Wireless Personal Communications (2020) 115:993–1018
https://doi.org/10.1007/s11277-020-07608-4

Development of Reinforcement Learning and Pattern Matching (RLPM) Based Firewall for Secured Cloud Infrastructure

J. Jeya Praise[1] · R. Joshua Samuel Raj[2] · J. V. Bibal Benifa[3]

Published online: 24 June 2020

Abstract
Cloud computing infrastructure is typically intended to store and deliver sensitive data and high performance computing resources through the internet. As the utility of cloud computing has increased to larger extend because of its sophisticated services, the security breaches also growing proportionately in terms of third party attacks. In order to mitigate the modern security attacks in the cloud environment, the traditional firewall rules and packet filtering methods are absolutely insufficient. Hence, a Deep Packet Inspection based firewall (RLPM) is developed to block the malicious attacks by validating the payload signature of arriving packets. RLPM combines the potential of Reinforcement Learning (RL) and parallel fast pattern matching simultaneously and it converges to an optimal solution at the earliest. RL method efficiently learns the environment and process the payload signature in a parallel manner. A two-way pattern matching algorithm is integrated with RL approach that validates the signature towards attaining the quick decisions. The performance results show that the RLPM is better as compared to the existing methods in terms of Response time, throughput and malicious attack blocking. As the firewall is deployed and tested in a real cloud computing environment, the response time is found to be 10% lesser while throughput is also increased about 10% than the existing state-of-the-art-methods.

Keywords Cloud infrastructure · Packet filtering · DPI · Signature generation · Pattern matching

✉ J. Jeya Praise
jeyapraise2018@gmail.com

[1] Anna University, Chennai, India

[2] Rajaas Engineering College, Nagercoil, India

[3] Indian Institute of Information Technology, Kottayam, India

1 Introduction

Cloud computing is a distributed model to dispense the computing, storage and networking resources without direct administration by the end-user. The release of cloud computing services to the end-user includes servers, storage, databases, networking tools and softwares through the internet [1]. Currently, every organization is intended to host their private data and specify the computing requirements in cloud infrastructures that are conversely vulnerable to security breaches for a large extend. A security breach leads to unlawful access of sensitive data, scientific applications, computing services, and networking services by thwarting their fundamental defense mechanisms [2]. A security violation happens once an application illegally pierces into a concealed or virtual IT perimeter and it is one of the primitive points of attack by an intruder. Primarily, security breaches are caused by malware [3], Distributed Denial of Service (DDoS) [4], Phishing [5], SQL injection [6], Trojan attacks [7], cross site scripting [8], port and IP based attacks [9]. Malware is a program that is deliberately designed to cause damage to a computing resource. It is well-known that several categories of malware exist in the computing environment that includes computer viruses, worms and Trojan horse [[3], [7]]. A DDoS attack is enforced by several compromised systems that attack a target (Server) and causes a denial of service for users who are intended to utilize the targeted computing resources [4]. Phishing is the fraudulent effort to gain sensitive data (Example: credit card details) by concealing oneself as a reliable entity in the communication systems [5]. SQL Injection is an injection attack that runs malicious SQL statements that compromises a database server exists behind a web application [6]. In the Cross-Site Scripting (XSS) attacks, the malicious scripts are injected into trusted websites. XSS attacks occur while an attacker exploits a web application to direct malicious code in the form of a browser side script to an end user [8]. A port scan is an attack that directs the client requests to a range of server port addresses on a host, with the goal of identifying an active port and exploiting a known vulnerability of that service [9]. In an IP address-based attack, an attacker forwards IP packets from a false source address in order to disguise itself. Further, DDoS attackers often use IP spoofing to overload the networks and devices with packets that appear to be from legitimate source IP addresses [9]. Hence, an efficient DPI based firewall system is needed to mitigate the security breaches in a real cloud computing environment and the deployment of typical cloud firewall scheme is presented in Fig. 1.

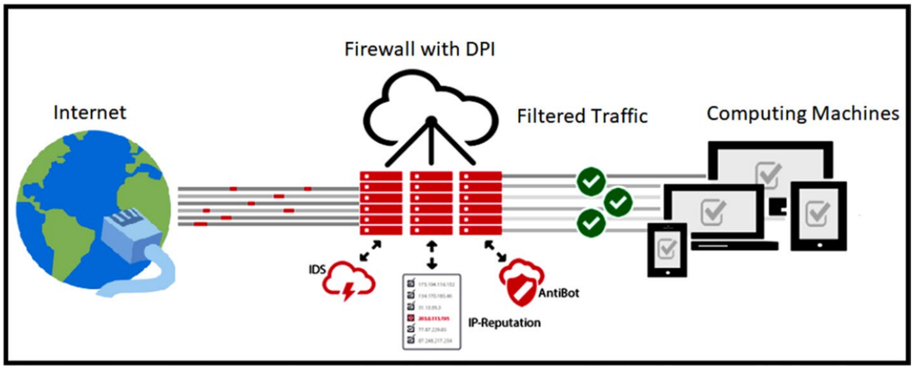


Fig. 1 Traffic flow between public and private network through Firewall

Firewalls are known to be the core facet to enhance network security [10]. The complication associated with the administration of firewall policy set bounds the security of cloud infrastructure. In addition to this, organization of firewall policies for protecting a data centre is ambiguous and error-prone task. Firewall filtering rules have to be established and dispersed in a subtle manner in order to circumvent policy anomalies that might persuade the data centre towards vulnerable attacks. Therefore, amending filtering rules in any policy set necessitate meticulous investigation to conclude the appropriate rule placement in the firewalls. A firewall rule includes set of fields (also called network fields) such as protocol type, source IP address, destination IP address, source port, destination port, and an action field.

Every network field is a distinct value or set of values where filtering actions are to accept or to deny a particular packet. The packet is allowed or blocked using a definite rule only if the header information matches with every fields of this rule. Otherwise, the rule is further evaluated and the process is iterated until an identical rule is found or the default action is executed. Deep Packet Inspection (DPI) is a complex method of investigating packets and handling the network traffic. It is a form of packet filtering technique that establishes, recognizes, categorizes, and reroutes or blocks the packets with malicious code payloads rather than conventional packet filtering methods that scrutinize only the packet headers [11]. DPI is a critical component in next generation networks for ensuring security, content filtering, traffic monitoring, load balancing, lawful interception, targeted advertising, data leakage prevention, and application-aware routing. DPI scrutinizes the payload of packets moving towards a specified checkpoint and formulates decisions based on the policies allocated by an Internet Service Provider (ISP) or network administrator. DPI facilitates the service provider to examine the internet traffic and to distinguish them according to the payload information. The packet header contains the origin and destination addresses and other information relevant to move the packet across the network. In the conventional packet filtering method, the “payload” of the packet includes the text, images, and files or applications transmitted by the user that are not considered to be a serious concern to the operator. DPI also allows the network operators to scan the payload of IP packets as well as the header information. The domain of packet inspection adopted in internet protocols is displayed in Fig. 2.

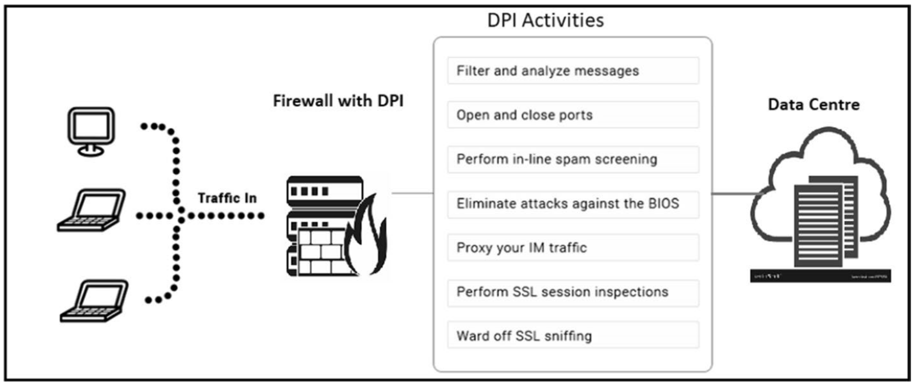

DPI Activities
Firewall with DPI
Traffic In
Filter and analyze messages
Open and close ports
Perform in-line spam screening
Eliminate attacks against the BIOS
Proxy your IM traffic
Perform SSL session inspections
Ward off SSL sniffing
Data Centre

Fig. 2 Application of DPI-based firewall in real time processing

Generally, DPI systems employ expressions or features to characterize the patterns of interest in a specific network data stream. In the proposed work, a firewall filtering method is devised to inspect the packets based on the generated rules. It combines the RL approach and Pattern Matching method to secure the cloud computing infrastructure. It would be useful to mention about the road map of the proposed work at this point. Here, Sect. 2 presents the related works and Sect. 3 highlights the proposed methodology. Then, Sect. 4 enumerates the features of experimental setup utilized along with the results & discussion and Sect. 5 narrates the concluding remarks.

