Performance Analysis of Machine Learning and Pattern Matching Techniques for Deep Packet Inspection in Firewalls

J.V. BibalBenifa
Indian Institute of Information Technology: International Institute of Information Technology
Saravanan Krishnann
Anna University Chennai
Hoang Long (hoangviet.tdtu.1979@gmail.com)
Ton Duc Thang University
Raghvendra Kumar
GIET: Gandhi Institute of Engineering and Technology
David Taniar
Monash University

Research Article

Keywords: Deep packet inspection, Classifiers, Pattern matching, Internet attacks, Security and privacy.

Posted Date: September 7th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-260788/v1

Performance Analysis of Machine Learning and Pattern Matching Techniques for Deep Packet Inspection in Firewalls

J.V. BibalBenifa[1], Saravanan Krishnann[2], Hoang Viet Long[3,4,*], Raghvendra Kumar[5], David Taniar[6]

[1]Department of Computer Science and Engineering, Indian Institute of Information Technology, Kottayam, India
benifa.john@gmail.com
[2]Department of Computer Science and Engineering, Anna University Regional Campus, Tirunelveli, Tamilnadu, India
saravanan.krishnan@auttvl.ac.in
[3]Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam;
[4]Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam.
hoangvietlong@tdtu.edu.vn
[5]Department of Computer Science and Engineering, GIET University, India
raghvendra@giet.edu
[5]Faculty of Information Technology, Monash University, Melbourne, Australia. Email:
David.Taniar@monash.edu

*Corresponding author: Hoang Viet Long (+084 988107432)

Abstract: Malware is essentially one of the major security issues that have the potential to break the computer operations instantly. Majority of the internet attacks are caused by malwares that are being distributed through HTTP over the Internet. A Firewall is essential to prevent such internet attacks for enhancing the security measures. The most efficient method to prevent Intrusion in the network is Deep Packet Inspection (DPI), which is presently implemented in advanced firewalls. This research work intends to detect and prevent the intrusion in the network using a hybrid method with DPI, Pattern Matching (PM), and Machine Learning (ML) techniques. In this present work, a hybrid method which involves the functionalities of both DPI and ML is used for classification and identification of attacks. Here, DPI is done by Boyer-Moore-Horspool (BMHP) pattern matching algorithm and ten ML algorithms such as Support Vector Machines (SVM), Linear-SVM (L-SVM), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Decision Tree (DT), Random Forest (RF), AdaBoost (Ada), Gaussian Naive Bayes (GaNB) and Bernouli Naive Bayes (BeNB) are employed for classification. Subsequently, the proposed work is

evaluated in a sequential and parallel manner and it is customized for identifying the fuzzy, impersonation and Denial of Service (DoS)-based attacks. The proposed system is analyzed in different dimensions such as performance of ML methods and role of DPI in attack identification including the pattern matching efficiency. From the investigation, it is identified that BMHP algorithm has the least time and memory consumed values about 0.0028 sec and 125.4 Mib respectively. Similarly, SVM has the accuracy of 99.91% with the least time and memory consumed values about 18.185 sec and 303.5 MiB respectively.

Keywords- Deep packet inspection; Classifiers; Pattern matching; Internet attacks; Security and privacy.

1. Introduction

On the Internet, thousands of significant and sensitive information are exchanged everyday [1]. The network needs the security against attackers and hackers who misuse information. Attackers can steal data, damage user's resources, and influence systems to gain physical access [2]. This gives substantial threat to the resources of the organization and proficient network security protocols secures the data there by preventing the outside interferences [3].

An Intrusion Detection System (IDS) is typically a hardware oriented device or a software utility that observes the entire network traffic and alerts the user when there is an unlawful attempt or access to the resources [4]. The purpose of IDS is to ensure that a user is informed when there is a probability of network attack. It continuously inspects the data navigating between the computing systems within the network. It inspects network traffic and generates an alert when distrustful action or known threats are identified, so the users can scrutinize more closely and appropriate actions will be initiated to mitigate the attacks [5].

In computer networks an attack is any effort to uncover, modify, restrict, abolish, snip or gain unlawful access to a computing asset. A cyber-attack is a category of attack that intends to affect the performance of information systems, computing resources and networks [6]. There are basically two types of attacks that include Active attack and Passive attack. An Active attack involves modifying the system resources or disturbs their basic operations. It also makes some changes to the data stream for creating incorrect statements. Types of active attacks include Masquerade, Modification of messages, Repudiation, Replay and Denial of Service. The nature of passive attack is it learns from the system by snooping or monitoring but it does not create any

modification to the resources. The major objective of the network attacker is to attain the facts that are being transferred. The types of passive attacks are release of message content and monitoring or traffic analysis.

Deep Packet Inspection (DPI) is an unconventional technique for investigating the network traffic flow [7]. DPI investigates the packet payloads efficiently and determines the attacks. DPI works at the application layer that exists in the reference model such as the Open Systems Interconnection (OSI) model. It inspects the payloads of packets moving through a given checkpoint by pattern matching the data with the attack signatures. Traditional packet filtering techniques focus only on the header information which has limitations. Recently the organizations are more focussed on DPI based firewall for securing the sensitive resources. DPI can be employed as a network security tool for the discovery and capturing of information with respect to malicious traffic [8]. It can also be adapted for network management for stabilizing the traffic flow

The goal of ML techniques is to make machines act like humans. ML in network security enables collecting massive volumes of digital information to analyse, envisage and find insights so as to forecast and mitigate cyber-attacks [9]. When ML methods are integrated with DPI, it provides added advantage in enhancing the security of the resources connected to the network. In the proposed work a hybrid method which includes the functionalities of both ML and DPI is integrated and Implemented for Network Intrusion detection. The signatures used in DPI are the signatures of attacks such as Dos, Impersonation and Fuzzy and the pattern matching algorithm used is BMHP. Different ML models such as SVM, L-SVM, KNN, MLP, DT, RF, AdaBoost, GaNB and BeNB are employed for classification.

2. Related Work

A ML based IDS for Mobile Internet of Things (IoT) proposed by Amouri et al (2020) [10]. The system proposed is a two-stage cross layer-based IDS and in the initial stage one encapsulates RF classifier mounted on five dedicated sniffers (DSs). The five DSs are employed for collecting data from MAC or network layer and subsequently these data is then fed to a RF classifier. The evaluation metric used is a classification report which detects Blackhole and DDoS Attack. Fang et al (2020) proposed a ML method for Intrusion detection system where it incorporates Elman

Neural Network (ENN)accompanied with Robust SVM. Here, Network packet feature extraction and digitization techniques are discussed along with the implementation [5].

Lee et al (2020) proposed a novel IDS for in-vehicle networks by using remote frames [11]. The training datasets for Dos, fuzzy and impersonation is provided. And also the classification of the attacks is done using key-value pair data models. The effectiveness of the classification is also discussed. Rawat et al (2019) proposed an IDS using classical ML techniques along with hybrid unsupervised feature learning and deep neural network-based approach [12]. Different ML algorithms used for the classification of attacks such as Decision Tree, Extra Tree, and Ensemble Extra Tree are used and the evaluation metric used for the validation of model's testing and training is accuracy and classification Report.

A DL based framework is proposed for creating the firewall functional rules and it contains data acquisition from the firewall log by adopting the feature identification and classification process [13]. The Long Short-Term Memory (LSTM), Bi-directional LSTM (Bi-LSTM) and SVM are employed for classification purposes. From the results, it is inferred that the DL exhibits classification accuracy about 97.38%. Praise et al (2020) proposed a Reinforcement Learning and Pattern Matching (RLPM) based firewall system [14]. RLPM is potential to mitigate seven types of attacks and proves ML methods are efficient in identifying the various internet-based attacks.

