ISSN 2278-3083

Volume 8, No.6, November –December 2019

International Journal of Science and Applied Information Technology

Available Online at http://www.warse.org/ijsait/static/pdf/file/ijsait26862019.pdf

https://doi.org/10.30534/ijsait/2019/26862019

Detecting BHP Flood Attacks in OBS Networks: A Machine Learning Prospective

Adel Rajab

College of Computer Science and Information System,

Najran University, 1988,

Najran, Saudi Arabia

adrajab@nu.edu.sa

ABSTRACT

The Optical Bust Switching (OBS) network has become the most promising switching technology for building the next generation of internet backbone infrastructure. However, an OBS network still faces a number of security and Quality of Service (QoS) challenges, particularly from Burst Header Packet (BHP) flood attacks. If a source node (ingress) becomes compromised by an attacker, overloading the network with malicious BHPs, the network resources will be reserved without proper utilization. This prevents legitimate BHPs from reserving the required resources, and can lead to severe issues, such as burst loss and Denial of Service (DoS) among others. One way to prevent a BHP flood attack is to detect the misbehaving edge nodes overloading the network with malicious BHPs, and taking the proper action to secure and sustain the QoS performance in an OBS network. A powerful and promising approach in identifying misbehaving edge nodes causing BHP flooding attacks is Machine Learning (ML), and in particular, classification techniques. A classification technique learns models by applying them to a large historical data set derived from an edge node's performance during a simulation run. The data set contains behavior traces from a number of edge nodes, with respect to input data characteristics, sensitivity, efficiency performance, predictive performance, and model content. The learned model can then be utilized to single out (classify) misbehaving edge nodes based on their future performance as accurately as possible, hence disciplining them. In this paper, we investigate the BHP flood attack problem by evaluating a number of ML techniques in classifying edge nodes, and determine the most suitable method to prevent this type of attack. Specifically, we evaluate Decision Tree (C4.5), Bagging, Boosting (AdaBosst), Probabilistic (Naïve Bayes), Rule Induction (RIppleDOwn Rule Learner-RIDOR), Neural Network (NN-MultilayerPerceptron), Logistic Regression, and Support Vector Machine-Sequential Minimal Optimization (SVM-SMO) on a real dataset to identify the method(s) most appropriate to combat the BHP flood attack problem in OBS networks.

Key words : Burst Header Packet **(**BHP) flood attack, Classification, Computer Network; Machine Learning, Network Security, OBS Network

.

1. INTRODUCTION

An optical network (ON) is a known medium for data transmission, adopting an Optical Burst Switching (OBS) network for the Internet [1]. In an OBS network, burst header packets (BHPs) are transmitted in advance to allocate enough resources prior to sending the actual data bursts (DBs), ensuring network management and Quality of Service (QoS). This enables attackers to flood the network with malicious BHPs, reserving the network resources without proper use. In this case, malicious BHPs continue to reserve the network resources without sending the actual DBs, hindering the performance of the OBS network, in some cases causing Denial of Service (DoS) [2]. Therefore, it is essential to prevent BHP flooding attacks in OBS networks by blocking misbehaving ingress nodes that continuously transmit malicious BHPs, and preventing the legitimate BHPs from reserving the required resources at the intermediate core switch.

Limited research works detecting BHP flooding attacks in OBS networks exist, e.g. [3, 4, 5]. In [3], a data flow classification architecture was implemented at the optical layer to combat BHP flooding attacks. This method distinguishes between the offset time inside the BHP and the recorded delay between this BHP and its related DB. [4] utilized optical code words to single out malicious BHPs sent by ingress nodes in an OBS network. The authors used statistical data analysis related to packets sent and dropped to detect the possibility of BHP flooding attacks. [5] developed a new security model to be implemented into the OBS core switch to prevent BHP flooding attacks. The countermeasure security model can detect malicious ingress nodes based on their behavior, alongside the amount of reserved resources that are not being utilized, and block any malicious ingress nodes until the threat ceases. The reported results using the NCTUns network simulator showed that the security method of [5] was able to effectively differentiate among legitimate and malicious ingress nodes, thus maintaining good network performance.

Despite the few recent studies on BHP flooding attacks, the detection rate is still low. Further, the entire process relies on the domain experts' knowledge and experience. Therefore, there is a need for a more efficient detection system that can engage the core switch in OBS network, thus identifying misbehaving ingress nodes in an automated manner as early as possible. One promising approach to accomplish this is the Machine Learning (ML) method. This uses the historical performance of source nodes during data transmission to construct classification models known as classifiers. The classifiers then predict whether the source nodes are sending legitimate BHPs or not, and filter out malicious BHPs that might cause flooding attacks. The outcomes of the ML method will enable security administrators to quickly block misbehaving ingress nodes until they change their behaviors. (It is the firm belief of the authors) that classifying ingress nodes using ML to counter BHP flooding attacks is yet to be studied within an OBS network.

This study examines the performance of ML methods to counter the risks associated with BHP flood attacks in OBS networks. The problem studied is a typical predictive task in classification, in which different variables linked with ingress nodes' performances are collected whilst sending BHPs (in simulation runs), and are saved in a training dataset. Examples of variables are not limited to iteration number, but can include the sending node label, packets sent, packets dropped, delay time, and so on. More details on the complete dataset of variables can be found at [6], and are briefly explained in Section 3.1. The ML role involves processing the different variables in the dataset to obtain concealed information useful for prediction (classifier). This classifier is then used to categorize ingress nodes in certain future scenarios as accurately as possible, improving the manual classification which indeed requires care, time and experience.

The ultimate aim of this study is to examine the applicability of ML to the problem of BHP flooding attacks in OBS networks. To achieve this, we extensively investigated various ML techniques that adopt different learning approaches to the research problem considered. We seek to identify the most relevant ML technique(s) for solving the issue of BHP flooding attacks, in addition to revealing the reasons behind the relevancy. Thus, we endeavor to answer the following research questions:

- Can ML be used as a BHP detection approach in an OBS network?
- Which ML techniques improve detection rate and time performance?
- Which ML technique is more suitable to end-users, and why?

The ML approaches considered in this study are Logistic Regression, Naïve Bayes, RIDOR, SVM-SMO, NN-MultilayerPerceptron, C4.5, AdaBoost, and Bagging [9, 5]. The diversity of the ML approaches strengthens the confidence in the results, hence our recommendations (see Sections 3 & 4). The performance of the wide range of ML techniques has been measured using different metrics, against a published dataset at UCI (University of California-Irvine) repository [7]. Specifically, we utilized classification accuracy, classifiers' construction time in milliseconds (ms), precision, recall, and the harmonic mean among other measures (Section 3 gives further details) [8].

