I. J. Computer Network and Information Security, 2012, 1, 1-16
Published Online February 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2012.01.01

M_2KMIX: Identifying the Type of High Rate Flooding Attacks using a Mixture of Expert Systems

Arun Raj Kumar, P. and S. Selvakumar
Department of Computer Science and Engineering
National Institute of Technology (NIT), Tiruchirappalli – 620015. Tamil Nadu State, INDIA
E-mail address: {park, ssk}@nitt.edu

***Abstract*—High rate flooding attacks such as SYN flood, UDP flood, and HTTP flood have been posing a perilous threat to Web servers, DNS servers, Mail servers, VoIP servers, etc. These high rate flooding attacks deplete the limited capacity of the server resources. Hence, there is a need for the protection of these critical resources from high rate flooding attacks. Existing detection techniques used in Firewalls, IPS, IDS, etc., fail to identify the illegitimate traffic due to its self-similarity nature of legitimate traffic and suffer from low detection accuracy and high false alarms. Also, very few in the literature have focused on identifying the type of attack. *This paper focuses on the identification of type of high rate flooding attack with High detection accuracy and fewer false alarms.* The attack type identification is achieved by training the classifiers with different feature subsets. Therefore, each trained classifier is an expert in different feature space. High detection accuracy is achieved by creating a mixture of expert classifiers and the ensemble output decisions are identified by our proposed Preferential Agreement (PA) rule. Our proposed classification algorithm, M_2KMix (mixture of two Multi Layer Perceptron and one K-Nearest Neighbor models) differs from the existing solutions in feature selection, error cost reduction, and attack type identification. M_2KMix was trained and tested with our own SSE Lab 2011 dataset and CAIDA dataset. Detection accuracy and False Alarms are the two metrics used to analyze the performance of the proposed M_2KMix algorithm with the existing output combination methods such as mean, maximum, minimum, and product. From the simulation results, it is evident that M_2KMix algorithm achieves high detection accuracy (97.8%) with fewer false alarms than the existing output combination methods. M_2KMix identifies three types of flooding attacks, viz., the SYN Flood, UDP flood, and HTTP Flood, effectively with detection accuracy of 100%, 93.75%, and 97.5%, respectively.**

Index Terms - High Rate Flooding, Neural Networks, Machine Learning, Ensemble of Classifiers.

I. Introduction

The first massive high rate flooding attack happened at commercial organizations such as CNN, yahoo, etc., in the year 2000. Since then, these attacks are considered to be pernicious threat to the Internet Services. The attack rate has increased to 100 Gbps as is evident from the Arbor Networks Survey Report [1]. Majority of the attack traffic reported in [2] are TCP SYN flood (33.8%), UDP flood (32.5%), and DNS flood (9.5%). The destination server alone is not affected by these high rate flooding attacks. Also, other infrastructure elements such as routers, switches, etc., suffer collateral damage along the path of an attack. Distributed Denial of Service (DDoS) attack is broadly classified into resource depletion and bandwidth depletion attack [3]. In resource depletion attack, attackers attempt to tie up the critical resources (memory and processor) making the victim unable to process the legitimate service. In bandwidth depletion attack, attackers flood the victim with large traffic that prevents the legitimate traffic and amplifies the attack by sending messages to broadcast IP address. A flood attack involves the zombies sending large volumes of traffic (UDP or ICMP) to a victim system, to congest the victim system's bandwidth.

High rate flooding attacks such as HTTP flood, SYN flood, and UDP flood have been posing a serious threat to Web servers, DNS servers, Mail servers, VoIP servers, etc. Though the critical services are well protected by Firewall and Antivirus software, attackers exploit the application vulnerabilities to target the server. Also, Firewall itself falls victim to such high rate flooding attacks leading to poor performance in offering firewall services and consequently bringing down the performance/services of a network. Today, attackers find it tedious to perform network layer attacks as the current defense mechanisms are efficient enough to filter or rate limit the attack traffic. As new applications are evolving in the internet on a daily basis, the vulnerabilities are also increasing leading to application layer based attacks. Attackers use the vulnerabilities present in the new applications and exploit the server completely. Nowadays, by disabling IP broadcasts in Firewalls, host computers prevent amplification in ICMP Flood and Smurf attacks. Though some of the DDoS attacks and attack tools have become obsolete now and some of the old attacks are detected by existing defense mechanisms, the attacker still changes the attack techniques using new attack tools, varies the traffic pattern, and deluges the traffic to shutdown the target server. This is a challenging issue as existing detection mechanisms are not capable of detecting zero-day attacks. Also, Source and Destination IP addresses are spoofed. Therefore, it is tedious to find the true origin of the attack.

Misuse detection methods identify packets that match a known pattern or signature. However, these methods fail to detect unknown anomalies. Anomaly detection methods are used to identify the traffic patterns that

deviate from the modeled normal traffic behavior. Anomaly detection methods [4], [5] are based on statistical measures of normal traffic with audit records and network level data. False positives are more in Anomaly detection methods. Rule based systems [6], [7], [8] consist of a set of rules that describe normal behavior. These systems raise an alarm whenever a rule is broken. The challenge in these systems is how efficiently rules are generated. Prior knowledge of the attacks and its behavior are required for rule generation. These systems may not detect new attacks. *Hence, existing detection algorithms for high rate flooding attacks in the literature suffer from more false positives and less detection accuracy.* Cost of misclassifications varies depending on the application. In intrusion detection, cost of unauthorized user attempting to exploit the root access is more expensive than the user doing scanning activities such as port scan. Hence, cost of false alarms is important in intrusion detection systems. To minimize the false alarms, Neyman Pearson approach [9] is used in the post training process to select the classifiers with high detection accuracy. Existing detection mechanisms have focused mostly in the classification of network traffic into normal and attack classes. *But, in this paper, identification of type of flooding attacks has been focused, besides classification of network traffic.* Therefore, it is useful for administrators to take appropriate action on the server in less time. For example, if attack type is identified as HTTP flood, it means Webserver with port 80 or 443 is under attack. Therefore, administrator filters or rate limits the malicious requests passing through this port. From source end, launching such attacks is done by different techniques, viz., sending SMS, injecting flood scripts in browser, attaching malicious URL shortening link in facebook, etc. But, whatever may be the techniques and technologies from source end, the requests finally arrive at the target server. Hence, *a victim or target based detection solution is proposed in this paper.*

The contributions of this paper include the following:

- Implementation of proposed classification algorithm, M_2KMix
- Identification of High Rate flooding attack (SYN, UDP, and HTTP flood) by novel Preferential Agreement (PA) rule
- Selection of Statistical Features from the lab generated/created Dataset and CAIDA dataset
- Normal and Attack Traffic Generation from our Smart and Secure Environment Project Lab (2 Mbps MPLS VPN Cloud) similar to Planet lab like environment
- A classification accuracy of (M_2KMix+PA)
  - 97.8% when testing on the SSELab Dataset
    - 100% accuracy in identifying the SYN Flood Traffic
    - 93.75% accuracy in identifying the UDP Flood Traffic
    - 97.5% accuracy in identifying the HTTP Flood Traffic

The rest of the paper is organized as follows: Existing Feature Extraction methods, Soft Computing techniques for intrusion detection, and Existing ensemble methods are discussed in Section II. Proposed M_2KMix algorithm is elucidated in Section III. The experiments conducted and their results are discussed in Section IV. Section V concludes the paper.

II. Related works

Since 2000, the three publicly available datasets for intrusion detection are the DARPA/Lincoln Labs packet traces, the KDD Cup dataset [10] derived from DARPA traces, and Cooperative Association for Internet Data Analysis (CAIDA) [11], [12]. There were criticisms on DARPA data [13]. The KDD Cup dataset was created by processing the tcpdump portions of the 1998 DARPA Intrusion Detection System (IDS) Evaluation dataset. CAIDA dataset is available by user request. The reason for the lack of public DDoS attack datasets is that the deep inspection of network traffic reveals highly sensitive information such as confidential communications, user's network access patterns, etc. Any breach of such information can be disastrous for both the organization and service providers. Solution is to create synthetic data through simulation, emulation, or anonymization of shared real time datasets. Thus, we have generated our own dataset in a distributed testbed using new attack tools and used the same for the experiments on classification and identification.

Constructing the features is integrated into the preprocessing stage which includes standardization, normalization, etc. Feature Selection is divided into Filter methods and wrapper methods [14]. In filter methods, selection is based on distance and information measures in the feature space. In wrapper methods, selection is based on classifier accuracy. In this paper, wrapper method has been used for the feature selection. If too many features exist, the learned hypothesis may fit the training set very well, but fails to generalize the new examples. In [15], 248 features were given and 1 feature was used to describe the class (normal or attack). Computation of all the 248 features [16] took approximately two days on a dedicated System Area Network. More number of features led to better accuracy. But, computation of more number of features in real time causes more overhead and time consuming. So, less number of features is suitable for better pattern classification in real time. *Hence, six features are selected for identification of attack traffic in this paper.*

Nowadays, it is tedious to defend the high rate flooding attacks as the attack traffic looks similar to legitimate network traffic that can easily circumvent the existing defense mechanisms. Hence, there is a need of deep packet inspection. The existing defense devices include intrusion detection systems (IDSs), anti-virus systems, anti-spam systems, and Web filtering systems. Identification includes checking for attacking signatures

or discovering anomalous behaviors. Signature based detection may miss unknown attacks (false negatives), but anomaly analysis may lead to false positives if normal traffic behaves unusually. So, there is a trade-off between false positives and false negatives. Signature-based schemes could lead to more false negatives, while behavior-based schemes have potentially higher false positives. The latter is more serious than the former. The mostly used anomaly detection algorithms in the literature are Machine Learning algorithms due to its generalized learning ability.

