Securing Software-Defined Networks Through Dos Attack Type Classification And Mitigation Framework Using Lrs^{2}btm

Amaresan Venkatesan
v.amaresan@gmail.com

Abstract:
Software-Defined Networking (SDN) optimizes network management by detecting and mitigating Denial of Service (DoS) attacks. Traditional techniques fail to address specific DoS attack types like DOS Hulk, Benign, DOS Slow HTTP Test, DOS Slowloris, and DOS GoldenEye, leading to ineffective mitigation and network vulnerabilities. To classify DoS attack types and enhance network security, Lasso-Resilience-Ridge Swishmax-based Bidirectional Long-Short Term Memory (LRS^{2}BTM) and Chaos-Shannon-based Elliptical Curve Cryptography (CS-ECC) are used. Initially, the user registration and key generation are done, followed by logging in to the DoS attack detection phase. In this phase, the attacks are categorized using LRS^{2}BTM. From the categorized attacks, DOS Hulk and GoldenEye attacks due to high traffic rates are mitigated using FIFO-TBA; also, DOS Slow HTTP Test and Slowloris attacks are blocked as they cause service disruption. Non-attacked (benign) and mitigated data are then secured with an Encryption Time (ET) of 2571ms. Thus, the proposed work outperformed the existing methodologies in enhancing network security and efficiency.

Keywords: First In First Out based Token Bucket Algorithm (FIFO-TBA), Diversity-Preserving Perturbation based Secretary Bird Optimization Algorithm (DP^2-SBOA), Elliptical Curve Cryptography (ECC), Bidirectional Long Short-Term Memory (BiLSTM), Software Defined-Networking (SDN), Data Security, Attack Detection (AD).

1. INTRODUCTION

SDN has emerged as a transformative approach in the realm of network management and architecture (Valdovinos et al., 2021). By decoupling the control plane from the data plane (AbdelAzim et al., 2021), SDN offers centralized control over network resources (ElSayed et al., 2021). However, this paradigm shift had drawbacks related to security measures (Wang et al., 2022), particularly the susceptibility to DoS attacks (Eliyan & Di Pietro, 2021). Therefore, the detection and mitigation of DoS attacks are necessary to overcome the security vulnerabilities in SDN.

Most of the prevailing works used techniques like Support Vector Machine (Alashhab et al., 2022), Gated Recurrent Unit (GRU), and Recurrent Neural Networks (RNN) for DoS-AD (Bahashwan et al., 2023) and mitigation processes. However, these techniques had drawbacks like failing to adapt to evolving attack patterns (Alshra'a et al., 2021), overfitting, and slow convergence (Ali et al., 2023, Rios et al., 2022). Also, these methods failed to classify the DoS attack types for enhanced mitigation processes. Hence, to overcome the drawbacks, the proposed work used LRS^{2}BTM, FIFO-TBA, and CS-ECC techniques for securing data through SDN.

1.1 Problem Statement

The drawbacks of existing methodologies are as follows,

- None of the works focused on categorizing the DoS attack types for enhanced mitigation processes that caused vulnerabilities in network security.
- In (Li et al., 2022), numerous features were extracted for DoS-AD that significantly increased the processing time.
- Increased network latency and bottlenecks in (Yeom et al., 2022) were caused due to inefficient traffic distribution in SDN.
- The absence of data security in most of the prevailing works caused data breaches and unauthorized access.

The objectives of the proposed work are described below,

- The proposed work effectively categorized the DoS attack types using LRS^{2}BTM.
- Optimal feature selection is done using DP^2-SBOA.
- DoS traffic attacks in SDN are mitigated using the FIFO-TBA technique.
- Data is secured in SDN using the CS-ECC technique.

The rest of the paper is organized as: section 2 discusses the related works, section 3 describes the proposed methodology, section 4 presents the results and discussion, and finally, section 5 concludes the proposed work with future development.

2. LITERATURE SURVEY

(Li et al., 2022) detected and mitigated DoS attacks in SDN using DoSGuard. This work utilized a monitor to track switch-host interactions, a detector to identify attacks through OpenFlow message and flow features, and a mitigator to filter malicious packets. But, DoSGuard's reliance on OpenFlow features lacked the ability to block specific flows and failed to detect attacks on the application plane.

(Yeom et al., 2022) introduced a Long-Short Term Memory (LSTM)-based collaborative source-side DoS-AD system. This framework utilized adaptive thresholds based on irregular network traffic patterns and established a collaborative network across multiple detection sites to enhance detection accuracy. However, the framework's dependence on heterogeneous graph embedding methods lacked a precise representation of complex site relationships, thus degrading the overall work performance.

(Wang et al., 2023) presented a DoSDefender framework to prevent DoS attacks in SDN by verifying connection attempts in the Transmission Control Protocol. Also, the framework verified the migrating connections and relaying packets in kernel space for defending the DoS attacks. Nevertheless, this work had potential scalability issues and performance impacts due to high network traffic scenarios.

(Bhayo et al., 2023) accomplished an effective framework named DoS attack detection in SDN using ML techniques. This model used Naive Bayes, Decision Tree, and Support Vector Machine to accurately detect DoS attacks in the SDN environment. Also, this model optimized memory and CPU usage for effective DoS attack detection. However, the extraction of a large number of features increased the processing time for DoS-AD, thereby degrading the model's effectiveness.

(Swami et al., 2023) developed a defense solution for detecting and mitigating spoofed flooding DoS attacks in an SDN controller. It employed the Interquartile Range (IQR) statistical measure for DoS detection and utilized existing SDN capabilities for mitigation. Despite enhanced detection, the lack of data security resulted in unauthorized access.

