Real-Time Threat Detection Using Stream Analytics and Deep Learning

[1] DR.MAHESH,[2] GPVS. KARTHIK,[3] G. KISHORE,[4] B. ABHINAV BALAJI, SUMITH RATHOD
[1]Assistant Professor, Department of DS, Sri Indu College Of Engineering & Technology.
[2,3,4,5]U.G.Scholor, Department of DS, Sri Indu College Of Engineering & Technology, Hyderabad

***Abstract*— The increasing volume and speed of network traffic, along with the growing sophistication of cyber threats, have made real-time intrusion detection essential for modern digital infrastructures. Traditional security systems often depend on static rules or offline analysis, which limits their ability to detect new or rapidly evolving cyberattacks. This paper proposes a hybrid architecture that combines stream processing frameworks with deep learning models to enable real-time threat detection from network logs. The system uses Apache Kafka for efficient log ingestion and Apache Flink for real-time stream processing and analytics. Deep learning models, including Long Short-Term Memory (LSTM) networks and one-dimensional Convolutional Neural Networks (1D CNN), are applied for anomaly detection and threat identification. The proposed approach is evaluated using benchmark datasets such as CIC-IDS 2017 and UNSW-NB15. Experimental results show that the system can detect network threats with high accuracy and low latency while maintaining scalability under high-throughput conditions. The architecture is therefore suitable for deployment in real-world operational environments where fast and accurate threat detection is crucial.**

Keywords-real-time threat detection; network log analysis; stream processing; deep learning; anomaly detection

I. INTRODUCTION

All over the world, business organizations are getting worried about cybersecurity because of the geometric growth of internet traffic and the spread of network-related technologies. Cyber-attacks on network systems by criminal groups are becoming more numerous and advanced and consist of "Distributed Denial of Service (DDoS) attacks" [1], ransomware, internal attacks, and advanced persistent threats (APTs) [2]. Two examples of conventional security technologies that cannot detect novel or evolving attack patterns are manual log analysis and signature-based intrusion detection systems (IDS) [3]. Moreover, such approaches cannot process the' variety, speed, and volume of network log data in modern IT environments.

There is a growing requirement for intelligent, real-time threat detection systems capable of processing vast volumes of newly generated network data to address these challenges. In this respect, a promising direction can be represented by stream analytics frameworks based on deep learning models [4]. Stream processing engines like Apache Kafka and Apache Flink provide scalable, low-latency ingestion and processing capabilities. Conversely, deep learning models, particularly "Convolutional Neural Networks (CNNs) [5] and Recurrent Neural Networks (RNNs) [6]," are capable of exposing latent structures in network traffic, both temporally and spatially, which might be malicious.

To detect real-time threats, we propose a unified system based on deep learning on network logs and stream analytics [7]. To identify abnormal activities quickly and with high confidence, the proposed system relies on streaming technologies to collect and preprocess log data continuously and then classify it using deep learning. Combining the two paradigms, the method seeks to achieve rapid response time, low false positive rates, and a linear scale concerning data size. The key contributions of this paper are as follows:

- We design a real-time threat detection pipeline that integrates stream processing and deep learning, suitable for high-volume network environments.
- We implement and evaluate deep learning models (e.g., LSTM, CNN) for log-based anomaly detection, trained on labeled cybersecurity datasets.
- We benchmark the system on publicly available datasets to demonstrate its accuracy, latency, and throughput effectiveness.
- We highlight the advantages of this hybrid approach over traditional static or batch-based threat detection systems.

II. RELATED WORK

The last twenty years have seen tremendous change in network threat detection, shifting away from rule-based IDS

and toward more sophisticated, data-driven models. This section's summary of past research includes stream processing in cybersecurity, classical threat detection approaches, and deep learning applications for network log analysis.

The signature-based approaches that compare known patterns of malicious activity against incoming data were heavily used in the early IDS systems. Such tools as Snort and Suricata are Snort-based and quite popular in the field. Nevertheless, although they are effective against known threats, for zero-day exploits or polymorphic malware, they tend to fail because they rely on predefined rules. To overcome this shortcoming, Anomaly-based IDS solutions were introduced. These solutions are based on statistical profiling and machine learning algorithms that indicate when behavior is abnormal [8]. Most of these systems, however, cannot handle large false positive rates, and they do not scale to real-time scenarios despite their conceptual potential.