2 Related Work

The related works pertaining to intrusion detection scheme that employs digital signature mechanism to resolve the problems of data loss and malicious attacks are studied elaborately in this section. In the recent works, a malicious node is assumed to exist as an intermediate node between the source and destination nodes [12, 13]. These malicious nodes illegitimately separate the packets before transferring it to the next node and thereby linking it to the source node through a fabrication process [14]. If the source receives the acknowledgement signal from the destination within a predefined time then it is assumed that a transmission is completed or else, it is declared as unsuccessful. Techniques for IDS implementation also employ a Multi Agent System IDS known as MASID, where a collection of predefined agents carry out the function of perceiving an anomalous behavior [15, 16]. In a particular case, if there is no reason exists to prove the malicious behavior, then the local agents are combined in a cooperative manner to justify the abnormal behavior by providing additional information. One of the successful techniques to classify the malicious dataset is Multiple Kernel Fuzzy C-means (MKFC) algorithm that extends the fuzzy-cas algorithm [17].

An anomaly detection method known as delayed Long Short-Term Memory (dLSTM) constructs a predictive model using time-series (training) data. This model performs anomaly detection based on the prediction error computed from the observed data. Here, multiple states exist in the waveforms of normal data that leads to lower prediction accuracy [18]. A Machine Learning (ML) based method employs Artificial Neural Networks (ANN), decision trees, nearest neighbor, and Support Vector Machine (SVM) for intrusion detection application along with distributed firewall logs [19]. However, the efficiency of the aforementioned methods depends on consistency of training dataset. Delayed Signature Matching (DSM) is a DPI method and it reduces the signature matching attempt while creating the firewall rules [20]. COmpression INspection (COIN) method is used for multi-pattern matching over compressed traffic and it does not recheck the patterns within the compressed data segment that was matched previously. It is being tested in a traffic generated by Alexa top sites and little improvements have been observed during the experiments [21]. A distributed packet tracing method is proposed by Lukashin et al. (2014) using MapReduce model to process terabytes of data and it has shown some improvements in the prediction of malicious behaviour in real data [22].

A self adaptive method is proposed by Junior et al. (2018) to deploy the distributed firewall that overcomes the software vulnerabilities exist in traditional distributed firewall systems [23]. The proposed Self-Adaptive Distributed Firewall (SADF) monitors the hosts by employing a vulnerability assessment system with firewall rules that identifies the susceptible attacks on affected hosts. Here, the system is tested in a simulated environment and it is well-known that the performance of SADF in real-distributed environment is indefinite. Hence, the uncertainties exist in the real-distributed environment were not included in the SADF based investigation. Recently, the ML based methods for NIDS applications have been revealed greater improvements in malicious packet classification [24]. Khan and Gumaei (2019) employed various ML methods for intrusion detection and methods such as Decision Tree (DT), Random Forests (RF), Hoeffding Tree (HT), and K-Nearest Neighbors (KNN) classifiers are applied on KDD99 and UNSW-NB15 datasets where it shows reasonable accuracy for malicious packet classification.

Amanullah et al. (2020) reviewed various Deep Learning (DL) methods to address the issues associated with the security enhancement of cloud linked IoT devices [25]. Previously, DL based framework was proposed by Ertam (2019) for generating firewall rules and it includes data acquisition from Firewall by following the feature selection and classification processes [26]. Here, the LSTM, Bi-directional LSTM and SVM are utilized as classifiers. It is observed from the results that the DL approach is more successful than the conventional SVM classifiers where the highest classification accuracy about 97.38% can be achieved by Bi-LSTM-LSTM based hybrid networks. As a summary, few state-of-the-art DPI methods and their potential applications in network security are listed in Table 1.

3 Proposed Work

The programme of activities and sequence of operations followed in the proposed work are presented in Fig. 3.

1. Initially, the input data passes through the firewall and from the input data log information, the log processor classifies and generates new rules.
2. By comparing the generated new rules with network performance log data, optimized rules are generated and updated in the firewall rule dataset. Subsequently, based on the updated rules in firewall rule dataset, the packet is either allowed or denied to enter into the network.
3. The above process is iterated for every packet data that flows in the network. The relevant packets are identified by payload signature matching process with valid signature.
4. The irrelevant packets are also classified in the same fashion where the payload signature does not match with the valid signature or it matches with the attack-based signatures exist in the database.

3.1 Creation of Training Data Set

Initially, the training dataset are generated for Malware [35], DDoS [36, 40], Phishing [37], SQL injection [38, 39], cross site scripting [40], Trojan behavior [41], port and IP scanning [41, 42] using Elliptic Curve Digital Signature Algorithm (ECDSA) (Refer Algorithm 1) [43]. In addition to this, the basic attack signature including HTTP, SSL, TCP Stream, ICMP and Layer 3/Layer 4 address-Based Signatures are also added along with the dataset. ECDSA is an efficient algorithm for signature generation and it provides similar results (in terms of security) like other DSA algorithms however, the key value is comparatively smaller. Here, the anomalous behaviour signature in training datasets are balanced and collectively made available in a common database. The custom signatures are generated and updated for each case specified in the common database. Figure 4 presents the sample

Table 1 Applications of DPI in network security

Sl. No	Method	Application	Feature	Algorithm	Scope
1	Static Application Signature [27]	P2P applications	Signature	Manual Signature identification and annotation	Traffic classification
2	Automated Construction of Application Signatures (ACAS) [28]	Classification of FTP, SMTP, POP3, HTTP, SSH	Header information	Bayes and Adaboost	Traffic classification
3	Autograph [29]	Intrusion detection	Heuristics	Content based signature creation	Traffic classification
4	Hamsa [30]	Intrusion detection	Payload signature	Greedy algorithm	Classification of worms
5	Light Weight- Deep Packet Inspection (LW-DPI) [31]	Chat, Streaming and Mail	Number of Packets and length of payload	String Matching Algorithm	Classification
6	LASER [32]	Limewire & BiTorrent	Number of Packets and length of payload	Least Common subsequence	Classification
7	Speculative parallel pattern Matching (SPPM) [33]	Packet Inspection	Regular Expressions	stride-*k* DFA	Pattern Matching
8	Space efficient deep packet inspection [34]	Traffic Analysis	Number of Packets	compressed web traffic	Traffic classification

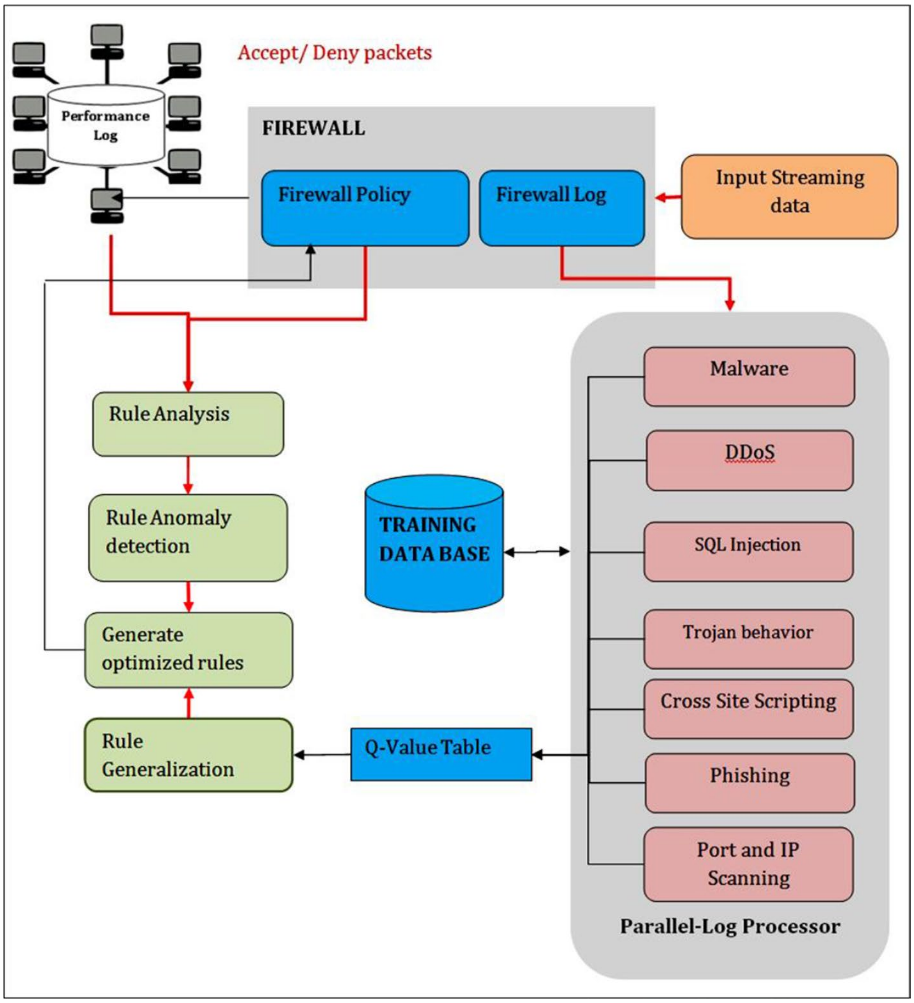
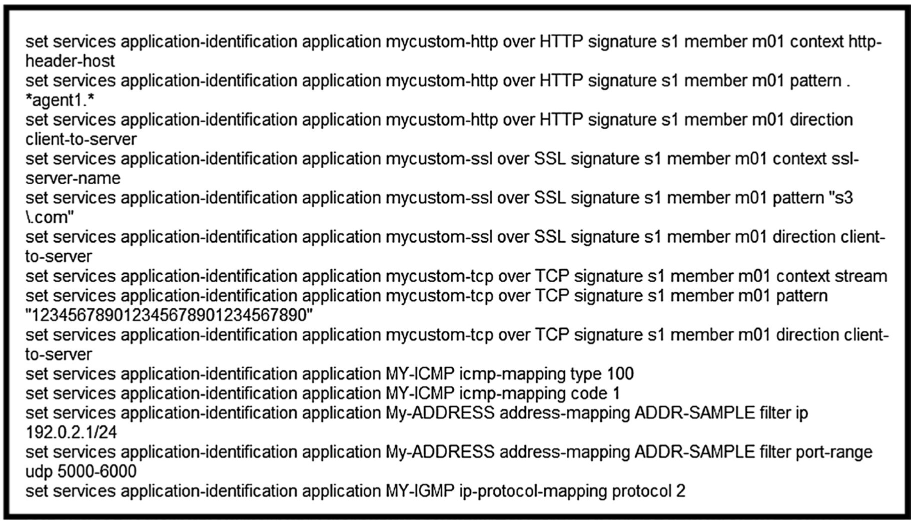