Machine Learning is mostly focused on finding relationships in data and analyzing the process for extracting such relations. Machine learning paradigms are classified as Supervised Learning (SL) and Unsupervised Learning (UL). In SL, the algorithm attempts to learn some function with given input vector and actual output. In UL, the algorithm attempts to learn only with given input vector by identifying relationships among data. Several Machine Learning (ML) algorithms have been proposed [9], [17], [18], [19], [20], [21], [22] for DDoS attack detection. These ML algorithms applied to DDoS attack detection have not considered identifying the type of flooding attack. Some algorithms [23], [24] have been used to classify the instances into multiple classes such as Normal, Probe, DoS, U2R, and R2L. But, the datasets (KDD Cup'99) used in these algorithms are obsolete. In [23], the threshold values are used in the decision making process to identify the different classes. However, more number of features such as intrinsic, content, and traffic features have been used in [23], [24]. Solutions proposed for multiclass problem may not be applicable to binary classification problem. Hence, none of the methods have focused explicitly in identifying the type of high rate flooding attack. In this paper, the existing Machine Learning algorithms, viz., Multi Layer Perceptron [25], Support Vector Machine [26], K-Nearest Neighbor [27], Decision tree (C4.5) [28], and Naïve Bayes [29] have been used for the experiments.

Single classifier makes error on different training samples. So, by creating an ensemble of classifiers and combining their outputs, the total error can be reduced and the detection accuracy can be increased. In [30], [31], the potential of high-performance parallel processors is exploited to operate several anomaly detection algorithms in parallel. Hence, the output from each algorithm is normalized and then aggregated to produce a single anomaly metric. Bagging [32], Boosting [33], and AdaBoost [33] are the existing ensemble algorithms used for pattern classification. Decision boundaries of each classifier have to be uniquely different from others. To achieve this diversity, ensemble of classifiers can be constructed by manipulating training data, feature sets, and injecting randomness. Existing ensemble algorithms achieve classifier diversity by manipulating the training dataset over the same feature space. *In this paper, ensemble construction by manipulating feature sets was chosen, as it would correctly detect the deviations and identify the type of attack as each classifier is trained with the different dataset and an expert in local feature space*. In order to construct the ensemble by manipulating input feature sets, the input feature set is divided into smaller feature subsets and each classifier is trained with the smaller feature subsets.

Classifier combination is divided into two categories:

- Classifier selection, where each classifier is trained to become an expert in some local area of the total feature space.
- Classifier fusion, where all classifiers are trained over the same feature space.

In this paper, classifier selection has been used. Selection of output combination method depends on the nature of the input classifiers and the feature space. Classifier outputs can be combined by methods such as Maximum rule [34], Minimum rule (WMV) [34], Product rule [35], Mean [35], etc. By combining expert learning systems, the ensemble model accuracy is always improved comparing to single-model solutions. In [35], sum rule outperformed the other existing combination rules in classifying the handwritten digit recognition. Majority vote and Sum rule are the output combination rules used mostly in the literature for pattern classification and it has achieved high detection accuracy. If existing combination rules are used, one or two classifier decisions may be ignored. But, in this paper, each classifier decision is significant in the ensemble and hence cannot be ignored. Borda Count [34] also gives importance to each classifier decision, in particular it provides support to the non-winning class. But, it will be useful for multiclass problems. In case of binary classification problems, Borda Count is similar to Simple Majority Voting. Hence, Borda Count is not suitable for identifying the type of flooding attack. In this paper, it has been experimentally proved that none of the existing rules were able to classify the samples correctly and these existing rules produced less detection accuracy than the single classifier model. *Hence, a novel ensemble output decision rule, Preferential Agreement, has been proposed in this paper.*

III. DESIGN OF PROPOSED M_2KMIX ALGORITHM

The proposed M_2KMix based Ensemble system consists of the following two stages:

- Preprocessing
- Classification

A. Preprocessing

Preprocessing refers to the process of extracting information about packets from network traffic for the construction of new statistical features. In preprocessing module as shown in Figure 1, different feature subsets are selected for the identification of different attacks. The input to the preprocessing module is the network traffic consisting of both labeled normal and labeled attack dataset. The output of this module is normalized dataset. The preprocessing module consists of feature selection, feature values extraction, and normalization. This stage prepares the data for training the different Machine Learning algorithms during training phase.

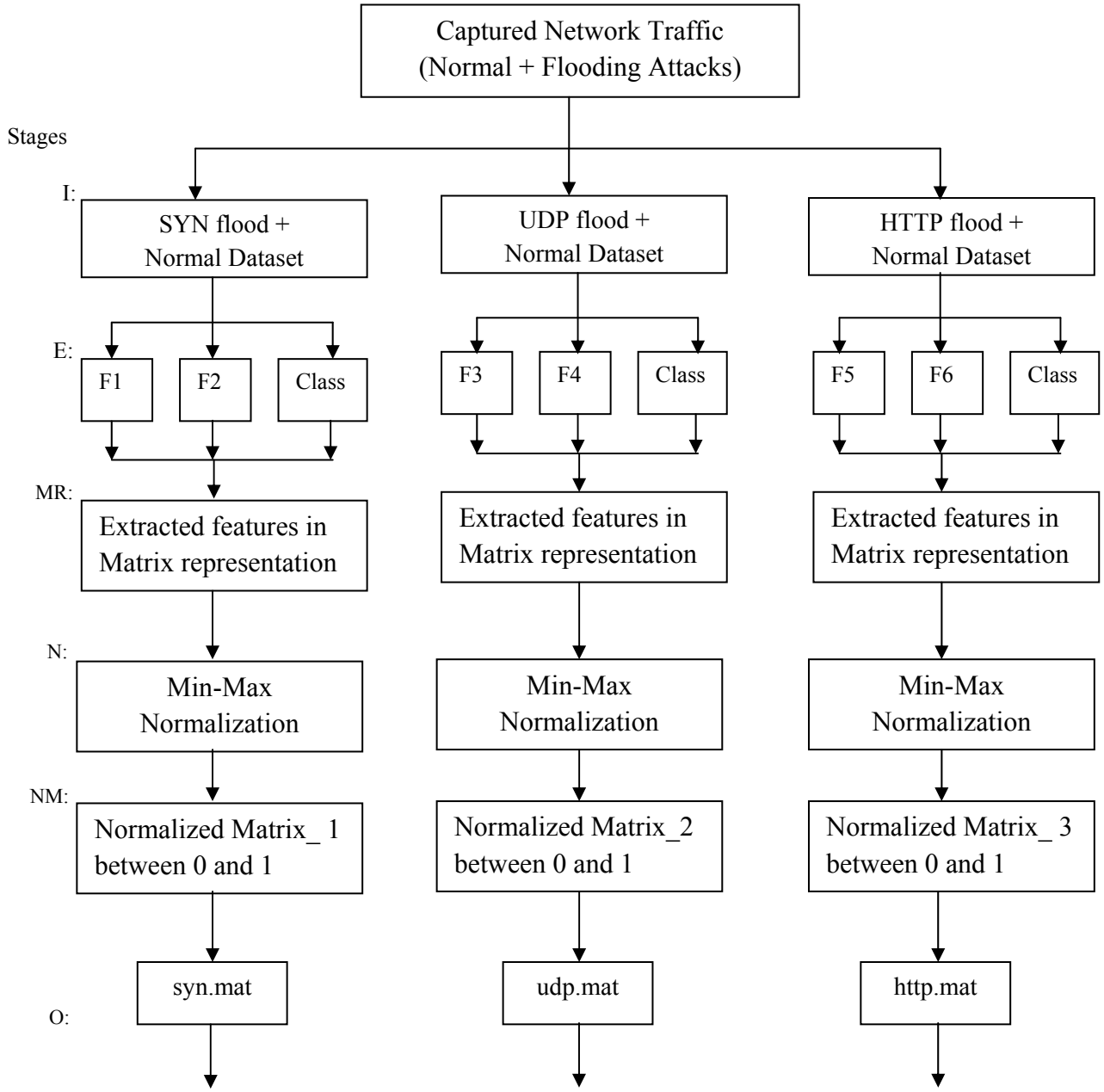

Figure 1. Preprocessing module for training phase

The preprocessing steps are explained as follows:

- The captured network traffic consisting of labeled normal and labeled flooding attacks is split into different attack type datasets such as SYN flood, UDP flood, and HTTP flood as shown in Figure 1, Stage I. The splitting of datasets into three subsets has been implemented in this paper as the scope of attack type classification is restricted to SYN flood, UDP flood, and HTTP flood.
- The split datasets are given as an input to the feature extraction module to extract the feature values, as shown in Figure 1, Stage E. These features quantify the behavioral characteristics of a connection in terms of ratio and number of various data items with respect to time.
- The features present in Table I detect TCP SYN flood, TCP SYN+ACK flood, TCP Spoofed SYN flood, TCP ACK Flood, HTTP flood, HTTPS flood, and UDP flood attack.
- Extracted feature values from each dataset are represented in matrix form consisting of first two columns as feature values and the last column as class label, Figure 1, Stage MR. From Figure 1, F1 and F2 are extracted from SYN flood dataset. Similarly, F3, F4, & F5, F6 are extracted from UDP Flood and HTTP flood dataset respectively.
- The matrix for each dataset consists of both discrete and continuous values and some redundant information. Hence, there is a need to

normalize the matrix values between 0 and 1 using min-max normalization.