Cantone et al (2003) proposed Boyer-Moore string matching algorithm [15]. New variants of Boyer-Moore string matching algorithms such as Horspool and Quick search are analysed. The Boyer-Moore algorithm is discussed and implemented and its performance is analyzed. Cho et al (2015) presented a computationally fast algorithm for order-preserving based pattern matching process [16]. It is a strategy for concluding the order-isomorphism between two strings even when there are the same characters as discussed. Then, it is proposed that the bad character rule is included in the Horspool algorithm for generic pattern matching problems. Finally, the bad character rule is integrated with the KMP- algorithm to enhance the worst-case time such that the efficiency is increased.

From the literature, the limitations of DPI are observed and they are: i) DPI is effective for buffer overflow, denial of service (DoS) and malware and it can lead to novel vulnerabilities 2) DPI is highly complex and needs periodic updates regarding the rules. 3) Third, DPI can decrease the network speed because it increases complexity on firewall processors.

3. Proposed Methodology

The high-level architecture of the proposed work is presented in Figure1, where implementation of DPI-based firewall along with ML is carried out through two stages. They are (***i***) **Stage1: Packet Pre-processing**, and (***ii***) **Stage 2: Classification using the ML modules**.

The attacks that are monitored in the architecture are [13] (i) Denial of Service (DoS), (ii) Fuzzy and (iii) Impersonation. DoS attack tries to prevent a legitimate user from accessing the service in the target machine and may contribute to the reduction in network capacity. Fuzzy attack spoofs the ID and Data values in the network packet. An impersonation attack effectively assumes the uniqueness of one of the valid parties in a communications protocol.

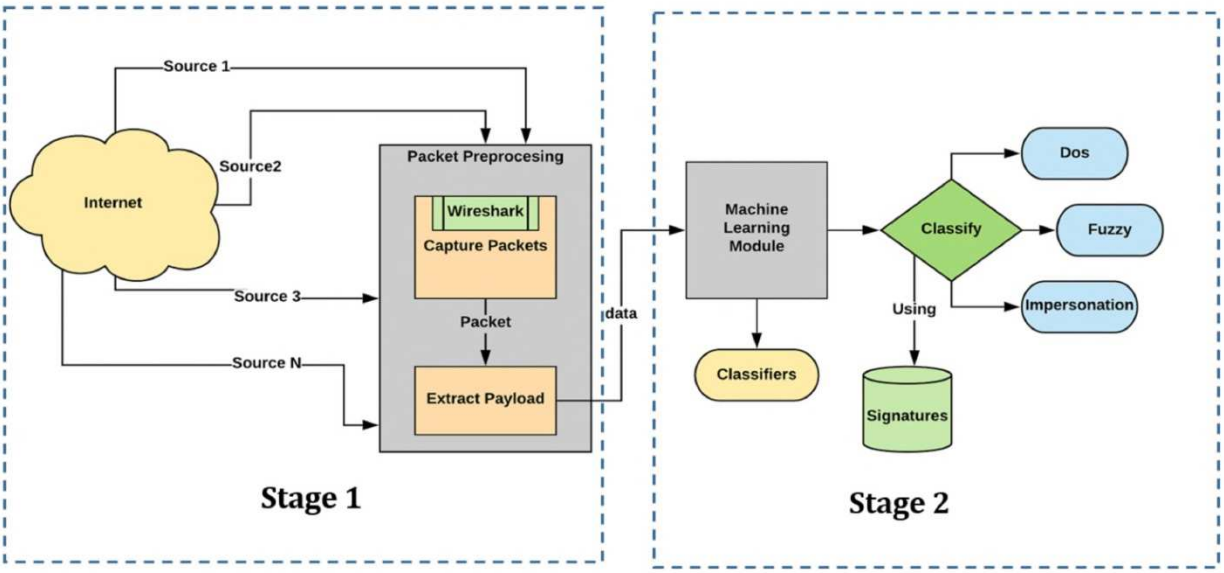

Figure 1. High Level workflow of the proposed work

The detailed workflow of the proposed method is presented in Figure 2 and a detailed description for the stages is presented subsequently.

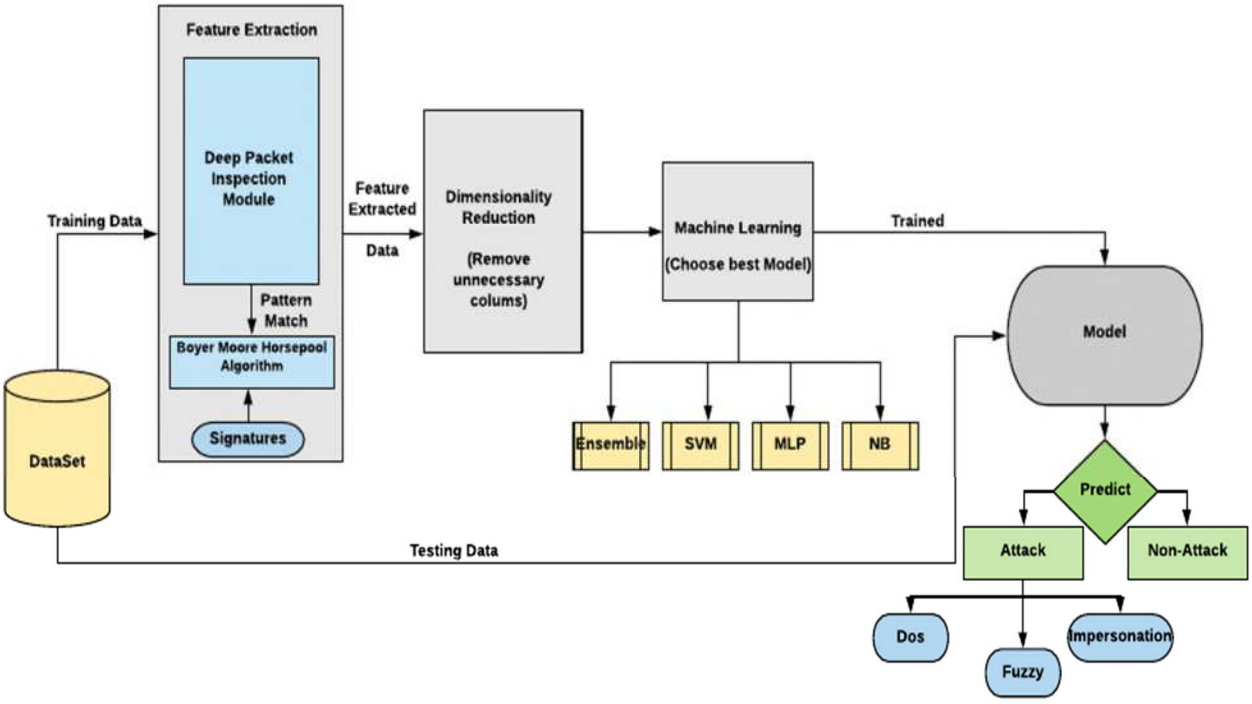

Figure 2. Detailed workflow of the Machine Learning Model

3.1. Stage 1: Packet Pre-processing

The packets over the internet are captured and monitored using a network monitoring tool known as Wireshark. The description of the captured packets from the live internet packet is presented in Figure 3. The packets that are captured in the wireshark are processed in the pre-processing phase. The captured packets are stored in a xyz.pcap file for further processing. The protocols of the arrived packets can be identified using scripting in t-shark with the following command that displays the desired protocol packets.

```
tshark -r "xyz.pcap" -Y usb -z conv,protocol
tshark -r "your/capture/file" -Y usb -z follow,tcp,hex,session\_no. $>$ session\_no.txt
```

In-order to extract the payload from the packet, scripting over t-shark is employed. The payload is saved in text files in sessions and each session contains the payload that is transferred between source and destination IP addresses. Subsequently the text files are processed and the desired payloads are extracted. The command used for extracting the payloads is as follows.