3. PROPOSED METHODOLOGY FOR SECURING SDN THROUGH DOS ATTACK TYPE CLASSIFICATION AND MITIGATION PROCESS

The structural diagram of the proposed work using LRS^{2}BTM and FIFO-TBA techniques is depicted in Figure 1,

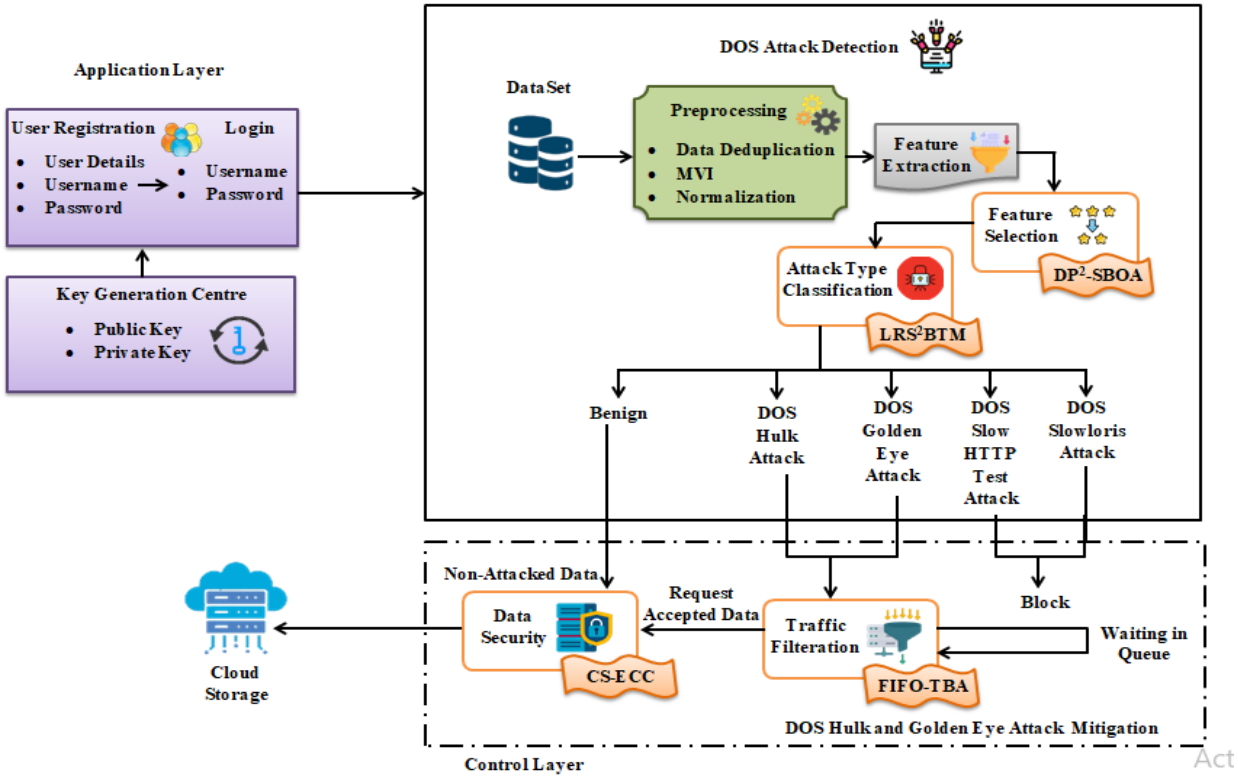


Figure 1: Structural Diagram of the Proposed Work

Application Layer

In the application layer, the user registration and login are done, which are explained further.

3.1 User Registration and Login

The proposed work starts with a registration process that contains user details, usernames, and passwords. Thus, the (s)numbers of registered users (H) are expressed as,

$$H = H^1, H^2, \ldots\ldots, H^s \tag{1}$$

The registered user then logs in to begin the process of storing data in the cloud through the SDN framework.

Key Generation Centre

Here, during registration, the private and public keys (p^k, q^k) are generated using Chaos dynamics with Shannon information theory (CS) to enhance the network security as shown below,

$$p^k = r^{con} \cdot U^{reg} \cdot \left(1 - U^{reg}\right) \tag{2}$$

$$q^k = p^k \times \mathrm{E} \tag{3}$$

Here, (r^{con}) represents the control parameter in CS and (E) represents the point on an elliptical curve.

Data Layer

In the data layer, the DoS attack types are categorized using DP^2-SBOA and LRS^{2}BTM techniques as explained in the below sections.

3.2 Data Collection

Initially, the data is collected from two datasets, namely Application Layer DoS Attack (AL-DoS-Attack) and Canadian Institute for Cybersecurity and Communications Security Establishment - Intrusion Detection System 2018 (CSE-CIC-IDS2018) for detecting DoS attack types. The (g) numbers of collected data $(D^{collect})$ are represented as,

$$D^{collect} = D^1, D^2, \ldots\ldots, D^g \quad where\ (collect = 1\ to\ g) \tag{4}$$

The collected data $\left(D^{collect}\right)$ is then preprocessed for further classification.

3.3 Preprocessing

The preprocessing is performed using various techniques, such as Data Deduplication (DD), Missing Value Imputation (MVI), and Normalization, which are discussed below,

- DD reduces storage redundancy by eliminating duplicate copies. Thus, the reduced data $\left(R^{da}\right)$ using the original data size $\left(D^{collect}\right)$ and the deduplicated data size $\left(F^{dup}\right)$ are expressed as,

$$R^{da} = \frac{\left(D^{collect} - F^{dup}\right)}{D^{collect}} \tag{5}$$

- MVI then fills in absent data to ensure completeness as shown below,

$$V^{\varepsilon} = \varpi\left(R^{da}\right) \tag{6}$$

Here, $\left(\varpi\right)$ represents the process for filling the missing values and $\left(V^{\varepsilon}\right)$ represents the data after the imputation of missing values.

- After that, the data are normalized $\left(D^{nor}\right)$ using minimum and maximum values as shown below,

$$D^{nor} = \frac{V^{\varepsilon} - V^{\varepsilon}_{\min}}{V^{\varepsilon}_{\max} - V^{\varepsilon}_{\min}} \tag{7}$$

Here, $\left(V^{\varepsilon}_{\max}, V^{\varepsilon}_{\min}\right)$ represent the maximum and minimum values $\left(\max, \min\right)$ of $\left(V^{\varepsilon}\right)$. Hence, the preprocessed data is denoted as $\left(D^{pro}\right)$.

3.4 Feature Extraction and Feature Selection

From $\left(D^{pro}\right)$, the features, such as Destination_Port, Total_Length_of_Fwd_Packets, Total_Length_of_Bwd_Packets, Flow_Duration, Active_Mean, Active_Max, Active_Min, Idle_Mean, Idle_Std, Label, and more are extracted and denoted as $\left(E^{fea}\right)$.

After that, the optimal features are selected from $\left(E^{fea}\right)$ using the Secretary Bird Optimization Algorithm (SBOA) by balancing exploration and exploitation. However, the premature convergence in SBOA leads to local optima trapping. Therefore, the Diversity Preserving Perturbation (DP^2) is used, which introduces random perturbations to maintain population diversity. The algorithmic steps of DP^2-SBOA are described further,

Population Initialization

At first, the population of secretary birds $\left(S_b\right)$ from $\left(E^{fea}\right)$ is initialized with the $\left(A\right)$ number of Secretary Birds (SB) and the $\left(m\right)$ number of dimensions as,

$$S_b = \begin{bmatrix} S_{b(1,1)} & \cdots & S_{b(1,j)} & \cdots & S_{b(1,m)} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ S_{b(i,1)} & \cdots & S_{b(i,j)} & \cdots & S_{b(i,m)} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ S_{b(A,1)} & \cdots & S_{b(A,j)} & \cdots & S_{b(A,m)} \end{bmatrix} \tag{8}$$

Here, $\left(S_{b(i,j)}\right)$ represents the value of the $\left(j^{th}\right)$ variable for $\left(i^{th}\right)$ SB's random position.