- Normalization is a process of ensuring that each attribute value in a database is suitable for further querying, which is free from certain undesirable characteristics and eliminates the effect of scale difference.
- The extracted features as shown in Table I and its values are input to the normalization module as shown in Figure 1, Stage N.
- The feature values are scaled to the range [0, 1] using (1), where 'i (t)' denotes the value of the feature, 'min (i)' denotes the minimum value, and 'max (i)' denotes the maximum value. Thus, data available for the classifier are real numbers between 0 and 1, Stage NM.

$$i_{norm}(t) = \frac{i(t) - \min(i)}{\max(i) - \min(i)} \quad (1)$$

- These normalized three matrix files are saved as syn.mat, udp.mat, and http.mat as shown in Figure 1 and given as input to the machine learning algorithm. Thus, all the .mat files consist of normalized values between 0 and 1, Stage O.

B. Design of Proposed Classification Algorithm - M_2KMix

Figure 2 shows the training module of the M_2KMix algorithm. The output, syn.mat file of the preprocessing module consisting of normalized feature values is given as input to the classification algorithms, viz., C_1, C_2, …, C_5, where C1, … C5 represents Machine Learning algorithms such as Decision tree (C4.5) [28] (C1), Support Vector Machine [26] (C2), Multi Layer Perceptron [25] (C3), K-Nearest Neighbor [27] (C4), and Naïve Bayes [29] (C5). Similarly, udp.mat and http.mat files are also given as input to the classification algorithms, viz., C_1, C_2, …, C_5. In this paper, the attack type identification is restricted to the three flooding attacks such as SYN flood, UDP flood, and HTTP flood. Proposed classification algorithm, M2KMix, is shown in Figure 3.

B.1 TRAINING

First dataset (DS1) consists of TCP SYN, TCP SYN+ACK, TCP ACK attack traffic, and normal traffic. Second dataset (DS2) consists of UDP flood traffic and normal traffic. Third dataset (DS3) consists of HTTP and HTTPS attack traffic and normal traffic. Samples are chosen randomly from DS1, the first algorithm (C1) is trained, and the hypothesis h_m is obtained. The error of h_m is computed using equation (2) as shown in Figure 3.

$$\varepsilon_t = \sum_1^n [h_m(x_i) \neq y_i] \quad (2)$$

If the computed error is greater than false alarm threshold value, then the hypothesis is dropped. If the computed error is less than false alarm threshold value, the normalized error is calculated using (3).

$$\beta_m = \varepsilon_t / 1 - \varepsilon_t \qquad 0 < \beta_t < 1 \quad (3)$$

TABLE I. LIST OF FEATURES

F. No.	Feature Description
F1.	Ratio of the number of incoming SYN+ACK packets and during specified time window
F2.	Ratio of the number of outgoing SYN+ACK packets and ACK packets during specified time window
F3.	Number of UDP echo packets to a specified port during specified time window
F4.	Number of UDP echo packets to port 53 during specified time window
F5.	Number of HTTP request packets to port 443 during specified time window
F6.	Number of HTTP request packets to port 80 during specified time window

As mentioned in the Neyman Pearson Statistical approach [36] (NeP), the total number of misclassified samples from both the classes for the specific algorithm is calculated using (4).

$$R_{jm}(h) = 1/n_j \sum I \{h_m(i) \neq j\} \quad (4)$$

All learning algorithm's True Negative (TN), True Positive (TP), False Positive (FP), False Negative (FN), and normalized error for this DS_1 are computed. Similarly, the above training steps are carried out for the other four algorithms for the same dataset DS_1. Then, using equation (5), optimum threshold is decided from the given set of classifiers based on minimum false alarms.

$$h_m^* = \arg\min \{R_{jm}(h): h \in H_0\} \quad (5)$$

The classifier that gives the minimum false alarms and high detection accuracy is chosen as the best learning algorithm, C_S, for the detection of SYN flood. For the remaining datasets (DS2 and DS3), the best learning algorithms, C_U and C_H, are chosen in a similar manner as shown in Figure 2. Hence, the output of the training phase yields the best learning algorithms, C_S, C_U, and C_H for the three types of attacks considered.

B.2 TESTING

The Preprocessing stages, I, E, MR, N, NM, and O in Figure 4 perform the same function as in Figure 1. The preprocessing module for testing phase prepares the data for attack type identification proposed using Preferential Agreement (PA) rule. Proposed PA rule is shown in Figure 5. The prepared data (new traffic), represented in matrix form consisting of six normalized feature values, are given as input to the set of classifiers (C_S, C_U, C_H) as shown in Figure 6. First two feature (F1 and F2) values are passed as input to C_S. Next two feature (F3 and F4) values are passed as input to C_U. The remaining two (F5 and F6) feature values are passed as input to C_H. C_S in the ensemble identifies the test instance as N (Normal) or S (SYN attack).

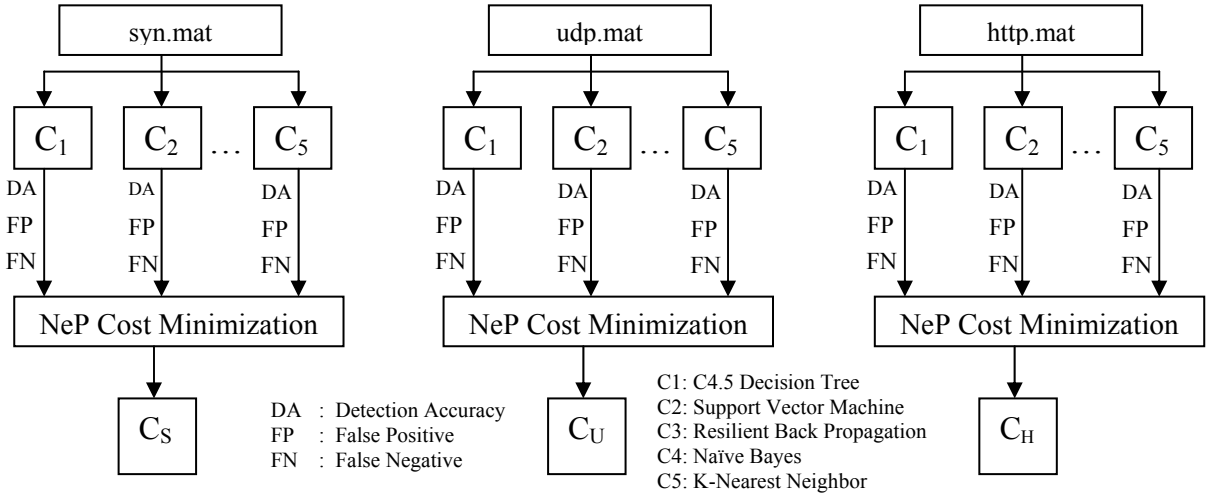

Figure 2. Training Module

Similarly, C_U in the ensemble identifies the test instance as N or U (UDP attack) and C_H in the ensemble identifies the test instance as N or H (HTTP attack). The output of the PA rule be N, S, U, H, SU, UH, SH, or SUH as shown in Figure 6. For example, a testing data consists of 'T' instances as shown in Figure 5. Each instance consists of six normalized feature values and given as input to the Ensemble. First instance in the matrix is passed as input to the ensemble. According to the PA rule, if all the classifiers in the ensemble output 'N' as their decision, then the final classification decision is the normal class. Similarly, if all the classifiers output S, U, or H, then the final classification decision is S, U, or H. If one or two classifiers output 'N' and the other one classifies as its attack, then the final classification decision is the attack type, viz., S, U, or H. The performance of the PA rule is measured by the metrics detection accuracy and false alarms for 'T' instances consisting of normal and flooding attacks.

IV. Experimental Results

Eight Institutes/Universities (sites) situated in different geographical locations are working collaboratively on a Smart and Secure Environment (SSE) project as shown in Figure 7. Our Institute (Site 4) is one among the sites connected through 2 Mbps Multi Protocol Label Switching (MPLS) Virtual Private Network (VPN) cloud. Each site maintains web, mail, DNS, and proxy servers. Attacks such as SYN flood, SYN+ACK flood, ACK flood, UDP flood (DNS Cache Poisoning), and HTTP flood were generated in SSE Testbed. Our proposed detection module, M_2KMix, analyzes the network traffic and classifies whether it is normal or attack. Proposed PA rule identifies the type of attack. The SSE environment that has been used for generating network traffic is similar to Planet Lab [37]. Inside each site, the link bandwidth is 1 Gbps. Between the core routers (CE) and edge routers (PE), the link bandwidth is 2 Mbps which is a dedicated MPLS VPN network.

