

AnomalyX: An Adaptive Threat Detection System using Time-Series Data Analysis

SHWETA BABU P.¹, AKSHAT GUPTA², AMOL VYAS³

¹Dept. Computer Science and Engineering (Professor, e-mail: shwetababup@rvce.edu.in)

²Dept. Computer Science and Engineering (USN: 1RV23CS027, e-mail: akshatgupta.cs23@rvce.edu.in)

³Dept. Computer Science and Engineering (USN: 1RV23CS032, e-mail: amolvyas.cs23@rvce.edu.in)

This work was supported by RV College of Engineering, Bangalore, India

ABSTRACT In modern network environments, the rapid growth of connected devices and high-volume traffic has significantly increased the complexity of detecting anomalous and malicious activities. Traditional rule-based and signature-driven intrusion detection systems struggle to identify zero-day attacks, evolving threat patterns, and subtle behavioral deviations in dynamic traffic streams. This paper presents *AnomalyX*, an adaptive network anomaly detection framework that leverages advanced time-series forecasting and confidence-driven statistical analysis to enable real-time detection of abnormal network behavior.

The proposed system integrates a scalable microservice-based architecture using FastAPI and an asynchronous data pipeline to ingest network flow metrics from distributed services. Time-series forecasting is performed using the TimeGPT model provided by Nixtla, enabling accurate prediction of expected traffic behavior without extensive model training. Deviations between observed values and forecast confidence intervals are analyzed to compute anomaly scores dynamically, allowing the system to detect both sudden spikes and gradual behavioral shifts. The framework utilizes the UNSW-NB15 dataset for feature modeling and evaluation, ensuring coverage of diverse real-world attack scenarios and traffic patterns.

AnomalyX emphasizes modularity, low-latency inference, and deployment simplicity by avoiding heavy-weight deep learning training pipelines and instead leveraging pre-trained forecasting intelligence combined with lightweight statistical decision logic. Experimental results demonstrate reliable anomaly identification with minimal computational overhead, making the system suitable for real-time monitoring environments. The proposed approach highlights the effectiveness of combining modern time-series intelligence with practical system engineering to deliver a scalable, adaptive, and production-ready anomaly detection platform.

INDEX TERMS Anomaly detection, cybersecurity, time-series analysis, TimeGPT, network traffic analytics, UNSW-NB15, real-time monitoring, machine learning

I. INTRODUCTION

THE exponential growth of cloud computing, Internet-of-Things (IoT) devices, and high-speed enterprise networks has resulted in massive volumes of heterogeneous network traffic. This growth has significantly increased the attack surface for cyber threats, making traditional security mechanisms such as signature-based intrusion detection systems, rule-driven firewalls, and static threshold monitoring insufficient for modern environments. These conventional approaches are effective only for previously observed attack signatures and fail to generalize to zero-day exploits, polymorphic malware, and slow-moving stealth attacks that evolve over time.

Machine learning has emerged as a promising solution for adaptive anomaly detection by learning behavioral patterns

directly from data rather than relying on handcrafted rules. However, many existing machine learning solutions require large labeled datasets, extensive model training, and complex hyperparameter tuning, making them expensive to deploy and difficult to maintain in production systems. Additionally, deep learning pipelines often introduce high latency and operational complexity, which limits their applicability in real-time monitoring scenarios where rapid response is critical.

Time-series modeling provides a natural framework for network anomaly detection since network metrics such as packet rates, flow counts, latency, and bandwidth utilization exhibit strong temporal dependencies. Forecasting-based anomaly detection predicts the expected behavior of traffic streams and flags deviations as potential anomalies. Recent advances in transformer-based forecasting models have sig-

nificantly improved prediction accuracy and generalization, enabling zero-shot or minimal-training deployment for new environments.

In this work, we propose *AnomalyX*, a scalable and adaptive network anomaly detection system that combines modern time-series forecasting with lightweight statistical decision logic and a production-oriented microservice architecture. The system utilizes TimeGPT, a transformer-based time-series forecasting model, to generate probabilistic forecasts of network behavior. Observed traffic values are continuously evaluated against forecast confidence intervals to dynamically compute anomaly scores without requiring supervised labeling or model retraining. This approach enables rapid deployment while maintaining strong detection capability across diverse traffic patterns.