Fitness Calculation

The fitness value $\left(C^{fit}\right)$ for choosing the optimal features is calculated based on maximum classification accuracy $\left(M^{Class}\right)$, which is given as,

$$C^{fit} = \max\left(M^{Class}\right) \tag{9}$$

For optimal solution, the SB uses two strategies, namely exploration (hunting strategy) and exploitation (escape strategy).

Exploration

The exploration phase broadly searches, consumes, and attacks the feature space to identify promising regions for optimal feature selection. Here, the SB searches its prey by updating the position using DP[2] to overcome the suboptimal issues as shown below,

$$S_{b(i,j)}^{search-prey} = S_{b(best)} + F^{mut} \times \left(S_{b(rand(1))} - S_{b(rand(2))}\right) + \ell \times \aleph(0,1) \tag{10}$$

$$S_{b(i,j)}^{consume-prey} = S_{b(best)} + e\left(\left(t/T\right)^4\right) \times \left(R^B - 0.5\right) \times \left(S_{b(best)} - S_{b(i,j)}\right) \tag{11}$$

$$S_{b(i,j)}^{attack-prey} = S_{b(best)} + \left(\left(1 - t/T\right)^{2 \times t/T}\right) e\left(\left(t/T\right)^4\right) \times S_{b(i,j)} \times R^L \tag{12}$$

$$S_{b(i,j)}^{new} = S_{b(i,j)}^{search-prey} \times S_{b(i,j)}^{consume-prey} \times S_{b(i,j)}^{attack-prey} \tag{13}$$

Here, $\left(S_{b(i,j)}^{search-prey} \times S_{b(i,j)}^{consume-prey} \times S_{b(i,j)}^{attack-prey}\right)$ represent the search, consume, and attack prey operations, respectively, $\left(S_{b(i,j)}^{new}\right)$ represents the updated position of SB, $\left(S_{b(best)}\right)$ represents the random best solution, $\left(F^{mut}\right)$ represents the scaling factor, $\left(S_{b(rand1)}, S_{b(rand2)}\right)$ represents the randomly chosen distinct vectors from the population, $\left(\ell\right)$ is the small perturbation factor, $\left(\aleph(0,1)\right)$ represents the Gaussian noise term with mean [0] and standard deviation [1], $\left(t, T\right)$ represents the current and maximum iterations, $\left(e\right)$ represents the exponential function, and $\left(R^L, R^B\right)$ represents the randomness in Brownian and levy flight operations.

Exploitation

The exploitation phase refines feature selection by focusing on improving solutions near the current best position as,

$$S_b^* = S_{b(best)} + \left(2 \times R^B - 1\right) \times \left(1 - \frac{t}{T}\right)^2 \times S_{b(i,j)} \tag{14}$$

Here, $\left(S_b^*\right)$ represents the new position of SB. The best solution is then selected based on the conditions as shown below,

$$\gamma = \begin{cases} S_b^* & \forall \left(S_b^* \le S_b\right) \\ S_b & else \end{cases} \tag{15}$$

Thus, by updating the position of SB, the best features $\left(\gamma\right)$ are selected.

3.5 Attack Type Classification

Then, from $\left(\gamma\right)$, DoS attack types are categorized using BiLSTM, which captures intricate patterns in both forward and backward directions. However, BiLSTM has drawbacks like slow convergence and potential overfitting due to doubled dimensionality. Therefore, the Lasso-Resilience-Ridge (LR^2) technique with Swishmax Activation (SA) is utilized to enhance training efficiency and reduce overfitting risks in large datasets. The LRS^2BTM classifier is shown in Figure 2,