The remaining of this paper is organized as follows. Section 2 reviews the related research studies and the considered ML approaches. Section 3 is devoted to experimental settings, data description, and results analysis. Lastly, Section 4 will offer concluding remarks.

2. LITERATURE REVIEW

2.1 Studies Related to Application of Machine Learning in Detecting and Classification Tasks

A limited amount of studies adopt ML techniques in attempting to counter BHP flooding attacks in OBS networks, e. g.. Despite the scarcity of literature, this section highlights these studies and others related to primarily utilizing ML in different types of computer networks. Developed rule sets based on experience to counter the problem of BHP flooding attacks in OBS networks. The rules are developed using statistical analysis of the ingress nodes' performance during a series of simulated runs, using different numbers of nodes. The rules then are used to categorize ingress nodes into two types: Behaving and Misbehaving. Experimental results showed that the domain experience rules can be enhanced if the classification systems built by ML techniques are adopted, since they are typically more accurate in detection than domain specific classifications.

[9] investigated the problems of BHP flood attacks in OBS networks to differentiate the types of data bursts, i.e. congestion or contention. A new metric named "number of bursts between failures" (NBBF) was proposed to detect which type of data bursts losses occur. In the process of classifying these data bursts, the authors applied two methods: unsupervised expectation maximization (EM) and a supervised Hidden Markov Chain (HMC). Reported results showed that when both methods are integrated, the accuracy of distinguishing among types of bursts losses is increased.

[24] investigated the Distributed Denial-of-Service (DDoS) flood attacks on the transport and application layers, and developed a detection mechanism that analyzes the traffic according to types of packets, packet arrival rate and server capacity. The detection mechanism relies on recording and monitoring information related to address pair (source and destination), the type of packet, the port addresses of the source and destination among others. The key to success of [10]'s method is the predefined setting value of the server capacity. No experiments have been conducted to reveal the pros and cons of the detection method of DDoS flood attacks.

[11] investigated the problems of reducing flood attacks and other service attacks in computer networks using ML. These types of attacks normally belong to DDoS flooding attacks, and other risk that impair Internet security. The aim was to

identify the misbehaving sources (nodes) in order to block their messages from their intended destinations. In the learning model proposed, elements of the network share behavior information about the network's performance, so the classifier may amend or enhance the model's behavior by blocking potentially detrimental messages. Reported experimental results revealed a 95% detection rate using a probabilistic classifier.

[12] reviewed different learning mechanisms utilized to detect DDoS flooding attacks, in particular, SYN flooding. This type of flooding attack harms the network performance: when packets flood the network, many users may suffer server access delays. In some cases, the server shuts down entirely from SYN flooding attacks. The authors of [12] critically analyzed different approaches related to ML, statistical analysis, and router based among others.

[13] adopted the Naïve Bayes (NB) probabilistic classification algorithm [14] to detect the type of Internet traffic. Before applying NB, features related to traffic flow such as port identification, elapsed time between two consecutive flows, and the flow length among others, were collected. The type of traffic flow variable was assigned by a domain expert in the dataset, and NB was applied to generate probabilistic classification systems to predict the traffic flow variable. The classification system derived by NB shows low predictive rates, but when the authors utilized feature selection methods prior to the training phase, the accuracy rate of the classification systems was improved.

The IP traffic classification problem was studied in the context of ML by [15]. The authors surveyed and compared the performance of supervised and unsupervised ML algorithms, and highlighted the role of feature assessment in pre-processing the IP traffic dataset. Results showed that NB, EM and decision tree algorithms often produce consistent results, with high classification accuracy for the IP Internet traffic problem. Moreover, a number of recommendations have been highlighted based on the survey, such as:

1) ML algorithms generate different results for the IP traffic problem because of the different learning mechanisms they employ in deriving the classification systems. Hence, hybrid learning seems appropriate for future investigation
2) Different requirements are sought by ML algorithms because learning environments differ from one algorithm to another, as well as configurations
3) It is essential to investigate real time learning, at least for the IP Internet traffic classification problem, in which the ML will, while in progress, derive the classifiers rather than using static datasets
4) Feature selection methods can be useful in some Internet application problems such as IP Internet traffic classification

The majority of recent research contends that utilizing ML techniques in computer networks relates to DDoS flood attacks using primarily adaptive distributed mechanisms, while other studies investigated data traffic analysis. This study investigates an entirely new issue – BHP flood attacks in OBS networks. We believe that ML has not yet been adopted to develop predictive models to counter BHP flood attacks in OBS networks.

2.2 The Considered Machine Learning Techniques

Since the BHP flooding attack is a typical prediction problem, classification methods in ML seems appropriate to identify malicious and legitimate edge nodes. In classification problems, a model called the classifier is constructed from historical labelled dataset(s). The learned classifier is then employed to forecast the class label in datasets that are unlabeled, known as test datasets [16, 5]. The quality of the classifiers extracted by ML methods rely primarily on the classification accuracy, as well as other known evaluation metrics such as recall, precision, and harmonic mean [18]. In addition, classifiers formed after data processing differ based on the ML techniques used. For instance, rule induction classifiers contain rules, and Naïve Bayes classifiers hold just class memberships in a probability format [19]. In this section, we highlight eight different ML techniques that generate different type of classifiers. Specifically, we investigate classifiers extracted by Logistic Regression, Probabilistic-Naïve Bayes, Rule Induction-RIDOR, Support Vector Machine -Sequential Minimal Optimization (SVM-SMO), Neural Network-NN-MultilayerPerceptron, Decision Tree-C4.5, Boosting-AdaBoost, and Bagging [20, 21, 22, 23, 24, 25, 26]. The choices of these techniques are mainly based on the following facts:

1) Different learning methodologies are employed for data processing
2) Different classifier formats are presented to the end-user
3) Applicability and usage in previous domains in particular computer networks, computer security among others, i.e. [18,.27,28]

Steps of machine learning are shown in Figure 1, and are briefly explained below.

1) Data pre-processing (Optional): In this step, any noise related to the training dataset, such as missing values, duplications, and feature selection are completed. The output of this step is a processed dataset.

2) Training: In this step, the ML technique processes the data for knowledge or patterns. In classification techniques, the classifier is constructed in this step.

3) Evaluation: The classifier is evaluated on a test dataset to measure its effectiveness. This step results in different evaluation metrics.

