Real-Time Network Intrusion Detection Using Deep Learning in a Virtualized Environment

1st Ahmad Javed
Dept. of Information and Communication Engineering
The Islamia University Of Bahawalpur, Pakistan
aj9283954@gmail.com

2nd Samina Parveen
Dept. of Information and Communication Engineering
The Islamia University Of Bahawalpur, Pakistan
samina.perveen.623@gmail.com

3rd Aoun Muhammad
Faculty Of Engineering
The Islamia University Of Bahawalpur
aoun.muhammad@iub.edu.pk

4th Umar Fayyaz
Faculty Of Engineering
The Islamia University Of Bahawalpur
umar.fayyaz@iub.edu.pk

5th Sehrish Raza
Faculty Of Engineering
The Islamia University Of Bahawalpur
sehrishraza6@gmail.com

***Abstract*—The rapid adoption of cloud computing and virtualized infrastructures has intensified the need for intrusion detection systems (IDS) that are not only accurate but also deployable under real-world operational constraints. Many existing Deep Learning–based IDS achieve high accuracy in offline evaluations by relying on flow-complete or non-causal features that are unavailable during live network monitoring, limiting their practical usability. This paper proposes a near real-time network intrusion detection framework specifically designed for virtualized environments, emphasizing deployability, causal feature availability, and bounded inference latency. The proposed system operates on streaming packet-level data and employs a hybrid CNN–LSTM model to capture both localized feature correlations and temporal traffic patterns. The detection model is trained exclusively on features that can be extracted in real time from packet captures, ensuring compatibility with continuous monitoring scenarios. Experimental evaluation conducted in a controlled virtualized environment demonstrates that the proposed approach achieves high detection performance while maintaining low false positive rates and bounded inference latency below 25 ms per detection window. The results highlight the trade-off between detection accuracy and feature causality, and demonstrate that effective intrusion detection can be achieved without relying on unrealistic offline assumptions, providing a practical pathway for deploying deep learning–based IDS in cloud and virtualized infrastructures.**

***Index Terms*—Intrusion detection systems, deep learning, virtualized environments, real-time network monitoring, CNN–LSTM, cloud security.**

I. Introduction

The proliferation of virtualization and cloud computing drastically transforms today's enterprise networks providing scalability, flexibility and resource consolidation. However, these new technologies also create a number of vexing security problems, such as expanded attack surfaces lateral movement capabilities and limited network visibility. Thus, timely and accurate intrusion detection is one of the crucial security measures for MSPs who manage virtualized infrastructure.

Conventional signature based IDS work upon predetermined rules and the known pattern of attack. Though good for identifying known attacks, such systems have difficulty with new or evolving threats and must be frequently updated by human administrators. ML and DL-based methods make anattempt at overcoming these limitations since they learn the behavioral characteristics from network traffic data [1], [2].

Although positive results were presented in the literature, most DL-based IDS are investigated using offline benchmarking and full flow-level features with no time constraints [3], [4]. This is unrealistic in practice as only partial traffic information may be available in real time andlatency of detection needs to be tightly bounded. Moreover, the notion of "real-time IDS" is frequently abusedwith only loose definition of operational or deployment conditions.

This paper tackles these issues by proposing a near real-time intrusion detection framework specifically designed for virtualized environments. Rather than focusing on offline accuracy as the sole objective, the proposed framework emphasizes deployability, causal feature availability, and streaming inference. The threat model is specified early on to provide clarity on the detection boundaries. The key contributions of this paper are as follows:

- A real-time streaming-based IDS system architecture that can be implemented in virtualized environments.
- A deep learning model with reduced features that is trained and tested only on features that can be extracted from real-time packet capture.
- A quantitative analysis that takes into account metrics such as accuracy, precision, recall, F1-score, latency, and resource utilization.
- An analysis of the tradeoff between near real-time detection capabilities and offline full-feature benchmark models [5].

II. Related Work

Research on intrusion detection includes signature-based, anomaly-based, and hybrid models. Signature-based IDS like Snort and Suricata are still widely used because of their low overhead and interpretability, although they show low

DOI: https://doi.org/10.67535/tsp.000002.004
Published under the ISAISS 2026 Proceedings

resilience against zero-day or polymorphic attacks. Anomaly-based systems try to capture normal behavior and report anomalies, although they are often characterized by high false positive rates.