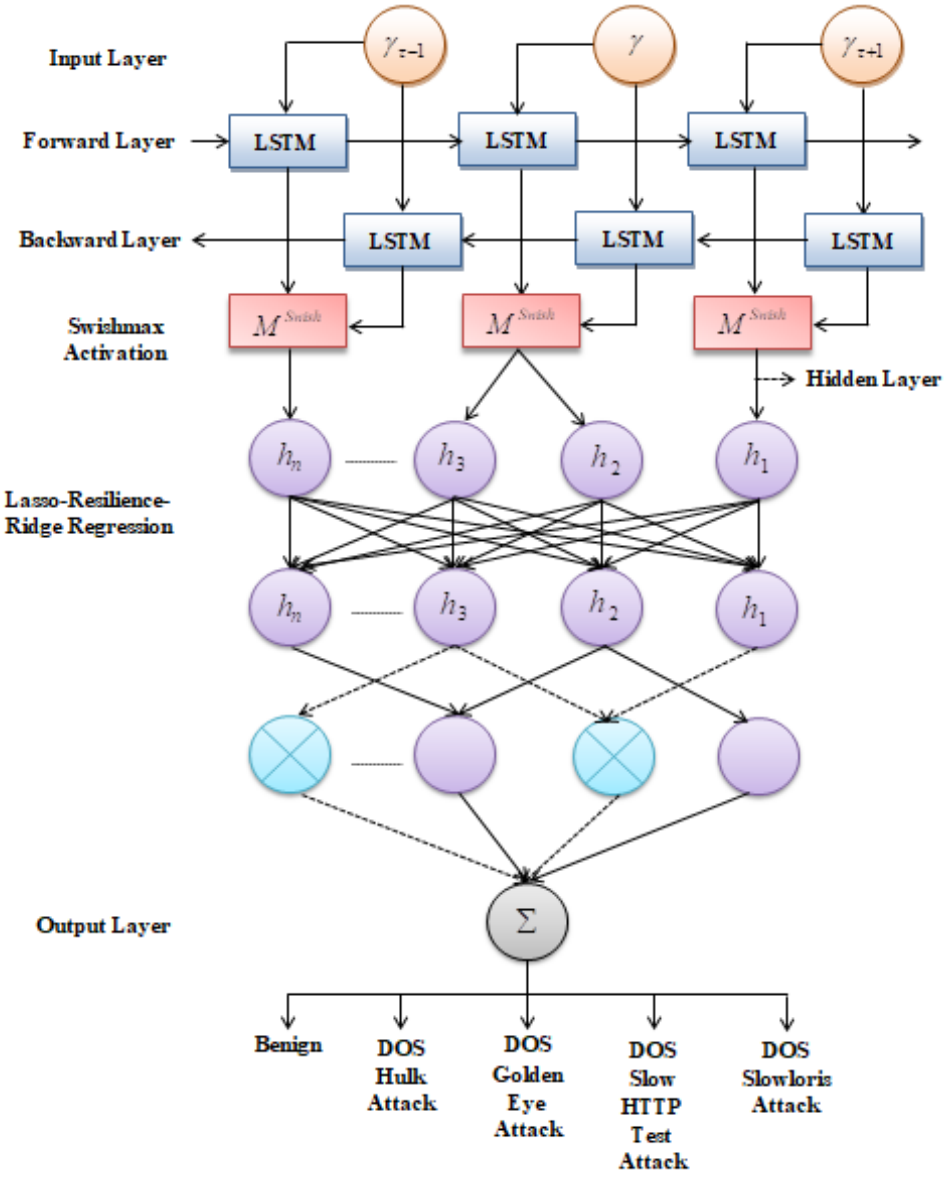


Figure 2: LRS²BTM classifier

The algorithmic steps of the proposed LRS²BTM are described below,

- At first, the selected features (γ) are initialized to start the detection of DoS attack types as,

$$\gamma = \gamma^1, \gamma^2, \ldots\ldots, \gamma^{\varphi} \tag{16}$$

Here, (φ) represents the total number of (γ).

- Now, the gate calculations and hidden state calculations are defined below. At first, the input gate (u_τ) with time (τ) regulates the flow of new information as,

$$u_\tau = \mathrm{M}^{swish}\left(w_{u_\tau} * [h_{\tau-1}, \gamma_\tau] + b_{u_\tau}\right) \tag{17}$$

Where, (w_{u_τ}, b_{u_τ}) represents the weights and biases of (u_τ), $(h_{\tau-1})$ represents the hidden state with time $(\tau-1)$, and (M^{swish}) represents the SA activation, which is formulated as,

$$\mathrm{M}^{swish} = \max\left(\gamma, \gamma \cdot \left(\frac{1}{1+e^{-\gamma}}\right)(\beta \cdot \gamma)\right) \tag{18}$$

Here, (β) represents the hyper-parameter that controls the smoothness of a non-linearity function.

- Then, the forget gate (f_τ) regulates what information to discard and the output gate (o_τ) controls what information to pass. The formula is thus represented as,

$$f_\tau = \mathrm{M}^{swish}\left(w_{f_\tau} * [h_{\tau-1}, \gamma_\tau] + b_{f_\tau}\right) \quad (19)$$

$$o_\tau = \mathrm{M}^{swish}\left(w_{o_\tau} * [h_{\tau-1}, \gamma_\tau] + b_{o_\tau}\right) \quad (20)$$

Here, $\left(w_{f_\tau}, b_{f_\tau}, w_{o_\tau}, b_{o_\tau}\right)$ represents the weights and biases in (f_τ, o_τ).

- After that, the candidate cell state $(\hat{c}_\tau)$ determines new information to be stored as given below,

$$\hat{c}_\tau = \mathrm{M}^{swish}\left(w_{\hat{c}_\tau} * [h_{\tau-1}, \gamma_\tau] + b_{\hat{c}_\tau}\right) \quad (21)$$

Here, $\left(w_{\hat{c}_\tau}, b_{\hat{c}_\tau}\right)$ represents the weights and biases in $(\hat{c}_\tau)$. Then, the updated cell state $\left(c_\tau^*\right)$ is given as,

$$c_\tau^* = f_\tau \circ c_{\tau-1} + u_\tau \circ \hat{c}_\tau \quad (22)$$

- Then, the forward $\left(\vec{h}_\tau\right)$and backward $\left(\overleftarrow{h}_\tau\right)$ hidden states are concatenated $\left(\overleftrightarrow{h}_\tau\right)$ using the LR^2 technique to overcome the overfitting issues as,

$$\overleftrightarrow{h}_\tau = L(\theta) + \lambda_1\left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \theta^*\right] + \lambda_2 + \left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \theta^{2^*}\right] + \lambda_3\left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \left(\theta^* - \mu\right)^2\right] \quad (23)$$

Here, $(\lambda_1, \lambda_2, \lambda_3)$ represents the regularization parameters for the Lasso, Ridge, and Resilience term, $(L(\theta))$ represents the original loss function, $\left(\theta^*, \theta^{2^*}\right)$ represents the Lasso-Ridge penalty, and (μ) represents the mean value of the resilience term.

Pseudo code of LRS^{2}BTM

Input: Selected features, (γ)

Output: Classified DoS attack types $\left(\Re^B, \Re^H, \Re^G, \Re^{SH}, \Re^S\right)$

Begin

Initialize iterations $\left(\tau, \tau^{\max}\right)$

While $\left(\tau < \tau^{\max}\right)$

Initialize (γ)

Calculate $\left(u_\tau, f_\tau, o_\tau\right)$

Evaluate activation, $\left(\mathrm{M}^{swish}\right)$

$$\mathrm{M}^{swish} = \max\left(\gamma, \gamma \cdot \left(\frac{1}{1+e^{-\gamma}}\right)(\beta \cdot \gamma)\right)$$

Update cell state, $\left(c_\tau^*\right)$

End while

Concatenate $\left(\vec{h}_t\right)$and $\left(\overleftarrow{h}_t\right)$ using LR^2

$$\overleftrightarrow{h}_\tau = L(\theta) + \lambda_1\left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \theta^*\right] + \lambda_2 + \left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \theta^{2^*}\right] + \lambda_3\left[\frac{\vec{h}_\tau}{\overleftarrow{h}_\tau} \times \left(\theta^* - \mu\right)^2\right]$$

Return $\rightarrow \left(\Re^B, \Re^H, \Re^G, \Re^{SH}, \Re^S\right)$

End

Therefore, the classified DoS attack types, such as Benign $\left(\Re^B\right)$, DoS Hulk Attack $\left(\Re^H\right)$, DoS Golden Eye Attack $\left(\Re^G\right)$, DoS Slow HTTP test attack $\left(\Re^{SH}\right)$, and DoS Slowloris Attack $\left(\Re^S\right)$ are obtained.

Control Layer

In the control layer, the traffic is mitigated and data is secured as discussed below,

3.6 Data Security

Here, $(\Re^B)$ are secured using the ECC technique that ensures efficient encryption and authentication mechanisms using key generation. However, the random method for generating private keys may lead to potential security vulnerabilities if not properly managed or implemented. Therefore, Chaos and Shannon-based techniques are employed that generate private keys to enhance robustness and security in SDN environments.

