Journal of Computer Science

Research Article

Autoencoder-Based Intrusion Detection for ARP Poisoning in SDN Controllers

Kago Seemela, Banyatsang Mphago, Thabo Semong and Phephisa Methula

Department of Computing and Informatics, Botswana International University of Science and Technology (BIUST), Palapye, Botswana

Article history
Received: 20-03-2026
Revised: 15-05-2026
Accepted: 04-06-2026

Corresponding Author:
Thabo Semong
Department of Computing and Informatics, Botswana International University of Science and Technology (BIUST), Palapye, Botswana
Email: Semongt@biust.ac.bw

Abstract: Address Resolution Protocol (ARP) poisoning is still a critical control plane attack in Software Defined Networks (SDNs), and spoofing of ARP mappings can trick SDNs controller and accomplish man in the middle, eavesdropping and denial of service attacks. Current SDN intrusion detection systems of ARP attacks rely on supervised learning and labelled attack data which is rather expensive and not necessarily a good representation of evolving traffic. Hence, this paper presents an unsupervised IDS that uses autoencoder to detect ARP poisoning attacks on the SDN controller. The model is trained only on benign ARP data from the public ARP–SDN dataset and ARP poison/flood labels are used only for the evaluation. The method used is the reconstruction error and an anomaly score threshold is chosen from the 99th percentile of the reconstruction errors of benign training errors. The proposed model attains a binary-accuracy of 99.83%, a macro F1-score of 0.9978 and 0 false-negative on the test data which it did not trained. For tenfold cross-validation, mean accuracy of 0.9974 and macro F1-score of 0.9965 have been achieved. The results indicate that autoencoder based anomaly detection can yield a good ARP poisoning prediction accuracy with less need for labelled attack data, but more testing on real SDN deployments is still necessary.

Keywords: Software Defined Networking, ARP Poisoning, Intrusion Detection System, Autoencoder, Anomaly Detection, Unsupervised Learning

Introduction

Software Defined Networking (SDN) separates the control plane from the data plane and gives network administrators a programmable and logically centralized view of the network. This design simplifies the deployment of policy and network management, but it also increases the importance of protecting the SDN controller because a compromised controller view can affect forwarding decisions across the network (Dabbagh et al., 2015; Semong et al., 2018; 2020).

One persistent threat in both traditional and SDN environments is Address Resolution Protocol (ARP) poisoning. ARP poisoning, also known as ARP spoofing, occurs when a malicious host injects forged IP–MAC mappings into ARP caches and impersonates a legitimate gateway or host Dhawan et al. (2015); Hijazi and Obaidat (2018). Once the attacker is placed on the traffic path, the attack can support eavesdropping, session hijacking, MAC flooding and denial-of-service attacks Rietz et al. (2018). In SDN, the impact can be more severe because ARP information may be processed by, or reflected in, the controller's network view and host-location logic Shah and Cosgrove (2019).

Intrusion Detection Systems (IDSs) are widely used to detect such abnormal behaviour. Misuse-based IDSs rely on known signatures, while anomaly-based IDSs identify deviations from normal network behaviour Cannady (1998); Al Jallad et al. (2020). Existing machine learning studies show that supervised classifiers can detect SDN attacks with promising accuracy Li and Li (2018); Chou et al. (2019); Nanda et al. (2016). Deep learning models such as long short-term memory networks, convolutionalneural networks and autoencoders have also been used to model complex traffic patterns Javaid et al. (2016); Liu et al. (2019); Said et al. (2020); Mhamdi et al. (2020).

For ARP attacks in SDN, previous studies have mainly adopted supervised learning. Unal et al. (2018) compared supervised classifiers such as logistic regression, Bayes Net, decision trees and random forests, reporting that the

best-performing model achieved detection rates of about 80% Unal et al. (2018). Sebbar proposed a MitM CBNA-RF approach that achieved accuracy above 90%, but it still required labelled traffic Sebbar et al. (2020). Ahuja et al. (2022a) introduced the ARP–SDN dataset and evaluated several supervised classifiers on benign, ARP poison and ARP flood traffic Ahuja et al. (2022b). Although these studies are important, their dependence on labelled attack traffic limits their practicality when labels are scarce or when attack patterns evolve.