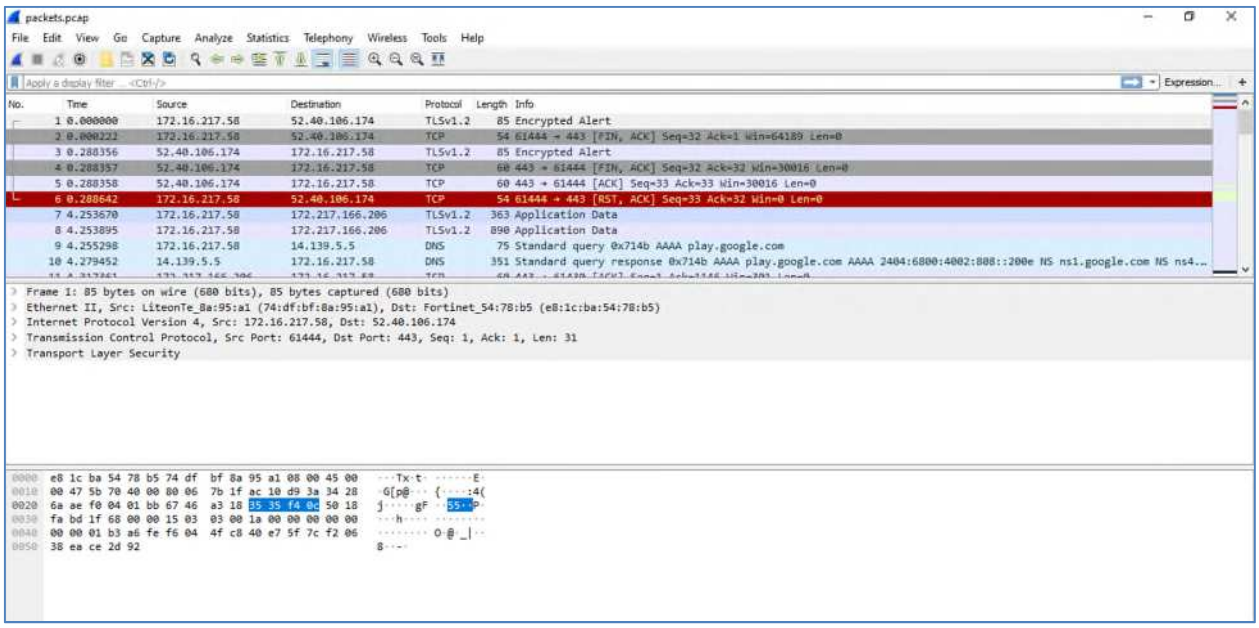

Figure 3. Captured packets using Wireshark

3.2. Stage 2: Classification using the Machine Learning Modules

The Machine Learning module involves (*i*) Feature Extraction, (*ii*) Dimensionality Reduction, (*iii*) Model creation using ML. The payload and protocols extracted in the Packet pre-processing stage are inserted into a data set. The data set used for classification is shown in Figure 4.

	protocol	sourceIPAdress	destinationIPAdress	timeStamp	len	data	target
0	HTTP	167.50.30.156	2.30.147.253	08-02-2019 10:10:28	70	d0ecc25418e1aca2	3
1	TCP/IP	178.76.229.13	113.144.16.192	01-06-2019 20:33:48	79	69b5faa8cd472e60	3
2	HTTP	237.224.185.211	2.108.37.18	17-06-2019 05:58:46	69	501457d8bff174bf	3
3	SMTP	251.227.74.148	220.3.101.137	05-01-2020 02:40:00	78	0000da9dd1551a42	0
4	UDP	202.178.50.102	214.157.79.86	30-01-2020 21:22:00	83	6c5fa73810164f11	1
...	...	...	...	...	...	...	...
10863	UDP	75.9.29.33	8.142.189.33	10-09-2019 05:29:49	95	4bfc3dc2f657f1b5	3
10864	TCP/IP	83.219.154.118	175.228.202.120	28-04-2019 17:47:15	86	74d619eee002c033	2
10865	FTP	195.235.123.98	240.86.210.116	10-03-2020 02:17:29	71	3800002006dfab19	0
10866	UDP	176.142.74.84	161.7.87.34	05-05-2019 14:04:11	76	bc7423805a250548	2
10867	FTP	21.164.14.96	25.12.5.126	14-04-2019 22:35:22	58	7e900002964a6098	0

Figure 4. Sample view of extracted payloads

3.2.1. Feature Extraction

For extracting the features DPI is used where the signatures of the attacks such as Dos, Fuzzy, Impersonation attacks are extracted from the data column using PM. The PM algorithm used is BMHP where the extracted feature(signature) is shown in Figure 5.

	protocol	sourceIPAdress	destinationIPAdress	timeStamp	len	data	signature	target
0	HTTP	167.50.30.156	2.30.147.253	08-02-2019 10:10:28	70	d0ecc25418e1aca2	no	3
1	TCP/IP	178.76.229.13	113.144.16.192	01-06-2019 20:33:48	79	69b5faa8cd472e60	no	3
2	HTTP	237.224.185.211	2.108.37.18	17-06-2019 05:58:46	69	501457d8bff174bf	no	3
3	SMTP	251.227.74.148	220.3.101.137	05-01-2020 02:40:00	78	0000da9dd1551a42	0000	0
4	UDP	202.178.50.102	214.157.79.86	30-01-2020 21:22:00	83	6c5fa73810164f11	0164	1
...	...	...	...	...	...	...	...	...
10863	UDP	75.9.29.33	8.142.189.33	10-09-2019 05:29:49	95	4bfc3dc2f657f1b5	no	3
10864	TCP/IP	83.219.154.118	175.228.202.120	28-04-2019 17:47:15	86	74d619eee002c033	02c0	2
10865	FTP	195.235.123.98	240.86.210.116	10-03-2020 02:17:29	71	3800002006dfab19	0000	0
10866	UDP	176.142.74.84	161.7.87.34	05-05-2019 14:04:11	76	bc7423805a250548	05a2	2
10867	FTP	21.164.14.96	25.12.5.126	14-04-2019 22:35:22	58	7e900002964a6098	0000	0

Figure 5. Sample view of extracted features

The inputs to the Algorithm are pattern (signature) and text (data) from the pattern to be searched, meanwhile the output is a Boolean value. The Boyer-Moore-Horspool (BMHP) algorithm [18] is shown in Algorithm 1.

Algorithm 1. Pseudocode for BMHP Algorithm

Procedure BMHP (Pattern p, Text t)

```
m= length of pattern
n= length of text
k= index of each element in pattern
if (m>n) return error
while(p) do
      value=m-k-1
if p matched then return (true)
else return (false)
```

The processing time of BMHP algorithm is linear it depends on the size of the string which is searched. The search operation completion time far lower than the other string matching algorithms. It has a property named Bad Match table, using this property BMHP skips some of the matching operation. Another one property of BMHP is when the sub string becomes longer, the algorithm operates in a very fast manner. The reason is the BHMP employs the Bad Match table to remove the positions where the sub string cannot match for each unsuccessful trial to identify a match between the sub string and the string, the algorithm uses the Bad Match table to rule out positions where the sub string cannot match. The BHMP is faster, simpler and optimized for searching the sub strings such that worst-case the performance of the BHMP algorithm is O(mn). Here, '*m*' signifies the length of the pattern and '*n*' denotes the length of the text [17],[18].

3.2.2. Dimensionality Reduction

Once the feature is extracted from the data, the dataset should be processed to dimensionality reduction where unnecessary columns are removed. Based on the reduced dimensionality, seven columns are retained in the dataset. When comes to large packet firewall data, it is necessary to reduce the columns. Network firewall dataset usually contains large number of columns having intermediate IP addresses. Here, we use feature extraction for removing the columns, which are not point of our interest. The removal of columns depends on the criteria that it does not affect the accuracy of the model. Subsequently, the pre-processed data is utilized for training and testing by splitting the data set into training (80%) and testing (20%) sets and then classified using different classifiers.