- At first, $(\Re^B)$ are securely stored based on the elliptical curve (δ) as shown below,

$$u^2 = v^3 + a''u + b'' \tag{24}$$

Here, (a'', b'') represents the parameters of (δ) and (u, v) represents the coordinates in (δ).

- After that, using (p^k, q^k) from (Equation 2,3), the secret key (s^k) is generated as,

$$s^k = p^k \times q^k \tag{25}$$

- Then, $(\Re^B)$ are converted to an unreadable form so that the authorized parties can only access and understand it. Therefore, the encrypted data (Ω) is equated as,

$$\Omega = \left(\Re^B + s^k\right) + \left(q^k(u,v) \times p^k(u,v)\right) \tag{26}$$

Pseudo code of CS-ECC

Input: Non-attacked data, $(\Re^B)$

Output: Encrypted data, (Ω)

Begin

Initialize iterations $(\tau, \tau^{\max})$

While $(\tau < \tau^{\max})$

Initialize $(\Re^B)$

Evaluate (p^k) using CS,

$$p^k = r^{con} \cdot U^{reg} \cdot \left(1 - U^{reg}\right)$$

Generate (p^k, q^k, s^k)

Encrypt $(\Re^B)$

$$\Omega = \left(\Re^B + s^k\right) + \left(q^k(u,v) \times p^k(u,v)\right)$$

End

Decrypt $(\Re^B)$

$$\Psi = \left(\Omega + s^k\right) + p^k$$

End

- The decryption is then carried out for authorized users as,

$$\Psi = \left(\Omega + s^k\right) + p^k \tag{27}$$

Here, (Ψ) represents the decrypted data.

3.7 Traffic Filtration

Meantime, for $(\Re^H, \Re^G)$, the traffic filtration takes place using FIFO-TBA since $(\Re^H, \Re^G)$ causes a high traffic rate. FIFO-TBA prioritizes data requests based on minimum duration time, ensuring efficient and timely processing by managing request flow and security.

Here, the tokens are generated (y^{tok}) for data requests at a constant rate (r) with time (τ). Thus, the (x^a) number of tokens (Z^{tok}) available in a bucket at (τ) is given as,

$$Z^{tok} = \min\left(B^{\max}, \tau^{\max}(\tau - 1) + (\tau - \tau_{last}) \cdot r\right) \quad (28)$$

Here, $(\tau^{\max}, \tau_{last})$ represents the time when the first and last token was added to the bucket and $(B^{\max})$ represents the maximum bucket capacity.

Then, based on the conditions $(B^{\max})$, the request acceptance $(Z^{tok} \geq x^a)$ and declined (wait in queue or later retry) $(Z^{tok} < x^a)$ criteria are performed as shown below,

$$\wp = \begin{cases} if \;\; Z^{tok} \geq x^a, \; then \; Z^{accept-request} \\ if \;\; Z^{tok} < x^a, \; then \; Z^{decline} \end{cases} \quad (29)$$

Here, $(Z^{accept-request})$ represents the accepted requests, which are then secured as shown in (section 3.6), and $(Z^{decline})$ represents the declined request. This process continues until all requests are processed. Simultaneously, $(\Re^{SH}, \Re^{S})$ are blocked as they provide incomplete requests, thus causing service disruption and false positives. Finally, the secured data is stored in cloud storage for future usage or analysis.

4. RESULTS AND DISCUSSION

This section compares the performance assessment of the proposed technique with traditional techniques. The entire work is implemented in the PYTHON platform.

4.1 DATASET DESCRIPTION

The proposed work used two types of datasets, namely AL-DoS-Attack and CSE-CIC-IDS2018 that are gathered from publicly available sources. Totally, the datasets contain 2542915 data with 77 features and 5 classes, namely Benign, DOS Hulk, DoS Slow HTTP Test, DoS Slowloris, and DoS GoldenEye attacks. From the datasets, the proposed work used 2034332 data (80%) for training and 508583 data (20%) for testing.

4.2 Performance analysis of the proposed work

Here, the performance of the proposed LRS^{2}BTM, DP^2-SBOA, and CS-ECC are compared with traditional techniques. The performance assessment of the proposed LRS^{2}BTM is shown in Figure 3,