Fig. 3 High level flow of the proposed work (RLPM)

commands for HTTP, SSL, TCP Stream, ICMP and Layer 3/Layer 4 Address-Based Signatures/Filtering options.

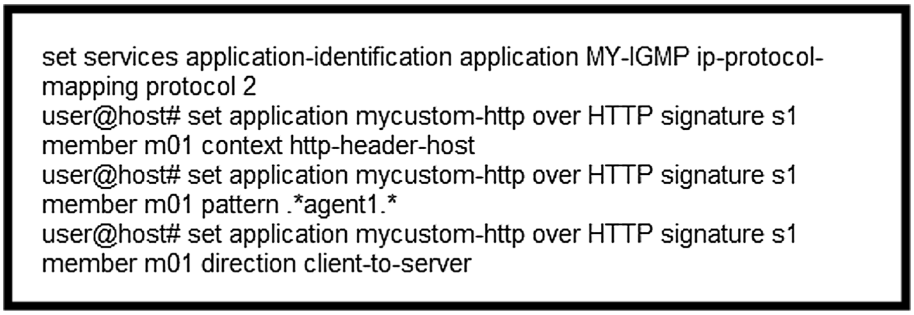
The configurations of basic HTTP, SSL, and TCP stream based custom signatures are presented in Figs. 5, 6, and 7 respectively. In addition, Fig. 8 shows the ICMP and Layer 3 or Layer 4 based custom signatures. Similarly, other advanced attack signatures are also updated to the common database for initial firewall rules.

Initially, the concrete firewall rules are generated from the publicly available dataset [31–38]. The created firewall policy has the fields such as Order, Protocol, Source IP, destination IP, source port, destination port and the payload signature to be denied. Hence, every incoming packet undergoes initial inspection through concrete firewall rules. The sample view of initially generated firewall rules is presented in Table 2. If any of the fields matches with the incoming packet flow then such packets will be discarded or allowed immediately by the firewall based on the action value. Meanwhile, if any irrelevant packets



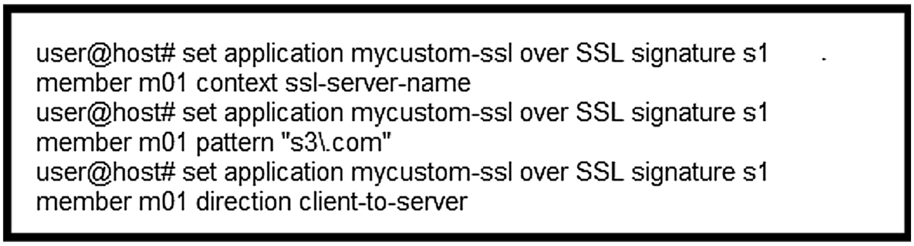
```
set services application-identification application mycustom-http over HTTP signature s1 member m01 context http-
header-host
set services application-identification application mycustom-http over HTTP signature s1 member m01 pattern .
*agent1.*
set services application-identification application mycustom-http over HTTP signature s1 member m01 direction
client-to-server
set services application-identification application mycustom-ssl over SSL signature s1 member m01 context ssl-
server-name
set services application-identification application mycustom-ssl over SSL signature s1 member m01 pattern "s3
\.com"
set services application-identification application mycustom-ssl over SSL signature s1 member m01 direction client-
to-server
set services application-identification application mycustom-tcp over TCP signature s1 member m01 context stream
set services application-identification application mycustom-tcp over TCP signature s1 member m01 pattern
"123456789012345678901234567890"
set services application-identification application mycustom-tcp over TCP signature s1 member m01 direction client-
to-server
set services application-identification application MY-ICMP icmp-mapping type 100
set services application-identification application MY-ICMP icmp-mapping code 1
set services application-identification application My-ADDRESS address-mapping ADDR-SAMPLE filter ip
192.0.2.1/24
set services application-identification application My-ADDRESS address-mapping ADDR-SAMPLE filter port-range
udp 5000-6000
set services application-identification application MY-IGMP ip-protocol-mapping protocol 2
```

Fig. 4 Sample configuration of HTTP, SSL, TCP Stream, ICMP and Layer 3/Layer 4 Address-Based Signatures/Filter options



```
set services application-identification application MY-IGMP ip-protocol-
mapping protocol 2
user@host# set application mycustom-http over HTTP signature s1
member m01 context http-header-host
user@host# set application mycustom-http over HTTP signature s1
member m01 pattern .*agent1.*
user@host# set application mycustom-http over HTTP signature s1
member m01 direction client-to-server
```

Fig. 5 Configuration of HTTP signature

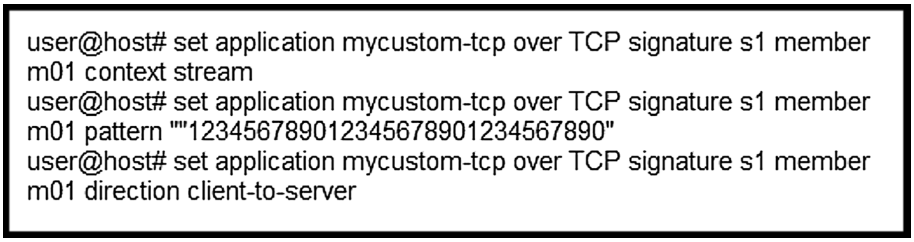


```
user@host# set application mycustom-ssl over SSL signature s1
member m01 context ssl-server-name
user@host# set application mycustom-ssl over SSL signature s1
member m01 pattern "s3\.com"
user@host# set application mycustom-ssl over SSL signature s1
member m01 direction client-to-server
```

Fig. 6 Configuration of SSL signature

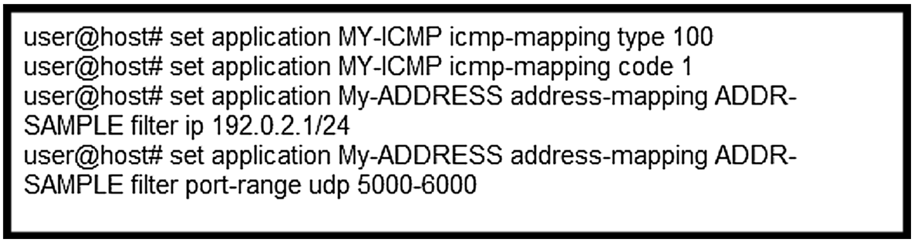
arrive then it is subjected to the log processor for inspection. The process of configuring the initial signatures is presented in Figs. 4, 5, 6, 7, and 8.