3.2.3. Model Creation using ML

In this section, we review a number of classifier methods that we used in the experimentations. They include: KNN, SVM, Decision Tree, Random Forest, Multilayer Perceptron, AdaBoost, and Naïve Bayesian.

(a) K-Nearest Neighbours (KNN)

KNN algorithm uses the concept of close proximity, which is the distance between points on a space [19]. In the Euclidean space, the Euclidean distance is used for finding the proximity. The Euclidean distance between two points (in an *n*-dimension) $d(x,y)$ is expressed in Equation (1)

$$d(x,y) = \sqrt{\sum_{n=1}^{\infty} (x_i - y_i)^2} \quad (1)$$

The boundaries of each class are decided by a parameter *K* and it becomes smooth while *K* values are increased. The experiments have to be conducted for various *K*-values because the training and validation error rates would be considerably changed at different *K*-values. In order to obtain the optimal value of *K*, the training and validation datasets are to be segregated from the initial dataset. The *K*-value has to be increased and iterated until there is no change in the predicted class. The other distance metrics suitable for this algorithm are Chebyshev, cosine similarity and Euclidean Distance is considered to the most efficient one.

(b) Support Vector Machines (SVM)

In the SVM, data points are plotted as vertices in an *n*-dimensional space where '*n*' represents the total number of extracted features. The magnitude of each feature is represented by the position of a particular coordinate [20]. The grouping of features into different categories or classes is performed by estimating the ideal hyperplane which distinguishes the two groups efficiently. A hyperplane is a margin that classifies the data points into their corresponding or specific classes in a high dimensional space. On both sides of the hyperplane, the data points exist that can be assigned to different classes. Depending upon the *n*-dimensional space, the hyperplane is chosen. To be more precise in 2 dimensions the hyperplane will be a straight line, in 3 dimensions it will be a plane in n dimensions it is known as hyper-plane. Considering the line equation (2)

$$y = ax + b \quad (2)$$

Let us consider the variables *x* and *y* as features and designating them as x_1, x_2, …, x_n, Equation (2) can be expressed as a revised form in Equation (3),

$$ax_1 - x_2 + b = 0 \quad (3)$$

If $x = (x_1, x_2)$ and $w = (a, -1)$ we get,

$$w.x + b = 0 \quad (4)$$

Hence the Equation of a hyper-plane is $w^T x = 0$

The hyper-plane is expected to be away from a data point, hence it can be considered as an optimal hyperplane. SVM is basically a large margin classifier where the space between the line and the Support Vectors (SV) is maximized. The margin is estimated based on the orthogonal distance measured from the line with its SV. The principal objective of SVM is to identify an ideal splitting characteristic of hyperplane or capitalize the boundary of the training data being considered.

The hypothesis function '*h*' is defined as in Equation (5),

$$h(x_i) = \begin{cases} +1 & if \ w.x + b \geq 0; \\ -1 & if \ w.x + b < 0; \end{cases} \quad (5)$$

The data point over the hyperplane can be grouped as a positive class, and the data point lower to the hyperplane can be grouped as a negative class. SVM has base factor *C* which relates to the regularization of the hyperplane and has a default value $c = 1$ and γ corresponds to points to be considered. If the value of γ is small, then the data points are located remote from the hyperplane is considered for fine tuning the hyperplane.

Using SVM, when the dataset has a higher dimension, the processing a lot of time. To solve this problem, **Linear Support Vector Machines** (**L-SVM**) can be used. L-SVM uses several kernel functions (i.e.)., using the defined dot product according to the function in L-SVM [21].

In our experimentations, we use both SVM, and L-SVM.

(b) Decision Tree (DT)

In order to define a path to scan a dataset and also to define the tree-like path towards the projected outcomes, DT is an effective method [22]. A feature selection process is an empirical approach to define the classification criteria. Attribute selection measure is employed for partitioning the data that provides the data into separate classes. These are considered to be splitting rules because they specify the split criterion on a DT. The feature with maximum score related to the assigned selection criteria is considered as the root node for the assumed datasets. The various attribute selection measures are Information Gain, Gain Ratio, and Gini Index.

Information Gain designates the splitting attribute with respect to the volume of information essential to define the tree and reduces the information required to classify the data points. These criteria project the lowest arbitrariness exists in these partitions. Information Gain is also called as the Entropy of the given data points as expressed in Equation (6) and(7).

$$Info\text{-}gain(D) = -\sum_{}^{m} p_i \log_2(p_i) \tag{6}$$

$$Info\text{-}gain_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} Info\text{-}gain(D_j) \tag{7}$$

where, p_i is the likelihood that an arbitrary data point exists in dataset 'D' corresponding to class 'C_i'. Finally, the Information Gain can be calculated as expressed in Equation (8),

$$Info\_gain(A) = Info\_gain(D) - Info\_gain_A(D) \tag{8}$$

The information gain is typically a selection measure that is highly influenced through test attributes with several end results. Thus, it chooses the attributes selection based on the vast number of values. Hence, *Gain Ratio* is considered as an effort to recover from this problem.

$$Split\text{-}info_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2\left(\frac{|D_j|}{|D|}\right) \tag{9}$$

$$Gainratio(A) = \frac{Gain(A)}{SplitInfo(A)} \tag{10}$$

The *Gini-Index* reflects a binary split for every feature and it is expressed in Equation (11) and (12)

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2 \tag{11}$$

Where, 'p_i' is the probability that a random data point presents in the dataset 'D' fits to class 'C_i'.

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2) \tag{12}$$
$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

(d)Random Forest (RF)

Random forest (RF) is an ensemble tree-based learning algorithm. It consists a group of decision trees from arbitrarily identified subset of training dataset. It collections the votes from various decision trees to agree on the ultimate class of the test data. It generates decision trees on each random data sample and then obtains the prediction from them. Subsequently the best solution is decided based on voting.

It aggregates numerous DT with unique set of observations and the splitting nodes are arrived depends on the various the features. Furthermore, the final forecasting of the RF are made by summarizing the predictions of each individual tree.

(e)Multi-Layer Perceptron (MLP)

A Multilayer Perceptron is defined as a determinate and well directed acyclic graph [24]. The input neurons are not considered as a target point for any connection while the output neurons cannot be considered as a source. Alternatively, hidden neurons are the nodes consist of neither input neurons nor output neurons. Here, every directed link is weighted with a physical number (i.e.), for the connection $i \rightarrow j$, the weight is represented as w_{ji}. When the connection $i \rightarrow j$ exists, such (i) represents the set of entire 'j' neurons present in the model. Similarly, *Pred*(i) is the

function where the set of total neurons '*j*' present in a connection $j \rightarrow i$. Standard perceptrons use a discontinuous function for the activation (i.e.)., a step function as expressed in Equation (13).

$$x_i = f_{step}\left(w_0 + \langle w_i, x_i \rangle \right) \tag{13}$$

But due to technical reasons MLP's use a smooth variant i.e.,

$$x_i = f_{\log}\left(w_0 + \langle w_i, x_i \rangle \right) \tag{14}$$

where

$$f_{\log}(z) = \frac{1}{1 + e^{-z}} \tag{15}$$

(f) AdaBoost

AdaBoost (Adaptive Boosting) is a prevalent improving technique that integrates multiple weak classifiers to construct single strong classifier [25]. In real world, a single classifier cannot accurately determine the class to which an object belongs. So multiple weak classifiers are grouped together and each classifier learns from the wrongly classified instances so that a successful and strong classifier can built.

(g) Naive Bayes

Naive Bayes classifiers is the supervised classification algorithms derived from Bayes' Theorem [26]. It determines the probability of occurrence of an event that is assigned with the probability of another event happened previously. Bayes' theorem is stated mathematically as the following equation:

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{16}$$

where, P(A) is the priori of A and the evidence is an attribute value of an unidentified instance. P(A|B) is posteriori probability of B

$$P(y|X) = \frac{P(X|y)\,P(y)}{P(X)} \quad (17)$$

where, '*y*' is known as a class variable and '*X*' is a response feature vector of size '*n*'. Let us consider, $X = (x_1,x_2,...,x_n)$.To determine the probability of a set of inputs corresponding to the magnitudes of the class variable '*y*', the output with supreme probability is selected. It can be expressed mathematically as follows,

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y) \quad (18)$$

We also used two variants of Naïve Bayes, which are **Gaussian Naïve Bayes** (**GaNB**), and **Bernouli Naïve Bayes** (**BeNB**).