Input:
- Training Data 'DS_i' of size 'T' with correct labels $y_j \in \Omega = \{ y_1, y_2)$
- Number of Learning Algorithms (M)

Initialize:
- $\mu = 0.5$ // False Alarm Threshold
- L = 2 // Number of classes
- k = 3 // Number of Flooding attacks

Training:

For each DS_i, where i = 1….. k

1. Choose samples from DS_i

For m = 1… M // Machine Learning algorithms

2. Train 'm' with DS_i and obtain hypothesis h_m
3. Compute error of h_m: $\varepsilon_t = \sum_1^n [h_m(x_i) \neq y_i]$
4. If $\varepsilon_t > \mu$, then drop hypothesis

Else

Normalized error (β_t): $\beta_m = \varepsilon_t / 1 - \varepsilon_t$

$0 < \beta_t < 1$

m = m + 1;

5. Split DS_i into 2 Classes (Attack and Normal)

For j = 0 and 1

6. $n_j = \sum_{i=1}^{n} I\{Y_i = j\}$ // Total number of instances for each class
7. $R_{jm}(h) = 1/n_j \sum_{i=Y_i=j} I\{h_m(i) \neq j\}$

End

End

8. $h_m^* = \arg\min \{R_{jm}(h): h \in H_0\}$ // Selection of best Machine Learning algorithm
9. i = i + 1;

Add the classifier h_m^* to the Ensemble 'E'

End

Figure 3. Classification Algorithm

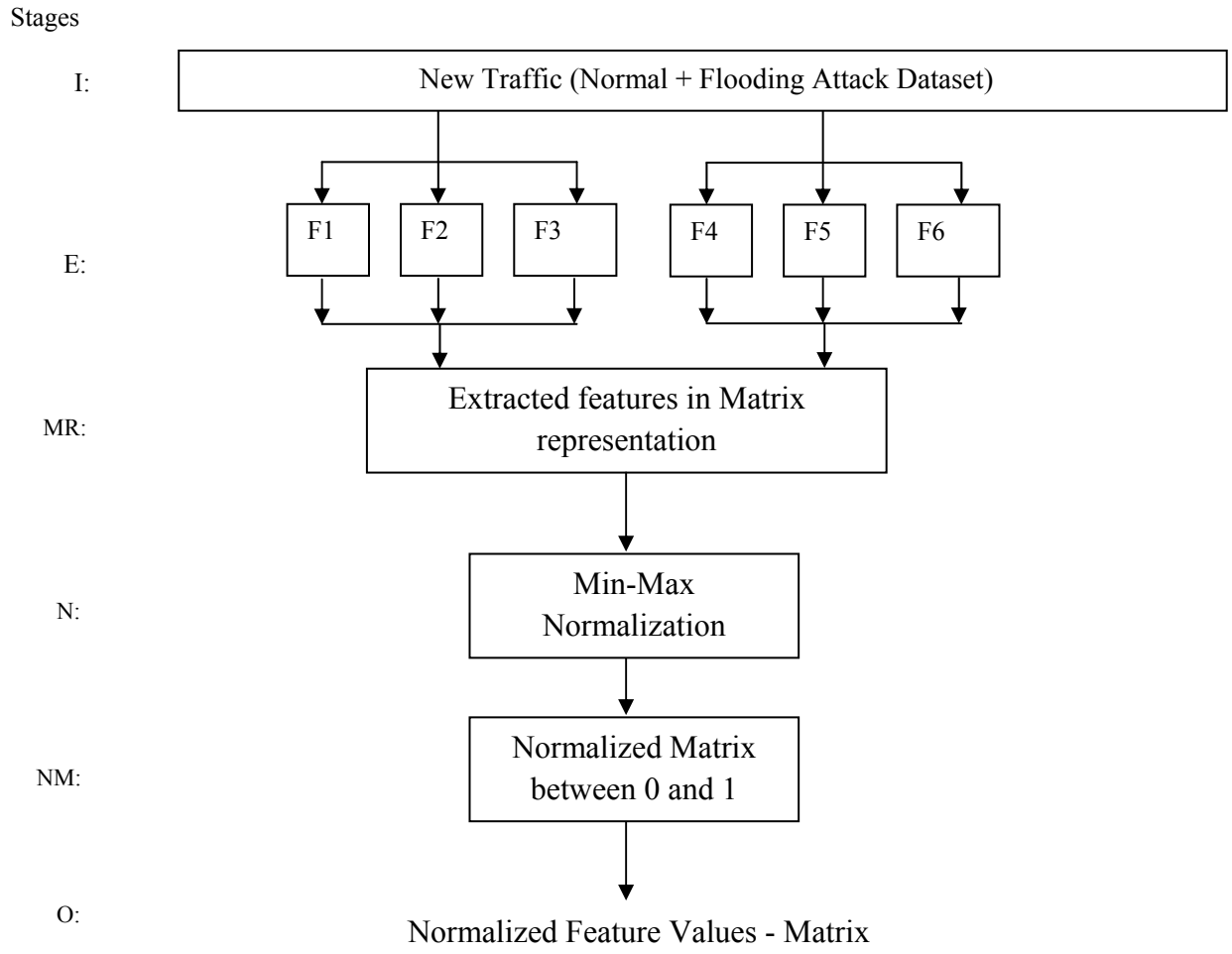

Figure 4. Preprocessing module for Testing Phase

Input:
- Testing Data 'TDS' of size 'TX6' // T instances of 6 features represented in matrix (F1,F2,F3,F4,F5,F6)
- C_S, C_U, C_H // Output of Training Phase

Testing:
1. Testing Data 'TDS' are passed as input to the Ensemble. F1 and F2 values are passed as input to C_S. F3 and F4 values are passed as input to C_U. F5 and F6 values are passed as input to C_H.
2. If all classifiers in the ensemble output 'N', the final classification decision is Normal class. Similarly, if all the classifiers output S, U, or H, then the final classification decision is S, U, or H.
3. If one or two classifiers output 'N' and the other one classifies as its attack, then the final classification decision is the attack type, viz., S, U, or H.

Output:
1. Calculate the detection accuracy and false alarms for Testing data

Figure 5. Preferential Agreement during Testing Phase

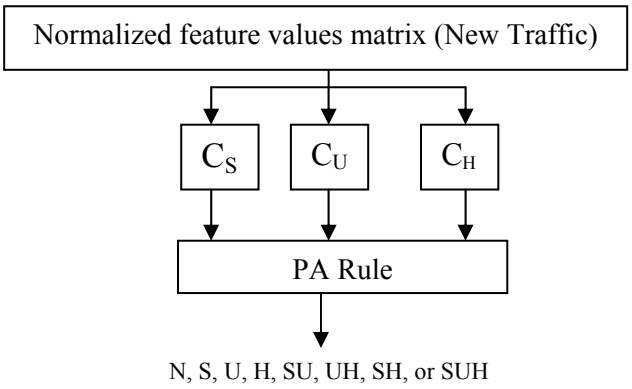

Figure 6. Testing Module

Setup seen for Site 1 barring E1 link in Figure 7 is the same in all the sites. Target server was configured in Site 4. There were three client machines in each site where one client machine was used for generating normal traffic and other two were used for generating attack traffic. In a single client machine, many virtual clients were created with spoofed and random IP addresses to generate attack traffic using Netwag tool [38].