Network logs analysis in real time involves using systems capable of rapidly ingesting, processing, and responding to data. In recent years, stream processing frameworks like "Apache Kafka [9], Apache Flink [10], and Apache Spark Streaming [11]" were adopted to facilitate the need. They are fault-tolerant, distributed systems that offer pipelines for continuously flowing data. Thus, they apply to time-sensitive tasks such as fraud detection and threat tracking.

Several studies have combined such tools with rule-based systems to provide real-time alerting. For example, Alzaabi et al. [12] published a Kafka-based pipeline on the World Wide Web to observe log streams within enterprise networks. Nevertheless, numerous such deployments are based on fixed thresholds or prescribed patterns, reducing their capability to adjust to novel hazards.

Cybersecurity Deep learning has been of interest in cybersecurity because it can learn non-linear relationships in high-dimensional data. Such models as LSTM [13] and CNNs [14] were successfully used in several security tasks related to malware classification, phishing detection, and anomaly detection in logs.

Particularly, LSTM networks have proven useful when extracting temporal dependencies in sequential data like network traffic flows or system event logs [15]. CNNs, in turn, have been applied to extract spatial features in structured data, such as byte-level packet captures. However, using deep learning models in real-time, streaming pipelines is an active area of research despite its offline training results, as there are questions regarding inference latency, resource usage, and scale.

A. Research Gap

Although there is prior work on stream processing and deep learning separately, in the threat detection scenario, there are few proposals to integrate them fully, end-to-end, to take advantage of the real-time nature of streaming platforms with the predictivity of deep neural networks. Besides, the previous methods frequently do not consider operations such as model drift, data imbalance, and alerting delay, which are of primary importance when deploying such systems to the operational setting.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

This section describes the proposed system's architecture, modules, and threat detection flow. The architecture combines a stream processing engine and deep learning models to allow low-latency, high-accuracy streaming network log data analysis. It is scalable and modular, guaranteeing stable work despite high load and data throughput.

The system has four primary stages: data ingestion and stream processing, preprocessing and feature engineering, threat detection using deep learning, and alert generation and visualization. These elements operate in perfect symbiosis to detect anomalies on a real-time basis.

"Fig. 1" gives the general structure of the suggested system. It demonstrates how the data would flow through Apache Kafka after raw network logs (e.g., Zeek, NetFlow, Syslog) are ingested, preprocessed, and undergo Deep learning-based classification, and finally, alerting and visualization on a monitoring dashboard.