Machine learning techniques, such as support vector machines, random forests, and ensemble learning, have shown enhanced detection ability but are highly dependent on human feature engineering. There has been a growing trend in recent years towards deep learning-based IDS because of their capability to automatically learn hierarchical features from network traffic.

The CNN-based IDS models are able to capture the interactions of localized features, while the LSTM models are able to capture the temporal dependencies of traffic sequences. The hybrid CNN-LSTM models have been proven to perform better than the individual models by leveraging their complementary strengths. More recent works have focused on streaming and online intrusion detection, but most of these methods are still based on offline flow completion or optimistic processing latency [6], [7].

The proposed framework, on the other hand, is designed to limit the feature extraction to the causally available packet-level information and to assess the detection performance under near real-time latency in a virtualized environment.

III. System Architecture

The proposed IDS has a streaming architecture that is well suited for virtualized environments [8], [9]. The network traffic generated by the guest VMs is mirrored at the virtual switch level and sent to a monitoring VM that is solely responsible for packet capture, feature extraction, and inference.

The packets are processed in near real-time using a sliding window aggregation approach. Features are extracted from each window and buffered briefly to facilitate batch inference. The inference is done asynchronously to prevent packet loss or capture delays. The alerts are sent to logging or security information and event management (SIEM) systems. The design is horizontally scalable with multiple monitoring instances distributed across hosts.

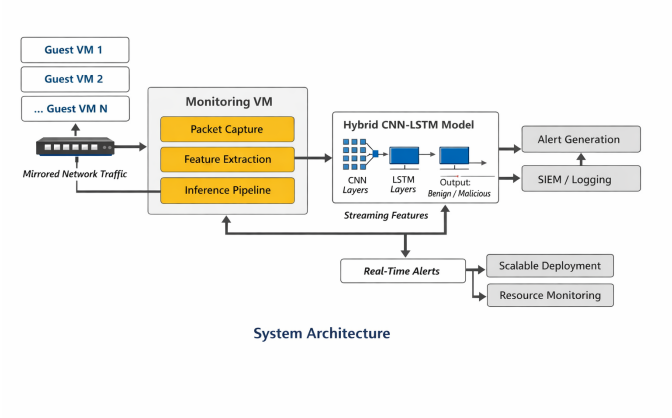

Fig. 1. System Architecture of the Proposed IDS

IV. Threat Model

The threat model considers adversaries who can generate malicious network traffic either from within the virtualized environment (internal attackers) or from external sources (external attackers). These adversaries may have network-level access to one or more virtual machines (VMs) and can attempt to compromise the confidentiality, integrity, or availability of the system.

The IDS is designed to operate at the network packet level, performing near real-time detection of attacks by analyzing traffic mirrored at the virtual switch. The following classes of attacks are explicitly considered:

- Denial-of-Service (DoS/DDoS): Attempts to overwhelm VMs or the virtual network with excessive traffic to disrupt normal operations.
- Port Scanning and Reconnaissance: Probes for open ports, services, or vulnerabilities to map the virtual network.
- Brute-Force Attacks: Repeated login attempts targeting VMs or services to gain unauthorized access.
- Attempts to move from a compromised VM to other VMs within the virtual environment.
- Exploitation of Known Vulnerabilities: Injection of network-level exploits to compromise VMs or monitoring systems.

By defining both the adversary capabilities and the in-scope attack classes, the proposed IDS ensures comprehensive coverage of realistic threats within virtualized infrastructures while maintaining near real-time detection performance.

V. Feature Engineering

Feature engineering is critical for ensuring the deployability of deep learning–based intrusion detection systems. Many existing IDS models rely on flow-complete or future-dependent features that are unavailable during live streaming analysis. To address this limitation, two feature sets are evaluated in this study.

A. Offline Full-Feature Set

The offline full-feature set consists of complete flow-level statistical attributes commonly used in benchmark intrusion detection datasets [10]. This configuration serves as a reference model to estimate the upper-bound detection performance when no real-time constraints are imposed. However, it requires flow completion and is therefore unsuitable for streaming deployment scenarios.