AnomalyX is implemented using a distributed architecture where a FastAPI-based inference service asynchronously collects historical metrics from upstream services, performs forecasting, and produces anomaly assessments in near real time. The framework leverages the UNSW-NB15 dataset for experimental validation, ensuring realistic representation of contemporary attack categories and network behaviors. Emphasis is placed on modularity, low operational overhead, and system robustness, making the platform suitable for academic experimentation as well as real-world deployment.

The primary contributions of this paper are summarized as follows:

- Design of a lightweight anomaly detection pipeline based on transformer-driven time-series forecasting rather than heavy supervised model training.
- Integration of an asynchronous microservice architecture enabling scalable real-time inference and efficient data ingestion.
- A confidence-driven anomaly scoring mechanism that adapts dynamically to evolving traffic behavior.
- Experimental validation using the UNSW-NB15 benchmark dataset demonstrating reliable detection performance with low computational latency.

II. RELATED WORK

Network anomaly detection has been extensively studied across multiple research domains, including intrusion detection systems, time-series analysis, and machine learning-driven cybersecurity. Existing work can be broadly categorized into three major directions: benchmark dataset development, deep learning-based intrusion detection models, and forecasting-based anomaly detection approaches.

A. NETWORK INTRUSION DETECTION DATASETS

The availability of realistic and representative datasets is critical for evaluating network anomaly detection systems. Early benchmark datasets such as KDD99 and NSL-KDD provided foundational resources for intrusion detection research but suffer from outdated traffic patterns and limited attack diversity. To address these limitations, Moustafa and Slay introduced the UNSW-NB15 dataset, which incorporates modern

synthetic attack behaviors alongside real normal traffic, providing a comprehensive feature set for contemporary evaluation. The dataset includes multiple attack categories such as exploits, fuzzers, denial-of-service, reconnaissance, and backdoors, making it suitable for validating generalization performance across heterogeneous traffic scenarios. Due to its balanced structure and feature richness, UNSW-NB15 has become a widely adopted benchmark for modern intrusion detection research and serves as the primary dataset used in this work.

B. MACHINE LEARNING AND DEEP LEARNING FOR ANOMALY DETECTION

A large body of research explores supervised and unsupervised machine learning techniques for detecting network anomalies. Traditional machine learning models such as support vector machines, decision trees, and ensemble methods rely heavily on feature engineering and labeled data. While effective in controlled environments, these approaches struggle with scalability and adaptability in dynamic networks.

Deep learning techniques, including convolutional neural networks, recurrent neural networks (RNNs), and long short-term memory (LSTM) architectures, have demonstrated improved performance in capturing temporal dependencies in network traffic. Several studies have reported high detection accuracy using LSTM-based sequence modeling for intrusion detection. However, deep learning pipelines often require extensive training data, computational resources, and continuous retraining to maintain accuracy under evolving traffic patterns. This operational overhead limits their practical deployment in resource-constrained or rapidly changing environments.

Recent research has also explored hybrid frameworks combining deep learning with optimization techniques and ensemble strategies to improve robustness. While these systems achieve high accuracy, they introduce architectural complexity, increased inference latency, and challenging maintenance requirements.

C. TIME-SERIES FORECASTING AND TRANSFORMER-BASED MODELS

Forecasting-based anomaly detection approaches model normal behavior as a time series and identify anomalies as deviations from predicted values. Classical statistical models such as ARIMA and Prophet have been widely used for this purpose but struggle with non-stationary patterns and complex temporal dependencies. Transformer-based architectures have recently demonstrated superior forecasting performance by leveraging self-attention mechanisms to model long-range temporal relationships.

TimeGPT, a transformer-based large-scale forecasting model, enables zero-shot forecasting across multiple domains without requiring task-specific training. Its ability to generalize across datasets and handle multivariate temporal signals makes it particularly suitable for real-time anomaly detection in network environments. Forecast-driven anomaly detection

reduces the dependency on labeled datasets and simplifies system deployment compared to fully supervised learning pipelines.

D. SYSTEM-ORIENTED AND REAL-TIME DETECTION FRAMEWORKS

Several studies emphasize real-time anomaly detection frameworks integrating streaming data pipelines, scalable storage engines, and automated alerting mechanisms. Systems leveraging time-series databases and asynchronous processing pipelines enable low-latency detection but often rely on static thresholding or complex deep learning inference stacks. Recent research has also explored retrieval-augmented generation and large language models for explainability and automated reasoning; however, such systems typically increase system complexity and computational cost.