The action value ‘1’ indicates the packet to be denied and ‘0’ indicates the packet that can be allowed. For signature generation of the incoming packets, ECDSA signature



```
user@host# set application mycustom-tcp over TCP signature s1 member
m01 context stream
user@host# set application mycustom-tcp over TCP signature s1 member
m01 pattern ""123456789012345678901234567890"
user@host# set application mycustom-tcp over TCP signature s1 member
m01 direction client-to-server
```

Fig. 7 Configuration TCP signature



```
user@host# set application MY-ICMP icmp-mapping type 100
user@host# set application MY-ICMP icmp-mapping code 1
user@host# set application My-ADDRESS address-mapping ADDR-
SAMPLE filter ip 192.0.2.1/24
user@host# set application My-ADDRESS address-mapping ADDR-
SAMPLE filter port-range udp 5000-6000
```

Fig. 8 Configuration of ICMP mapping and Layer 3/4 address

Table 2 Sample view of Firewall Rule set

Protocol	SIP	SP	DIP	DP	SIGNATURE	Action
IP	114.192.137.2	5008	140.192.13.1	4569	Sdffg1234	1
UDP	167.192.12.7	5169	161.172.117.2	6732	2435sdfgj	1
TCP	140.192.37.2	4567	140.34.12.0	9045	Sad4566gh	0

generation algorithm is employed and it is presented in Algorithm 1. The ECDSA algorithm is employed because of its prompt and lightweight nature. The notations used for various algorithms are presented in Table 3.

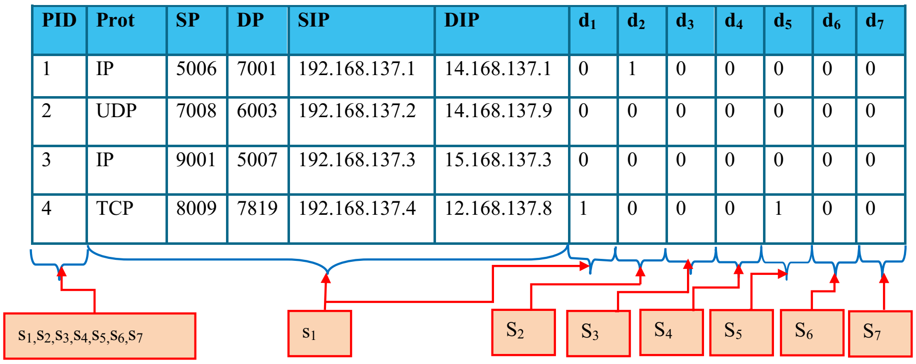
```
Algorithm 1: SignatureGeneration
Input: Packet $P_i$, PayloadData$PL_i$
Output: Generated Signature $G_i$
Compute $e = H(PL_i)$
Select a random integer $c \in [1, h-1]$
Compute $W = (x_1, y_1)$
Compute $r = x_1 \bmod h$
Compute $G_i = c^{-1}(e + zr) \bmod h$
Return $G_i$
```

Table 3 Notations used in the research work

Notation	Description
e	Hash value
c	Random integer
W	Curve point
G_i	Signature of payload
z	Key value
S	State set
$\{s_1, s_2, ..s_n\}$	Slave agents
A	Action value
D	Database
FR	Firewall rule
T	Time period
P	Packet
H	Header
PL	Payload
x	Pattern search string
DL-RT-F	Download response time with firewall
DL-RT-WF	Download response time without firewall
UL-RT-F	Upload response time with firewall
UL-RT-WF	Upload response time without firewall
RT	Response time
RT-F	Response time with firewall
RT-WF	Response time without firewall
Q	Q-value table
Q_g	Generalized Q value table

3.2 Reinforcement Learning Based Pattern Matching

Reinforcement learning is a robust and adaptive method that learns the environment and converges to an accurate solution [44]. Initially, seven different slave agents are used to process the incoming payloads and headers. The header data is subjected to ‘d_1’ database that contains the malicious IP and Port numbers. The payload data is processed and its signature is sent to remaining six agents that compares with the databases d_2, d_3,…,d_7. These databases contain information about the various Malware [35], DDoS [36], Phishing [37, 38], SQL injection [39], Trojan behavior [41], and cross site scripting [40]. The overall workflow is presented in Algorithm 2 and the pattern matching process is separately presented in Algorithm 3.



PID	Prot	SP	DP	SIP	DIP	d_1	d_2	d_3	d_4	d_5	d_6	d_7
1	IP	5006	7001	192.168.137.1	14.168.137.1	0	1	0	0	0	0	0
2	UDP	7008	6003	192.168.137.2	14.168.137.9	0	0	0	0	0	0	0
3	IP	9001	5007	192.168.137.3	15.168.137.3	0	0	0	0	0	0	0
4	TCP	8009	7819	192.168.137.4	12.168.137.8	1	0	0	0	1	0	0

Fig. 9 Q-Value Table and parallel update of slave agents to different fields

In Algorithm 2, initially the header and payload is segregated and the header is sent to slave agent 's_1'. The slave agent compares the header with the 'd_1' using pattern matching algorithm and the Packet ID, SP, SIP, DP, and DIP along with the action to be performed are updated to the Q-Value Table. The layout of Q-Value table and the field updating process is presented in Fig. 9. Similarly, the signature of payload is generated and compared using pattern matching function. Further, packet ID and the corresponding compared results are also updated to the Q-Value Table (Fig. 9). For example, if the slave node s_2 processing the malware database d_2, and observes that the signature G_i matches with the database (d_2) then the action value '1' is returned to the Q-Value table. In the same fashion, if there are no matches found then the value '0' is returned to the Q-Value table.

Algorithm 2. Reinforcement Learning based Pattern Matching

Input: Slave agent set $S=\{s_1,s_2,..s_n\}$, Action A= $A=\{Accept=0,Deny=1\}$, Database $Firewall\ Rule\ FR$, Time Period T, Packet P, Header H, Payload PL,

Output: Generate Q-Value Table and Q_g- Value Table

```
For each packet $P_i$
    Extract Header $H_i$
        Extract $SIP_i$, $SP_i$, $DIP_i$, DP from $H_i$
        Send $(P_i, SP_i, SIP_i, DP_i, DIP_i)$ to $s_1$
    Extract Payload $PL_i$
        ECDSA()
        Generate $G_i$
        Send ( $G_i$ ) to $s_2,s_3,s_4,s_5,s_6,s_7$
For slave agent $s_1$ do
    Receive $(P_i, SP_i, SIP_i, DP_i, DIP_i)$
    PatternMatching()
    Receive_Action $(P_i, SP_i, SIP_i, DP_i, DIP_i)$
    Update $Q \leftarrow (P_i, SP_i, SIP_i, DP_i, d_1) \wedge A(P_i, SP_i, SIP_i, DP_i, d_1)$
For each slave agent $s_2\ to\ s_7$ do
    Receive ()
    PatternMatching()
    For slave agent $s_{i-1}$ do
        Receive_Action $(P_i, d_{i-1})$
        Update $Q \leftarrow (P_i, d_{i-1})$
For every time Period $T_i$
Generate $Q_1$- Table
        For each $P_i$ in Q
Update $Q_1$, FR
```

```
**Algorithm 3: PatternMatching**(x,D)
**Input** $x = \{SP_i, SIP_i, DP_i, DIP_i, G_i\}$, $D = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$
**Output** $P_i, SP_i, SIP_i, DP_i, DIP, Action\,value = \{0,1\}\ for\ P_i$
Receive $(P_i, SP_i, SIP_i, DP_i, DIP_i) \wedge G_i$
Serach_pattern $(SP_i, SIP_i, DP_i, DIP_i) \,||\, d_1 \wedge G_i \,||\, d_2 to\, d_7$
$l_1 \leftarrow length(SP_i)$
$l_2 \leftarrow length(DP_i)$
$l_3 \leftarrow length(SIP_i)$
$l_4 \leftarrow length(DIP_i)$
$l_5 \leftarrow length(G_i)$
For each $l_i$
    $(n_1, k_1) \leftarrow \max\_suf(l_1, \leq)$
    $(n_2, k_2) \leftarrow \max\_suf(l_1, \subseteq)$
    $if\,(n_1 \geq n_2)\,then\,\{$
    $n \leftarrow n_1; k \leftarrow k_1$
    $\}\,else\,\{$
    $n \leftarrow n_2; k \leftarrow k_2$
    $\}end\,if$
    $if\,(n < l_i/2\,and\,x[1]...x[n]$
    Is a suffix of $x[n+1]....x[n+k]\,then\,\{$
    $B = \phi; pos \leftarrow 0; m \leftarrow 0$
    $while\,pos + l_i \leq |d_i|\,do\,\{$
    $i \leftarrow \max(d_i, m) + 1$
    $while(i \leq l_i)\,and\,x[i] = d_i[pos + i]\,do\,i \leftarrow i + 1$
    $if\,(i \leq l_i)\,then\,\{$
    $pos \leftarrow pos + \max(d_i - n, m - k + 1);$
    $m \leftarrow 0$
    $\}\,else\,\{$
    $j \leftarrow n$
    $while(j > m\,and\,x[j] = d_i[pos + j]\,do\,j \leftarrow j - 1$
    $if\,(j \leq m) then\,add\,pos\,to\,B$
    $pos \leftarrow pos + k;$
    $m \leftarrow l_i - k$
    $\}end\,if$
    $\}end\,while$
    $return(B);$
    $\}else\{$
    $q := \max(d_i, l_i - n) + 1$
    $B := \phi; pos \leftarrow 0;$
```

```
        $while(pos + l_i \le |d_i|)\, do \{$
        $i \leftarrow n+1$
        $while(i \le l_i)\, and\, x[i] = d_i[pos+i]\, do\, i \leftarrow i+1$
        $d_i\,(i \le l_i)\, then \{$
        $pos \leftarrow pos + \max(d_i - n, m - k + 1);$
        $m \leftarrow 0$
        $\}\, else \{$
        $j \leftarrow n$
        $while(j > 0\, and\, x[j] = d_i[pos + j]\, do\, j \leftarrow j-1$
        $if\,(j \le 0)\, then\, add\, pos\, to\, B$
        $pos \leftarrow pos + q;$
        $\}end\, if$
        $\}end\, while$
        $return\,(B)$
        $\}end\, if$
Return $P_i, SP_i, SIP_i, DP_i, DIP, Action\, value \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$
end PatternMatching
```