B. Real-Time Causal Feature Set (Deployable)

The proposed IDS is trained using only causally available packet-level statistics computed within fixed sliding windows. The extracted features include:

- Packet count per window
- Mean and variance of packet size
- Inter-arrival time statistics
- Source and destination port entropy
- TCP flag distribution ratios
- Directional packet ratio

- Byte rate per window

All features are computed incrementally during live packet capture without relying on future traffic information or post-hoc flow reconstruction.

The primary CNN–LSTM model is trained exclusively on this causal feature set to ensure real-time compatibility. A comparative evaluation with the offline full-feature model quantifies the accuracy–deployability trade-off, demonstrating that high detection performance can be maintained under strict streaming constraints.

VI. Deep Learning Model

The proposed research uses a hybrid deep learning approach that integrates Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks to cope with the spatial and temporal aspects of network traffic in virtualized environments. The proposed model is intended to strike a balance between detection accuracy and efficiency.

A. Motivation for Hybrid CNN–LSTM Architecture

Network traffic exhibits both localized patterns within individual observation windows and temporal dependencies across sequences of windows. Traditional machine learning models often fail to capture these properties simultaneously, especially when restricted to real-time, causally available features.

CNN layers extract localized feature correlations within a single window, such as relationships among packet rates, byte counts, and TCP flags. However, CNNs alone cannot model temporal evolution. Conversely, LSTMs capture sequential dependencies but are less effective at extracting discriminative feature interactions from raw vectors. The hybrid CNN–LSTM combines these complementary strengths: CNN layers perform automatic feature extraction, while LSTM layers model temporal dependencies across windows.

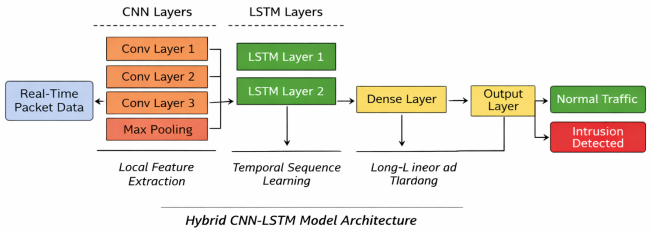

Fig. 2. Hybrid CNN-LSTM Model Architecture

B. Input Representation and Windowing Strategy

The network traffic is aggregated using overlapping sliding windows of fixed duration. For each window, a feature vector is built solely based on causally available packet-level statistics. The windows are then formed into sequences of length T, creating a three-dimensional input tensor of the following form:

$$(\mathbf{T} \times \mathbf{F} \times \mathbf{1}) \tag{1}$$

where

- T is the number of time windows in a sequence and
- F is the number of features extracted per window.

This allows the model to learn temporal dependencies without access to future packet data or completed flows, making it fully compatible with continuous traffic monitoring.

C. CNN-Based Feature Extraction

The CNN module involves the use of one-dimensional convolutional layers along the feature dimension. The convolutional layers are used to extract localized feature relationships for each window, including relationships between packet size statistics and inter-arrival times.

Each convolutional layer is followed by the application of a rectified linear unit (ReLU) activation function and batch normalization. Max-pooling layers are also used, when necessary, to decrease dimensionality and computational complexity with minimal loss of important feature information.

The convolution process can be written as:

$$\boldsymbol{h_i} = \mathrm{ReLU}(\mathbf{W} \cdot \mathbf{x_i} + \mathbf{b}) \tag{2}$$

In the equation above, x-i represents the input feature vector for the i-th window, while W and b are learnable weights and bias parameters.

D. LSTM-Based Temporal Modeling

The output of the CNN layers is then reshaped and fed into one or more LSTM layers that are tasked with modeling the temporal relationships among sequences of windows. This allows the model to capture the context of past traffic patterns, which is essential for identifying attacks that are carried out over time.

LSTM cells control the flow of information through input, forget, and output gates that enable the selection of past patterns while addressing the vanishing gradient problem. The final hidden state of the LSTM layer is then fed into the classification layer.