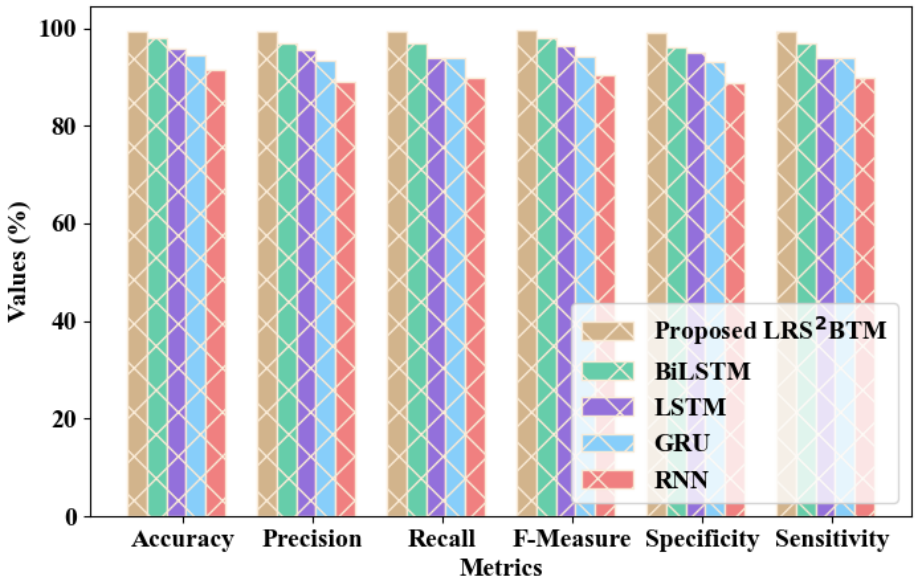


Figure 3: Performance assessment of the proposed LRS^{2}BTM

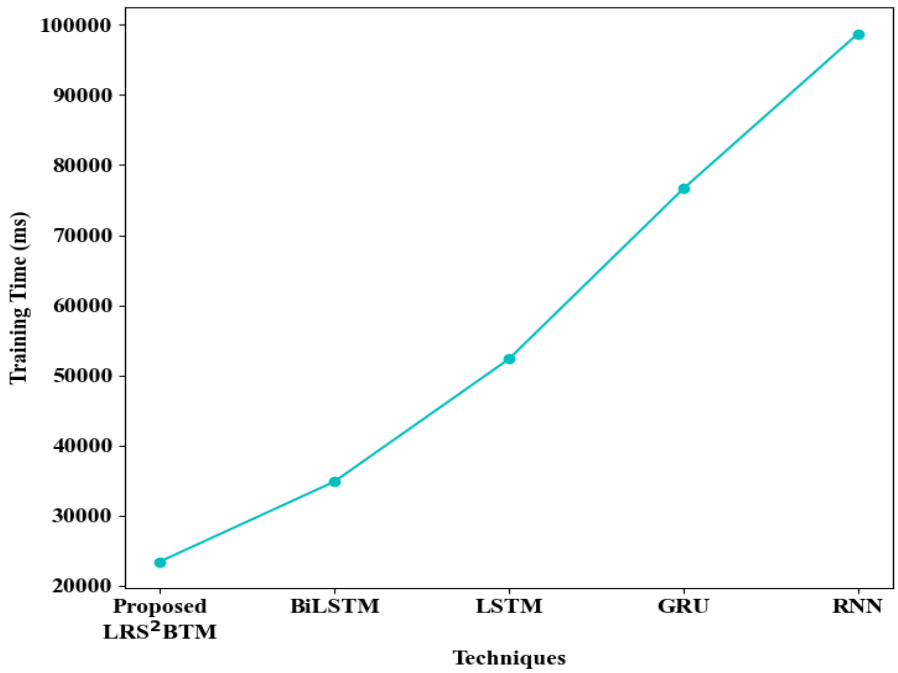


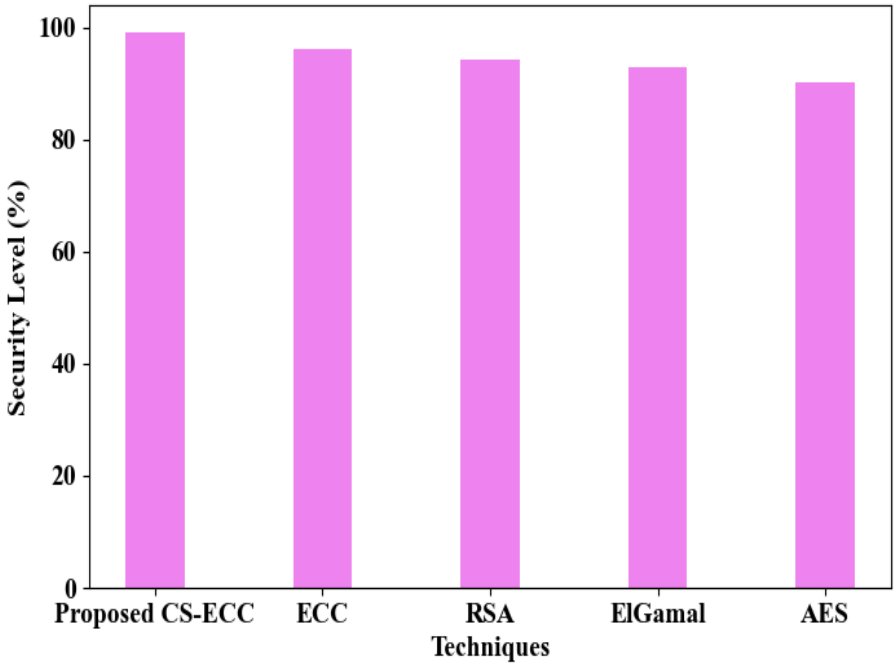
Figure 4: Training Time

Figures 3 and 4 compare the performance assessment of the proposed LRS^{2}BTM and traditional BiLSTM, LSTM, GRU, and RNN techniques. Here, as shown in the figures, when compared to traditional techniques, the proposed work attained high Accuracy (99.48%), Precision (99.39%), Recall (99.31%), F-measure (99.53%), Specificity (99.09%), and Sensitivity (99.31%) with minimum Training Time (23453ms). As LRS^2 usage in BiLSTM overcomes high computational costs and overfitting issues, the proposed technique's performance is enhanced.

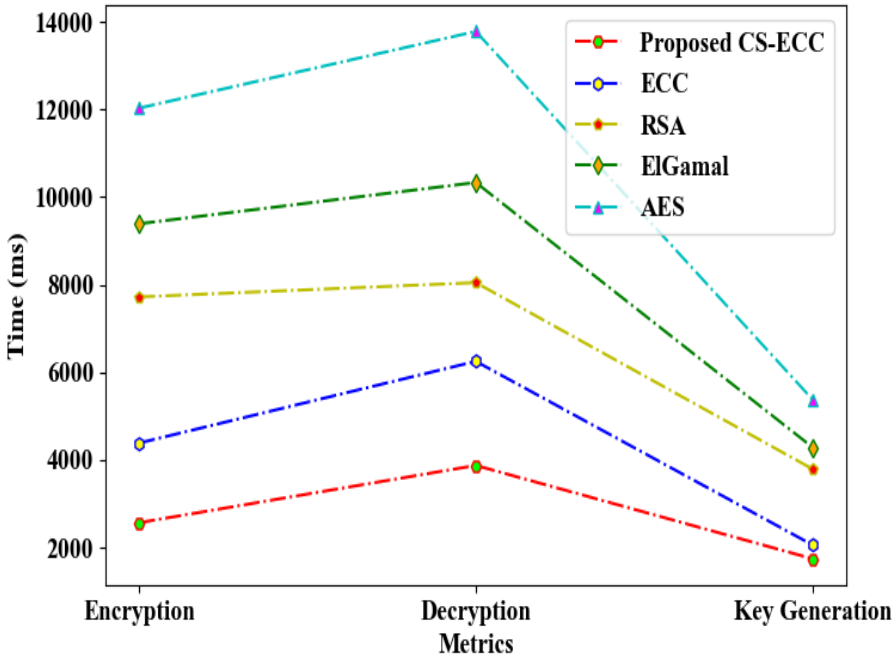
Table 1: Comparative Analysis based on FPR and FNR

Techniques	FPR	FNR
Proposed LRS^{2}BTM	0.0192	0.0337
BiLSTM	0.0573	0.0838
LSTM	0.0887	0.1323
GRU	0.1669	0.1783
RNN	0.2006	0.2103

Table 1 shows the enhanced performance of LRS^{2}BTM in the context of False Positive Rate (FPR) and False Negative Rate (FNR). Here, the proposed work attained minimum FPR and FNR values, while the prevailing techniques had maximum FPR and FNR values. Hence, the reduced performance degraded the entire work process.



