijing, China, ISBN:978-1-4799-5878-8, pp: 670-674.

Koc, L., T.A. Mazzuchi and S. Sarkani, 2012. A network intrusion detection system based on a hidden Naive Bayes multiclass classifier. Expert Syst. Appl., 39: 13492-13500.

Kumar, S. and A. Yadav, 2014. Increasing performance of intrusion detection system using neural network. Proceedings of the 2014 International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), May 8-10, 2014, IEEE, Ramanathapuram, India, ISBN:978-1-4799-3915-2, pp: 546-550.

Lin, W.C., S.W. Ke and C.F. Tsai, 2015. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. Knowl. Based Syst., 78: 13-21.

Lippmann, R.P., D.J. Fried, I. Graf, J.W. Haines and K.R. Kendall *et al*., 2000. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), January 25-27, 2000, IEEE Computer Society Press, Los Alamitos, CA, pp: 12-26.

Malialis, K. and D. Kudenko, 2015. Distributed response to network intrusions using multiagent reinforcement learning. Eng. Appl. Artif. Intell., 41: 270-284.

Pal, D. and A. Parashar, 2014. Improved genetic algorithm for intrusion detection system. Proceedings of the 2014 International Conference on Computational Intelligence and Communication Networks (CICN), November 14-16, 2014, IEEE, Bhopal, India, ISBN:978-1-4799-6930-2, pp: 835-839.

Paul, S., T. Makkar and K. Chandrasekaran, 2016. Extended Game Theoretic Dirichlet Based Collaborative Intrusion Detection Systems. In: Computational Intelligence, Cyber Security and Computational Models, Senthilkumar, M., V. Ramasamy, S. Sheen, C. Veeramani and A. Bonato *et al*., (Eds.). Springer, Singapore, pp: 335-348.

Rahmatian, M., H. Kooti, I.G. Harris and E. Bozorgzadeh, 2012. Hardware-assisted detection of malicious software in embedded systems. IEEE. Embedded Syst. Lett., 4: 94-97.

Sahu, S.K. and S.K. Jena, 2016. A multiclass SVM classification approach for intrusion detection. Proceedings of the International Conference on Distributed Computing and Internet Technology, January 15-18, 2016, Springer, Bhubaneswar, India, pp: 175-181.

Shanmugavadivu, R. and N. Nagarajan, 2011. Network intrusion detection system using fuzzy logic. Indian J. Comput. Sci. Eng., 2: 101-111.

Soni, M., M. Ahirwa and S. Agrawal, 2015. A survey on intrusion detection techniques in MANET. Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN), December 12-14, 2015, IEEE, Jabalpur, India, ISBN:978-1-5090-0077-7, pp: 1027-1032.

Wang, G., J. Hao, J. Ma and L. Huang, 2010. A new approach to intrusion detection using artificial neural networks and fuzzy clustering. Expert Syst. Applic., 37: 6225-6232.

Wang, J., Q. Yang and D. Ren, 2009. An intrusion detection algorithm based on decision tree technology. Proceedings of the Asia-Pacific Conference on Information Processing APCIP-2009, Vol. 2, July 18-19, 2009, IEEE, Shenzhen, China, ISBN:978-0-7695-3699-6, pp: 333-335.

Wang, Y., 2008. Statistical Techniques for Network Security: Modern Statistically-Based Intrusion Detection and Protection. Information Science Reference, New York, USA., ISBN:978-1-59904-708-9, Pages: 395.