4) Pattern Visualization (Optional): In this step, the outcomes as well as its quality measures are presented to the end-user in a non-technical manner to ease decision making.

The next section briefly summarizes known ML learning approaches that this study investigates to be utilized in solving the BHP flood attacks problem.

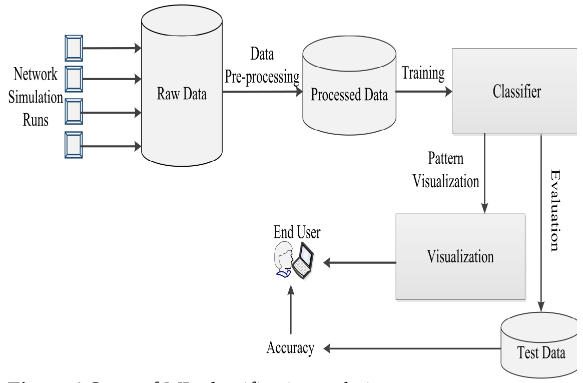

Figure 1. Steps of ML classification technique

A. Rule Induction - RIDOR

Rule induction is a classification approach that normally extracts If-Then rules in a sequential fashion. Typically, a rule induction technique divides the input dataset into splits according to the available class values. Then, for each class split, the induction technique learns and derives If-Then rules based on mathematical metrics, such as a rule's expected accuracy (Equation (1)). Data examples in a split, for instance A, are positive examples for the class of A, and are considered negative examples for the other class labels in the other data splits. For a data split, the induction technique builds an empty rule, and then adds items to the rule's antecedent (left hand side/body) until the rule meets a termination condition. When this occurs, the rule is generated, and all data examples that the rule classifies are discarded. Then, the induction technique learns the next rule from the same split until the data split becomes empty. Following this, the induction technique moves to the next data split until all data splits become empty, or no more rules with acceptable accuracies can be discovered [29]. Common rule induction techniques are RIDOR and RIPPER [8].

RIDOR, for example, derives a default rule class, and then learns all the exceptions for that default rule using Incremental Reduced Error Pruning (IREP) [31], a learning method. An exception is a rule able to forecast the class label other than the default class. IREP eliminated one exhausting phase of an earlier rule induction technique called Reduced Error Pruning (REP), saving substantial training time. In RIDOR, the training dataset is divided into pruning (1/3) and growing (2/3) subsets. Then, RIDOR builds incremental rules one at a time. When a rule is about to be evaluated for possible pruning, its training data examples in the pruning and growing subsets are removed, and the rule gets extracted. During pruning, RIDOR considers deleting items from the rule's body and terminates the pruning phase when removing an item from a rule cannot improve the rule's accuracy.

$$r's\ Expected\ Accuarcy = (P/T) \tag{1}$$

where P = the # of positive instances covered by a rule r (both antecedent and consequent)

T= the total # of instances covered by r's antecedent

B. Decision Tree Rules – C4.5

C4.5 is a decision technique utilizing Entropy and Information Gain (IG) (Equations 2-3 below) to construct tree based classifiers for prediction. To build a classifier, initially, the IGs for all variables in the training dataset, other than the class variable, are computed, and a root with the highest IG is selected. The IG is calculated based on how informative a data variable is in dividing the examples in the training dataset with respect to the class label. When a root is chosen, the algorithm excludes it in the next iteration and repeatedly calculates the IGs for the other available variables, until the tree cannot be built any further or the remaining data examples are linked with just a single class. In the formed decision tree, a path from the root node to any leaf denotes a rule, and the leaf denotes a decision (class label).

$$Gain\ (T, f) = Entropy\ (T) - \sum\ ((|\ T_f|\ /\ |\ T\ |)\ *\ \text{Entropy}\ T_f)) \tag{2}$$

$$Entropy\ (T) = \sum -P_c\ \log_2\ \ P_c \tag{3}$$

where P_c = Probability that T belongs to class l, T_f = Subset of T for which feature F has value $f_{a.}$, $|T_f|$ = Number of examples in T_f, and $|T|$ = Size of T.

C. Probabilistic Methods- Naive Bayes

In classification, when a test example requires a class label, an efficient way to classify the test example is to use NB technique, which is based on Bayes theorem. NB calculates the probability of the test example with respect to each class label using prior knowledge of the test example's variables, and their appearances with each class in the training dataset. The frequency of each variable and the class in the training dataset is obtained in addition to the frequency of each class label. Then, all probabilities are multiplied by each other and the test data example is given the class with the highest probability score (Equation 4 below). NB predicates independent assumptions for variables and the class, which is not necessarily true in real application data [32]. Nevertheless, this probabilistic technique is highly efficient in deriving classifiers in contrast to other ML techniques [33].

Given a test data example as a vector A = (a_1, a_2, …, a_m) where each a is a variable, using NB, the conditional probability can be obtained as:

$$P(C_n|A) = \frac{P(C_n).(A|C_n)}{P(A)} \tag{4}$$

The test data example will be given the class with the greatest probability $P(C_n|A)$.

D. Boosting and Bagging

Bagging and Boosting learning approaches use the training dataset in multiple trails to produce numbers of weak classifiers, that are then merged to form a global classifier [34]. The idea is to utilize both the weak and the strong classifiers in predicting the class label of test data.

In Boosting, a weak classifier is simply built from the input dataset, and then utilized to assign class labels to the training data examples. The next weak classifier is built from the training data, and training examples that have not been correctly classified by the previous weak classifier are selected more often to be re-classified by the current weak classifier, improving the model's predictive accuracy. The below steps clarify how Boosting algorithms, such as AdaBoost [10, 34], work:

1) Select a base ML algorithm for learning such as a rule based classifier
2) The base algorithm learns a weak classifier from the training dataset and assigns an equal weight for each training data example
3) When there are misclassification cases (incorrectly classified data examples), we re-apply the base ML algorithm, and pay more attention to the unclassified data examples to improve the predictive performance
4) Repeat steps 2-3 until the intended accuracy has been derived
5) Merge the weak classifiers to produce a strong classifier
6) When a test data needs to be classified, use a voting mechanism to assign the class label from the strong classifier and the weak classifiers.

In the Bagging classification approach [5], sample data examples are generated for each trail (iteration) from the original training dataset (often with the same size of the original training dataset). Then, a base ML algorithm is used to generate a classifier from the sample, and the process is repeated a number of times. Finally, all derived classifiers are aggregated together to form a global (strong) classifier. When test data is about to be classified in the Bagging approach, the class is assigned based on a voting mechanism using both the global and weak classifiers, similar to the Boosting approach. The difference between Bagging and Boosting approaches is that in Bagging, when the data sample is produced from the training dataset, the resembling process is not reliant on the performance of any previously derived classifiers, as it is in Boosting.