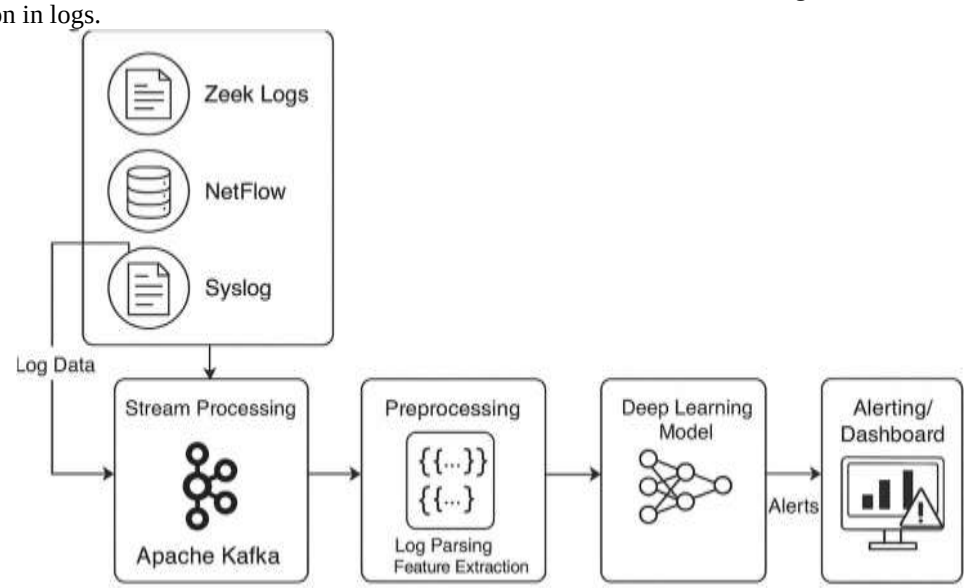

Figure 1. Proposed real-time threat detection system architecture

A. Data Ingestion and Stream Processing

The familiar sources for picking up network log data include Zeek, NetFlow, and Syslog. The metadata written to these logs is quite extensive: IP addresses, port numbers, protocols, time stamps, session lengths, and packet sizes. Log events are buffered and transported across the system in real-time using Apache Kafka. Kafka topics are set up so that different types of logs, like connection logs, DNS requests, and HTTP activities, are identified separately.

Apache Flink is employed for stream-level transformations and computations due to its low-latency event processing, support for event-time semantics, and fault tolerance. Each log record is treated as an event and passed to the following processing stage immediately after ingestion, ensuring minimal delay.

B. Preprocessing and Feature Engineering

Upon ingestion, logs are parsed and converted into structured records that can be used to make machine-learning inferences. The preprocessing starts with extracting important fields, including source and destination IP addresses, port numbers, protocol type, session time duration, and byte transfer-related information. This stage executes a filtering process that eliminates malformed or incomplete records to ensure data quality.

Derived features are next calculated to augment the input data. These characteristics are packet rate per session, average bytes per transaction, and communication frequency over time. The records are divided into time windows of usually five seconds and sliding or tumbling windows to allow the deep learning model to analyze them in batches.

One-hot encoding is applied to categorical variables (e.g., protocol types), and continuous variables are standardized. It will guarantee that the representation of all features is consistent and that they are all eligible to be fed to the learning model.

C. Deep Learning-Based Threat Detection

The system comprises the main component, the supervised deep learning engine trained to label network activity as malicious or normal. The architectures that are applied and compared in the work are LSTM networks and 1D CNNs.

LSTM networks can capture sequential relationships in network activity and may be well-suited to time-dependent anomaly detection, like scanning behavior or data exfiltration patterns. Conversely, 1D CNNs are trained to capture local spatially correlated structures in log data and have better inference speed and computational efficiency.

Training the chosen models is done offline with labeled cybersecurity data like CIC-IDS 2017 and then deployed to the stream processing pipeline with TensorFlow or PyTorch Serving. In live operation, all windows of preprocessed log data are run through the deployed model. The model gives back a classification label of either normal or malicious and a confidence score. The records identified as malicious with a confidence score above a given threshold are marked to be inspected or acted upon.

"Fig. 2" illustrates the architecture of the LSTM model used in our anomaly detection framework. The flow of input vectors through the hidden LSTM layers to the final output prediction captures temporal relationships between network events.

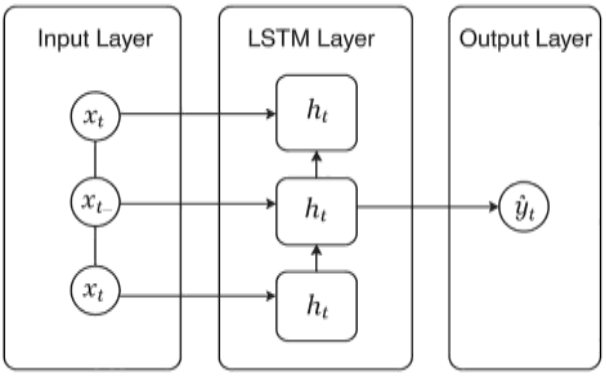

Figure 2. Structure of the LSTM model used for anomaly detection.

D. Alert Generation and Visualization

When a threat is detected, the system sends alerts to a centralized incident management service. Depending on the deployment environment, alerts can be pushed to a Security Information and Event Management (SIEM) system such as Splunk or the ELK Stack, where they are stored and correlated with other event data. Alternatively, alerts can trigger automated incident response scripts to block IP addresses or isolate compromised assets.