3.3 Pattern Matching Algorithm

The problem of pattern matching in strings can be efficiently carried out by regular and renowned techniques from automata theory. Here, the pattern to be explored is considered as a fixed value and the strings in database files are assumed to be variable values. The two-way string-matching algorithm (Algorithm-3) works well by comparing the pattern 'x' with the text files 'd_i' in both directions [45]. The starting point of the scan is decided based on the critical position 'n' of 'x' that satisfies, $n < k(x)$, Where, $k(x)$ is the period of 'x'. A positive integer 'k' is called as the period of 'x' if,x[i]= x[i + k], two letters of 'x' at the distance of 'k' always coincide. The variable 'i' is used as a cursor on the pattern to perform the matching. To discover an occurrence of the pattern 'x' at a given position of the text file d_i, the algorithm abstractly divides the search process into two consecutive phases. The initial phase consists of matching 'x' only against d_i and the letters of 'x' are scrutinized from left to right direction. When a mismatch is identified during the initial phase, next phase will be discarded and the pattern always moves to the right. The shift brings the critical position in the right side of the letter in d_i that causes the mismatch. If no mismatch happened in the first phase, then the secondary phase starts and the left part of the pattern 'x_l' is compared against the text files. The word 'x_l' is scanned from right to left as similar to the Boyer and Moore approach [46]. If a mismatch is identified during the scanning process, then the pattern is shifted to a number of places equal to the period of 'x'. Similarly, if any string matches found then the value '1' is returned else the value '0' is returned.

Table 4 Generalized Q-value table represented as Q_g

PID	Protocol	SP	DP	SIP	DIP	Action
1	IP	5006	7001	192.168.137.1	14.168.137.1	1
2	UDP	7008	6003	192.168.137.2	14.168.137.9	0
3	IP	9001	5007	192.168.137.3	15.168.137.3	0
4	TCP	8009	7819	192.168.137.4	12.168.137.89	1

Table 5 Optimized rules after anomaly detection

PID	Protocol	SP	DP	SIP	DIP	Action
1	IP	5006	7001	192.168.137.1	14.168.137.1	1
2	UDP	7008	6003	192.168.137.2	14.168.137.9	0
3	IP	9001	5007	192.168.137.3	15.168.137.3	0
4	TCP	8009	7819	192.168.137.4	12.168.137.89	1
5	IP	7056	8719	192.145.132.1	12.156.139.0	1

3.4 Rule Generalization and Rule Anomaly Detection

The Q-value table is generalized by classifying a packet to accept or deny and it is subsequently updated to Q_g. The sample view of generalization process is presented in Table 4. The performance of the network is monitored and suspicious source IP and destination IP will be identified and updated by the performance monitor. Subsequently, the optimized rules are formulated by combining the results from performance monitor and Q_g. Then, the optimized rules are updated to firewall policy and packets are accepted or denied to pass through the entire network. The optimized rules after the anomaly detection process are shown in Table 5.

4 Experimental Analysis

4.1 Experimental Setup

The performance of RLPM based firewall is tested in a real cloud environment hosted at Noorul Islam Centre for Higher Education (NICHE), India [47]. The architecture of the cloud infrastructure setup along with the firewall is presented in Fig. 10. Three computing servers exist at the data centre which is equipped with 512 GB RAM and 96 TB internal storage configurations. A 24 node computing machines are also attached to the network and the experimental results are observed during the time of high traffic density. The configuration of firewall used for the experimental analysis is presented in Table 6. The RLPM and other firewall techniques compared in this section are developed using Phalcon PHP (v 7.2) framework.

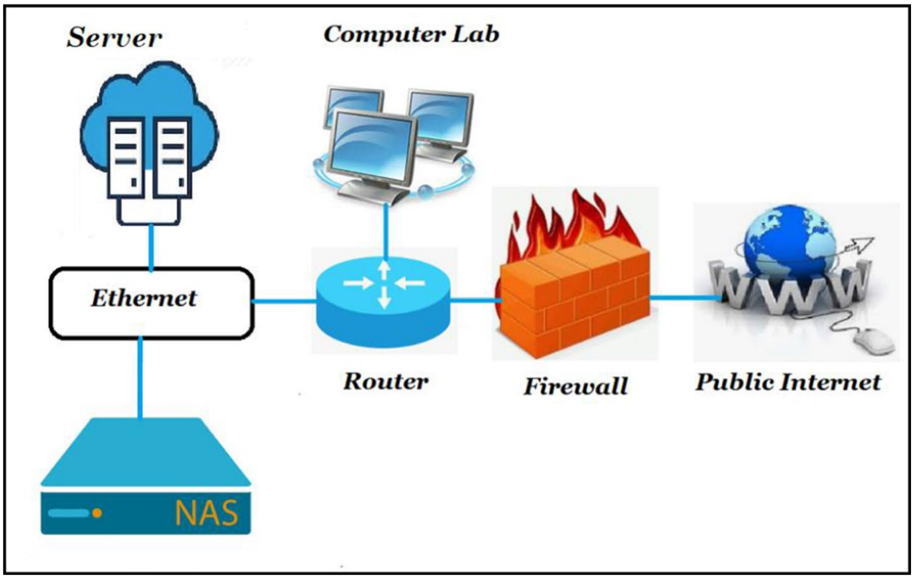


Fig. 10 Architecture of the cloud infrastructure setup

Table 6 Description of Firewall used for experimental purpose

Components	Specifications
Processor	X 86 - 64 bit Core Multi Threaded
LAN Port	6
RAM	4 GB
Flash Storage	16 GB
Operating System	FreeBSD

4.2 Results and Discussion

The experiments are conducted for evaluating various test scenarios that are listed below:

- Initially, the actual user traffic with and without the usage of firewall is observed. In the former case (with firewall), the firewall along with security policies are enforced in the cloud infrastructure. In the later case, every incoming and outgoing packet is forwarded without any security inspection through the router.
- In the second scenario, the firewall performance (Response Time (RT), Throughput & Latency) is tested using specific applications such as e-mail (download & upload) and http page requests.
- In the third scenario, the attack blocking capability of firewall is tested by blocking the malicious packets and allowing the cleaned web traffic only to pass through the firewall.

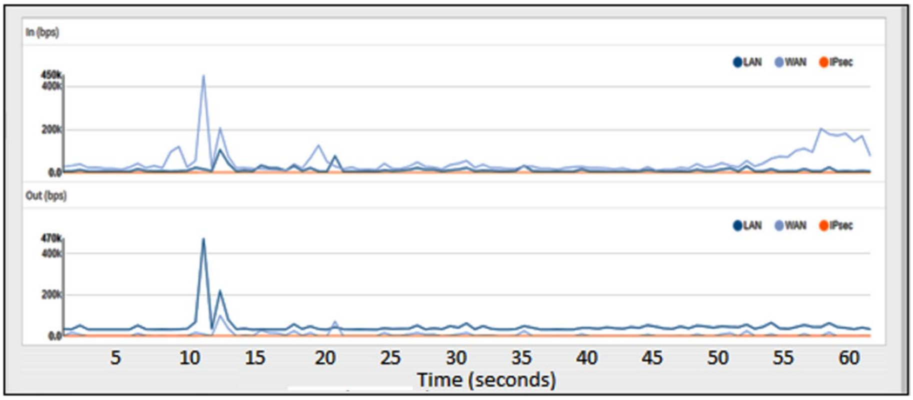


Fig. 11 Traffic flow without firewall scenario

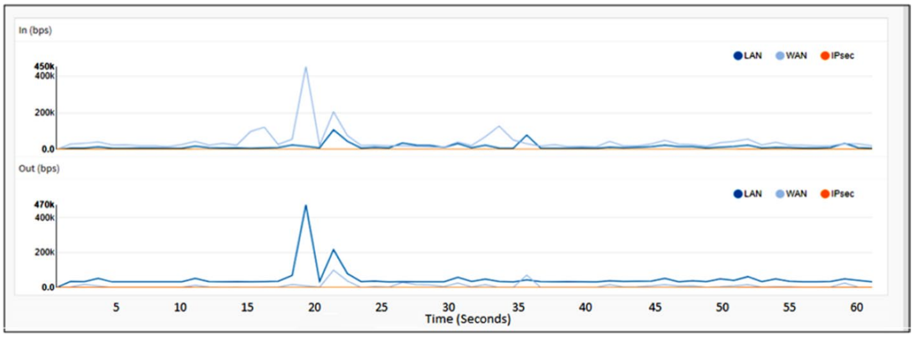


Fig. 12 Traffic flow with Firewall scenario

- The fourth scenario is to test the pattern matching efficiency and computational time of the Pattern Matching Algorithm employed in this research work.
- In the fifth scenario, the signature generation time of the ECDSA algorithm is evaluated.