E. ANN

An Artificial Neural Network (ANN) consists of interconnected neurons that transform a set of input examples into desired output (class) without having to reveal the transformation details [20]. The ANN advantage comes from choosing the right numbers of the hidden neurons, and the results often rely on the input variables features and weights associated with their interconnections. Nevertheless, determining the numbers of hidden neurons and other important thresholds prior to data processing is fundamental to the quality of the outcome in ANN algorithms. Questions such as, what is the right number of hidden layers, epoch size, and acceptable learning rate, among others, need to be set by a domain expert in order to generate fair and acceptable classifiers. Overall, researchers still utilize train-and-error methods to tune the aforementioned parameters since there is no clear methodology for setting these up [35]. ANNs utilize sigmoid functions during constructing classifiers, in which weights are repeatedly amended to come up with the desired error rate that the domain expert had set prior to the beginning of the learning phase.

F. SVM

SVM is a classification approach proposed to enhance the predictive performance of classic classification techniques [36]. This approach depends on hyperplanes, which divide data examples based on class memberships. The SVM learning mechanism sorts data examples using mathematical functions known as kernels. A kernel computes the similarity of data examples using the available classes in the training dataset [36]. Often, kernels are determined by SVM experts, and then utilized for the classification phase.

SMO trains SVM on a large quadratic programming (QP) optimization problem [37]. SMO decomposes the QP problem into a number of smaller problems, and then solves them by avoiding a numerical QP inner loop. The computing resource needed in the particular memory for SMO is linear in the training dataset size, which permits the SMO algorithm to process larger input datasets. Reported experimental results revealed that SVM algorithms such as SMO generate high predictive classification systems in multiple domains, especially text categorization rather than probabilistic, and induction [37, 38].

G. Logistic Regression

When the target variable in classification dataset is continuous, (numeric) classic ML methods such as rule induction, decision trees, and covering are not able to produce a classifier. Linear regression can solve such a problem by offering methods describing the training dataset in the context of a predictive task, by revealing the relationships between independent variables and the class variable (dependent). Unlike linear regression, in Logistic regression, the class variable is not continuous, but is rather categorical (predefined possible values) [37, 18].

Logistic regression is formulated based on Equation 5 below:

$$p = \frac{e^{a+bX}}{1+e^{a+bX}} \tag{5}$$

where p = probability of Y = 1

e = base of the natural logarithm(around 2.718)

a and b = inputs parameters of the logistic model

Due to the curvilinear correlation between p and X, b in (Equation 5) is different than b in a typical linear regression model.

We can linearize the logistic regression model by converting the dependent variable from a likelihood (probability) to a logit, as shown in Equation 6.

$$ln\left(\frac{p}{1-p}\right) = a + bX \tag{6}$$

$$ln\left(\frac{p}{1-p}\right) = \text{logit (log odds) of } Y = 1 \tag{7}$$

where

a and b = inputs of the logistic model

The logit (Equation 7) is often named a *link* function, because it gives a linear conversion of the logistic regression model.

3. DATA AND EXPERIMENTAL RESULTS

3.1 Experimental Settings and Data

This section investigates the ML algorithm’s performance on a simulated dataset generated by the NCTUns simulator for over a thousand runs on NSFNET topology [39]. The aim is to enhance the performance of UDP on OBS networks by automatically detecting misbehaving ingress nodes that may cause BHP flood attacks, helping to manage the network’s resources. By employing NSFNET topology, we can insert and simulate with any number of nodes in order to investigate different scenarios. The simulation parameters for the OBS network configuration are displayed in Table 1. The simulator may need to run for 15 to 45 minutes to obtain the result for just “one second” depending on the load assigned.

All experiments have been conducted utilizing a recently developed simulated dataset that belongs to the authors. This can be obtained from the UCI Machine Learning Repository (University of California-Irvine) dataset [40]. This contains twenty-two variables related to flooding attacks, including the class variable. The variables collected during the NCTUns simulator directly associate with the OBS network’s performance. The dataset size consists of 1075 examples, and each example denotes one iteration (a simulation run) in which an ingress node is sending data over the OBS network. Different scenarios, including BHP flood attacks without pre-setting values, have been generated during the simulation, ensuring that ingress nodes have random levels of BHP flood attacks. This is essential to show situations of occupied network resources without proper utilization and with different occupancies. During the simulated runs, two ingress nodes were used. In addition, for each simulation run, the bandwidth of the node was initially assigned to 100 Mbps, and then incrementally increased to 200 Mbps, 300 Mbps, 400 Mbps, and so forth, until the maximum bandwidth, i.e. 1000 Mbps, is reached.

Table 1. NCTUns Network Simulator parameter of the OBS Network configuration in evaluation

Parameter	Value
Link bandwidth	1000Mb/s
Propagation delay	1 µs
Bit error rate	0
Maximum burst length	1500 bytes
Number of BHP channels	1
Number of DB channels	2
Use of Wavelength Conversion	No
Use of Fiber Delay Line (FDL)	No
Transport Layer Protocol	UDP

For illustration purposes, Table 2 depicts eight variables with five iterations exhibiting how the ingress nodes for every simulation run were used to transmit data. The table displays iterations that demonstrated behaving and misbehaving edge nodes. The dataset contains four possible class labels (Block, No Block, Misbehaving-No-Block, Misbehaving-Wait), and thus the problem is a multi-class classification. In Table 2, at iteration #1, ingress node 3 was permitted to send data, since it was classified as a behaving node. Ingress node 9, associated with a low BHP flooding rate, was slightly misbehaving, yet because of its low packet dropping rate, it was not blocked. However, at iteration #5, ingress node 3 was blocked, since this node was causing high BHP flooding, its BHPs reserving bandwidth without utilization. At the same iteration, despite node 9 misbehaving, it was still permitted to send data (misbehaving but no block). A trickier scenario is illustrated at iteration #4, in which both ingress nodes are misbehaving, yet are not reaching a BHP flooding attack. Therefore, node 3, due to its higher BHP flood rate, delays until node 9 transmits its data.