In Gaussian Naïve Bayes (GaNB), a Gaussian distribution is also called Normal distribution where continuous values corresponding to each feature are expected to be distributed based on Gaussian distribution [27]. The probability of the features is expected to be Gaussian, hence, the conditional probability is expressed by:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^{\,2}}} exp\left(-\frac{(x_i - y_i)^2}{2\sigma_y^{\,2}}\right) \quad (19)$$

where, μ_i represents the mean and σ_i represents standard deviation.

In Bernouli Naïve Bayes (BeNB), the attributes are assumed to be independent Booleans that express the inputs as the binary term occurrence attributes instead of term frequencies [28]. All these classifiers are evaluated based on evaluation metrics which are shown in "Results and Discussions" chapter. The best classifier with the good evaluation metrics is chosen for training and testing the data (i.e.)., classify the attacks. The trained model is saved for classifying the new data into mentioned attacks and non-attacks thus, preventing the intrusion and providing security to the systems. In this way, the functionalities of both DPI and Machine Learning are used effectively to prevent the Internet attacks.

4. Results and Discussion

4.1 Experimental Environment

The experiments are conducted on a Windows 64-bit machine with a RAM size of 8GB. The language used for coding is python (version 3.x). The network monitoring tool used is Wireshark. The various ML algorithms are implemented and operated in a sequential and parallel mode to test the performance. The algorithms are evaluated for its efficiency while using DPI and without DPI.

4.2 Data set Preparation

Based on the setup mentioned above, random attacks such as DoS, Impersonation, Fuzzy, non-attack respectively were run in the simulation environment. Standard dataset or public dataset is not available for the specified purpose. Hence, dataset is created for the proposed machine learning model which consists of 10,868 records with seven columns namely protocol, source IP Address, destination IP Address, timestamp, length, data payload, target (refer Figure 4). Corresponding to the data each target class is set. The types of classes set in the data set are class'0', class'1', class'2', class'3' representing DoS, Impersonation, Fuzzy, Non-Attack respectively. This data set is then pre-processed using DPI using the signatures of the attacks and then used for predicting the type of attack using Machine Learning classifiers. The number of attacks in each category (i.e.)., throughput is as follows: DoS - 2233, Fuzzy - 4316, Impersonation - 2187, Non-Attack - 2132.

4.3 Performance Analysis

4.3.1. Throughput with and without firewall

In this section, we would like to analyse the throughput with and without firewall. Figure 6 shows the download throughput of a 1 GB file from the Internet. Here, the without firewall case and throughput observed with firewall (various ML algorithms) are included. It is observed that an average of 150 Mbps traffic flows and while firewall with ML is employed the throughput is comparatively similar to the without firewall case. While comparing the various ML classifiers, KNN has the exact similar results of without firewall cases. It is also noted that KNN for the firewall problem has been implemented with tuned parameters and hence the throughput is similar to without a firewall case.

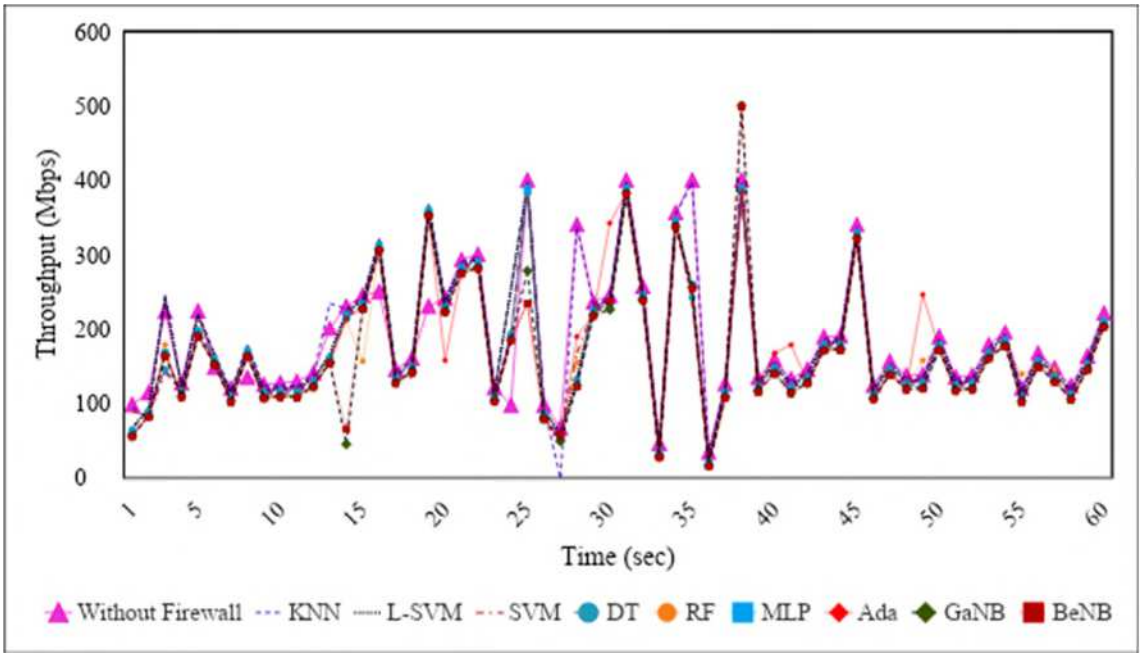

Figure 6. Throughput comparison with and without firewall scenario

4.3.2. Comparison of Pattern Matching Algorithms

We analyze the feature extraction of various pattern matching algorithms, including Boyer-Moore (BM) [29], Knuth-Morris-Pratt (KMP) [30], BMHP, and the Naïve methods. Table 1 shows the comparison results, in terms of time and memory used. The results show that the BMHP algorithm consumes the least time (0.028 sec) and memory (125.4 Mib). Consequently, our architecture adopts the BMHP algorithm for pattern matching.

Table 1. PM algorithm Time and Memory usage statistics

	Pattern Matching Method			
	BM	KMP	**BMHP**	Naive
Sequential Time (s)	0.0054	0.0210	**0.0034**	0.0190
Parallel Time (s)	0.0038	0.0180	**0.0028**	0.0130
Sequential Memory (MiB)	126.5	127.2	**125.4**	127.4
Parallel Memory (MiB)	128.8	130.4	**126.7**	131.5

4.3.3. Comparisons of Various Classifiers

The metrics used for evaluation of the classifiers used in the experiments are (*i*) accuracy, (*ii*) time consumed for training and testing, and (*iii*) memory for training and testing. The accuracy of various classifiers is measured with and without DPI. All the parameters used by the classifiers are default parameters. When DPI is not employed, the accuracy of DT is 63.34 % which is highest among the ten classifiers compared. When DPI is employed, the majority of the classifiers perform well, such that an accuracy of 99.91% is attained by KNN, SVM. L-SVM, DT, RF, MLP, GaNB and BeNB. Subsequently, AdaBoost achieves an accuracy of about 82.52 which is 40% higher without DPI. It shows that DPI plays a major role in boosting the performance of ML classifiers in firewalls. Figure 7 presents the accuracy of various classifiers while used with and without DPI.

Next, the training and testing time consumed by the various ML methods are analyzed. It is observed that SVM consumes a less time of about 18.185 seconds for training and testing. In case of time consumption SVM yields good performance compared to other ML methods used in the present investigation. The testing and training time consumption of various ML techniques are displayed in Figure 8. Further the memory used by the classifier to train and test is presented in Figure 9. Here, it is observed that SVM consumes about 303.5 MiB which is comparatively lesser than the other ML methods used for the investigation.