(i) Traffic flow (with and without Firewall)

Initially, the traffic with and without firewall scenario is observed and the results are displayed in Figs. 11 and 12. For the without firewall scenario, all the packets are allowed to pass though the network without any intermediate blocking. Further, the same traffic flow is allowed through firewall for the purpose of validating the results. Figure 12 indicates the throughput which is slightly affected because of the presence of firewall and it can be neglected.

(ii) Response Time

E-mail applications are employed for the RT investigation against the e-mail downloads and uploads with and without firewall cases. A 200 MB file is downloaded with the link

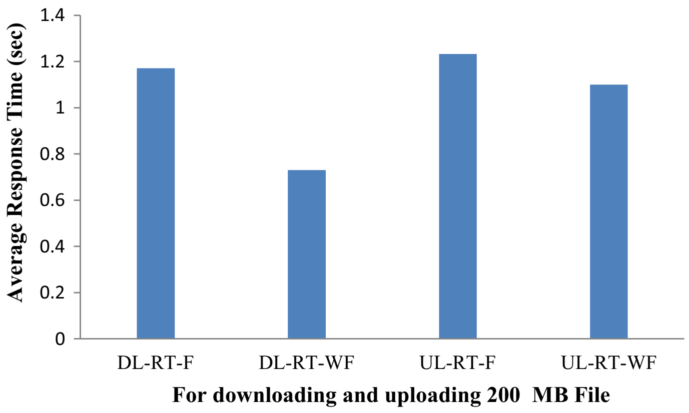


Fig. 13 Response time during E-mail upload and downloads

speed of 2 Gbps and evaluated against the RT performance metrics. The average RT of with and without firewall cases is recorded for the download duration and it is plotted in Fig. 13. Similarly, the identical 200 MB file has been uploaded using the same link speed and it is also presented with and without firewall cases in Fig. 13.

It is observed that the download average RT is low as compared to the upload average RT for with and without firewall cases. During the investigation, the maximum RT observed for DL-RT-F is about 1.57 s and for DL-RT-WF is about 1.3 s. Similarly, the maximum RT for e-mail upload with and without firewall cases is about 1.6 s and 1.2 s respectively. Therefore, it is confirmed that the download speed is comparatively higher while the packets are not blocked. Firewalls sorting traffic is a straightforward process and the packet headers are investigated in the primary stage itself. Subsequently, the payloads are compared against the firewall policy set and every packet will be either blocked or admitted based on the policy set. If a packet is allowed to pass, then it is pushed through the firewall towards its destination before the packet behind could overwrite it. This reason creates an overhead on the router and it results in a larger value of RT about 1.57 s while downloading the data from an email server.

The process of packet filtering should be strictly carried out based on the origin and destination of the data packets. The users experience a delay while uploading the files into the email server because of packet filtering process and it results in a peak RT of 1.6 s. The firewall iterates through the ordered rules and stops at the first rule that matches the packet being held. Several rules may apply to a specific packet and the default action is applied while a mismatch occurs. When stringent security rules are enforced on the router, it consumes extra time to decide an action and delays the file uploading process. In the without firewall scenario, the average RT is comparatively low since, there is no processing of packets occur in the intermediate hardware and thereby enhances the system performance.

Next, the performance of the network while accessing http page with and without firewall case is also investigated and the results are presented in Fig. 14. In the without Firewall case, the RT observed is less than the with firewall case because of the free flow of traffic across the network. The latency in processing the packets during the HTTP page request is observed and displayed in Fig. 15. The enforced security policies as well as the packet latency time imposed over the firewall results in extra processing time thereby degrading

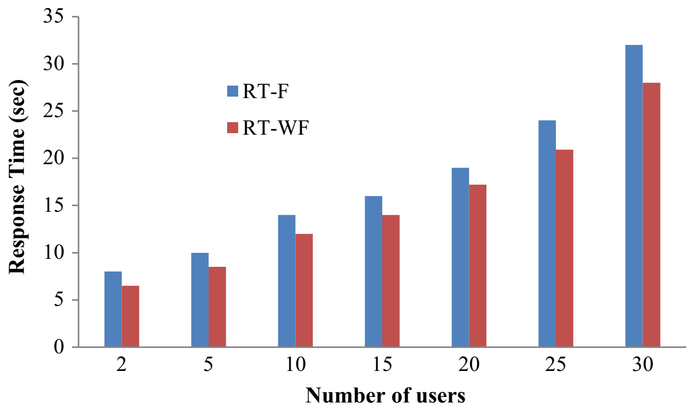


Fig. 14 Page response time (with & without Firewall)

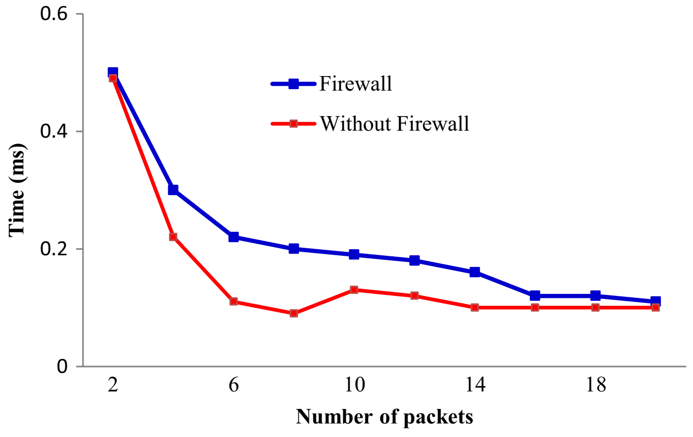


Fig. 15 Latency in the network performance

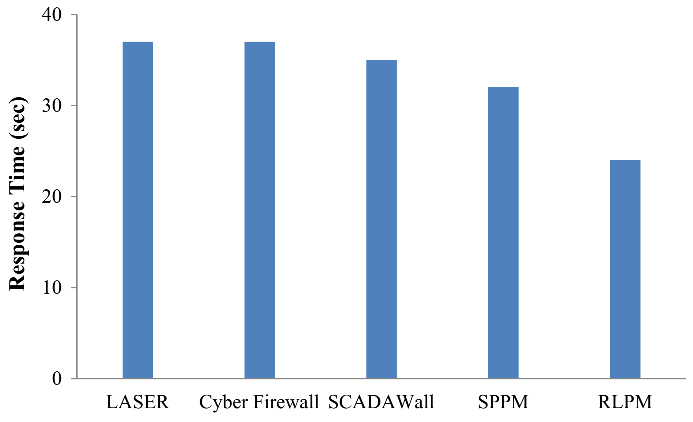


Fig. 16 Response Time comparison of RLPM with other methods

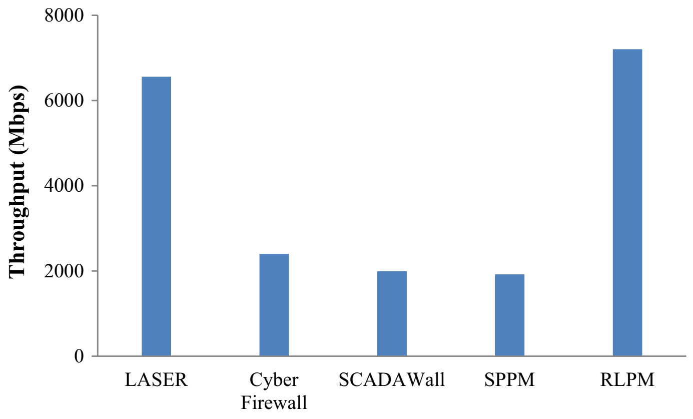


Fig. 17 Throughput comparison of RLPM with other methods

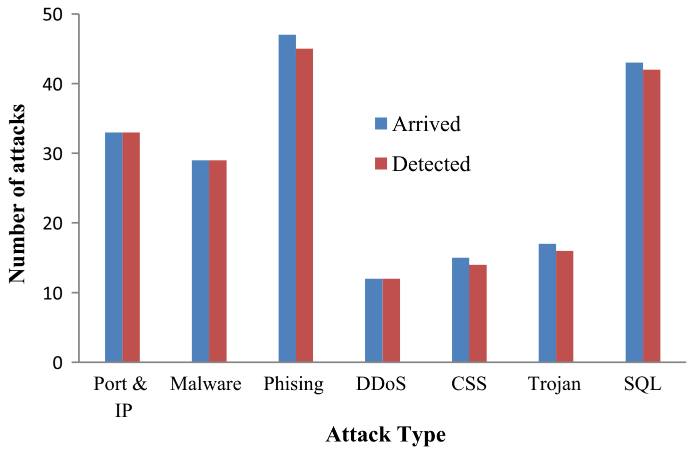


Fig. 18 Attack blocking capability of RLPM

Table 7 Pattern matching Efficiency

No of packets	Patter matching algorithm			
	Brute force	Boyre Moore	FPM	RLPM
16	79.13	72.87	85.75	99
32	78.41	71.93	83.19	98.7
64	76.36	70.56	81.34	96.43
128	74.67	69.43	79.04	96.41
256	72.19	65.36	76.56	96.4

the entire network performance. The overall page RT increases because of latency and it has a value of 9.40 s for accessing http pages by two users as shown in Fig. 14.