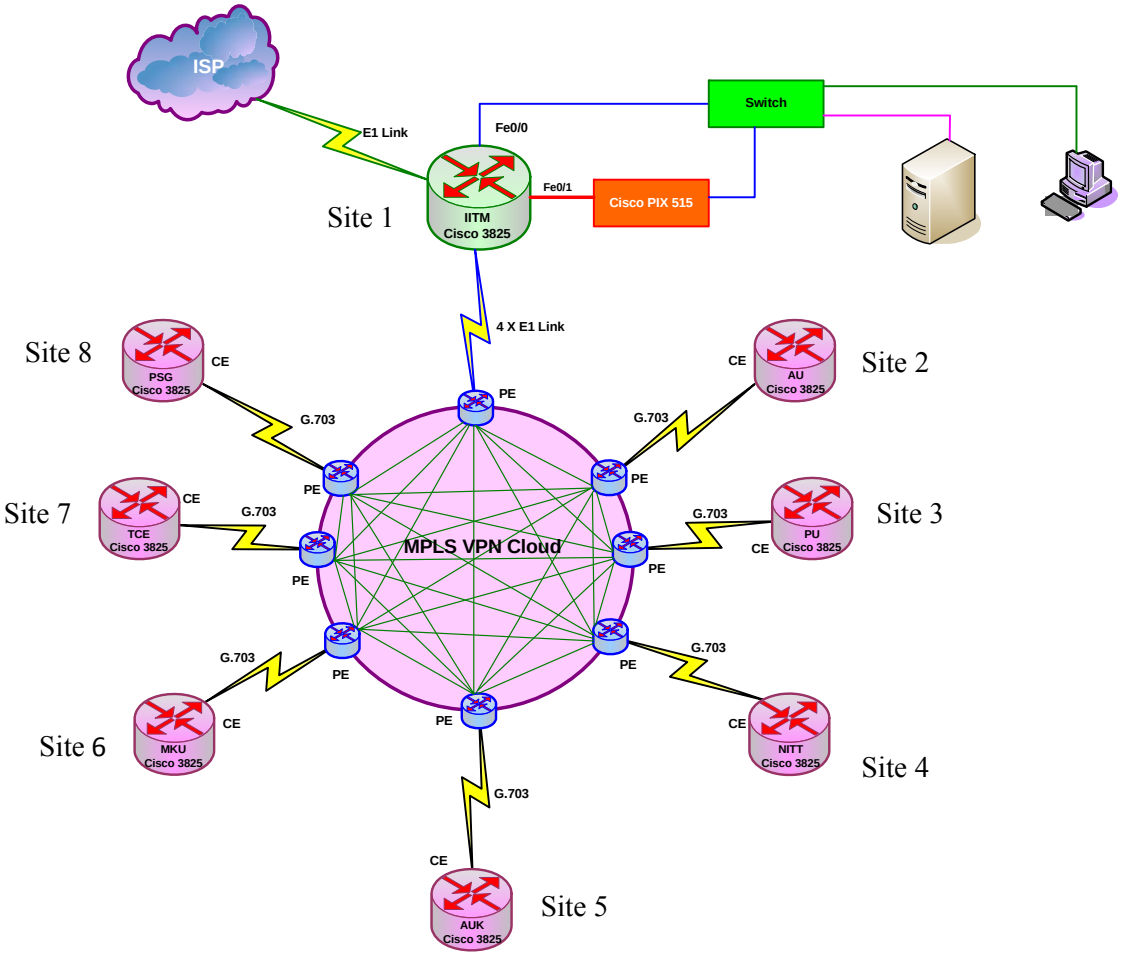

Figure 7. SSE Environment

HTTP traffic was generated using HTTPTrafficGen [39], a HTTP load generation tool, and using Low Orbit Ion cannon Tool [40]. Normal traffic consists of FTP access, webpage access, DNS requests and replies, e-mail access, video conferencing (UDP traffic), etc. Client machines were laptops and desktops. Attacks were launched on the web server present in Site 4. Traces were collected in Site 4 in tcpdump format. TCP port (80) traffic was generated by the clients in all sites including target server site. The traffic was collected during working hours. Simulations and the analysis of experimental data were performed with the use of MATLAB. Learning algorithms such as Multi Layer Perceptron-Resilient Back Propagation (MLP-RBP), Support Vector Machine (SVM), K-Nearest Neighbor, Naïve Bayes, and Decision Trees (C4.5) were used for training and testing the different attack datasets.

The objective is to find the best classifier for each attack type and to identify the attack type during test run. Hence, these learning algorithms were trained with different attack datasets. And for each dataset, the learning algorithm that produced high Detection Accuracy and fewer False Alarms was chosen as a best classifier by NeP and added to the ensemble. Further, the attack type is identified by our proposed novel Preferential Agreement rule. Confusion matrix is shown in Table II. For the simulation experiments conducted, detection accuracy was calculated using (6).

TABLE II. CONFUSION MATRIX

	Normal	**Attack**
Normal	TN	FP
Attack	FN	FN

Accuracy = (TP +TN) / (TP+FP+TN+FN) (6)

True Positive (TP) = Number of samples correctly predicted as attack class

False Positive (FP) = Number of samples incorrectly predicted as attack class

True Negative (TN) = Number of samples correctly predicted as normal class

False Negative (FN) = Number of samples incorrectly predicted as normal class

The number of training and testing instances used in the experiments is shown in Table III. The experiments conducted are as follows:

1. Exp. 1: Selection of Best Classifier for SYN Flood traffic
2. Exp. 2: Selection of Best Classifier for UDP Flood traffic
3. Exp. 3: Selection of Best Classifier for HTTP Flood traffic
4. Exp. 4: Comparison of PA rule with existing output combination methods

TABLE III. TRAINING AND TESTING INSTANCES USED FOR EXPERIMENTS 1, 2, AND 3

Exp. No.	Total No. of Conns. (30 minutes)	Dataset 1 (DS1)				Dataset 2 (DS2)				Dataset 3 (DS3)			
		Training Instances		Testing Instances		Training Instances		Testing Instances		Training Instances		Testing Instances	
		A	N	A	N	A	N	A	N	A	N	A	N
1.	368,648	175	125	130	170	-	-	-	-	-	-	-	-
2.	297,462	-	-	-	-	160	140	145	155	-	-	-	-
3.	356,312	-	-	-	-	-	-	-	-	135	165	140	160
A – Attack N- Normal													

A. Exp. 1: Selection of best classifier for SYN flood dataset (DS1)

Different attacks were generated in different time period. Total number of connections is listed in Table III for different experiments. From Table III, it can be seen that for experiment 1, both attack and normal traffic were generated for a period of 30 minutes where 368,648 packets were captured at a packet rate of 200 packets/second approximately. Similarly, the number of connections for Exp. 2 and Exp. 3 are shown in Table III. Five classification algorithms were tested and the results are tabulated in Table IX.

A set T of training data consists of 2 classes. A test based on feature F_i of the training data was split into subsets. **C4.5** selects the next attribute (based on a greedy principle) according to the information gain measure. C4.5 finds the normalized information gain for each feature. In the tree structure, each leaf node represents a classification decision and each internal node evaluates the corresponding attribute. Objective is to minimize the number of nodes that produced misclassifications. Confusion matrix values obtained during the experiments for C4.5 Decision tree are shown in Table IV. From the experiments, the detection accuracy achieved for the best decision tree classifier was seen to be 91.3%.

For problems that cannot be linearly separated in the input space, **SVM** performs a mapping from the input space to higher dimensional feature space through the use of a kernel function, where an optimally separating hyperplane is determined. In our experiments, the kernel function used for SVM was Radial Basis function (RBF) [25] which is a feed forward neural network. RBF networks are nonlinear hybrid networks consisting of single hidden layer processing elements. It uses Gaussian transfer function (which varies from 0 to 1 over the same range with a maximum at 0) than sigmoid function (which varies from -1 to +1 over a range of $-\infty$ to $+\infty$) which was used in MLP. The clusters used during each simulation were 8, 16, 32, and 64. From the simulation, it was observed that the model with 10 neurons in hidden layer, 16 clusters, and learning rate = 0.1 performs best in terms of detection accuracy. Confusion matrix values obtained during the experiments for SVM are shown in Table V. From the experiments, the detection accuracy achieved for SVM classifier was seen to be 86.7%.

A three layer **MLP** neural network was simulated. The number of neurons in the input layer was 2 (equal to number of features). Number of neurons in the output layer (sigmoid function) was 2 (normal and attack type). From the simulation experiments, it was found that the Resilient Back Propagation algorithm performed well in classifying the patterns. Initial weights were randomly assigned. Each epoch consisted of both normal and attack connection records. Simulation experiments were carried out to find the number of nodes in hidden layer by varying the values from 10 to 40 in increments of 10. The learning rate parameter (L_r) determines how fast the system should adapt to new instances. A higher value of L_r puts more weight on the instance and puts less weight on the existing instances. The performance is evaluated by varying the L_r from 0.1 to 1.0. Confusion matrix values obtained during experiments for MLP are shown in Table VI. From the experiments conducted, it was inferred that lower L_r values produced less false positive rate and less detection accuracy. Higher L_r values produced more false positive rates and stable detection accuracy. It was observed that the model with 10 neurons in hidden layer with learning rate = 0.1 produces detection accuracy of 95.3% with less false alarms.

In **Naïve Bayes** conditional independence and training data, the probabilities of a packet belonging to normal or attack class are computed. If the probability that a packet belongs to normal class is more than attack class, then the packet is classified as normal class. It distinguishes the normal and attack traffic packets by its conditional probabilities. For each feature, it estimated a class-conditional distribution using a histogram. Assuming independence of features, the per-class output was computed as a product of per-feature class conditional densities. Confusion matrix values obtained during the experiments for Naïve Bayes are shown in Table VII. From the experiments conducted, the detection accuracy was seen to be 90.3 %.

TABLE IV. CONFUSION MATRIX OF C4.5 DECISION TREE EXPERIMENT RESULTS FOR DS1

	Normal	**Attack**
Normal	158	12
Attack	14	116

TABLE V. CONFUSION MATRIX OF SVM EXPERIMENT RESULTS FOR DS1

	Normal	**Attack**
Normal	150	20
Attack	20	110

TABLE VI. CONFUSION MATRIX OF MLP EXPERIMENT RESULTS FOR DS1

	Normal	**Attack**
Normal	162	8
Attack	6	124

TABLE VII. CONFUSION MATRIX OF NAÏVE BAYES EXPERIMENT RESULTS FOR DS1

	Normal	**Attack**
Normal	154	16
Attack	13	117

TABLE VIII. CONFUSION MATRIX OF K-NN EXPERIMENT RESULTS FOR DS1

	Normal	**Attack**
Normal	162	8
Attack	12	118

K-Nearest Neighbor (K-NN) classifies a sample based on the majority vote of their nearest neighbors. The neighbors were identified with the representation of samples in multidimensional feature space. As it is binary classification problem, the K values vary in odd numbers ranging 11, 21, 31, and 41.