In contrast, the proposed AnomalyX framework focuses on achieving practical real-time detection through a lightweight forecasting-driven architecture combined with statistical confidence modeling and asynchronous microservices. By avoiding heavy training pipelines and emphasizing operational simplicity, AnomalyX balances detection accuracy, scalability, and deployment efficiency, addressing limitations observed in prior approaches.

III. METHODOLOGY

This section describes the overall design methodology of the proposed AnomalyX framework, including system architecture, data modeling, forecasting pipeline, anomaly scoring mechanism, and database schema design. The framework is engineered to support real-time anomaly detection with minimal computational overhead while maintaining scalability and robustness in distributed environments.

A. OVERALL SYSTEM ARCHITECTURE

AnomalyX follows a microservice-oriented architecture composed of three primary layers: data ingestion, forecasting and anomaly analysis, and persistence. Network flow metrics are continuously generated by upstream services and exposed through a RESTful interface. A FastAPI-based inference service asynchronously retrieves historical and real-time metrics using non-blocking HTTP clients. This design ensures high throughput and low latency under concurrent workloads.

The forecasting engine utilizes the TimeGPT model provided by Nixtla to predict future values of network metrics based on historical time-series windows. The predicted distribution is used to establish confidence bounds for expected behavior. Observed values are evaluated against these bounds to compute anomaly scores dynamically. Detected anomalies and metadata are persisted through a structured database layer implemented using Prisma ORM.

The modular design allows each component to evolve independently while preserving system stability and scalability.

B. DATA FLOW AND PROCESSING PIPELINE

The operational pipeline consists of the following sequential stages:

- 1) **Metric Collection:** Network statistics such as packet counts, byte volumes, protocol distributions, and flow durations are retrieved from upstream services at fixed sampling intervals.
- 2) **Window Construction:** Historical observations are organized into fixed-length sliding windows to preserve temporal continuity.
- 3) **Forecast Generation:** TimeGPT generates probabilistic forecasts for the next time step or horizon.
- 4) **Deviation Analysis:** Actual observations are compared with forecast confidence intervals to compute deviation scores.
- 5) **Anomaly Decision:** Deviations exceeding a dynamic threshold are labeled as anomalies.
- 6) **Persistence:** Results are stored for auditing, visualization, and downstream analytics.

C. TIME-SERIES FORECASTING MODEL

Let $X = \{x_1, x_2, \dots, x_t\}$ represent a univariate or multivariate time series of observed network metrics. The forecasting model learns a mapping function:

$$\hat{x}_{t+1} = f(X_{t-k:t}) \quad (1)$$

where k denotes the historical window size and $f(\cdot)$ is the transformer-based TimeGPT forecasting function.

TimeGPT produces both a point forecast \hat{x}_{t+1} and a confidence interval $[\ell_{t+1}, u_{t+1}]$ corresponding to a predefined confidence level. This probabilistic output enables robust uncertainty-aware anomaly detection without supervised labeling.

D. CONFIDENCE-BASED ANOMALY SCORING

An anomaly score A_t is computed by measuring the deviation between the observed value x_{t+1} and the forecast interval:

$$A_t = \begin{cases} 0, & \ell_{t+1} \leq x_{t+1} \leq u_{t+1} \\ |x_{t+1} - \hat{x}_{t+1}|, & \text{otherwise} \end{cases} \quad (2)$$

A dynamic threshold τ is maintained using rolling statistics of recent deviation values:

$$\tau = \mu_A + \lambda \sigma_A \quad (3)$$

where μ_A and σ_A denote the rolling mean and standard deviation of anomaly scores, and λ controls sensitivity.

An observation is classified as anomalous if:

$$A_t > \tau \quad (4)$$

This adaptive thresholding mechanism enables the system to automatically adjust to evolving traffic behavior while minimizing false positives.