Problem Statement

Most existing ARP poisoning detection models for SDN are trained in a supervised manner and require labelled benign and attack samples Unal et al. (2018); Sebbar et al. (2020); Ahuja et al. (2022c). In realistic deployments, benign ARP traffic is abundant, while labelled ARP poisoning examples are limited, costly to collect and may not cover new attack variants Altalhan et al. (2025); Myakala (2025); Kumar and Dash (2024); Towhid et al. (2024). Therefore, a practical IDS should learn normal ARP behaviour from benign traffic and detect deviations without depending on labelled attack data during training.

Research Questions and Contributions

To strengthen the clarity of the work, this study is based on the following research questions:

- RQ 1: Is it possible for an autoencoder trained only on benign ARP traffic to be able to detect ARP poison and ARP flood flows in an SDN dataset
- RQ 2: How stable will the autoencoder IDS performs under different benign training sizes, and the 10 fold cross validation
- RQ 3: How is the proposed unsupervised detector compared to ARP detection approaches in the literature that are supervised

The main contributions of this paper can be summarized as:

- This work presents an autoencoder based IDS for ARP poisoning detection in the SDN controllers, trained with one class learning with benign ARP flows
- Revise and clarify ARP-SDN data source, original classes, topology and selected input features
- Discuss an an anomaly detection procedure based on a threshold, which is the 99th percentile of the benign reconstruction errors
- Present a performance evaluation using holdout testing, training size sensitivity analysis and ten fold cross validation
- Carry out comparison with the existing supervised learning ARP poisoning detection methods and discuss real world deployment issues

Hypothesis

The hypothesis of this study is that a benign ARP autoencoder will be able to reconstruct normal ARP flows with small error, and will be able to recognize ARP poisoning or flooding as high error flows. The model is not trained with a supervised classification objective and labels are used only to isolate benign samples for training and to assess the end result of the predictions.

Materials and Methods

SDN Security and ARP Poisoning

SDN places control logic in a controller that manages network behaviour using programmable interfaces. This makes the controller central to network operation and also makes its control-plane information a high-value target. ARP poisoning can mislead the controller about host locations or address mappings, allowing an attacker to redirect or disrupt traffic at scale Dhawan et al. (2015); Shah and Cosgrove (2019).

Machine Learning for SDN Intrusion Detection

Several studies have used machine learning to detect abnormal SDN traffic. Lightweight classifiers have been proposed for flow-based anomaly detection, while supervised models have been used to classify network flows into benign and malicious categories Li and Li (2018); Nanda et al. (2016). In ARP-focused work, Unal et al. (2018) compared multiple supervised classifiers for ARP poisoning detection and Sebbar proposed a random-forest-based MitM detection approach Unal et al. (2018); Sebbar et al. (2020). Ahuja et al. (2022a) created a specialised ARP–SDN dataset and evaluated supervised models on ARP poison and ARP flood traffic Ahuja et al. (2022b).

Autoencoders and Unsupervised Anomaly Detection

Autoencoders reconstruct their input through a compressed latent representation. When trained on normal traffic, they usually reconstruct benign patterns with low error, while unseen or abnormal patterns produce larger reconstruction errors. This property makes autoencoders suitable for one-class anomaly detection, especially when labelled attack data are scarce Liu et al. (2019); Mhamdi et al. (2020); Said et al. (2020). Most autoencoder-based SDN studies focus on DDoS or general anomaly detection, leaving ARP-specific autoencoder detection underexplored.

ARP–SDN Dataset and Preprocessing

Dataset Source and System Setup

The Experiments are conducted based on the public ARP–SDN dataset created by Ahuja et al. (2022a-c) as depicted on Table 1. The source data are generated in

SDN environment by using Mininet and Ryu controller. The public repository is a tree topology with depth 3 and fanout 3 made up of 27 hosts, 13 switches and one controller.

There are 3 original class labels in the source dataset namely benign, ARP poison, ARP flood. In the current study, only the benign records are used to train the autoencoder. ARP poison and ARP flood data are combined in the evaluation as the proposed model is a one-class anomaly detector and not a supervised multiclass classifier.

Feature Selection and Preprocessing

The nine-feature input vector was selected using three criteria: Availability after preprocessing, numerical suitability for autoencoder training and relevance to ARP/SDN control-plane behaviour as depicted on Table 2. The retained features capture switch-level forwarding context, port-level movement, controller interaction through packet-in statistics, ARP operation behaviour and network performance indicators such as round-trip time and packet loss.