Table 2. Sample of five iterations of the multi-class training dataset

Iteration	Node #	Drop-Rate	Bandwidth-Use	Delay	Node Status	BHP Flood	Class
1	3	0.11	0.793	0.00009	B	0.000	No Block
1	9	0.22	0.703	0.00009	B	0.008	M-No Block
2	3	0.45	0.559	0.0005	M	0.369	M-No Block
2	9	0.42	0.589	0.0007	M	0.3697	M-Wait
3	3	0.466	0.543	0.0005	M	0.2887	M-Wait
3	9	0.416	0.593	0.0006	M	0.2541	M-No Block
4	3	0.476	0.532	0.0006	M	0.3621	M-Wait
4	9	0.415	0.594	0.0008	M	0.3112	M-No Block
5	3	0.482	0.526	0.0005	M	0.4337	Block
5	9	0.414	0.596	0.0008	M	0.3848	M-No Block

The Waikato Environment for Knowledge Analysis (WEKA) tool was adopted to process the dataset using ML [41]. This tool is a Java based open source, containing various methods related to ML, data mining, visulization, data filtering, and variable selection among others. For all considered ML algorithms, a 10-fold cross validation (10 fold-CV) method was employed during the training phase [41]. 10 fold-CV is a common testing method in ML that ensures the input dataset splits into 10 folds. The algorithm is then trained on 9 folds, and evaluated against the remaining fold to generate the error rate. This procedure is repeated ten times, and all error rates are averaged to show the overall performance of the learning algorithm. The machine used to run all experiments is Intel® Xeon with 3.72 GHz 2 processors.

A number of ML algorithms have been selected to counter the risk of BHP flood attacks by detecting misbehaving ingress nodes. In particular, Simple Logistic Regression, Naïve Bayes, RIDOR, SVM-Sequential Minimal Optimization (SVM-SMO), NN-MultilayerPerceptron, C4.5, AdaBoost, and Bagging [37, 9, 12, 25, 32, 29, 10, 5]. We would like to evaluate the classification systems' predictive accuracies derived from the aforementioned ML algorithms on the BHP flood attack problem. The main metrics used in the ML algorithms' comparisons are:

1) Classification accuracy in %
2) True Positives (TPs) and False Positives (FPs)
3) *P*recision, *R*ecall and Harmonic Mean (*F*-measure)
4) Training time measured in milliseconds (ms) to build the classifiers
5) Classifiers content for the rule induction, Bagging and tree based algorithms

These evaluation measures mathematical descriptions are given below:

$$P = \frac{TP}{TP+FP} \quad (8)$$

$$R = \frac{TP}{TP+FN} \quad (9)$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \quad (10)$$

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (11)$$

where *TP* is the number of data examples correctly classified by class A, *TN* is the number of data examples correctly classified by class -A, *FP* is the number of A's examples incorrectly classified as -A, and *FN* is the number of -A examples incorrectly classified as A.

Prior to running the ML learning algorithms against the BHP flooding attacks dataset, we pre-processed the dataset using Correlation Features Sets (CFS) to determine the most influential features [41]. CFS is a well-known feature selection method which heuristically examines the correlation of each feature with the class label in order to discard any redundant or low correlated features. After running the CFS on the initial dataset, three features (Drop-Rate, Bandwidth-Use, BHP-Flood) were identified to be more effective to combat the BHP flood attack problem. Hence, we will utilize these features during the training phase for the classifiers.

3.2 Results Analysis

Figure 2 highlights the classification accuracies derived by the ML classifiers from the dataset. It is clear from the figure that the Bagging, rule induction (RIDOR), and decision tree (C4.5) classifiers have higher prediction rates than that of the remaining classifiers. Noticeably, the C4.5 algorithm outperformed the remaining algorithms when it comes to predictive accuracy. To be exact, its predictive accuracy is 4.66%, 14.52%, 20.84%, 1.68%, 26.42%, 39.07%, 18.05% higher than those of RIDOR, Naïve Bayes, Simple Logistic Regression, Bagging, SVM-SMO, AdaBoost, and NN-MultilayerPerceptron, respectively. The superiority of C4.5 may be due to the intensive backward and forward pruning implemented after constructing the tree. C4.5 trims sub-trees that lead to larger errors, replacing them with more accurate leaves, resulting in concise, yet highly predictive, classifiers. In addition, the C4.5 algorithm triggers an implicit discretization procedure based on Entropy, converting continuous variables into discrete ones prior to the training phase. This ensures small intervals for each continuous attribute, easing the data processing, and ensuring its efficiency. Finally, Bagging and RIROD classifiers seem competitive in the decision tree, both algorithms using effective pruning procedures to cut down the number of rules produced.

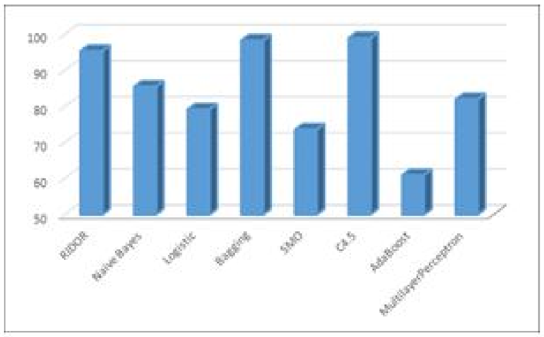

Figure 2. Classification accuracies in % derived by the ML algorithm

Figure 3 displays the classifiers' sizes for the top three predictive classifiers (C4.5, Bagging, RIDOR). It is clear from the figure that Bagging derives larger classifiers compared to both RIDOR and C4.5 algorithms. This is due to the generation of multiple local classifiers, and the integration step forming a final tree structure, which may lead to many branches and leaves. The classifier presented by RIDOR is the least predictive among those of the three algorithms, yet it contains a concise set of rules. From the user's perspective, a more concise set of rules could make it easier for network administrators to understand and manually control the BHP flooding attack problem. C4.5, on the other hand, offers moderate-sized classifiers that have superiority in classification accuracy over RIDOR and Bagging respectively. In fact, C4.5 covered more training examples than RIDOR, discovering more rules that may contribute to the increase in predictive performance.