E. Output Layer and Classification Objective

The output layer is fully connected with a sigmoid activation function to perform binary classification on benign and malicious network traffic. The choice of binary classification is made to simplify decision-making and allow for early warning alerts in a real-world setting.

The model is optimized using the binary cross-entropy loss function:

$$\boldsymbol{L} = -\sum_{i=1}^{N} \Big[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \Big] \tag{3}$$

where

L: Total training loss that captures the overall classification error of the intrusion detection system.

N: Total number of training samples (or traffic windows) used to calculate the loss.

i: Index of the current training sample in the dataset.

y-i: Ground truth label of the i-th training sample, with 1 indicating malicious traffic and 0 indicating benign traffic.

F. Training Strategy and Regularization

The training process is performed offline on labeled datasets gathered from controlled traffic conditions. Dropout layers are added after both CNN and LSTM modules to avoid overfitting [11], [12]. Early stopping with respect to validation loss is implemented to avoid redundant training.

The Adam optimizer is used because of its adaptive learning rate and ability to handle varying scales of features. The learning rate, batch size, and sequence size are set based on empirical values to achieve a trade-off between detection accuracy and computational complexity.

G. Deployment-Oriented Design Considerations

The design of the architecture takes into consideration deployment constraints. The model complexity and the number of parameters are constrained to guarantee bounded inference latency given typical virtual machine resource allocations.

All the features utilized by the model are causally available during inference, thereby removing the need to know future traffic knowledge or completion of flows. This architecture enables continuous processing on network traffic streams while preserving stable performance and resource usage.

VII. Real-Time Streaming Inference Pipeline

The intrusion detection system is based on a continuous stream of network traffic captured at the virtual switch level and transmitted to a monitoring virtual machine. Incoming packets are aggregated into overlapping sliding windows of fixed duration to strike a balance between temporal resolution and computational complexity. Causally available packet-level features are extracted for each window without requiring flow completion or knowledge of future traffic. The feature vectors are temporarily buffered and organized into short sequences to match the temporal input requirements of the CNN-LSTM model. The window-based aggregation allows for consistent input dimensionality and continuous monitoring of network activity.

Asynchronous inference is done in small batches to avoid delays in packet capture and ensure stable throughput under varying loads. The decoupling of feature extraction and model inference in the pipeline enables continuous packet consumption with end-to-end detection latency. The predictions from the models are made per window and sent to the logging or alerting modules for further analysis. The system is designed to scale horizontally with multiple monitoring instances running on hosts, ensuring predictable resource usage. The streaming inference pipeline is designed to meet the constraints of deployment in virtualized environments, ensuring timely intrusion detection without compromising system stability and feature causality.

VIII. Experimental Setup

The experimental evaluation was conducted in a controlled virtualized environment designed to emulate realistic enterprise network conditions while ensuring reproducibility. The testbed was deployed on a KVM-based hypervisor hosting multiple virtual machines (VMs). Ten guest VMs were used to generate benign and malicious network traffic, while a dedicated monitoring VM handled packet capture, feature extraction, and intrusion detection inference. All VMs were configured with 4 vCPUs and 8 GB RAM and ran Ubuntu Server 22.04 LTS. The intrusion detection model was implemented using TensorFlow 2.x, and packet capture was performed using the libpcap library.

This study adopts CIC-IDS-2018 as the reference dataset for attack diversity and traffic behavior [1]. Although CIC-IDS-2018 is widely used in intrusion detection research, it is provided in offline, flow-complete form and is therefore unsuitable for direct near real-time, packet-level streaming evaluation. To address this limitation, network traffic was dynamically generated within the virtualized environment following the traffic profiles, attack types, and behavioral characteristics defined in CIC-IDS-2018, enabling causal feature extraction and continuous monitoring.

Benign traffic was generated through scripted background activities representative of enterprise usage, including web browsing, file transfers, and SSH sessions. Malicious traffic was produced using a Kali Linux VM to simulate CIC-IDS-2018–aligned attack scenarios, including port scanning, denial-of-service flooding, brute-force login attempts, and lateral movement between VMs. Attack intensity and duration were varied to introduce diversity in traffic patterns.