Table 1: Class distribution and split used for one-class evaluation

Class	Train	Test	Total
Benign ARP flows	6739	33694	26955
ARP poison/flood	0	20061	20061
Total	26955	26800	53755

Raw MAC and IP address fields were excluded because they can encourage host-specific memorisation rather than learning general ARP behaviour. Preprocessing involved replacing missing values with zero, converting available categorical or hexadecimal fields into numerical representations, removing non-generalising identifiers and normalising all retained numerical features to the range [0,1] using min–max scaling:

$$zi = \frac{xi - \min(x)}{man(x) - \min(x)} \quad (1)$$

Proposed Autoencoded Intrusion Detection System

System Architecture

Figure 1 shows the proposed IDS workflow. ARP-related flow records are collected from the SDN environment, transformed into the selected feature vector, preprocessed and passed to the autoencoder. During deployment, the trained autoencoder reconstructs each incoming flow and computes a reconstruction error. If the error exceeds the threshold, the controller can alert, drop, rate-limit or further inspect the flow before allowing it to influence ARP cache updates.

Autoencoder Model

The autoencoder is a fully connected feed-forward neural network with nine input neurons corresponding to the selected features. The encoder compresses the input through 32 and 16 neurons into an 8-neuron bottleneck, and the decoder reconstructs the original nine-dimensional input. Hidden layers use ReLU activation, and the output layer uses linear activation. Table 3 summarises the architecture and hyperparameters.

Table 2: Selected features and rationale for inclusion Plummer (1982); Almes et al. (1999; 2016)

Selected feature	Rationale
Switch identifier	Captures where the ARP-related event is observed in the SDN topology.
Input port	Indicates the ingress point of the ARP flow and helps expose abnormal source-port behaviour.
Output port	Provides forwarding-context information used by the controller and switches.
ARP operation code	Distinguishes request/reply behaviour, which is important in poisoning scenarios.
Packet-in count	Measures controller interaction and helps detect abnormal ARP bursts or flooding.
Protocol code	Confirms protocol context and separates ARP-related records from other protocol events.
Packet loss	Captures degradation that may occur during flooding or misdirection.
Average RTT	Measures delay behaviour and helps identify abnormal network response patterns.
Total time/ ping statistics	Summarises timing behaviour associated with benign and attack traffic.

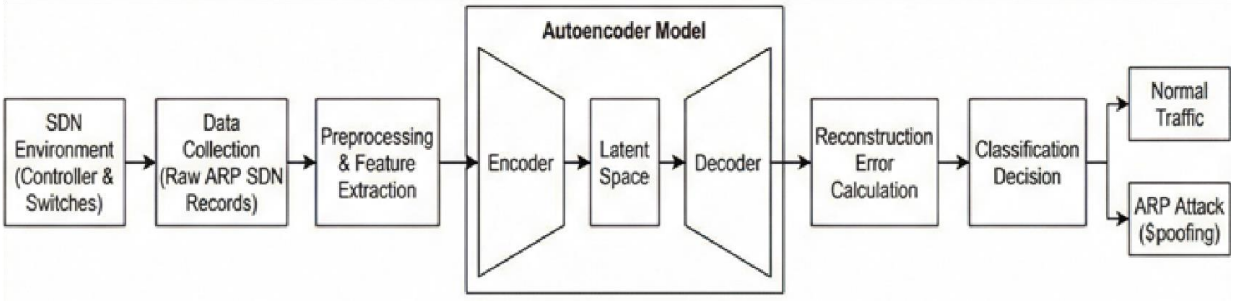

Fig. 1: Conceptual workflow of the autoencoder-based ARP poisoning IDS

Table 3: Autoencoder architecture and hyperparameters

Component	Setting
Input layer	9 neurons
Encoder hidden 1	32 neurons, ReLU
Encoder hidden 2	16 neurons, ReLU
Bottleneck	8 neurons, ReLU
Decoder hidden 1	16 neurons, ReLU
Decoder hidden 2	32 neurons, ReLU
Output layer	9 neurons, linear
Loss function	Mean squared error
Optimiser	Adam, learning rate $10\rightarrow^{3}$
Batch size	256
Epochs	100, with early stopping
Validation split	20% of benign training data