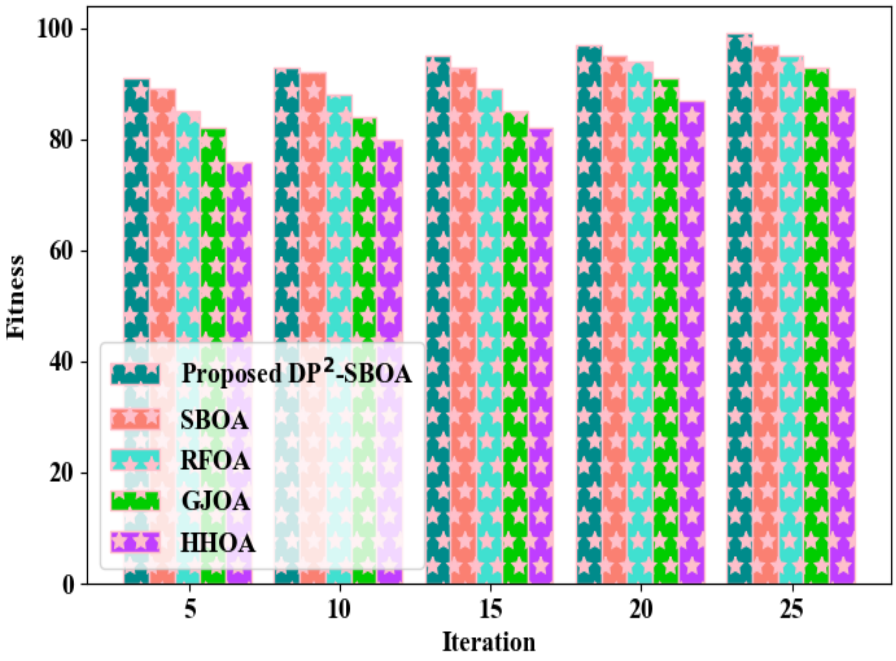
(a)



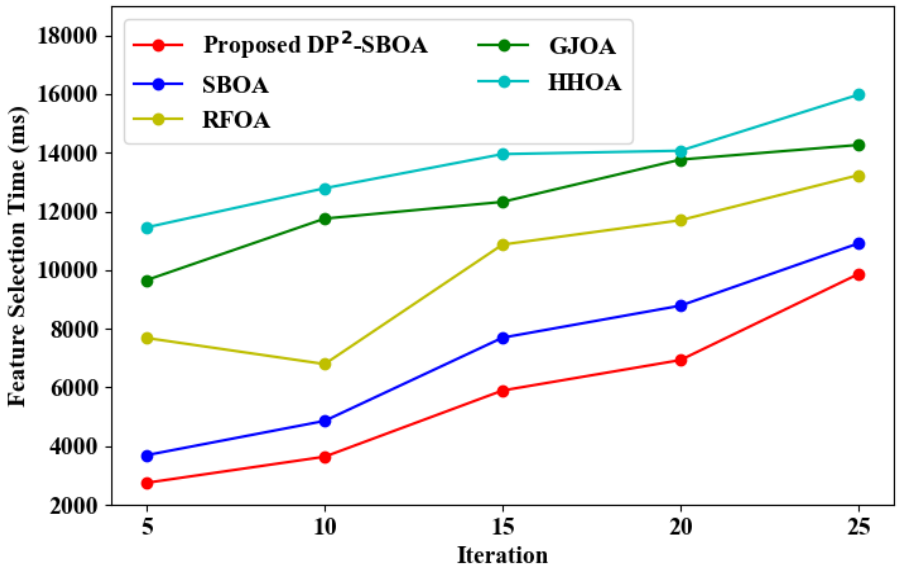
(b)

Figure 5: (a) Security Level and (b) Encryption, Decryption, and Key Generation time of the proposed CS-ECC Technique

Security Level (SL), ET, Decryption Time (DT), and Key Generation Time (KGT) of the proposed work are 98.97%, 2571ms, 3878ms, and 1748ms, respectively, as shown in Figures 5 (a and b). But, the prevailing ECC, Rivest-Shamir-Adleman (RSA), ElGamal, and Advanced Encryption Standard (AES) attained low SL of 96.07%, 94.11%, 92.83%, and 90.04%, respectively with maximum ET, DT and KGT. This is due to the randomness of generating private keys. But, in the proposed work, the CS technique is introduced to generate robust private keys using chaos dynamics and Shannon information theory.



(a)



(b)

Figure 6: Comparative analysis based on (a) Feature Selection Time and (b) Fitness Vs Iteration

The proposed DP^2-SBOA is compared with SBOA, Red Fox Optimization Algorithm (RFOA), Golden Jackal Optimization Algorithm (GJO), and Harris Hawks Optimization Algorithm (HHOA) techniques as shown in Figures 6 (a and b). Additionally, the Feature Selection Time (FST) and Fitness Vs Iteration (FVsI) of the proposed DP^2-SBOA are enhanced when compared with traditional techniques. This is due to the usage of DP^2, which introduced random perturbations to avoid local optima problems. Thus, the proposed work attained minimum FST (9861ms) and high Fitness value (99) in the 25th iteration. Thus, the proposed work was more efficient than traditional techniques.

Table 2: Comparative Analysis with Related Works

Study	Techniques	Accuracy (%)	F-measure (%)
Proposed Work	LRS^{2}BTM	99.48	99.53

(Wang & Wang, 2022)	CNN-ELM	98.92	98.74
(Wang et al., 2022)	DT	96.79	-
(Eliyan & Pietro, 2023)	EWMA	91.27	-
(Lent et al., 2022)	GRU	-	98.95
(ElSayed et al., 2021)	CNN	-	98.58

Table 2 compares the proposed work with related works. As shown in the table, the proposed work used the LRS^{2}BTM technique, which attained high Accuracy and F-measure due to the inclusion of LRS^2. So, the performance was enhanced with faster convergence. But, the prevailing works used Convolutional Neural Network and Extreme Learning Machine (CNN-ELM), GRU, and CNN techniques and had a low average F-measure of 98.75%. Also, (Wang et al., 2022 and Eliyan & Pietro, 2023) used DT and Exponentially Weighted Moving Average (EWMA) techniques as they had slow convergence and overfitting issues. Hence, when compared to prevailing works, the proposed work outperformed in detecting and mitigating the DoS attacks.

5. CONCLUSION

This research effectively secures SDN by detecting and mitigating the DoS attacks. At first, the user registers and logs in to detect the DoS-attacked data. Here, the DoS attack types were classified using LRS^{2}BTM with an F-measure of 99.53%. Before that, the optimal features were selected in 2749ms (5th iteration). Then, DoS Hulk and Goldeneye attacks were mitigated using FIFO-TSA and the DoS Slow HTTP and Slowloris attacks were blocked due to incomplete requests. Finally, the data is secured with 98.97% SL and then stored in the cloud for future usage.