TABLE IX. SIMULATION RESULTS OF MACHINE LEARNING ALGORITHMS FOR DS1

Classification Algorithm	**False Positive**	**False Negative**	**Detection Accuracy**
C4.5 Decision tree	7.1	10.8	91.33
SVM	11.76	15.38	86.67
Multi layer Perceptron	4.71	4.6	95.3
Naïve Bayes	9.41	10.0	90.3
K-Nearest Neighbor	4.71	9.23	93.33

Simulation experiments were conducted and from the results obtained, it can be seen from Table IX, that more detection accuracy of 93.33% was obtained when k=11. Confusion matrix values obtained during the experiments for K-NN are shown in Table VIII. From the experiments conducted for different classifiers for DS 1, it is evident from Table IX that the best classifier for dataset DS1 is MLP with high detection accuracy of 95.3%. The best classifier obtained from the experiments is added to the ensemble.

B. Exp. 2: Selection of best classifier for UDP flood dataset (DS2)

The second experiment was conducted to select the classifier that performed best in identifying the UDP flood attacks. The experiments conducted in this Section are similar to the experiments conducted in Section IV.A with a difference in inputs. Inputs given to the classifiers are the normal traffic and UDP flood attack traffic dataset. Each classifier is trained and tested with the same inputs. The same five classification algorithms were tested and the results are tabulated in Table XV. **C4.5** finds the normalized information gain for each feature. Confusion matrix values obtained during the experiments for C4.5 are shown in Table X. From the experiments, the detection accuracy achieved for the best decision tree classifier was seen to be 88%.

In **SVM**, the clusters used during each simulation were 8, 16, 32, and 64. From the simulation, it was observed that the model with 10 neurons in hidden layer, 8 clusters, and learning rate = 0.1 performs best in terms of detection accuracy. Confusion matrix values obtained during the experiments for SVM are shown in Table XI. From the experiments, the detection accuracy achieved for SVM classifier was seen to be 90%.

A three layer **MLP** neural network was simulated. Resilient Back Propagation algorithm was used as a base classifier. Simulation experiments were carried out to find the number of nodes in hidden layer by varying the values from 10 to 40 in increments of 10. The performance of the learning rate is evaluated by varying the L_r from 0.1 to 1.0. Confusion matrix values obtained

TABLE X. CONFUSION MATRIX OF C4.5 DECISION TREE EXPERIMENT RESULTS FOR DS2

	Normal	**Attack**
Normal	128	17
Attack	19	136

TABLE XI. CONFUSION MATRIX OF SVM EXPERIMENT RESULTS FOR DS2

	Normal	**Attack**
Normal	129	16
Attack	14	141

TABLE XII. CONFUSION MATRIX OF MLP EXPERIMENT RESULTS FOR DS2

	Normal	**Attack**
Normal	136	9
Attack	11	144

TABLE XIII. CONFUSION MATRIX OF NAÏVE BAYES EXPERIMENT RESULTS FOR DS2

	Normal	**Attack**
Normal	133	12
Attack	15	140

during the experiments for MLP are shown in Table XII. It was observed that the model with 10 neurons in hidden layer with learning rate = 0.1 produces detection accuracy of 93.3% with less false alarms. Confusion matrix values obtained during the experiments for **Naïve Bayes** are shown in Table XIII. From the experiments conducted, the detection accuracy was seen to be 91%.

K-Nearest Neighbor classifies a sample based on the majority vote of their nearest neighbors. Simulation experiments were conducted and from the results tabulated in Table XV, it is evident that more detection accuracy of 90.3% was obtained when k=11. Confusion matrix values obtained during the experiments for K-NN are shown in Table XIV. From the experiments conducted for different classifiers for DS2, it is evident from Table XV that the best classifier for dataset DS2 is MLP with high detection accuracy of 93.3%. The obtained best classifier is added to the ensemble.

TABLE XIV. CONFUSION MATRIX OF K-NN EXPERIMENT RESULTS FOR DS2

	Normal	**Attack**
Normal	129	16
Attack	13	142

TABLE XV. SIMULATION RESULTS OF MACHINE LEARNING ALGORITHMS FOR DATASET DS2

Classification Algorithm	**False Positive**	**False Negative**	**Detection Accuracy**
C4.5 Decision tree	11.72	12.3	88
SVM	11.03	9.03	90
Multi layer Perceptron	6.2	7.1	93.3
Naïve Bayes	8.3	9.7	91
K-Nearest Neighbor	11.03	8.38	90.3

C. Exp. 3: Selection of best classifier for HTTP flood dataset (DS3)

The third experiment was conducted to select the classifier that performed best in identifying the HTTP flood attacks. The experiments conducted in this Section are similar to the experiments conducted in Section IV.A and Section IV.B but with a difference in inputs. Inputs given to the classifiers are the normal traffic and HTTP flood attack traffic dataset. The same five classification algorithms were tested and the results are tabulated in Table XXI. **C4.5** finds the normalized information gain for each feature. Confusion matrix values obtained during the experiments for C4.5 decision tree are shown in Table XVI. From the experiments, the detection accuracy achieved for the best decision tree classifier was seen to be 89%.

In **SVM**, the clusters used during each simulation were 8, 16, 32, and 64. From the simulation, it was observed that the model with 10 neurons in hidden layer, 8 clusters, and learning rate = 0.1 performs best in terms of detection accuracy. Confusion matrix values obtained during the experiments for SVM are shown in Table XVII. From the experiments and the results tabulated in Table 21, the detection accuracy achieved for SVM classifier was seen to be 90%. A three layer **MLP** neural network was simulated. Confusion matrix values obtained during the experiments for MLP are shown in Table XVIII. It was observed that the model with 10 neurons in hidden layer with learning rate = 0.1 produces detection accuracy of 90.33% with less false alarms.

TABLE XVI. CONFUSION MATRIX OF C4.5 DECISION TREE EXPERIMENT RESULTS FOR DS3

	Normal	**Attack**
Normal	125	15
Attack	18	142

TABLE XVII. CONFUSION MATRIX OF SVM EXPERIMENT RESULTS FOR DS3

	Normal	**Attack**
Normal	123	17
Attack	13	147

TABLE XVIII. CONFUSION MATRIX OF MLP EXPERIMENT RESULTS FOR DS3

	Normal	**Attack**
Normal	127	13
Attack	14	146

TABLE XIX. CONFUSION MATRIX OF NAÏVE BAYES EXPERIMENT RESULTS FOR DS3

	Normal	**Attack**
Normal	126	14
Attack	11	149

TABLE XX. CONFUSION MATRIX OF K-NN EXPERIMENT RESULTS FOR DS3

	Normal	**Attack**
Normal	130	10
Attack	12	148

In **Naïve Bayes**, confusion matrix values obtained during the experiments are shown in Table XIX. From the experiments conducted, the detection accuracy was seen to be 91.67%. **K-Nearest Neighbor (K-NN)** classifies a sample based on the majority vote of their nearest neighbors. The neighbors were identified with the representation of samples in multidimensional feature space. As it is binary classification problem, the K values were varied in odd numbers ranging 11, 21, 31, and 41.

TABLE XXI. SIMULATION RESULTS OF MACHINE LEARNING ALGORITHMS FOR DS3

Classification Algorithm	**False Positive**	**False Negative**	**Detection Accuracy**
C4.5 Decision tree	10.7	11.25	89
SVM	12.14	8.13	90
Multi layer Perceptron	9.3	8.8	90.33
Naïve Bayes	10.0	6.8	91.67
K-Nearest Neighbor	7.14	7.5	92.67

Simulation experiments were conducted and it is evident from Table XXI that detection accuracy of 92.67% was obtained when k=11. Confusion matrix values obtained during the experiments for K-NN are shown in Table XX. From the experiments conducted for different classifiers for DS3, it can be seen from Table XXI that the best classifier for this dataset is K-NN with high detection accuracy of 92.67%. The obtained best classifier is added to the ensemble.

D. Observation from the Experiments 1, 2, and 3

The data consists of approximately 1 million preprocessed data packets. Five different machine learning algorithms were compared. After being trained with the training data, each model was evaluated with testing samples. In our simulation experiments, the dataset was split into training dataset and testing dataset. The testing dataset was served as completely unseen samples to the trained ensemble. Testing dataset contained some new attack patterns which were not included in the training dataset and poses a challenge to test the ability of M_2KMix algorithm in detecting the new attack types.