Experimental Setup

Threshold Selection and Classification

After training, the reconstruction error is computed for all benign training samples. Let Etrain = {*ei*} denote the set of benign reconstruction errors. The threshold ! is selected as the 99th percentile of Etrain. This percentile-based threshold is appropriate for one-class anomaly detection because it is derived only from benign traffic and does not require labelled attacks. It also controls the expected false-alarm rate on benign training traffic to approximately 1%, while retaining sensitivity to high-error attack flows:

$$e(x) = MSE(x, x) \tag{2}$$

$$label(x) + \begin{cases} Benign, & e(x) \le \tau, \\ Attack, & e(x) > \tau, \end{cases} \tag{3}$$

All experiments were implemented in Python using TensorFlow/Keras for the autoencoder and scikit-learn for preprocessing, train-test splitting, cross-validation and performance evaluation. The evaluation is binary: Benign versus attack, where the attack class combines ARP poison and ARP flood records. The reported metrics are accuracy, precision, recall, F1-score, macro F1-score and confusion matrix counts.

Results

Training Behaviour

Figure 2 shows that both training and validation reconstruction losses decrease sharply in the first epochs and then converge to low values. The close tracking between the two curves suggests that the model learns a stable benign representation without obvious overfitting.

Reconstruction Error Distributions

The benign training error distribution in Figure 3 is concentrated near zero, and the selected threshold is approximately 4.3 ↓ 10→5. Figure 4 shows that attack samples produce substantially larger reconstruction errors than benign samples, supporting the threshold-based anomaly decision.

Binary Classification Performance

Table 4 reports the binary classification metrics, and Figure 5 shows the confusion matrix. The model achieves 99.83% accuracy and macro F1-score of 0.9978. Most importantly, the test run records no false negatives for the attack class, meaning that all ARP poison/flood samples in the test set were detected.

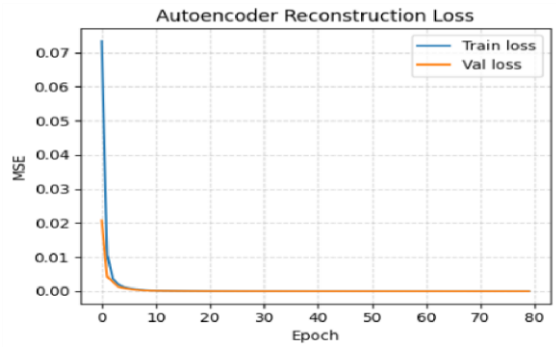

Fig. 2: Training and validation reconstruction loss of the autoencoder

Fig. 3: Reconstruction error distribution for benign training data

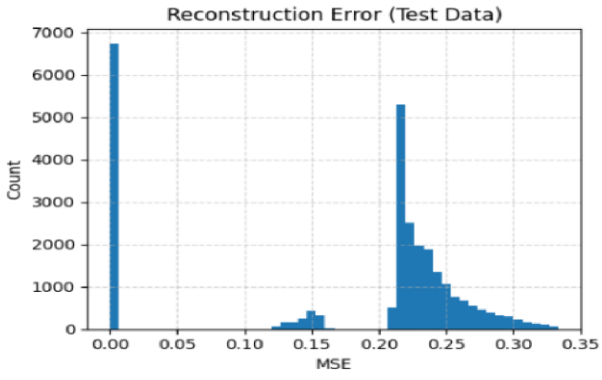

Fig. 4: Reconstruction error distribution on the test set

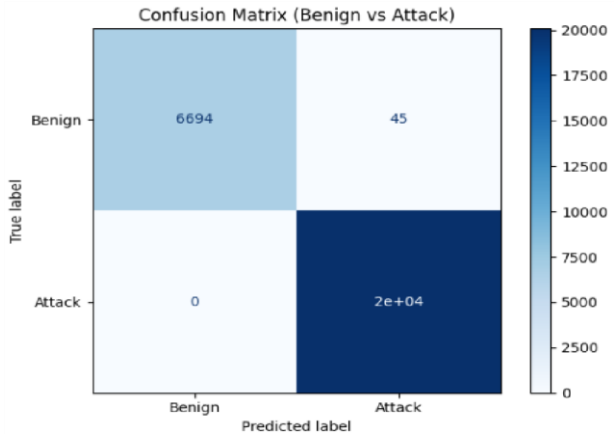

Fig. 5: Confusion matrix for binary classification

Table 4: Binary classification metrics on the test set

Class	Precision	Recall	F1	Support
Benign	1.0000	0.9933	0.9967	6739
Attack	0.9978	1.0000	0.9989	20061
Accuracy		0.9983		26800
Macro avg	0.9989	0.9967	0.9978	–
Weighted avg	0.9983	0.9983	0.9983	–

Effect of Training Dataset Size

Table 5 and Figure 6 show that the proposed detector maintains high accuracy even with 20% of the benign training data. The gains after 40% are marginal, suggesting that the model can learn a useful normal profile from a limited benign history

10-fold Cross-Validation

To address robustness, 10-fold cross-validation was performed on the complete dataset. In each fold, the autoencoder was trained only on benign samples from the training portion and evaluated on the corresponding test fold containing benign and attack samples. Table 6 shows low standard deviations, indicating stable performance across splits.