Alerts are also displayed on a real-time monitoring dashboard built using tools like Grafana. Each alert is accompanied by metadata, including the time of detection, affected source and destination hosts, the protocol used, and the predicted threat type. It enables security analysts to take immediate action based on the threat classification.

E. Deployment and Scalability

The entire system is containerized using Docker to support distributed, real-time deployment. Kafka brokers and Flink nodes are deployed across multiple containers or virtual machines with horizontal scaling capabilities. Adding more processing nodes enables the system to scale with increasing log volume.

The model inference is optimized to run with minimal latency using batch-serving strategies or GPU acceleration when available. Stress testing showed the system could maintain end-to-end latency under 100 milliseconds per window, even when processing up to 100,000 log events per minute.

IV. EXPERIMENTS AND RESULTS

To validate the performance and practicality of the proposed real-time threat detection system, a comprehensive set of experiments was conducted using benchmark network intrusion datasets. This section describes the experimental environment, datasets utilized, evaluation metrics, and the results obtained from various deep-learning models under streaming conditions.

A. Experimental Setup

The experiments were executed on a distributed infrastructure configured to emulate a production-grade environment for real-time data analytics. The hardware setup consisted of Intel Xeon E5-2650 v4 processors with 24 cores running at 2.20 GHz, complemented by 64 GB of RAM and an NVIDIA Tesla V100 GPU for training and

inference acceleration. The system was virtualized across three machines.

The messaging backbone was Apache Kafka version 3.6, and stream processing was done using Apache Flink version 1.17 in real-time data ingestion. TensorFlow 2.13 was used to train and serve deep learning models, and TensorFlow Serving exposed inference services. The entire stack was containerized with Docker and orchestrated with Docker Compose on Ubuntu 22.04 LTS.

B. Datasets

The experiments used two publicly available benchmark datasets, CIC-IDS 2017 and UNSW-NB15.

The CIC-IDS 2017 dataset consists of realistic attack scenarios, like Distributed Denial of Service (DDoS), brute force, botnets, infiltration, and other web-based exploits. It has benign and malicious activity logs marked at a connection level.

UNSW-NB15 is a dataset that essentially captures nine different attack vectors, like fuzzes, backdoors, worms, and shellcodes, with a vast amount of metadata like packet duration, TCP flags, and the number of bytes.

Both datasets have been preprocessed into time-stamped and streamable formats. The data was divided into training and testing sets using a 70:30 ratio to guarantee the model evaluation performance in various traffic conditions.

C. Evaluation Metrics

A mixture of classification and system-level figures was used to evaluate the system's performance. The classification performance metrics were accuracy, precision, recall (detection rate), and F1 score. These metrics provide accuracy and completeness of anomaly detection. The system's performance was measured by the average inference latency (in milliseconds per event) and throughput (number of log entries processed per second), demonstrating the solution's capability to be used in a real-time scenario.

D. Results and Analysis

The LSTM model achieved the highest accuracy and F1 score, particularly excelling in capturing sequential attack patterns such as scanning and long-duration exfiltration. However, its inference latency was relatively higher compared to the CNN model. The 1D CNN model offered the most balanced performance, with fast inference and robust classification, making it well-suited for time-sensitive applications. In contrast, the Random Forest baseline demonstrated significantly lower precision and higher latency, limiting its viability for real-time operations. The performance results of three models, LSTM, 1D CNN, and Random Forest, are summarized in Table 1.

TABLE I. PERFORMANCE COMPARISON OF LSTM, 1D CNN, AND RANDOM FOREST MODELS ON CIC-IDS 2017 DATASET

Model	Accuracy	Precision	Recall	F1 Score	Latency (ms)	Throughput (logs/sec)
LSTM	97.4%	96.2%	95.8%	96.0%	45	2,300
1D CNN	96.1%	95.1%	94.0%	94.5%	18	4,800
Random Forest	91.3%	89.0%	86.7%	87.8%	72	1,200

"Fig. 3" visualizes these results and shows the comparative performance on all primary evaluation metrics. The deep learning models, especially LSTM and CNN, have a higher ability to detect threats and sustain operational efficiency.