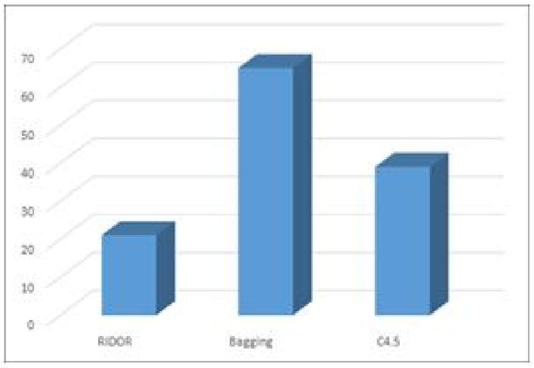

Figure 3. The classifier sizes of RIDOR, Bagging and C4.5 algorithms

Figures 4a – 4d show the true positives (TPs), false positives (FPs), true negatives (TNs) and false negatives (FNs) respectively for the considered algorithms on the BHP flood attack dataset. The TPs and TNs are consistent with the classification accuracy rates derived beforehand, in which C4.5, Bagging and RIDOR achieved higher TPs and TNs than that of the remaining algorithms. For example, RIDOR correctly classified "Block", "No Block" and "M- No Block" class labels without any error. However, for the hard-to-detect cases, i.e. the ones which belong to the "M-Wait" class, 28 instances have been misclassified by RIDOR as the "M=No Block" label. For the TNs results, AdaBoost algorithm seems have the rates because it was unable to clearly differentiate among the four class labels in particular NB-Wait, which its instances have been completely misclassified to NB-No-Block class label.

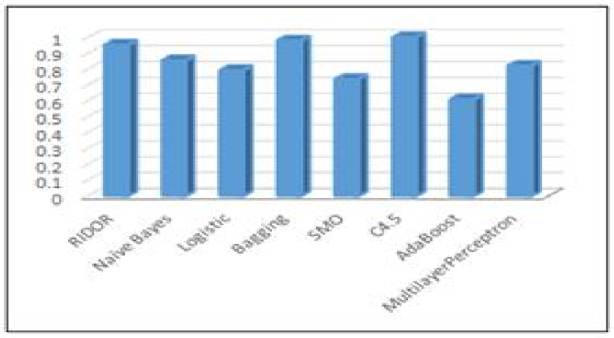

Figure 4a. The TPs of the ML algorithms

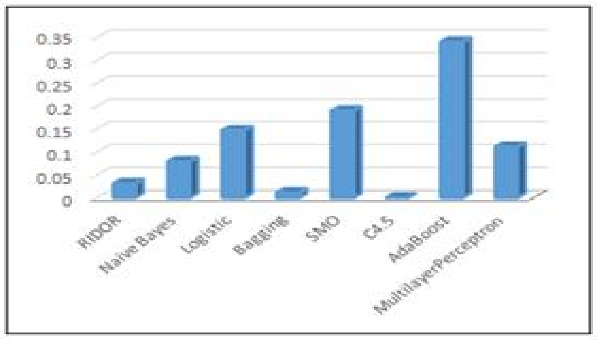

Figure 4b. The FPs of the ML algorithms

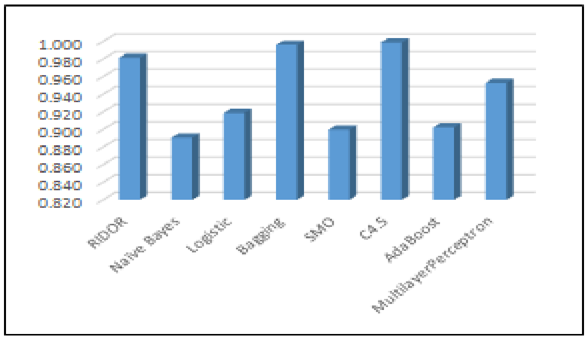

Figure 4c. The TNs of the ML algorithms

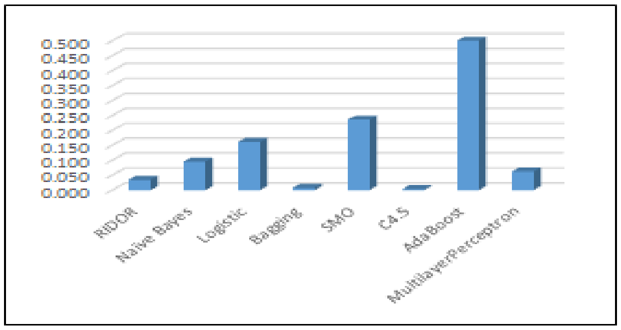

Figure 4d. The FNs of the ML algorithms

The results of the TPs, TNs, FNs and FPs show that "Block" and "No Block" cases are easy to detect by the ML algorithms except AdaBoost, but cases that belong to class labels "M-Wait" and "M-No Block" are harder to be detected, due to overlaps between these two class labels. To be precise, in terms of FPs and FNs, the three least performed algorithms (AdaBoost, SVM-SMO, Logistic) are associated with 300, 204, and 254 misclassifications respectively. These figures clearly reveal the reasons behind the low predictive rates of these three algorithms in detecting difficult-to-classify cases of "M-Wait" and "M-No Block". To overcome this issue of overlapping between class labels, more data cases covering "M-No Block" and "M-Wait" are needed, so the ML algorithms can further distinguish between them during the learning phase. This is due to the fact that the misbehaving nodes are further decomposed in the dataset into three sub-class labels, in order to reflect the true nature of the problem and reduce overfitting during the learning phase. Moreover, and in terms of FNs, decision tree and Bagging algorithms consistently derived good results when compared with the remaining algorithms. To be exact, Bagging algorithm only wrongly classified 11 instances 8 of which belong to the hard to classify class NB-Wait. Typically, we do not desire to end up with a binary classification problem in which the ML algorithm decides whether the ingress node is behaving or misbehaving. However, we do aim to understand to which degree the node is misbehaving, and if two nodes are misbehaving, which may be allowed to transmit data, and which should delay in using their flooding or network utilization rates. Therefore, it was necessary to further split the misbehaving class into multiple class labels during the data collection phase.

Figure 5 shows three more types of measures: precision, recall, and F-measure. The precision results displayed in Figure 5 shows a consistency with classification accuracy rates, and highlights that malicious ingress nodes are harder to be detected than behaving ingress nodes, at least for the dataset and algorithms used. Usually, high precision rates, such as in C4.5, RIDOR and Bagging, relate to their low FPs. C4.5 achieved the largest precision and AdaBoost the lowest. In the precision results, seven out of eight algorithms have consistent results when compared to their accuracies, except for the AdaBoost algorithm. The precision of AdaBoost declined significantly to 0.397 (39%) due to a large number of FP cases, as shown in Figure 5. Precision shows the number of correctly classified cases from all that have been classified. On the other hand, recall results in the same figure denotes the number of correctly classified cases in all cases intended to be correctly classified. In the recall results, all the ML algorithms have consistent results when compared to their predictive accuracies.