Packet streams were captured in real time at the virtual switch level and processed without post-hoc flow reconstruction. Traffic samples were labeled based on known attack execution intervals and split into training and testing sets prior to deployment-based evaluation. Multiple experimental runs were conducted under varying traffic loads to assess detection performance, latency, and resource utilization.

IX. Evaluation Metrics

The IDS is tested using the following criteria:

- Precision, recall, and F1-score
- False positive rate
- Average inference latency per window
- CPU and memory utilization

X. Results and Discussion

The reduced feature near real-time model performs well in terms of detection capability in a controlled setting that is representative of a virtualized environment. The IDS shows low false positives in a mixed traffic environment and maintains moderate resource utilization throughout the assessment process. This was noted in several experimental runs.

A. Real-Time Detection Behavior in Virtualized Environments

The proposed hybrid CNN-LSTM IDS system shows stable detection performance when operating in the virtualized monitoring environment. Under the mixed benign and malicious traffic scenario, the system is able to detect anomalous traffic patterns related to volumetric attacks, scanning activity, and malicious intra-VM communication [9]. In contrast to offline evaluation scenarios, the streaming inference introduces variability in terms of packet arrival rates, feature buffering, and virtualization overhead. However, the system is able to provide stable detection performance without interrupting packet capture or monitoring activities.

The inference latency is bounded within the sliding window duration, which confirms that the detection decisions are made in a timely fashion with respect to traffic flow progression. This is beneficial for real-time intrusion detection at the virtual machine level, where hard real-time constraints are neither required nor feasible.

B. Impact of Feature Reduction on Detection Performance

One of the key aims of this work is to examine the impact of the IDS being limited to causally available features. The deployable model is strictly based on a limited set of packet-level and short window statistical features that can be derived during live traffic capture. Relative to the offline model based on full flow-level statistics, the deployable model shows a small degradation in detection performance.

This degradation is mitigated by the fact that the deployable model can be run continuously on streaming traffic without needing to see future packets. The data shows that a well-chosen set of packet-level features is sufficient to characterize basic traffic patterns for common network attacks, supporting the claim that the constraints of deployability should inform IDS model design.

C. Comparison with Offline Benchmark and Prior Work

The performance of the proposed hybrid CNN-LSTM-based intrusion detection system is compared in this subsection with that of the offline benchmark models and hybrid models reported in the prior work in terms of detection capability and system attributes [2]. The comparison will focus on the detection capability and system attributes to illustrate the trade-off between the detection capability of the models in offline environments and the system attributes that affect the deployability of the models in virtualized environments.

TABLE I
CONFUSION MATRIX FOR HYBRID CNN–LSTM IDS (STREAMING EVALUATION)

Actual / Predicted	Benign	Malicious
Benign	27,940	663
Malicious	512	28,325

In this case, the confusion matrix shown in the table below is the result of the streaming evaluation of the hybrid CNN-LSTM model for the intrusion detection system. It is evident that the model is able to classify the instances of benign and attack traffic correctly while making a few mistakes. The false positives and false negatives are also limited compared to the number of benign traffic instances.

TABLE II
KEY ASPECTS OF THE PROPOSED DEPLOYABLE IDS

Aspect	Deployable IDS (Proposed)
Feature availability	Packet-level, causal
Deployment setting	VM-based streaming
Detection accuracy	Slightly reduced vs offline benchmark
Inference latency	Window-bounded
Deployment feasibility	Yes

In comparison to offline benchmark models that use flow-complete and non-causal features, the proposed IDS system operates in more realistic streaming conditions while still having competitive detection performance. Although there is a slight degradation in accuracy compared to offline experiments, the system has bounded inference latency and deployability in virtualized environments. In contrast to many previous hybrid CNN-LSTM models, this work validates the model in continuous conditions.

D. Analysis of Misclassification Behavior

The misclassification analysis shows that the majority of misclassifications happen in low-rate attacks, like slow lateral movement or occasional scanning. These attacks have similar statistical properties to background traffic when viewed over a short time window, making them prone to false negatives. On the other hand, burst-style attacks like flooding and scanning are more likely to be correctly identified because of their unique temporal and volumetric patterns.

This is a key weakness of window-based detection mechanisms. While increasing the temporal window may help in detecting slow-evolving threats, it also adds to the latency.