From the experiments, it was evident that the trained ensembles were able to detect new attacks. During training, the false positives were more and with successive rounds of boosting the network, the false positives were reduced gradually. When learning rate for MLP was too small, it took very long time to reach the global minimum. When learning rate was large, the gradient descent overshoot the maximum diverge and hence moved away from the local minimum. The number of hidden neurons which resulted in the minimum Generalization Error has been chosen as the optimal value in MLP algorithm. From the experimental studies, it was observed that setting more number of neurons in hidden layer resulted in Overfitting and less number of neurons resulted in Underfitting.

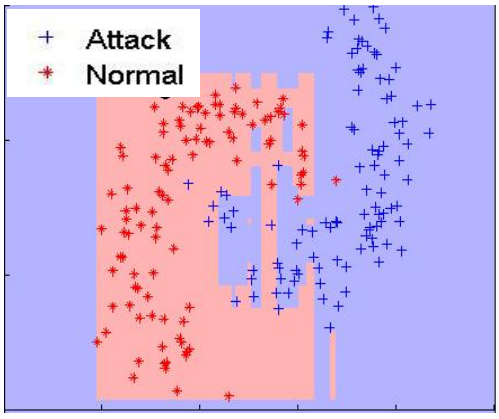

Figure 8. MLP classifier decision for Dataset1

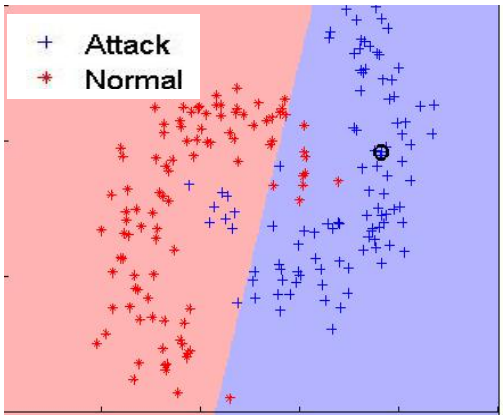

Figure 9. MLP classifier decision for Dataset2

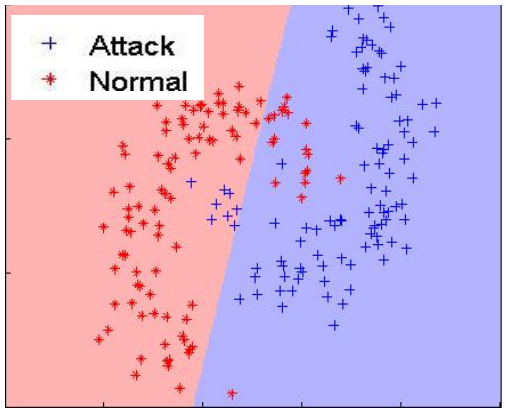

Figure 10. K-NN classifier decision for Dataset 3

The decision boundary became very complicated emphasizing very local changes of the class distributions for experiments conducted for Dataset 1. Eventually, MLP returns the network with lowest mean square error (MSE) on the testing set and the decision boundaries obtained from our experiment for MLP are visualized in Figure 8. MLP does not overfit training data when trained for a large number of epochs. 10 hidden units were used and the optimizations run for 1000 iterations. Similarly, among the set of experiments conducted for Dataset 2, MLP performed well in classifying the normal and attack traffic patterns and the decision boundaries are visualized as shown in Figure 9. But, for Dataset 3, K-NN classifier performed well in classifying the patterns than MLP and other ML algorithms. The decision boundary for K-NN classifier is shown in Figure 10. It can be seen from Figure 8, 9, and 10 that some normal samples are classified incorrectly as attack samples. These misclassifications were overcome when M_2KMix algorithm is combined with our proposed PA rule, which will be explained in Section IV.E. Due to space limitations, we restrict the classifier decision figures to only the best classifiers.

E. Exp. 4: Comparison of our Proposed PA rule with existing Output Combination Methods

The fourth experiment was conducted to analyze the performance of the proposed PA rule and to compare the results with the existing output combination methods. From the experiments 1, 2, and 3, MLP, MLP, and K-NN algorithms performed well in identifying the SYN, UDP, and HTTP attack traffic patterns respectively. Two MLP classifiers and One K-NN classifier were added to the ensemble. As our objective is towards identifying the attack type, proposed PA rule is implemented in this Section. Using PA rule, how effectively the M_2KMix algorithm identifies the different attack types can be seen from the experiments conducted with new dataset. Hence, a mixed traffic (CAIDA [21] and SSE Lab Datasets) was used to train and test the attack identification algorithm. The mixed traffic consisted of lab generated normal and attack datasets and CAIDA dataset. CAIDA DDoS Attack 2007 dataset contains approximately one hour of anonymized traffic traces from a DDoS attack on August 4, 2007. The one-hour trace is split up in 5-minute pcap files. Two 5 minute pcap files out of one hour trace were used. Only attack traffic to the victim and responses to the attack from the victim were included in the traces. DDoS attack traffic consisted of Ping ICMP flood, TCP SYN flood, and HTTP flood requests. Non- attack traffic also has been included in CAIDA dataset. To analyze the performance of proposed M_2KMix+PA, performance metrics such as detection accuracy and false alarms have been used for the experiments. Information coming from different data sources makes the ensemble a completely versatile classifier. The test data was not from the same probability distribution as the training data, and it included attack patterns not in the training data. *This made the classification task more realistic.*

The soft decisions output by three classifiers were then fused using existing classifier combination strategies. Four existing output combination methods such as product, minimum, mean, and maximum were applied under the assumption of equal priors and their results were compared. None of the existing output combination methods were able to improve the detection accuracy and decrease the classification error. The reasons are as follows: For example, suppose the first classifier

identifies a given instance as an attack class and the other two classifiers identify the same instance as a normal class. If max, prod, and sum rules are used, this instance will be misclassified as normal class as more votes are assigned for the normal traffic. *Thus the existing rules fail to detect the given instance as SYN flood attack.* In another instance, suppose the first classifier identifies the instance as normal class and the other two classifiers identify the instance as attack class. *If min rule is chosen, the instance is misclassified as normal class.* Hence, it may also fail to detect UDP and HTTP flood attacks. The results using these rules are not better than any of the individual classifiers in the ensemble as well. None of the existing output combination rules produced best identification results. It is evident from Table XXII that the detection accuracy is less for the existing fusion methods.

TABLE XXII. IDENTIFICATION ACCURACY OF EXISTING OUTPUT COMBINATION METHODS AND PROPOSED PA RULE

Ensemble Output Combination method	**SYN Flood**		**UDP Flood**		**HTTP Flood**	
	DA (%)	**FA (%)**	**DA (%)**	**FA (%)**	**DA (%)**	**FA (%)**
Preferential Agreement (PA)	100	0	93.75	6.25	97.5	2.5
Product	91.25	8.75	87.5	12.5	95	5
Maximum	92.5	7.5	85	15	88.25	11.75
Minimum	86.25	13.75	72.5	27.5	81.25	18.75
Mean	90	10	82.5	27.5	87.5	12.5

TABLE XXIII. CLASSIFICATION ERROR RATE OF EXISTING OUTPUT COMBINATION METHODS AND PROPOSED PA RULE METHODS

Ensemble Output Combination method	**Classification Error (CE) (%)**	**Overall Detection Accuracy (%)**
Product	8.3	91.7
Maximum	9.5	90.5
Minimum	19.4	80.6
Mean	12.7	87.3
Preferential Agreement	2.2	97.8

Unlike the existing methods, the proposed PA rule gives weightage to the classifier identifying the instance as attack type (S, U, or H). By using our proposed PA rule, highest detection accuracy was obtained for all the three flooding attacks as shown in Table XXII for unseen testing samples. The results in terms of performance metrics such as Overall Detection Accuracy and Classification Error are also tabulated in Table XXIII. The performance of PA rule is good in the proposed method as it is resilient to errors. *Proposed method is a simple technique of improving the reliability of decision making based on different classifier results.*

V. Conclusion

Proposed Ensemble, M_2KMix, incrementally and sequentially learns from data that consists of heterogeneous features by a mixture of expert classifiers and identifies the attack type through our proposed novel (PA rule) method. Different feature sets are obtained from different data sources. If there exists any new flooding attack, these new attack traffic be trained as a separate classifier and can be included to the trained ensemble without retraining the entire ensemble.

Our proposal differs from the existing methods in feature selection, error cost reduction, and attack type identification. The performance of the ensemble of MLP+KNN classifiers was evaluated based on the testing dataset which contains attack types not included in the training set. From the simulation experiments, it was evident that the trained ensembles were able to detect new attacks. Hence, the Learning ensemble learns the new data without forgetting the previously acquired knowledge, discarding the existing classifier, and retraining a new one.

As less number of features was selected for classification, the proposed M_2KMix attack identification algorithm is suitable for real time detection system. The focus in this paper is to identify the type of flooding attack and to minimize the false alarms. From the simulation results, it has been found that the proposed M_2KMix attack type identification algorithm results in high detection accuracy of 97.8%. It has been found that M_2KMix algorithm identifies three types of flooding attacks, the SYN Flood (100%), UDP flood (93.75%), and HTTP Flood (97.5%), effectively.