Future Recommendation

However, in the future, advanced mitigation strategies will be implemented against DoS Slow HTTP and Slowloris attacks for more enhanced SDN security.

REFERENCE

Dataset links:

1. https://www.kaggle.com/code/hamzasamiullah/ml-analysis-application-layer-dos-attack-dataset/input?select=train_mosaic.csv
2. https://www.kaggle.com/datasets/dhoogla/csecicids2018
3. AbdelAzim, N. M., Fahmy, S. F., Sobh, M. A., & Bahaa Eldin, A. M. (2021). A hybrid entropy-based DoS attacks detection system for software defined networks (SDN): A proposed trust mechanism. *Egyptian Informatics Journal*, *22*(1), 85–90. https://doi.org/10.1016/j.eij.2020.04.005
4. Alashhab, A. A., Zahid, M. S. M., Azim, M. A., Daha, M. Y., Isyaku, B., & Ali, S. (2022). A Survey of Low Rate DDoS Detection Techniques Based on Machine Learning in Software-Defined Networks. *Symmetry*, *14*(8), 1–30. https://doi.org/10.3390/sym14081563
5. Ali, T. E., Chong, Y. W., & Manickam, S. (2023). Machine Learning Techniques to Detect a DDoS Attack in SDN: A Systematic Review. *Applied Sciences (Switzerland)*, *13*(5), 1–27. https://doi.org/10.3390/app13053183
6. Alshra'a, A. S., Farhat, A., & Seitz, J. (2021). Deep Learning Algorithms for Detecting Denial of Service Attacks in Software-Defined Networks. *Procedia Computer Science*, *191*, 254–263. https://doi.org/10.1016/j.procs.2021.07.032
7. Bahashwan, A. A., Anbar, M., Manickam, S., Aladaileh, M. A., & Hasbullah, I. H. (2023). A Systematic Literature Review on Machine Learning and Deep Learning Approaches for Detecting DDoS Attacks in Software-Defined Networking. *Sensors*, *23*(9), 1–48. https://doi.org/10.3390/s23094441
8. Bhayo, J., Shah, S. A., Hameed, S., Ahmed, A., Nasir, J., & Draheim, D. (2023). Towards a machine

learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks. *Engineering Applications of Artificial Intelligence*, *123*, 1–17. https://doi.org/10.1016/j.engappai.2023.106432

9. Eliyan, L. F., & Di Pietro, R. (2021). DoS and DDoS attacks in Software Defined Networks: A survey of existing solutions and research challenges. *Future Generation Computer Systems*, *122*, 149–171. https://doi.org/10.1016/j.future.2021.03.011
10. Eliyan, L. F., & Pietro, R. Di. (2023). DeMi: A Solution to Detect and Mitigate DoS Attacks in SDN. *IEEE Access*, *11*, 82477–82495. https://doi.org/10.1109/ACCESS.2023.3301994
11. ElSayed, M. S., Le-Khac, N. A., Albahar, M. A., & Jurcut, A. (2021). A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of Network and Computer Applications*, *191*, 1–18. https://doi.org/10.1016/j.jnca.2021.103160
12. ElSayed, M. S., Le-Khac, N. A., Albahar, M. A., & Jurcut, A. (2021). A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique. *Journal of Network and Computer Applications*, *191*, 1–18. https://doi.org/10.1016/j.jnca.2021.103160
13. Lent, D. M. B., Novaes, M. P., Carvalho, L. F., Lloret, J., Rodrigues, J. J. P. C., & Proenca, M. L. (2022). A Gated Recurrent Unit Deep Learning Model to Detect and Mitigate Distributed Denial of Service and Portscan Attacks. *IEEE Access*, *10*, 73229–73242. https://doi.org/10.1109/ACCESS.2022.3190008
14. Li, J., Tu, T., Li, Y., Qin, S., Shi, Y., & Wen, Q. (2022). DoSGuard: Mitigating Denial-of-Service Attacks in Software-Defined Networks. *Sensors*, *22*(3), 1–17. https://doi.org/10.3390/s22031061
15. Rios, V. D. M., Inacio, P. R. M., Magoni, D., & Freire, M. M. (2022). Detection and Mitigation of Low-Rate Denial-of-Service Attacks: A Survey. *IEEE Access*, *10*, 76648–76668. https://doi.org/10.1109/ACCESS.2022.3191430
16. Swami, R., Dave, M., & Ranga, V. (2023). IQR-based approach for DDoS detection and mitigation in SDN. *Defence Technology*, *25*, 76–87. https://doi.org/10.1016/j.dt.2022.10.006
17. Valdovinos, I. A., Perez-Diaz, J. A., Choo, K. K. R., & Botero, J. F. (2021). Emerging DDoS attack detection and mitigation strategies in software-defined networks: Taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, *187*, 1–29. https://doi.org/10.1016/j.jnca.2021.103093
18. Wang, D., Zhao, Y., Zhi, H., Wu, D., Zhuo, W., Lu, Y., & Zhang, X. (2023). DoSDefender: A Kernel-Mode TCP DoS Prevention in Software-Defined Networking. *Sensors*, *23*(12), 1–11. https://doi.org/10.3390/s23125426
19. Wang, J., & Wang, L. (2022). SDN-Defend: A Lightweight Online Attack Detection and Mitigation System for DDoS Attacks in SDN. *Sensors*, *22*(21), 1–21. https://doi.org/10.3390/s22218287
20. Wang, S., Balarezo, J. F., Chavez, K. G., Al-Hourani, A., Kandeepan, S., Asghar, M. R., & Russello, G. (2022). Detecting flooding DDoS attacks in software defined networks using supervised learning techniques. *Engineering science and technology, an international journal*, *35*, 1-17. https://doi.org/10.1016/j.jestch.2022.101176
21. Wang, S., Balarezo, J. F., Chavez, K. G., Al-Hourani, A., Kandeepan, S., Asghar, M. R., & Russello, G. (2022). Detecting flooding DDoS attacks in software defined networks using supervised learning techniques. *Engineering Science and Technology, an International Journal*, *35*, 1–17. https://doi.org/10.1016/j.jestch.2022.101176
22. Yeom, S., Choi, C., & Kim, K. (2022). LSTM-Based Collaborative Source-Side DDoS Attack Detection. *IEEE Access*, *10*, 44033–44045. https://doi.org/10.1109/ACCESS.2022.3169616