Comparison With Supervised Approaches

Table 7 compares the proposed model with representative supervised approaches. Although evaluation protocols differ across studies, the comparison shows that the proposed one-class autoencoder achieves competitive or better detection performance while avoiding supervised training on labelled attack traffic.

Table 5: Effect of benign training dataset size on performance

Benign train fraction	Accuracy	Macro F1
20%	0.9976	0.9968
40%	0.9983	0.9977
60%	0.9980	0.9974
80%	0.9980	0.9974
100%	0.9977	0.9969

Table 6: 10-fold cross-validation summary

Metric	Mean	Std. dev.
Accuracy	0.9974	0.0004
Macro F1	0.9965	0.0006

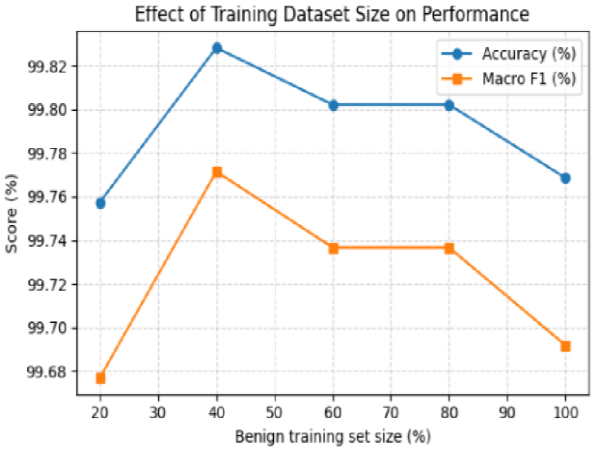

Fig. 6: Accuracy and macro F1 against benign training dataset size

Table 7: Comparison with representative ARP poisoning detection approaches in SDN

Study Approach	Learning type	Dataset / Scenario	Reported performance
Unal et al. (2018) Logistic regression and other ML classifiers	Supervised	SDN testbed ARP poisoning	Best detection rate about 80%
Sebbar et al. (2020) CBNA-RF / random forest	Supervised	Large-scale SDN MitM	Accuracy above 90%; labelled traffic required
Ahuja et al. (2022b) ML classifiers on ARP–SDN	Supervised	ARP-specific SDN dataset	Mid-high 90%range; labelled attacks used
This work Autoencoder anomaly detector	Unsupervised one-class	ARP–SDN dataset	99.83% accuracy, macro F1 0.9978, no missed attacks

Discussion

Benefits and Practical Implications

The results support the hypothesis that benign-only autoencoder training can detect ARP poison/flood traffic as high-error anomalies. The main practical benefit is reduced reliance on labelled attack data. In a deployment, the model could be integrated as a controller-side monitoring module that inspects ARP-related records before ARP cache updates and generates alerts, drop rules or rate-limiting actions for suspicious flows.

External Validity and Real-World Validation

This study uses the public ARP–SDN dataset because it is one of the few ARP-specific SDN datasets with documented poison and flood classes. The results should therefore be interpreted as controlled-dataset evidence rather than direct proof of production readiness. Future validation must include live controller deployments, heterogeneous topologies, additional controllers and traffic c collected from operational SDN or industrial network environments. The 10-fold cross-validation and training-size sensitivity analysis provide internal robustness, but they do not replace external real-world validation.

Limitations

The main limitations are:

(i) Poisoning and flooding are grouped into a single attack class
(ii) The threshold may need adaptation in networks with concept drift
(iii) The dataset was generated in a controlled Mininet/Ryu environment. These limits should be considered when interpreting the reported accuracy

Ethical Considerations

The study uses a public research dataset and does not process personally identifiable information. The proposed detector is intended for defensive monitoring of ARP traffic in SDN environments. However, IDS models can generate false positives, and automatic blocking may affect legitimate users. Therefore, practical deployment should include operator's review, logging, safe rollback of mitigation rules and compliance with local data-governance policies. The attack-generation information from the public dataset should be used only for authorised research, testing and defence.