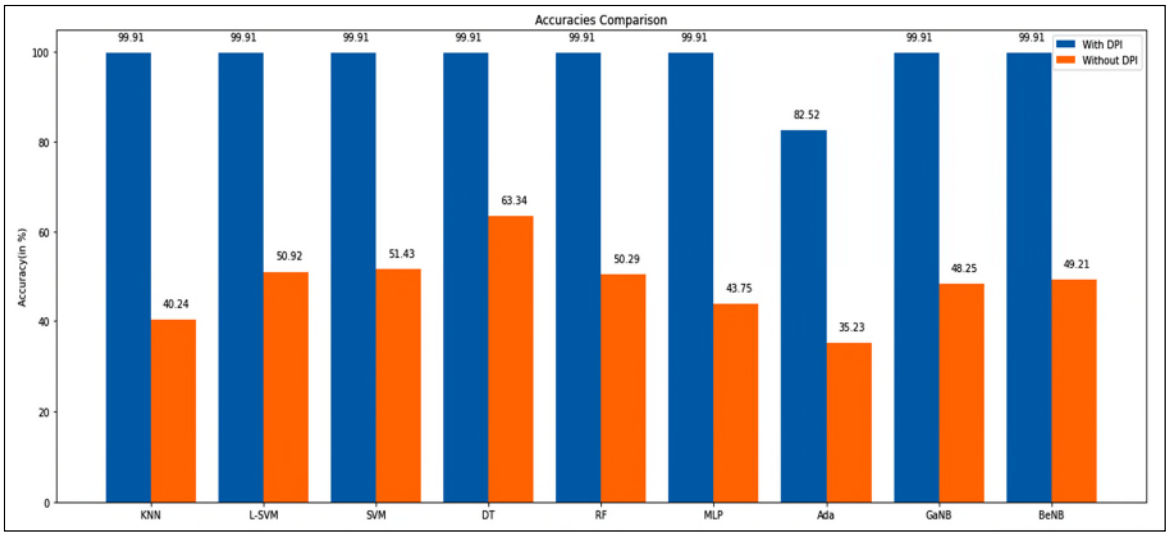

Figure 7. Accuracy of various classifiers with and without DPI

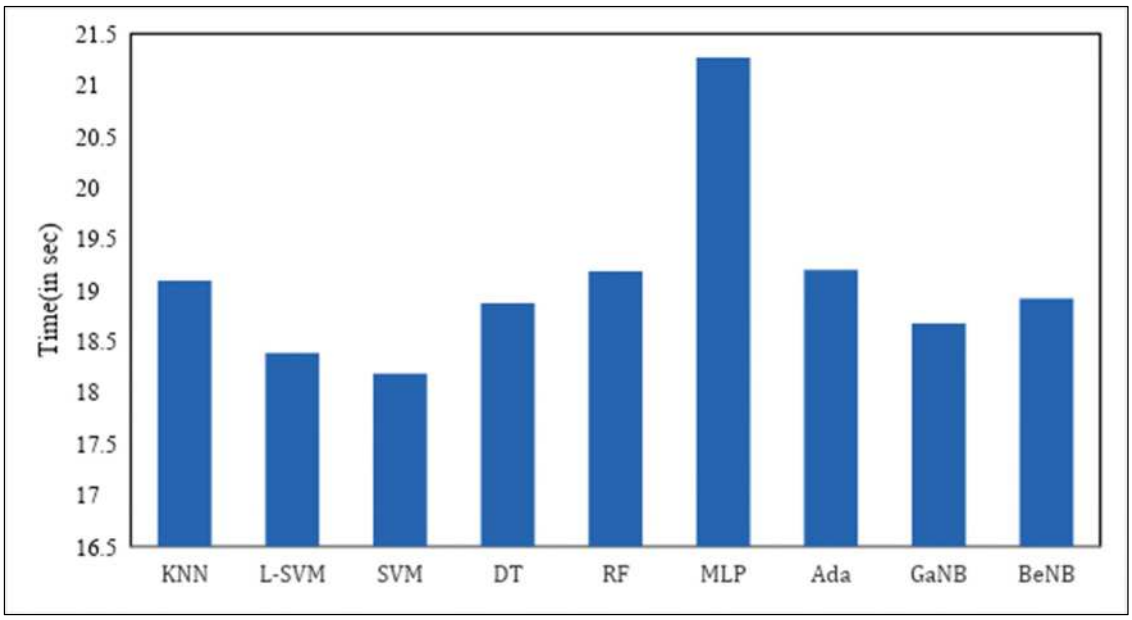

Figure 8. Time consumed by the classifier to train and test

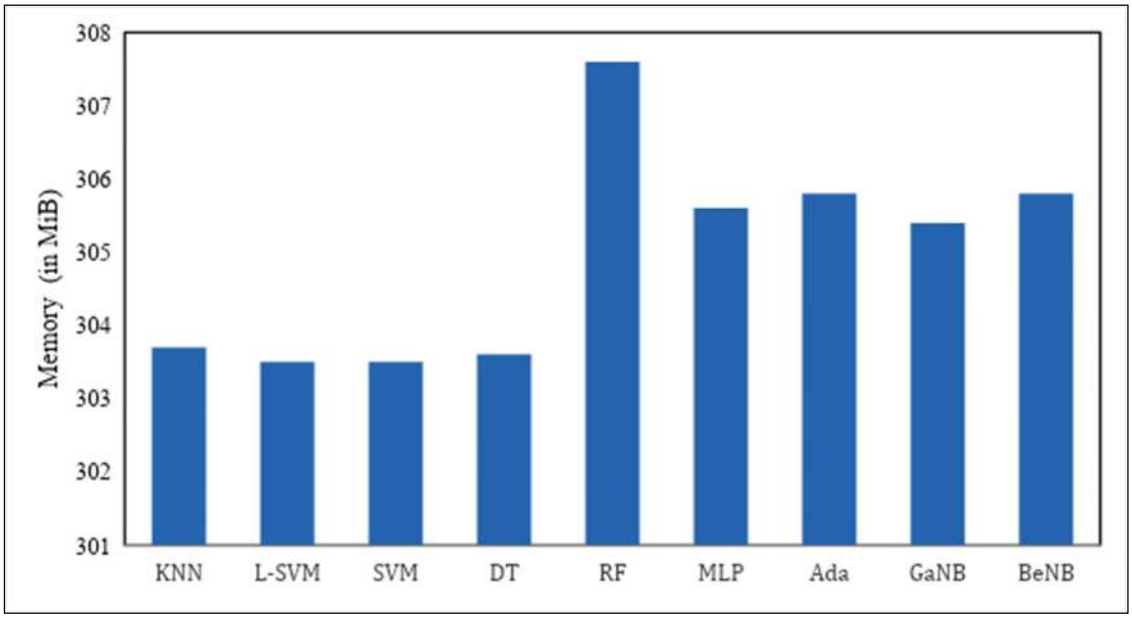

Figure 9. Memory used by the classifier to train and test

Table 2 presents the performance comparison of various ML techniques used in the present investigation. All the classifiers perform well while DPI is employed meanwhile SVM consumes less memory for doing the training and testing process. SVM also consumes less time

to converge for training and testing. Further, it is noted that only when the time required for testing is compared, KNN consumes less time as compared to other methods.