Acknowledgements

The authors are grateful for the sponsorship of this research work provided by the Government of India, New Delhi, under the Collaborative Directed Basic Research (CDBR) – Smart and Secure Environment (SSE) Project.

References

[1] Arbor Networks, “Worldwide Infrastructure Security Report”, Volume VI, November 2010.

[2] DDoS Security Reports, Arbor Networks, http://ddos.arbornetworks.com/.

[3] C. Douligeris, A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification", in Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 03), Darmstadt, Germany, pp. 190-193, Dec. 14-17, 2003.

[4] Wu, S., Chang, H., Jou, F., Wang, F., Gong, F., Sargor, C., Qu, D., and Cleaveland, R. Jinao: Design and implementation of a scalable intrusion detection system for the ospf routing protocol. To appear in Journal of Computer Networks and ISDN Systems. http://projects.anr.mcnc.org/JiNao/ JiNaoJournal.ps Accessed 30 December 2002.

[5] Lunt, T. F. Detecting Intruders in Computer Systems. In 1993 Conference on Auditing and Computer Technology (1993).http://www.sdl.sri.com/ papers/c/a/canada93/canada93.ps.gz Accessed 22 August 2002.

[6] Ye, N., and Li, X. Application of decision tree classifiers to computer intrusion detection. In DATA MINING 2000 Data Mining Methods and Databases for Engineering, Finance and Other Fields, July 2000, Cambridge, UK (Southampton, UK, 2000), N. Ebecken and C. Brebbia, Eds., WIT Press, pp. 381–90.

[7] Tsudik, G., AND Summers, R. Audes-an expert system for security auditing. Computer Security Journal 6, 1 (1990), 89–93.

[8] Teng, H. S., Chen, K., and Lu, S. C. Y. Adaptive real-time anomaly detection using inductively generated sequential patterns. In IEEE Symposium on Security and Privacy (1990), pp. 278–284.

[9] Arun Raj Kumar, P. and S. Selvakumar, "Distributed Denial of Service Attack Detection using an Ensemble of Neural Classifier", *International Journal of Computer Communications, Elsevier Publications, United Kingdom*, Volume 34, Issue 11, 2011, pp. 1328-1341.

[10] KDD data set, 1999; http://kdd.ics.uci.edu/databases/-kddcup99/kddcup99.html

[11] The CAIDA "DDoS Attack 2007" Dataset Paul Hick, Emile Aben, kc claffy, Josh Polterock, http://www.caida.org/data/passive/ddos-20070804_dataset.xml.

[12] CAIDA UCSD Network Telescope "Three Days Of Conficker" - < Nov. 2008 > , Paul Hick, Emile Aben, Dan Andersen and kcclaffy http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml.

[13] Sommer, R., Paxson, V., "Outside the closed world: On using machine learning for network intrusion detection", *In proceedings of the Symposium on Security and Privacy*, 2010.

[14] Park J S, Shazzad K M, Kim D S, "Toward modeling lightweight intrusion detection through correlation-based hybrid feature selection", Information Security and Cryptolo, First SKLOIS Conference, CISC 2005, Beijing, China, 2005, pp. 279–289.

[15] Moore A W, Zeuv D, "Discriminators for use in flow-based classification", *Intel Research Technical Report*, 2005.

[16] Auld T, Moore A W, Gull S F, "Bayesian neural networks for Internet traffic classification", *IEEE Transactions on Neural Networks, 2007*, Volume 8, No.1, pp. 223-239.

[17] Dimitris Gavrilis, Evangelos Dermatas, "Real time detection of distributed denial-of-service attacks using RBF networks and statistical features", *Computer Networks*, 44 (5) (2005) pp.235 – 245.

[18] Hoai-Vu Nguyen, Yongsun Choi, "Proactive detection of DDoS attacks using k-NN classifier in an Anti-DDoS Framework", *International Journal of Computer System Science and Engineering,* 2008, pp. 247-252.

[19] Rasool Jalili, Fatema Imani–mehr, Morteza Amini, Hamid Reza shahriari "Detection of DDoS attacks using statistical Preprocessor and unsupervised Neural Networks", *LNCS 2005*, pp.192-203.

[20] Stefan Seufert, Darragh O Brein, "Machine Learning for Automatic Defense against Distributed Denial of Service Attacks", *Proceedings of IEEE International Conference (ICC) 2007*, pp.1217-1222.

[21] Xu, L., "Bayesian Ying-Yang System and Theory as A Unified Statistical Learning Approach: (III) Models and Algorithms for Dependence Reduction, Data Dimension Reduction, ICA and Supervised Learning". Lecture Notes in Computer Science: Proc. of International Workshop on Theoretical Aspects of Neural Computation, May 26-28, 1997, Hong Kong, Springer-Verilag, pp. 43-60.

[22] Yang Xiang and Wanlei Zhou, "Mark-Aided Distributed Filtering by using Neural Networks for DDoS defense", *IEEE GLOBECOM* 2005, pp. 1701-1705.

[23] G. Giacinto, R. Perdisci, and F. Roli, "Network intrusion detection by combining one class classifiers," presented at the Int. Conf. Image Analysis and Processing, Cagliari, Italy, 2005.

[24] Devi Parikh and Tsuhan Chen, Data fusion and cost minimization for intrusion detection. *IEEE Transactions on Information Forensics and Security*, **3** 3 (2008), pp. 381–389.

[25] Haykin, S., Neural Networks: A Comprehensive Foundation, Prentice Hall, Upper Saddle River, NJ, 1994.

[26] Mukkamala, S. Lanaski, S.and Sung, A. "Intrusion detection using neural networks and support vector machines," in *Pmc.* ofthe *Int Joint Conf on Neural Networks (IJCNN 2002).* Honolulu. vol. *2.* 2002, pp. 1702 – 1707.

[27] Guo, G., Wang, H., Bell, D., Y. Bi, and Greer, K.,, "Using kNN model for automatic text Categorization", Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 10, No. 5, pp. 423- 430. 2006.

[28] Duda, R. O. and Hart, P. E., "Pattern Classification and Scene Analysis", New York: Wiley, 1973.

[29] Mitchell, T. *Machine Learning*, McGraw-Hill Education (ISE Editions), 1997.

[30] Shanbhag, S. and T. Wolf, Evaluation of an online parallel anomaly detection system, in Proc. of IEEE Global Communications Conference (GLOBECOM). 2008: New Orleans, LA.

[31] Shanbhag, S. and T. Wolf. Massively Parallel Anomaly Detection in Online Network Measurement. In Proceedings of 17th International Conference on Computer Communications and Networks, 2008. ICCCN '08. 2008.

[32] L. Breiman, "Bagging predictors," Machine Learning, vol. 24, no. 2, pp. 123–140, 1996.

[33] Y. Freund and R.E. Schapire, "Decision-theoretic generalization of on-line learning and an application to boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119–139, 1997.

[34] Polikar, R. "Ensemble based systems in Decision Making", *IEEE Circuits and Systems,* Vol. 6, September 2006, pp. 21 – 45.
[35] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas, "On Combining Classifiers", Pattern Analysis and Machine Intelligence, IEEE Transactions on, Vol. 20, No. 3. (1998), pp. 226-239.
[36] Clayton Scott, Robert Nowak, "A Neyman Pearson Approach to statistical learning", *Technical Report TREE 0407.*
[37] Planet Lab, http://www.planet-lab.org/
[38] Netwag Tool, http://ntwag.sourceforge.net/
[39] HTTPTrafficGen, HTTP Traffic Generator for testing Web Applications, http://www.nsauditor.com/web_tools_utilities/http_traffic_generator.html.
[40] LOIC, Low Orbit Ion Cannon Tool, http://sourceforge.net/projects/loic/.

Arun Raj Kumar, P., received a Bachelor of Engineering degree in Computer Engineering from Regional Engineering College (REC), Jaipur, India in 2002. He worked as a Faculty for one year and as a Lecturer for three years in the Department of IT in an Engineering College from 2003 to 2006. He received a Master of Technology degree in Computer Science and Engineering with Distinction from National Institute of Technology (NIT) Tiruchirappalli, India in 2008. Currently, he is pursuing Ph.D. in Computer Science and Engineering at National Institute of Technology (NIT) Tiruchirappalli, India. His research interests include Computer Networks, Wireless Sensor Networks, and Network Security.

S. Selvakumar obtained Ph.D. degree by the Indian Institute of Technology Madras, Chennai in 1999. Currently, he is a Professor in the Department of Computer Science and Engineering, National Institute of Technology (NIT), Tiruchirappalli, Tamil Nadu, India. He has 28 years of teaching experience and 18 years of research experience. His research interests include group communication in high-speed networks, routing, multimedia communication, scheduling for QoS guarantee, mobile networks, network security, and network computing. He has to his credit of publishing 16 research papers in International Journals, 3 research papers in National Journals, and 32 research papers in International and National Conferences. He has visited University of Alcala, Madrid, Spain, Iowa State University, Ames, USA, City University of New York, USA, Queensland University of Technology, Brisbane, Australia, University of Vienna, Austria, Singapore, Canada, and Switzerland. He is presently the member of All India Board of IT Education, AICTE, New Delhi.