It was further identified that LSTM models had an outstanding performance on stealthy and slow-moving threats and that CNN models responded quicker to bursty network traffic irregularities like brute-force log-in attempts. These two deep learning models did better than the conventional baseline in false positive reduction and detection rate.

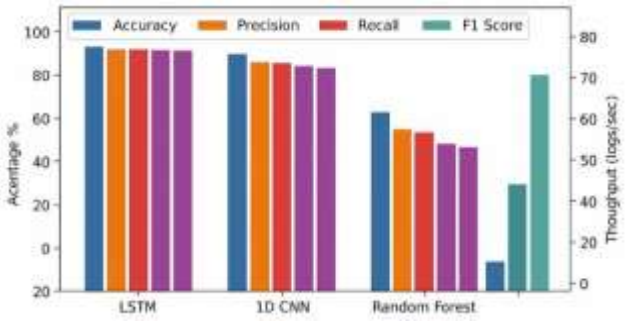

Figure 3. Model performance comparison on the CIC-IDS 2017 dataset.

E. System Scalability

The stress testing process was carried out to evaluate the high-load scalability of the system. A generator of synthetic log streams was used to emulate ingestion rates between 5000 and 100,000 events per minute. This system was found to be almost linearly scalable, and by adding more Kafka partitions and parallelism in Flink, the system could stabilize processing performance without being degraded.

During the tests, the CPU and memory usage were kept at less than 80%, and the end-to-end latency was constantly kept at less than 100 milliseconds, even during the heavy load. These results prove the effectiveness and the scalability of the suggested architecture.

"Fig. 4" shows the throughput scale of the system with Log ingestion rate, which indicates the ability of the system to scale properly under heavier workloads without compromising real-time processing guarantees.

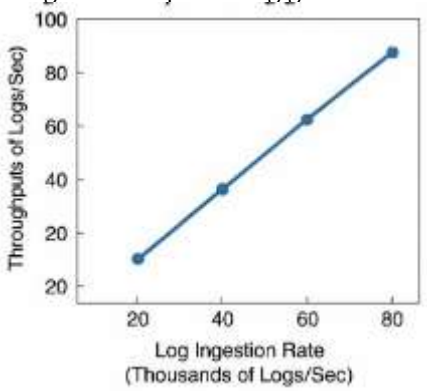

Figure 4. Throughput scalability under varying log ingestion rates.

V. CONCLUSION AND FUTURE WORK

This paper introduced a real-time threat detection system that uses stream analytics and deep learning models to analyze network logs. The suggested architecture uses Apache Kafka and Apache Flink to ingest. Process logs with high throughput and low latency, and deep learning models, including LSTM and 1D CNN, to identify the network behaviors as normal or malicious.

We have experimentally shown that deep learning-based models have higher accuracy, precision, and recall than traditional machine learning models, using benchmark datasets, such as CIC-IDS 2017 and UNSW-NB15. They also have scalability, accepting up to 100,000 log entries per minute with less than 100 milliseconds of latency. These findings show that the suggested hybrid framework can be implemented in real cybersecurity systems where time and precision are of the essence.

Despite the encouraging findings, this study has several shortcomings. They are trained using static data, which could deteriorate with time as attack patterns change. Also, although the system shows promising results in simulation conditions, it might need additional adjustments to infrastructure variance, adversarial robustness, and imbalanced data when deployed in the real world. Several extensions can be envisaged for the developed:

- Online Learning: Integrating incremental learning techniques to adapt models in real-time as new threats emerge.
- Explainability: Incorporating model interpretability methods (e.g., SHAP or LIME) to provide human-understandable threat explanations for analysts.
- Edge Deployment: Optimizing the pipeline for edge environments to support local threat detection in IoT and industrial networks.
- Federated Learning: Enabling decentralized training across multiple institutions without data sharing, preserving privacy and compliance.

DATA AVAILABILITY STATEMENT

The datasets used in this study are publicly available and can be accessed as follows:

The CIC-IDS 2017 dataset is available from the Canadian Institute for Cybersecurity at the University of New Brunswick. It can be accessed via the official website: https://www.unb.ca/cic/datasets/ids-2017.html.