Table 2. Performance Comparison of Classifiers

Metrics	Classifiers								
	KNN	**L-SVM**	**SVM**	**DT**	**RF**	**MLP**	**Ada**	**GaNB**	**BeNB**
Accuracy with DPI (%)	99.91	99.91	99.91	99.91	99.91	99.91	82.52	99.91	99.91
Accuracy without DPI (%)	40.24	50.92	51.43	63.34	50.29	43.75	35.23	48.25	49.21
Testing Time (sec)	1.209	2.237	2.978	1.485	1.518	3.172	1.278	1.348	1.299
Memory (MiB)	303.7	303.5	303.5	303.6	307.6	305.6	305.8	305.4	305.8

Figure 10 displays the various attacks handling potential of the ML based firewall. Totally 1697 attacks are imposed out of which 457 DoS attacks, 407 impersonation attacks and 833 fuzzy attacks and 477 non attacks are exactly identified by the SVM, KNN, L-SVM, DT, RF, MLP, GeNB and BeNB. AdaBoost has given some misclassification such that it returned 457 DoS, 200 impersonation, 348 fuzzy and 1169 non-attacks. The AdaBoosthas some misclassification even when DPI is imposed. Figure 11 displays the response time results of various ML-based firewalls in terms of packet counts. The DT method has less response time such as 5.291sec, 7.182 sec, 9.402 sec, 12.083 sec, 15.524 sec and 18.974 sec for packet counts of 1000, 3000, 5000, 7000, 9000 and 1100 respectively. The highest response time is offered by MLP such as 5.291, 8.126,10.588, 13.346, 16.859 and 20.347 for packet counts of 1000, 3000, 5000, 7000, 9000 and 1100 respectively. MLP is a neural network-based method and it consumes more training time but it is a powerful and accurate mechanism for several classification problems.

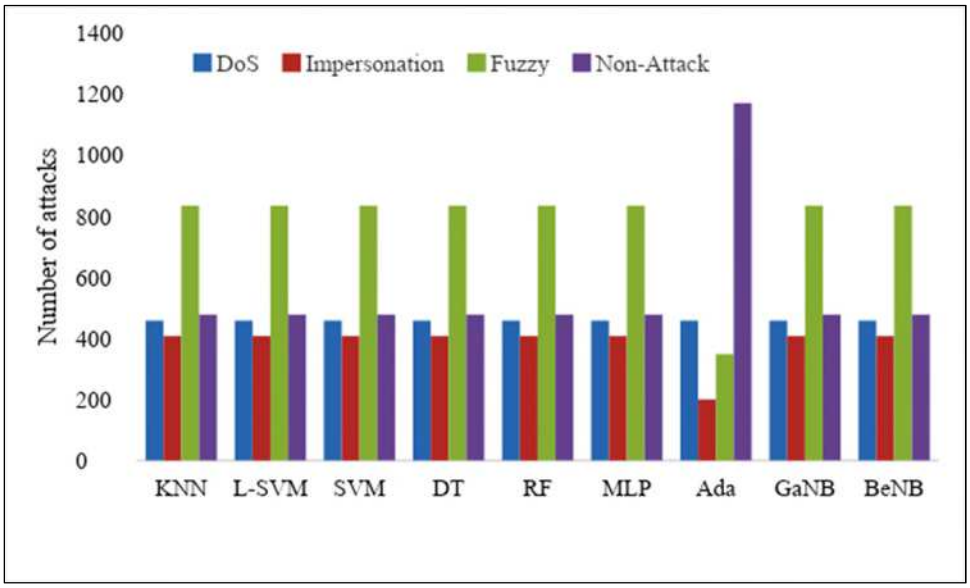

Figure 10. Count of attacks identified by various classifiers

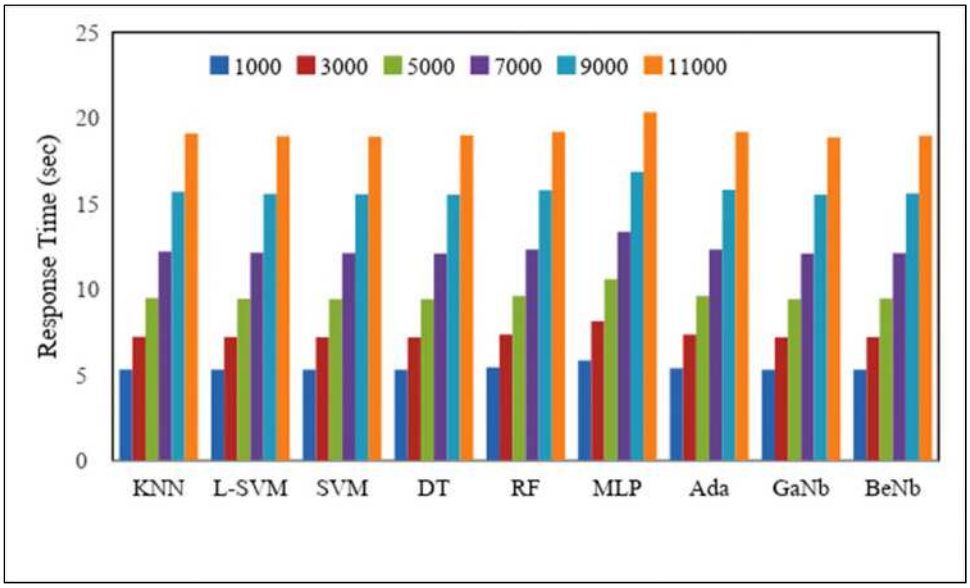

Figure 11. Response Time observed for Packet counts

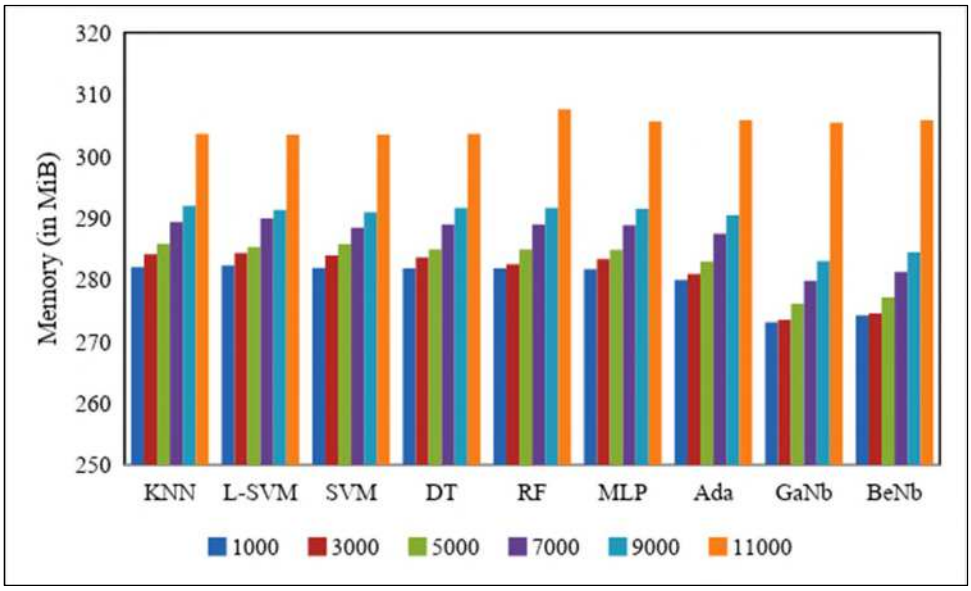

Figure 12. Packet Count Vs Memory Utilization

Figure 12 displays the memory utilization of various classifiers while handling different packets. It is observed that while handling the packets GaNB consumes a less memory of about 273.1 MiB, 273.5 MiB, 276.1 MiB, 279.8 MiB, 283 MiB, and 305.4 MiB for packet counts of 1000, 3000, 5000, 7000, 9000 and 1100 respectively. It shows that while considering individual packets GeNB performs well as compared to other ML methods under investigation. The maximum memory consumption by L-SVM is such that 282.3 MiB, 284.3 MiB, 285.3 MiB, 289.9 MiB, 291.3 MiB, and 303. 5 MiB for the packet counts of 1000, 3000, 5000, 7000, 9000 and 1100 respectively.