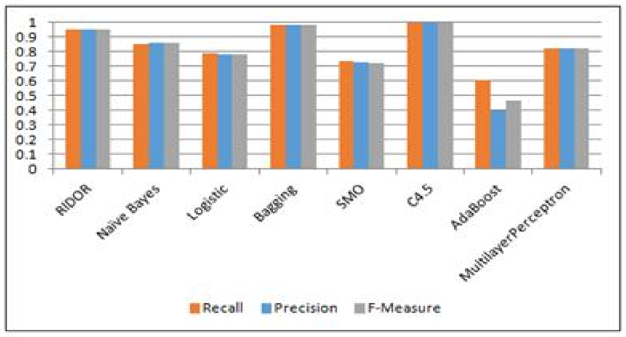

Figure 5. The Precision, Recall and F1 scores of the ML algorithm

To have a clearer insight into precision and recall alongside one another, we generated the scores when using the F1 measure. The F1 score takes the weighted average of recall and precision (false negatives and false positives) into consideration, especially when involving data such as our four unevenly distributed class labels. In our study, we can observe that C4.5, RIDOR and Bagging still generate highly competitive F1 scores compared to the remaining considered algorithms on the BHP flood attack dataset.

Lastly, Figure 6 depicts the runtime in millisecond (ms) taken from the ML algorithms in constructing the classifiers. Here, the fastest algorithms were Naïve Bayes and C4.5. Naive Bayes uses simple likelihood calculations for all variables in the test dataset using their frequencies in the training dataset, hence no rule learning being involved. Alternately, the C4.5 algorithm employs fast learning based on computing Entropy for the variables in the training dataset to build tree based classifiers. Hence, these two algorithms are quite efficient in building predictive classifiers in contrast to alternative ML algorithms. The MultilayerPerceptron NN algorithm was the slowest algorithm in building the classifier due to the exhaustive search this algorithm employs, which is based on pre-setting the desired expected error achieved. This often necessitates repetitive training dataset scans.

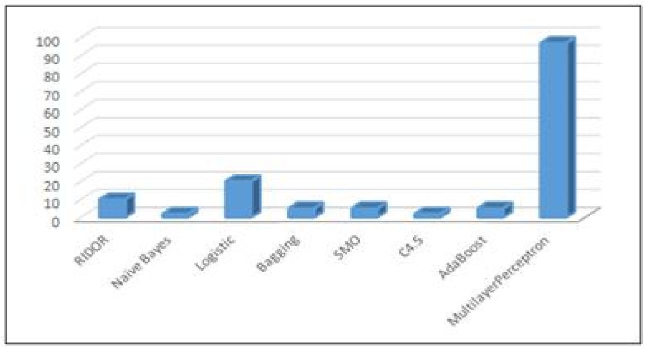

Figure 6. The time in ms needed to build the classifier of the ML algorithm

4. CONCLUSIONS

In spite of the many benefits of an OBS network, such as bandwidth efficiency, economic values and resiliency, OBS network can become vulnerable when burst loss occurs during ingress nodes sending data, causing BHP flood attacks. This fundamental issue in OBS networks may deteriorate the overall network's performance, due to the allocating of resources without proper usage. BHP flood attacks hinder the QoS of the OBS network, hence potentially causing a severe problem – the Denial of Service (DoS). This paper investigated the aforementioned issue by applying machine learning (ML) to automatically detect misbehaving ingress nodes, and blocking them in a preliminary stage. We evaluated various ML algorithms via simulation data, involving more than two ingress nodes and over 530 runs.

The aim was to classify ingress nodes as accurately as possible, using variables related to their performance, such as packet drop rate, bandwidth used, and average delay time among others. Experimental results from a processed dataset related to BHP flood attacks showed that rule based classifiers, in particular decision trees (C4.5), Bagging, and RIDOR, consistently derive high predictive classifiers compared to alternate ML algorithms, including AdaBoost, Logistic Regression, Naïve Bayes, SVM-SMO and NN-MultilayerPerceptron. Moreover, the harmonic mean, recall and precision results of the rule based and tree classifiers were more competitive than those of the remaining ML algorithms. Lastly, the runtime results measured in terms of millisecond showed that decision tree classifiers are not only more predictive, but are also more efficient than the rest of the algorithms. Thus, this is the most appropriate technique for classifying ingress nodes to combat the BHP flood attack problem. This paper is one of the initial attempts on adopting ML techniques to automatically classify ingress nodes in OBS networks.

In the near future, we intend to build a new rule-based classifier using the decision tree, and embed it inside the simulator to detect misbehaving nodes during the simulation phase.

REFERENCES

1. Y. Chen, C. Qiao, and X. Yu, **Optical burst switching: A new area in optical networking research**,*IEEE Network*,vol. 18, no. 3, pp. 16–23 Jun 2004,
2. A. Rajab, C. T. Huang, M. Al-Shargabi, and j. Cobb **Countering Burst Header Packet Flooding Attack in Optical Burst Switching Network**,*ISPEC 2016: Information Security Practice and Experience*, 2016, pp 315-329.
3. M. Sliti, M. Hamdi, and N. Boudriga, (2010) **A novel optical firewall architecture for burst switched networks**, *in Proc. 12th Intl. Conference on Transparent Optical Networks (ICTON)*,2010, pp. 1-5.
4. M. Sliti, and N. Boudriga, **BHP flooding vulnerability and countermeasure,***Photonic Network Communications*, vol. 29, no. 2, pp.198-213, 2015.
5. N. Abdelhamid, and F. Thabtah.**Associative Classification Approaches: Review and Comparison,***Journal of Information and Knowledge Management (JIKM)*, vol. 13, no. 03, pp. 145-227, 2014.
6. A. Rajab, **Burst Header Packet (BHP) flooding attack on Optical Burst Switching (OBS) Network Data Set,***University of California Irvine Data Repository*, 2017.
7. M. Lichman, UCI **Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California**, School of Information and Computer Science, 2013.
8. F. Thabtah, **A review of associative classification mining**,*The Knowledge Engineering Review*,vol. 22, no. 1, pp. 37-65, Mar 2007.
9. A. Jayaraj, T. Venkatesh, and C. Murthy, (2008) **Loss classification in optical burst switching networks using machine learning techniques: improving the performance of tcp**,*IEEE Journal on Selected Areas in Communications*,vol. 26, no. 6, pp. 45 –54, Aug 2008.
10. N. Patani1, R. Patel, **A Mechanism for Prevention of Flooding based DDoS Attack**,*International Journal of Computational Intelligence Research*, vol. 13, no. 1 pp. 101-107, 2017.
11. J. L. Berral, N. Poggi, J. Alonso, R. Gavaldà, J. Torres, and M. Parashar, (2008) **Adaptive distributed mechanism against flooding network attacks based on machine learning,***Proceedings of the 1st ACM Workshop on Workshop on AISec,* October 2008, pp.43–50.
12. R. C. Prathibha, R. R. Rejimol Robinson, **A Comparative Study of Defense Mechanisms against SYN Flooding Attack**,*International Journal of Computer Applications*, vol. 98, no.18, pp. 0975 – 8887, July 2014.
13. A. W. Moore, and D. Zuev, **Internet traffic classification using Bayesian analysis techniques**,*In ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS) 2005*, Banff, Alberta, Canada, June 2005, vol. 33, No 1, pp. 50-60.
14. R. O. Duda, and P. E. Hart, **Pattern Classification and Scene Analysis**, *New York: John Wiley & Sons*, vol. 3, pp. 731-739, 1973.
15. T. T. Nguyen, and G. Armitage, **A survey of techniques for internet traffic classification using machine learning**,*IEEE Communications Surveys and Tutorials*, vol. 10, no. 4, pp. 56-76, 2008.
16. N. Abdelhamid, A. Ayesh, and F. Thabtah, **Phishing**