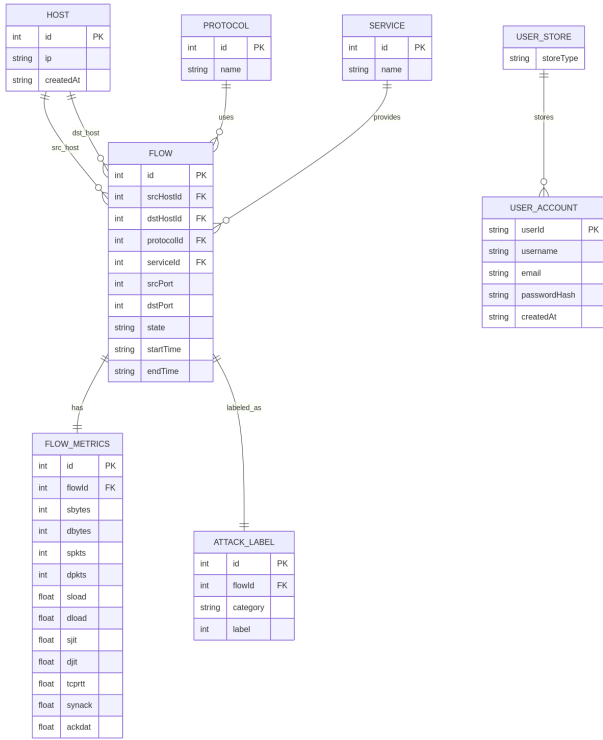


FIGURE 1. Entity Relationship Diagram of the AnomalyX database schema

E. DATABASE SCHEMA AND ER MODEL

Persistent storage is used to maintain historical metrics, forecast outputs, detected anomalies, and audit logs. The database schema is designed to support efficient querying, traceability, and extensibility. The primary entities include:

- **Metric:** Stores raw network measurements with timestamps and feature attributes.
- **Forecast:** Stores predicted values, confidence bounds, and associated model metadata.
- **Anomaly:** Stores anomaly scores, severity levels, and detection timestamps.
- **Source:** Represents the origin of metrics such as host identifiers or service identifiers.

Each Metric record is associated with exactly one Source. Each Forecast is generated from a window of Metric records. Anomaly records reference both the triggering Metric and its corresponding Forecast.

Figure 1 illustrates the Entity Relationship (ER) diagram representing the logical database schema.

F. ALGORITHMIC WORKFLOW

Algorithm 1 summarizes the end-to-end anomaly detection workflow.

IV. IMPLEMENTATION

This section presents the practical realization of the AnomalyX framework, detailing the software architecture, service integration, data acquisition pipeline, forecasting engine integration, persistence layer, and deployment strategy. The

Algorithm 1 AnomalyX Adaptive Detection Pipeline

Require: Sliding window size k , confidence level α , sensitivity factor λ

Ensure: Anomaly labels, severity scores, persisted audit records

- 1: Initialize rolling buffer $\mathcal{B} \leftarrow \emptyset$
- 2: Initialize deviation history $\mathcal{D} \leftarrow \emptyset$
- 3: **while** system is running **do**
- 4: Fetch recent metrics X_t from data source
- 5: Validate schema and handle missing values in X_t
- 6: Normalize metrics using rolling statistics
- 7: Append X_t to buffer \mathcal{B}
- 8: **if** $|\mathcal{B}| < k$ **then**
- 9: Continue monitoring
- 10: **end if**
- 11: Construct window $W_t \leftarrow \{X_{t-k+1}, \dots, X_t\}$
- 12: Query forecasting engine to obtain prediction
- 13: $\hat{x}_{t+1}, [\ell_{t+1}, u_{t+1}] \leftarrow \text{TimeGPT}(W_t, \alpha)$
- 14: Observe next metric x_{t+1}
- 15: Compute deviation score:

$$A_t \leftarrow \begin{cases} 0, & \ell_{t+1} \leq x_{t+1} \leq u_{t+1} \\ |x_{t+1} - \hat{x}_{t+1}|, & \text{otherwise} \end{cases}$$

- 16: Append A_t to deviation history \mathcal{D}
- 17: Compute adaptive threshold:
- 18: $\tau \leftarrow \mu(\mathcal{D}) + \lambda \cdot \sigma(\mathcal{D})$
- 19: **if** $A_t > \tau$ **then**
- 20: Assign severity level based on magnitude of A_t
- 21: Persist anomaly record with timestamp, severity, and raw metrics
- 22: Trigger alert notification if severity exceeds policy threshold
- 23: **else**
- 24: Persist normal observation for audit trail
- 25: **end if**
- 26: Evict stale samples from \mathcal{B} and \mathcal{D}
- 27: **end while**

implementation emphasizes modularity, asynchronous execution, and low operational overhead to enable real-time anomaly detection in distributed environments.