The RT performance of the proposed RLPM Firewall is compared with the other industrial product softwares as highlighted in Fig. 16. The RLPM Firewall is compared with LASER [32], CyberFirewall [48], SCADAWall [10] and SPPM [33] products. Interestingly, the RT of RLPM is found to be 10% lesser than LASER and 9% lesser than the SPPM as compared.

(iii) Firewall Throughput

Firewall Throughput is referred as Mbps (megabits per second) or Gbps (gigabits per second) volume of traffic that moves through the firewall at any time. The performance of a firewall strictly depends on supported throughput and based on the speed of internet connections. The firewalls that support an antivirus or intrusion prevention system engines deliberately scan the traffic for viruses, malware or anomalous behaviour. Figure 17 presents the throughput of the firewall system while compared with the latest firewall products (software) deployed in a common hardware presented in Table 6.

(iv) Firewall blocking capability

The results corresponding to initial attack blocking capability of RLPM firewall is presented in Fig. 18. It indicates that nearly 99% of the arrived attacks in the cloud environment were identified and blocked by the firewall. The signature of unidentified attacks has been generated and updated with the database. It shows the efficiency of firewall that depends on the training database and the pattern matching efficiency.

(v) Pattern Matching Time

The pattern matching algorithm operates in a linear time and uses constant memory space for processing. It also utilizes a limited buffer space while operated with real time applications. During pattern matching process, shift operations are carried out in a longer manner to preserve the computational time. The search for pattern 'x' using Boyer and Moore's algorithm neglects certain comparisons due to longer shift operations [46]. The time complexity of the function "positions" is directly proportional to the number of comparisons between the letters of 'x' (length of pattern) and $\mathrm{d_i}$. This pre-computation is done using Fast Pattern Matching (FPM) algorithm [47] and it is combined with the critical factorization computation of the RLPM method. It results in an optimum strategy that is globally linear in time and utilizes constant space. For the input words 'x' and $\mathrm{d_i}$, the function computes the set of positions $[p(x, d_i)]$ of the pattern 'x' inside the text $\mathrm{d_i}$ in time,$O(|t| + |x|)$. In such case, the computation employs $2|t| + 5|x|$ letter comparisons and 13 extra memory locations.

The pattern matching algorithm exhibits a time complexity that is linear while using FPM and in certain cases, it is found to be sub-linear similar to Boyre Moore (BM) [49]. A version of BM algorithm uses an additional information such as position values to compute the shifts of the pattern for the given function [50, 51]. FPM is one of the examples of a pattern matching algorithm that consumes linear time while operating in real time. In fact, the time utilized by the algorithm on a single symbol of the text file cannot be bounded

Table 8 Pattern matching time of 5 packets

Sl.No	Method	Matching Time (in Seconds)
1	Brute Force	0.32
2	Boyre Moore	0.283
3	FPM	0.28
4	RLPM	0.17

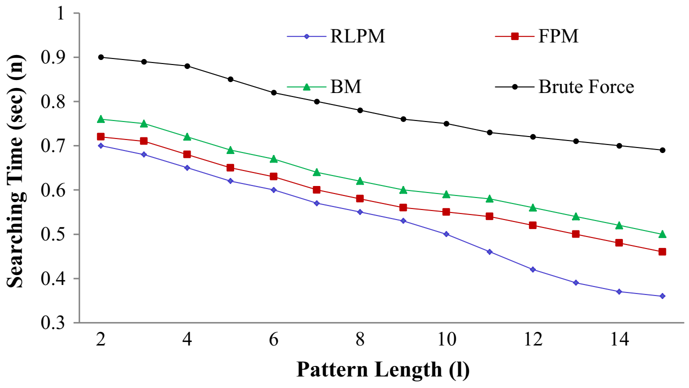


Fig. 19 Time complexity of the pattern matching algorithms

Table 9 Signature generation time

Sl. No	Method	Time (s)
1	SHA-1	2.13
2	EC Diffie Hellman	2.255
3	Scalar multiplication	1.032
4	ECDSA	1.004

by a constant value. The possibility of realizing the string-matching in linear time and constant memory space has been implemented with a multi-head deterministic finite-state automaton [52]. The efficiency of various pattern matching algorithms is compared with other competitive methods and it is presented in Table 7. Here, the efficiency is measured in terms of less iterations, time complexity and memory requirements of the algorithm. The pattern matching time of 5 sample packets while using different pattern matching algorithms is summarized in Table 8.

(vi) Time Complexity of Pattern Matching Algorithm

Time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function for the given input. The Brute Force, BM, FPM and RLPM algorithms are analyzed for the time complexity. In Fig. 19, the horizontal axis describes the length of the pattern considered for each algorithm with the comparative searching time in the vertical axis.

The RLPM has less time complexity because of the two way pattern matching strategy implemented in the parallel manner.

(vii) Efficiency of signature generation algorithm

The time required for signature generation process corresponding to 5 sample packets using ECDSA method is compared with the other signature generation algorithms as presented in Table 9. It is observed that ECDSA method is light weight in nature and uses a smaller key value that produces the signature as quickly as possible while compared to other methods. Hence, the suitability of ECDSA method for this research work is well justified in terms of performance.

5 Conclusion

In the presented work, a hybrid firewall mechanism named Reinforcement Learning and Pattern Matching (RLPM) is developed for securing cloud computing infrastructure. RLPM employs deep packet inspection that is known as an efficient strategy to inspect payload segment of the packets for preventing the network-based attacks. Here, DPI is accomplished through the two-way pattern matching algorithm that compares the signature of incoming packets and determines the attacks. RLPM operates in a parallel manner to inspect the packet payloads and detects the anomalous behaviour in the network flow towards cloud infrastructure. RLPM learns the attacks in a computing environment and converges to an optimal solution within the limited time step. At each iterative step, the signature of new attacks are determined and updated to the database. The experiments are conducted in multiple dimensions such as the performance aspects of firewall in terms of RT, throughput, latency, and attack resistance potential of RLPM. A comparative analysis is also done to evaluate the performance of various state-of-the-art methods with RLPM. The results are validated in a real and highly sensitive cloud data centre environment. It is observed that the RLPM firewall is efficient for detecting various malicious attacks and protects the cloud infrastructure. It is also observed that the throughput of RLPM is about 10% higher than LASER while the RT is 10% lesser. The proposed work can be easily extended further to mitigate the other contemporary attacks against the cloud data centre environment in future.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Kumar, R., & Goyal, R. (2019). On cloud security requirements, threats, vulnerabilities and counter measures: A survey. *Computer Science Review, 33,* 1–48.
2. Jeong, C. Y., TomLee, S. Y., & Lim, J.-H. (2019). Information security breaches and IT security investments: Impacts on competitors. *Information & Management*. https://doi.org/10.1016/j.im.2018.11.003.
3. Sibi Chakkaravarthy, S., Sangeetha, D., & Vaidehi, V. (2019). A survey on malware analysis and mitigation techniques. *Computer Science Review, 32,* 1–23.