detection based associative classification data mining,*Expert Systems with Applications*, vol. 41, no. 13, pp. 5948-5959, Oct 2014.

17. H. Abdel-Jaber, F. Thabtah, M. Woodward, A. Jaffar, and H. (2014). **Random Early Dynamic Detection Approach for Congestion Control,***Baltic Journal of Modern Computing*, vol. 2, no. 1, pp. 16-31, 2014.
18. F. Thabtah, W. Hadi, N. Abdelhamid, and A. Issa, **Prediction Phase in Associative Classification**. In *Journal of Knowledge Engineering and Software Engineering,*vol. 21, no. 6, pp. 855-876, 2011.
19. I. Qabajeh, F. Thabtah, and F.Chiclana, **A dynamic rule induction method for classification in data mining**,*Journal of Management Analytics*, vol. 2, no. 3, pp. 233–253, Jul 2015.
20. M. Sumner, E. Frank, and M. Hall, **Speeding up logistic model tree induction Knowl**,*Discov. Databases: Pkdd*, Vol. 37, no. 21, pp. 675–683, 2005.
21. B. R. Gaines, and P. Compton, (1995). **Induction of Ripple-Down Rules Applied to Modeling Large Databases,***J. Intell. Inf. System*, vol. 5, no. 3, pp. 211-228, Nov 1995.
22. J. Platt, **Fast training of SVM using sequential optimization**, *(Advances in kernel methods – support vector learning*, B. Scholkopf, C. Burges, A. Smola eds), MIT Press, Cambridge, 1998, pp. 185-208
23. D. E. Rumelhart, G. E., Hinton, and R. J. Williams, **Learning representations by back-propagating errors,***Nature*,vol. 323, no. 6088, pp. 533–536, Oct 1986.
24. J. Quinlan, **C4.5: Programs for machine learning**, San Mateo, 1993, CA: Morgan Kaufmann.
25. Y. Freund, and R. E Schapire, (1997) **A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting**,*Journal of Computer and System*, vol. 55, no. 1, pp. 119-139, Aug 1997.
26. L. Breiman, **Bagging predictors**,*Machine Learning*, vol. 24, no. 2, pp. 123-140, Aug 1996.
27. R. Bunker, and F. Thabtah, (2017) **A Machine Learning Framework for Sport Result Prediction,** *Applied Computing and Informatics Journal*, Vol. 15, no. 1, pp. 1-21, Jan 2017.
28. R. M. Mohammad, F.Thabtah, and L. McCluskey, **An Improved Self-Structuring Neural Network**,*Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2016, pp. 35-47.
29. F. Thabtah, I. Qabajeh, and F. Chiclana, **Constrained dynamic rule induction learning**,*Expert Systems with Applications*, vol. 30, no. 63, pp. 74-85, Nov 2016.
30. W. Cohen,**Fast Effective Rule Induction**. *In Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, California, 1995,pp. 115-123Morgan Kaufmann.
31. J. Fürnkranz, and G. Widmer, (1994). **Incremental reduced error pruning**,*In Machine Learning: Proceedings of the Eleventh Annual Conference*, New Brunswick, New Jersey,1994, pp. 70-77. Morgan Kaufmann.
32. N. A. Zaidi, J. Cerquides, M. J. Carman, and G. I.Webb, **Alleviating naive bayes attribute independence assumption by attribute weighting**,*Journal of Machine Learning Research*, vol. 14, pp. 1947 – 1988, 2013.
33. J. Zhang, C. Chen, Y. Xiang, W. Zhou, and Y. Xiang, **Internet traffic classification by aggregating correlated naive bayes predictions**,*IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 5–15, Oct 2013.
34. J. Quinlan, **Bagging Boosting and C4.5,** 2006.
35. R. Mohammad, F. Thabtah, and L. McCluskey, **Predicting Phishing Websites based on Self-Structuring Neural Network**,*Journal of Neural Computing and Applications*, vol. 25, no. 2, pp. 443-458, Aug 2016.
36. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, (2009). **The WEKA Data Mining Software: An Update,***In SIGKDD Explorations*, vol. 11, no. 1, pp. 10-18, Nov 2009.
37. M. Hall, **Correlation-based Feature Selection for Machine Learning**, Thesis, department of computer science, Waikaito University, New Zealand, 1999.
38. H. Joachims, **Large-scale support vector machine learning practical, Advances in kernel methods: support vector learning**, MIT Press, Cambridge, MA, 1999.
39. T. Joachims, **Text categorization with support vector machines: Learning with many relevant features**, *In European conference on machine learning*, Springer, Berlin, Heidelberg, April 1998, pp. 137-142.
40. S. Le Cessie, and J. C. Van Houwelingen, **Ridge Estimators in Logistic Regression,***Applied Statistics*,vol. 41, no. 1, pp. 191-201, Mar 1992.
41. J. Shawe-Taylor, and S. Sun, **A review of optimization methodologies in support vector machines**, *Neurocomputing*,vol. 74, no. 17, pp. 3609–3618, Oct 2011.
42. R. Schapire,**The strength of weak learnability,***Machine Learning*, vol. 5, no. 2, pp. 197-227, Jun 1990.
43. NSFNET : [Online]. Available: http://nsl.csie.nctu.edu.tw/nctuns.html [accessed August 25, 2017].