From the presented results, it is observed that the best accuracy achieved by the classifiers is about 99.91% by 8 out 9 classifiers with default parameters. In-order to choose the best classifier, it is also needed to consider the time and memory requirements of each classifier which are shown in Figure 8sand 9, and Figures 11 and 12 referring to the time and memory scales respectively. The comparison between the throughput and each attack classified by all the classifiers is shown in Figure 10. Table 2 shows the overall metrics used for choosing the best classifier in terms of accuracy, time and memory. Hence, DPI adds a good advantage to the classification of attacks using Machine Learning as the system monitors the packet and can exactly identify the attack preventing further damage to the system. As long as the signature of the attack is present in the encrypted data of the packet, it can be extracted and monitored using

DPI which is more efficient than many traditional methods. Furthermore, based on the results obtained from the conducted experiments it is evident that the proposed architecture classifies the attacks efficiently.

5. Conclusions

The rapid increase in the usage of the Internet requires the utmost network security that plays a vital role to protect the computing resources. Hence, new methods should be developed to make the completely secured networks in proportion to the increased usage of the Internet. In this paper, DPI and ML firewall is implemented to work in parallel for improving the security and computing efficiency.

Our results show that the BMHP algorithm is the best pattern matching algorithm used for DPI in terms of memory and least time taken. Similarly, the DPI-SVM classifier is preferred synchronously with the BMHP algorithm. DPI-SVM gives the highest accuracy about 99.91% in a less computation time of 18.185 sec, and less memory usage (303.5 MiB). While considering the testing time only, KNN yields good results about 1.209 sec. Hence, combining the functionalities of DPI, ML and operating these technologies in parallel enhances the security of the computing resources connected to the internet.

In the future, the signatures of several attacks other than the attacks mentioned in this proposed architecture will be collected for efficient processing.

Ethics declarations

Conflict of interest

The authors declare that they have no conflict of interest.

Authorship contributions: All authors contributed equally to this work.

References

1. M.Z.A. Aziz, M. Y. Ibrahim, A.M. Omar, R.A. Rahman, and M.I. Yusof, “Performance analysis of application layer firewall. In 2012 IEEE Symposium on Wireless Technology and Applications (ISWTA), pages 182–186.IEEE, 2012.

2. C.Y. Jeong, S.Y.T. Lee and J.H. Lim, “Information security breaches and IT security investments: Impacts on competitors”, Inf. Manag., 56(5)(2019), 681-695.
3. M.R. Lyu and L.K.Y. Lau, Firewall security: Policies, testing and performance evaluation. In Proceedings 24th Annual International Computer Software and Applications Conference. COMPSAC2000, pages 116–121. IEEE, 2000.
4. W. Gang, H. Jinxing, M. Jian, H. Lihua, “A new approach to intrusion detection using artificial neural networks and fuzzy clustering”, Expert. Syst. Appl., 2010; 37(9)(2021) 6225–6232.
5. W. Fang, X. Tan, and D. Wilbur, Application of intrusion detection technology in network safety based on machine learning, Safety Science, 124:104604, 2020.
6. M. Nawir, A. Amir, N. Yaakob, and O.B. Lynn, Internet of things (iot): Taxonomy of security attacks. 3rd International Conference on Electronic Design (ICED), pages 321–326. IEEE,2016.
7. I. Dubrawsky, Firewall evolution-deep packet inspection, Security Focus,29, 2003.
8. G.D.L.T. Parra, P. Rad, K.K.R. Choo, “Implementation of deep packet inspection in smart grids and industrial Internet of Things: Challenges and opportunities”, J. Network Comp. Appl., 135(2019) 32-46.
9. A. Gandomi and M. Haider, Beyond the hype: Big data concepts, methods, and analytics. International journal of information management, 35(2) (2015):137–144.
10. A. Amouri, V.T. Alaparthy, and S.D. Morgera. A machine learning based intrusion detection system for mobile internet of things. Sensors, 20(2) (2020):461.
11. H. Lee, S.H. Jeong, and H.K. Kim, Otids: A novel intrusion detection system for in-vehicle network by using remote frame.48 In 2017 15th Annual Conference on Privacy, Security and Trust (PST), pages 57–5709. IEEE, 2017.
12. S. Rawat, A. Srinivasan, R. Vinayakumar, Intrusion detection systems using classical machine learning techniques versus integrated unsupervised feature learning and deep neural network. arXiv preprint arXiv:1910.01114, 2019.
13. F. Ertam, “An efficient hybrid deep learning approach for internet security”, Physica A: 535(2019) 122492.

14. J.J. Praise, R.J.S. Raj, and J.V.B. Benifa, Development of Reinforcement Learning and Pattern Matching (RLPM) Based Firewall for Secured Cloud Infrastructure. Wireless Pers Commun, 115(2020) 993-1018.
15. D. Cantone and S. Faro. Fast-search: A new efficient variant of the Boyer-Moore string matching algorithm. In International Workshop on Experimental and Efficient Algorithms, pages 47–58. Springer, 2003.
16. S. Cho, J.C. Na, K. Park, and J.S. Sim. A fast algorithm for order-preserving pattern matching. Information Processing Letters, 115(2)(2015), 397-402.
17. R. Cole. Tight bounds on the complexity of the Boyer–Moore string matching algorithm. SIAM Journal on Computing, 23(5):1075–1091, 1994.
18. H.M. Mahmoud, R.T. Smythe, and M.R.́Egnier. Analysis of boyer-moore-horspool string-matching heuristic. Random Structures & Algorithms, 10(1-2)(1997):169–186.
19. S.O. Belkasim, M. Shridhar, M. Ahmadi, "Pattern classification using an efficient KNNR", Pattern Recognition, 25(10)(1992) 1269-1274.
20. M. Brown, S.R. Gunn, H.G. Lewis, "Support vector machines for optimal classification and spectral unmixing", Ecological Modelling, 120(2–3)(1999), 167-179.
21. G. Zhu, C.G. Yang, P. Zhang, "Linear programming ν-nonparallel support vector machine and its application in vehicle recognition", Neurocomputing, 215(2016) 212-216.
22. I.K. Sethi, J.H. Yoo, "Structure-driven induction of decision tree classifiers through neural learning", Pattern Recognition, 30(11)(1997), 1893-1904.
23. Y. Zhu, W. Xu, G. Luo, H. Wang, J. Yang, W. Lu, "Random Forest enhancement using improved Artificial Fish Swarm for the medial knee contact force prediction", Artificial Intelligence in Medicine, 103, 2020: 101811.
24. X. Feng, G. Ma, S.F. Su, C. Huang, M.K. Boswell, P. Xue, "A multi-layer perceptron approach for accelerated wave forecasting in Lake Michigan", Ocean Engineering, 211(2020): 107526.
25. P. Bahad, P. Saxena (2020) Study of AdaBoost and Gradient Boosting Algorithms for Predictive Analytics. In: Singh Tomar G., Chaudhari N., Barbosa J., Aghwariya M. (eds) International Conference on Intelligent Computing and Smart Communication 2019. Algorithms for Intelligent Systems. Springer, Singapore.

26. H.C. Kim, J.H. Park, D.W. Kim, J. Lee, “Multilabel naïve Bayes classification considering label dependence”, Pattern Recognition Letters, 136(2020) 279-285.
27. M.O. Ortega, A.L. Castellanos, G. Valente, R. Goebel, M.V. Sosa, “Fast Gaussian Naïve Bayes for searchlight classification analysis”, Neuro Image, 163(2017), 471-479.
28. S. Wang, L. Jiang, &C. Li, Adapting naive Bayes tree for text classification. Knowl Inf Syst 44(2015), 77–89.
29. D. Cantone and S. Faro, Fast-search: A new efficient variant of the Boyer-Moore string matching algorithm. In International Workshop on Experimental and Efficient Algorithms, pages 47–58. Springer, 2003.
30. D.E. Knuth, J.H. Morrjs, and V.R. Pratt, “Fast pattern matching in strings”, SIAM J. Comput. 6 (2)(1977), 323-350.