4. Manav, M. T. (2018). Defense mechanisms against distributed denial of service attacks: A survey. *Computers & Electrical Engineering, 72,* 26–38.
5. Rao, R. S., & Pais, A. R. (2019). Jail-Phish: An improved search engine based phishing detection system. *Computers & Security, 83,* 246–267.
6. McWhirter, P. R., Kifayat, K., Shi, Q., & Askwith, B. (2018). SQL Injection Attack classification through the feature extraction of SQL query strings using a Gap-Weighted String Subsequence Kernel". *Journal of Information Security and Applications, 40,* 199–216.
7. Boraten, T., & Kodi, A. (2018). Mitigation of hardware trojan based denial-of-service attack for secure NoCs. *Journal of Parallel and Distributed Computing, 111,* 24–38.
8. Moataz, A., & Ali, A. F. (2016). Multiple-path testing for cross site scripting using genetic algorithms. *Journal of Systems Architecture, 64,* 50–62.
9. YasinNur, A., & EnginTozal, M. (2018). Record route IP traceback: Combating DoS attacks and the variants. *Computers & Security, 72,* 13–25.
10. Li, D., Guo, H., Zhou, J., Zhou, L., & Wong, J. W. (2019). SCADAWall: A CPI-enabled firewall model for SCADA security. *Computers & Security, 80,* 134–154.
11. De La Torre, G., Parra, P. R., Kwang, K., & Choo, R. (2019). Implementation of deep packet inspection in smart grids and industrial Internet of Things: Challenges and opportunities. *Journal of Network and Computer Applications, 135*(1), 32–46.
12. Yang, X., & Liu, P. (2013). A new algorithm of the data mining model in cloud computing based on web Fuzzy clustering analysis. *Journal of Theoretical & Applied Information Technology*, *49*(1), 266–273.
13. Gang, W., Jinxing, H., Jian, M., & Lihua, H. (2010). A new approach to intrusion detection using artificial neural networks and fuzzy clustering. *Expert Systems with Applications, 37*(9), 6225–6232.
14. Kakkar, L., & Mehta, G. (2016). A review: Hadoop storage and clustering algorithms. *IOSR Journal of Computer Engineering, 18*(1), 23–29.
15. Wang, H., Wang, J. (2014) An effective image representation method using kernel classification. In: *2014 IEEE 26th international conference on tools with artificial intelligence (ICTAI)* (pp. 853–858). IEEE.
16. Zhang, S., Wang, H., & Huang, W. (2017). Two-stage plant species recognition by local mean clustering and Weighted sparse representation classification. *International Journal of Computers and Applications, 41*(4), 262–267.
17. Rustam, Z., & Talita, A.S. (2018). Fuzzy Kernel robust clustering for anomaly based intrusion detection. In *2018 Third International Conference on Informatics and Computing (ICIC)*, Palembang, Indonesia, (pp. 1–4). https://doi.org/10.1109/iac.2018.8780480.
18. Maya, S., Ueno, K., & Nishikawa, T. (2019). dLSTM: A new approach for anomaly detection using deep learning with delayed prediction. *International Journal of Data Science and Analytics, 8,* 137–164. https://doi.org/10.1007/s41060-019-00186-0.
19. Alnafessah, A., & Casale, G. (2019). Artificial neural networks based techniques for anomaly detection in Apache Spark. *Cluster Computing*. https://doi.org/10.1007/s10586-019-02998-y.
20. Yingpei, Z., Shanqing, G. (2018). Deep packet inspection with delayed signature matching in network auditing. https://doi.org/10.1007/978-3-030-01950-15.
21. Xiuwen, S., Hao, L., Dan, Z., Xingxing, L., Kaiyu, H., & Chengchen, H. (2019). COIN: A fast packet inspection method over compressed traffic. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2018.12.008.
22. Lukashin, A., Laboshin, L., Zaborovsky, V., & Mulukha, V. (2014). Distributed packet trace processing methodfor information security analysis. *LNCS, 8638,* 535–543.
23. Da Costa Júnior, E., da Silva, C., Pinheiro, M., et al. (2018). A new approach to deploy a self-adaptive distributed firewall. *Journal of Internet Services and Applications, 9,* 12. https://doi.org/10.1186/s13174-018-0083-6.
24. Khan, F. A., & Gumaei, A. (2019). A comparative study of machine learning classifiers for network intrusion detection. In X. Sun, Z. Pan, & E. Bertino (Eds.), *Artificial intelligence and security ICAIS 2019. Lecture notes in computer science* (Vol. 11633). Cham: Springer.
25. Amanullah, M. A., Habeeb, R. A. A., Nasaruddin, F. H., et al. (2020). Deep learning and big data technologies for IoT security. *Computer Communications*. https://doi.org/10.1016/j.comcom.2020.01.016.
26. Ertam, F. (2019). An efficient hybrid deep learning approach for internet security. *Physica A, 535,* 122492.
27. Sen, S., Spatscheck, O., & Wang, D. (2004). Accurate, scalable in network identification of p2p traffic using application signatures. In *Proceedings of the 13th international conference on World Wide Web*, New York.

28. Haffner, P., Sen, S., Spatscheck, O., & Wang, D. (2005). ACAS: automated construction of application signatures. In *MineNet '05 Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*, (pp. 197–202).
29. Kim, H.-A., & Karp, B. (2004). Autograph: toward automated, distributed worm signature detection. In *Proceedings of the 13th conference on USENIX Security Symposium*, (p. 19). San Diego, CA.
30. Li, Z., Sanghi, M., Chen, Y., Kao, M.-Y., & Chavez, B. (2006) Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilienc. In *IEEE Symposium on Security and Privacy*.
31. Fernandes, S., Antonello, R., Lacerda, T., Santos, A., Sadok, D., & Westholm, T. (2009). Slimming down deep packet inspection systems. In *NFOCOM Workshops 2009 IEEE*, (pp. 1–6).
32. Park, B. -C., Won, Y. J., Kim, M. -S., Hong, J. W. (2008). Towards automated application signature generation for traffic identification. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)* (pp. 160–167). Salvador. https://doi.org/10.1109/NOMS.2008.4575130.
33. Najam, M., Younis, U., & Rasool, R. (2015). Speculative parallel pattern matching using stride-k DFA for deep packet inspection. *Journal of Network and Computer Applications, 54,* 78–87.
34. Afeka, Y., Bremler-Barrb, A., & Korala, Y. (2012). Space efficient deep packet inspection of compressed web traffic. *Computer Communications, 35*(7), 810–819.
35. Malware Dataset. Retrieved August 15, 2020 from https://www.kaggle.com/nsaravana/malware-detection.
36. DDoS Dataset. Retrieved August 15, 2020 from https://www.caida.org/data/passive/ddos-20070804_dataset.xml.
37. Phishing Data. Retrieved August 15, 2020 from https://archive.ics.uci.edu/ml/datasets/phishing+websites.
38. SecList Data. Retrieved 8 August 15, 2020 from https://github.com/danielmiessler/SecLists.
39. SecList Data. Retrieved August 15, 2020 from https://iscxdownloads.cs.unb.ca/iscxdownloads/ISCX-URL-2016/#ISCX-URL-2016.
40. Cross Site Scripting. Retrieved August 15, 2020 from https://data.mendeley.com/datasets/9jhzmswdfj/1.
41. Anamoly dataset. Retrieved August 15, 2020 from https://ant.isi.edu/datasets/all.html.
42. Licheng, W., Xiaoying, S., Jing, L., Jun, S., & Yixian, Y. (2019). Cryptographic primitives in blockchains. *Journal of Network and Computer Applications, 127,* 43–58.
43. Elliptic Curve Digital Signature Algorithm. Retrieved August 15, 2020 from https://www.cs.miami.edu/home/burt/learning/Csc609.142/ecdsa-cert.pdf.
44. Bibal Benifa, J. V., & Dejey, D. (2018). Reinforcement learning-based proactive auto-scaler for resource provisioning in cloud environment. *Mobile Networks and Applications, Springer*. https://doi.org/10.1007/s11036-018-0996-0.
45. Maxime Crochemore and Dominique Perrin. Two-way string matching. http://www.quretec.com/u/vilo/edu/2002-03/Tekstialgoritmid_I/Articles/Exact/Two-way-p650-crochemore.pdf.
46. Klein, S. T., & Ben-Nissan, M. (2009). Accelerating boyer–moore searches on binary texts. *Theoretical Computer Science, 410*(37), 3563–3571.
47. https://www.nichecloud.in.duc.
48. Sari, A. (2019). Turkish national cyber-firewallto mitigate countrywide cyber-attacks. *Computers & Electrical Engineering, 73,* 128–144.
49. Knuth, D. E., Morrjs, J. H., & Pratt, V. R. (1977). Fast pattern matching in strings. *SIAM Journal on Computing, 6*(2), 323–350.
50. Horspool, N. (1980). Practical fast searching in strings. *Software: Practice and Experience, 10,* 501–506.
51. Sedgewick, R. (1988). Algorithms. Addison-Wesley, Reading, Mass. 2d edn.
52. Galil, Z., & Seiferas, J. (1983). Time space optimal string matching. *Computer and System Sciences, 26,* 280–294.

Ms. J. Jeya Praise is presently associated with Anna University, Chennai as a Ph.D. Research Scholar. She has obtained her BE and ME degrees in the domain of Computer Science and Engineering from Anna University, India. Her research interests are virtualization and cloud security.

Dr. R. Joshua Samuel Raj is presently associated with Rajaas Engineering College as Professor. He has obtained his BE and ME degrees in the domain of Computer Science an Engineering. He was awarded with Ph.D. from Kalasalingam University, India. His research interests are virtualization and cloud security.

Dr. J. V. Bibal Benifa is presently associated with Indian Institute of Information Technology, Kottayam, India as an Assistant Professor. She obtained her BE, ME and Ph.D. degrees in the domain of cloud computing from Anna University, India. Her research interest are cloud computing big data analytics, image processing and machine learning. She is presently involved in developing various societal applications.