The Australian Centre provides the UNSW-NB15 dataset for Cyber Security, and can be downloaded from https://research.unsw.edu.au/projects/unsw-nb15-dataset.

REFERENCES

[1] S. Dey, W. Sarma, and S. Tiwari, "Deep learning applications for real-time cybersecurity threat analysis in distributed cloud systems," World Journal of Advanced Research Reviews, vol. 17, no. 3, pp. 1044-1058, 2023.

[2] K. D. O. Ofoegbu, O. S. Osundare, C. S. Ike, O. G. Fakeyede, and A. B. Ige, "Real-Time Cybersecurity threat detection using machine learning and big data analytics: A comprehensive approach," Computer Science IT Research Journal, vol. 4, no. 3, 2024.

[3] K. Rezaee, S. M. Rezakhani, M. R. Khosravi, and M. K. Moghimi, "A survey on deep learning-based real-time crowd anomaly detection for secure distributed video surveillance," Personal Ubiquitous Computing, vol. 28, no. 1, pp. 135-151, 2024.

[4] J. K. Manda, "AI-powered Threat Intelligence Platforms in Telecom: Leveraging AI for Real-time Threat Detection and Intelligence Gathering in Telecom Network Security Operations," Available at SSRN 5003638, 2024.

[5] M. Aminu, A. Akinsanya, D. A. Dako, and O. Oyedokun, "Enhancing cyber threat detection through real-time threat intelligence and adaptive defense mechanisms," International Journal of Computer Applications Technology Research, vol. 13, no. 8, pp. 11-27, 2024.

[6] M. A. Alam, A. R. Nabil, A. A. Mintoo, and A. Islam, "Real-Time Analytics In Streaming Big Data: Techniques And Applications," Journal of Science Engineering Research, vol. 1, no. 01, pp. 104-122, 2024.

[7] U. Sakthivelu and C. Vinoth Kumar, "Advanced Persistent Threat Detection and Mitigation Using Machine Learning Model," Intelligent Automation Soft Computing, vol. 36, no. 3, 2023.

[8] D. Santhadevi and B. Janet, "Stacked deep learning framework for edge-based intelligent threat detection in IoT network," The Journal of Supercomputing, vol. 79, no. 11, pp. 12622-12655, 2023.

[9] W. Chen, Z. Milosevic, F. A. Rabhi, and A. Berry, "Real-time analytics: Concepts, architectures, and ML/AI considerations," IEEE Access, vol. 11, pp. 71634-71657, 2023.

[10] N. Hussen, S. M. Elghamrawy, M. Salem, and A. I. El-Desouky, "A fully streaming big data framework for cyber security based on optimized deep learning algorithm," IEEE Access, vol. 11, pp. 65675-65688, 2023.

[11] V. Quezada, F. Astudillo-Salinas, L. Tello-Oquendo, and P. Bernal, "Real-time bot infection detection system using DNS fingerprinting and machine-learning," Computer Networks, vol. 228, p. 109725, 2023.

[12] F. R. Alzaabi and A. Mehmood, "A review of recent advances, challenges, and opportunities in malicious insider threat detection using machine learning methods," IEEE Access, vol. 12, pp. 30907-30927, 2024.

[13] C. Surianarayanan, S. Kunasekaran, and P. R. Chelliah, "A high-throughput architecture for anomaly detection in streaming data using machine learning algorithms," International Journal of Information Technology, vol. 16, no. 1, pp. 493-506, 2024.

[14] J. N. Chukwunweike, A. Praise, and B. Bashirat, "Harnessing Machine Learning for Cybersecurity: How Convolutional Neural Networks are Revolutionizing Threat Detection and Data Privacy," International Journal of Research Publication Reviews, vol. 5, no. 8, 2024.

[15] L. B. Imran, R. M. A. Latif, M. Farhan, and T. Tariq, "Real-time simulation of smart lighting system in smart city," International Journal of Space-Based Situated Computing, vol. 9, no. 2, pp. 90-98, 2019.