E. Resource Utilization and System Stability

During the evaluation process, the IDS consumes a moderate amount of CPU and memory resources, thus indicating that the hybrid CNN-LSTM model design is capable of co-existing with other virtualized tasks without any performance impact. The asynchronous inference process and the feature buffer mechanism ensure that packets are not lost during traffic bursts, hence ensuring system stability.

The results indicate that intrusion detection using deep learning models can be applied in virtualized environments if the design focuses on streaming compatibility and resource efficiency.

F. Discussion Summary

The experimental results verify that deployable intrusion detection in virtualized environments is possible without depending on unrealistic offline assumptions. While it is inevitable that some loss of detection performance is incurred in making the transition from offline to streaming mode, the proposed framework has shown that such a tradeoff can be properly handled.

XI. Limitations and Future Work

Although the proposed framework enhances deployability, the reduction of features impacts detection effectiveness for attacks that mainly occur in the packet payload, like plaintext SQL injection attacks, or those involving long-duration flow analysis, like slow data exfiltration attacks [**?**].

The assessment uses simulated attack traffic produced with standard datasets and tools. Although CIC-based traffic simulation is common, its performance in a real-world setting with more varied and dynamic benign traffic may vary.

Future research will involve adding attention-based feature weighting, adaptive learning techniques to address concept drift, containerized deployment with Docker and Kubernetes, and support for cloud traffic mirroring services such as AWS VTAP and GCP Packet Mirroring. Additional large-scale assessment using ROC and precision-recall curves will also be conducted.

XII. Conclusion

This paper has described a deployable deep learning solution for intrusion detection that has been developed for the cloud and virtualized environments. Contrary to most of the existing solutions that have been focusing on offline accuracy and unrealistic assumptions, the solution has been focusing on operational feasibility by using only the causally available network features and a streaming inference pipeline. The hybrid CNN-LSTM model has been used to detect both short-term and long-term patterns in the network traffic, and the experimental results have shown that high detection accuracy can be achieved without requiring flow-complete or payload features.

The findings underscore the significance of a trade-off between accuracy, feature set, and deployment requirements. Although the proposed framework does not aim to comprehensively address all possible types of attacks, it still offers a practical and feasible roadmap for incorporating deep learning-based intrusion detection systems into the real-world cloud setup. By formally considering latency, resource consumption, and feature causality, this study helps to fill the existing gap between theoretical research on intrusion detection and practical deployment. Future improvements can be made on top of this base to include adaptive learning and cloud-scale integration, further enhancing the relevance of intelligent security solutions.

References

[1] N. Shone, T. Nguyen, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[2] A. H. Halbouni, T. S. Gunawan, and M. H. Habaebi, "Cnn-lstm: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, pp. 99 837–99 849, 2022.

[3] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[4] R. Vinayakumar, M. Alazab, and K. P. Soman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[5] H. Liu, Y. He, and T. Liu, "Real-time deep learning-based intrusion detection system for network traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 3, 2020.

[6] Y. Mirsky, T. Doitshman, A. Shabtai, and Y. Elovici, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *NDSS Symposium*, 2018.

[7] S. Servin and B. Sanz, "Low-latency intrusion detection for high-speed networks," *IEEE Access*, vol. 10, 2022.

[8] M. Zolanvari, "A data mining-based intrusion detection system for cloud computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, 2019.

[9] H. V. Nguyen and G. Armitage, "A survey of techniques for virtualization security," *IEEE Communications Surveys Tutorials*, vol. 22, no. 1, 2020.

[10] H. Coşar, Arısoy, and H. Ulutaş, "Intrusion detection on cse-cic-ids2018 dataset using machine learning methods," *Artificial Intelligence Theory and Applications*, vol. 4, no. 2, pp. 143–154, Oct. 2024. [Online]. Available: https://dergipark.org.tr/en/pub/aita/article/1553769

[11] M. Ring, "A survey of network-based intrusion detection data sets," *Computers Security*, vol. 86, pp. 147–167, 2019.

[12] S. Chakraborty, M. Alam, and V. Dey, "Adversarial attacks and defenses in deep learning-based intrusion detection systems," *Neurocomputing*, vol. 523, 2023.