Recommendations for Deployment

For practical use, the autoencoder should be integrated close to the SDN controller, retrained periodically on recent benign traffic, combined with lightweight rule-based checks for critical infrastructure hosts, and extended with temporal ARP-rate features to improve flood discrimination.

Conclusion

This paper presented an autoencoder-based intrusion detection system for ARP poisoning detection in SDN controllers. The model is trained only on benign ARP traffic and uses reconstruction error as an anomaly score. The study clarifies the dataset source, feature selection, threshold selection and evaluation design. The results show 99.83% binary accuracy, macro F1-score of 0.9978 and no missed attacks on the held-out test set, with stable10-fold cross-validation performance. Future work will focus on external validation on live SDN environments, adaptive thresholding, multiclass attack attribution and online learning.

Acknowledgment

This work was not funded by any commercial, public or non-profit funding agency.

Funding Information

This work was not funded by any commercial, public or non-profit funding agency.

Author's Contributions

Kago Seemela: Designed the study, implemented the experiments and drafted the manuscript.

Banyatsang Mphago: Supervised the work and reviewed the methodology.

Thabo Semong: Supervised the work and reviewed the manuscript.

Phephisa Methula: Supported problem formulation and implementation review. All authors approved the final manuscript.

Ethics

The study uses a publicly available research dataset and follows the ethical guidelines of the Journal of Computer Science.

Conflict of Interest

The authors declare no conflict of interest.

Data Availability

The ARP–SDN dataset used in this study is publicly available from Mendeley Data at https://data.mendeley.com/datasets/yxzh9fbvbj/2. The corresponding dataset-generation repository is available at https://github.com/nisha077/ARP-SDN-Dataset.

References

Ahuja, N., Singal, G., & Mukhopadhyay, D. (2022a). ARP Poisoning and Flood attack in SDN. *Mendeley Data*. https://doi.org/10.17632/yxzh9fbvbj.2

Ahuja, N., Singal, G., Mukhopadhyay, D., & Nehra, A. (2022b). Ascertain the efficient machine learning approach to detect different ARP attacks. *Computers and Electrical Engineering*, *99*, 107757. https://doi.org/10.1016/j.compeleceng.2022.107757

Ahuja, N. (2022c). ARP-SDN-Dataset. *GitHub*.

Al Jallad, K., Aljnidi, M., & Desouki, M. S. (2020). Anomaly detection optimization using big data and deep learning to reduce false-positive. *Journal of Big Data*, *7*(1), 68. https://doi.org/10.1186/s40537-020-00346-1

Almes, G., Kalidindi, S., & Zekauskas, M. (1999). *A Round-trip Delay Metric for IPPM*. https://doi.org/10.17487/rfc2681

Almes, G., Kalidindi, S., & Zekauskas, M. (2016). *A One-Way Loss Metric for IP Performance Metrics (IPPM)*. https://doi.org/10.17487/rfc7680

Altalhan, M., Algarni, A., & Turki-Hadj Alouane, M. (2025). Imbalanced Data Problem in Machine Learning: A Review. *IEEE Access*, *13*, 13686–13699. https://doi.org/10.1109/access.2025.3531662

Cannady, J. (1998). Artificial neural networks for misuse detection. *Proceedings of the 1998 National Information Systems Security Conference (NISSC'98*, 443–456.

Chou, L.-D., Liu, C.-C., Lai, M.-S., Chiu, K.-C., Tu, H.-H., Su, S., Lai, C.-L., Yen, C.-K., & Tsai, W.-H. (2019). Behavior Anomaly Detection in SDN Control Plane: A Case Study of Topology Discovery Attacks. *2019 International Conference on Information and Communication Technology Convergence (ICTC)*, 357–362. https://doi.org/10.1109/ictc46691.2019.8939903

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Software-defined networking security: pros and cons. *IEEE Communications Magazine*, *53*(6), 73–79. https://doi.org/10.1109/mcom.2015.7120048

Dhawan, M., Poddar, R., Mahajan, K., & Mann, V. (2015). SPHINX: Detecting Security Attacks in Software-Defined Networks. *Proceedings 2015 Network and Distributed System Security Symposium*, 1–15. https://doi.org/10.14722/ndss.2015.23064

Hijazi, S., & Obaidat, M. S. (2018). A New Detection and Prevention System for ARP Attacks Using Static Entry. *IEEE Systems Journal*, *13*(3), 2732–2738.

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. (2016). *A Deep Learning Approach for Network Intrusion Detection System*. 1–6. https://doi.org/10.4108/eai.3-12-2015.2262516

Kumar, M., & Dash, C. S. (2024). Detecting and Preventing ARP Spoofing Attacks Using Real-Time Data Analysis and Machine Learning. *International Journal of Innovative Research in Computer Science and Technology*, *12*(5), 47–55. https://doi.org/10.55524/ijircst.2024.12.5.7

Li, D., & Li, Z. (2018). A Lightweight Traffic Anomaly Detection Model in SDN Based on Decision Tree. *3rd International Conference on Communications, Information Management and Network Security*, 35–39. https://doi.org/10.2991/cimns-18.2018.8

Liu, Z., He, Y., Wang, W., & Zhang, B. (2019). DDoS attack detection scheme based on entropy and PSO-BP neural network in SDN. *China Communications*, *16*(7), 144–155. https://doi.org/10.23919/jcc.2019.07.012

Mhamdi, L., McLernon, D., El-moussa, F., Raza Zaidi, S. A., Ghogho, M., & Tang, T. (2020). A Deep Learning Approach Combining Autoencoder with One-class SVM for DDoS Attack Detection in SDNs. *2020 IEEE Eighth International Conference on Communications and Networking (ComNet)*, 1–6. https://doi.org/10.1109/comnet47917.2020.9306073

Myakala, P. K. (2025). Scaling AI with Limited Labeled Data: A Self-Supervised Learning Approach. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, *2*(1), 26–35. https://doi.org/10.62762/tetai.2025.607708

Nanda, S., Zafari, F., DeCusatis, C., Wedaa, E., & Yang, B. (2016). Predicting network attack patterns in SDN using machine learning approach. *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, 167–172. https://doi.org/10.1109/nfv-sdn.2016.7919493

Plummer, D. C. (1982). *An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. https://doi.org/10.17487/rfc826

Rietz, R., Cwalinski, R., König, H., & Brinner, A. (2018). An SDN-Based Approach to Ward Off LAN Attacks. *Journal of Computer Networks and Communications*, *2018*, 1–12. https://doi.org/10.1155/2018/4127487

Said, M. E., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2020). Network Anomaly Detection Using LSTM Based Autoencoder. *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, 37–45. https://doi.org/10.1145/3416013.3426457

Sebbar, A., ZKIK, K., Baddi, Y., Boulmalf, M., & Kettani, M. D. E.-C. E. (2020). MitM detection and defense mechanism CBNA-RF based on machine learning for large-scale SDN context. *Journal of Ambient Intelligence and Humanized Computing*, *11*(12), 5875–5894. https://doi.org/10.1007/s12652-020-02099-4

Semong, T., Maupong, T., Anokye, S., Kehulakae, K., Dimakatso, S., Boipelo, G., & Sarefo, S. (2020). Intelligent Load Balancing Techniques in Software Defined Networks: A Survey. *Electronics*, *9*(7), 1091. https://doi.org/10.3390/electronics9071091

Semong, T., Xie, K., Zhou, X., Singh, H., & Li, Z. (2018). Delay Bounded Multi-Source Multicast in Software-Defined Networking. *Electronics*, *7*(1), 10. https://doi.org/10.3390/electronics7010010

Shah, Z., & Cosgrove, S. (2019). Mitigating ARP Cache Poisoning Attack in Software-Defined Networking (SDN): A Survey. *Electronics*, *8*(10), 1095. https://doi.org/10.3390/electronics8101095

Towhid, M. S., Khan, N. S., Hasan, M. M., & Shahriar, N. (2024). Towards Effective Network Intrusion Detection in Imbalanced Datasets: A Hierarchical Approach. *2024 International Conference on Computing, Networking and Communications (ICNC)*, 254–258. https://doi.org/10.1109/ICNC59896.2024.1055613

Unal, E., Sen-Baidya, S., & Hewett, R. (2018). Towards Prediction of Security Attacks on Software Defined Networks: A Big Data Analytic Approach. *2018 IEEE International Conference on Big Data (Big Data)*, 4582–4588. https://doi.org/10.1109/bigdata.2018.8622524