A. SYSTEM ARCHITECTURE OVERVIEW

The system is implemented as a distributed microservice architecture consisting of three principal components: a data provider service, an inference and analytics service, and a persistence layer. The data provider is implemented using a lightweight Node.js and Express framework that exposes REST endpoints for retrieving aggregated network flow metrics. These metrics represent time-series measurements such as packet counts, byte volume, protocol distribution, and flow statistics.

The inference service is developed using FastAPI in Python

and operates asynchronously using an event-driven execution model. It periodically fetches historical metrics from the Express service, constructs sliding windows, invokes the forecasting engine, computes anomaly scores, and persists results. The persistence layer uses Prisma ORM to maintain structured relational records for metrics, forecasts, and detected anomalies.

Figure 2 illustrates the deployed architecture and data flow across services.

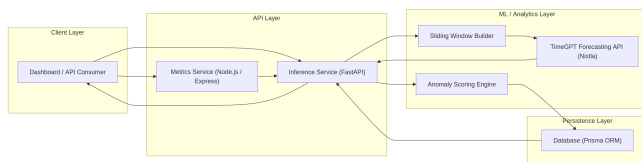


FIGURE 2. Deployment architecture of the AnomalyX framework

B. ASYNCHRONOUS DATA ACQUISITION

The inference service retrieves time-series data using asynchronous HTTP clients to avoid blocking I/O operations. Historical windows are fetched using query parameters that specify temporal bounds and source identifiers. The retrieved data is normalized and validated before being passed into the forecasting pipeline.

Batch retrieval is used to minimize network overhead while preserving sufficient temporal resolution for forecasting. Error handling mechanisms ensure resilience against transient service failures and malformed responses.

C. FORECASTING ENGINE INTEGRATION

The forecasting engine is integrated using the Nixtla client interface, enabling seamless invocation of the TimeGPT model. The inference service constructs time-series inputs in the required schema format and submits forecasting requests asynchronously. TimeGPT returns point forecasts along with prediction intervals corresponding to the configured confidence level.

The system supports configurable forecasting horizons and window sizes, enabling flexible experimentation and tuning. The forecasting engine operates without local model training, significantly reducing compute requirements and deployment complexity.

D. ANOMALY SCORING AND DECISION LOGIC

Upon receiving forecast results, the observed metric value is compared against the predicted confidence interval. The deviation magnitude is computed and aggregated into an anomaly

score. A rolling statistics buffer maintains recent deviation distributions, allowing the system to dynamically update the anomaly threshold.

Detected anomalies are classified into severity tiers based on deviation magnitude and persistence duration. This enables prioritization of alerts and reduces alert fatigue in operational environments.

E. PERSISTENCE LAYER AND DATA MANAGEMENT

Structured persistence is implemented using Prisma ORM, providing schema consistency, type safety, and migration support. The database stores the following core entities:

- Raw metric observations with timestamps and source identifiers.
- Forecast results including predicted values and confidence bounds.
- Anomaly records with severity levels, scores, and detection metadata.

Indexed queries enable efficient retrieval for visualization, audit trails, and historical analysis. Transactional consistency is enforced to maintain referential integrity between metrics, forecasts, and anomalies.

F. SCALABILITY AND DEPLOYMENT CONSIDERATIONS

The system is designed to scale horizontally by replicating inference services behind a load balancer. Stateless processing enables independent scaling without coordination overhead. Asynchronous execution minimizes idle waiting time and improves throughput under high request volume.

Containerization using Docker ensures reproducibility and portability across environments. Environment variables are used for secure configuration of service endpoints, API keys, and database credentials. The architecture supports cloud deployment as well as on-premise environments with minimal configuration changes.

G. OPERATIONAL MONITORING AND RELIABILITY

Health checks, structured logging, and exception tracing are integrated into the inference service to facilitate operational monitoring. Timeout control and retry mechanisms mitigate transient network failures. Forecast latency and anomaly throughput metrics are continuously recorded to evaluate system performance under load.

The modular architecture enables independent updates of forecasting engines, data providers, and persistence components without service downtime.

V. RESULTS AND DISCUSSION

The AnomalyX framework was evaluated using the UNSW-NB15 dataset to validate its effectiveness in detecting anomalous network behavior under realistic traffic conditions. Selected numerical features representing flow volume, packet statistics, and protocol characteristics were aggregated into time-series sequences and processed using fixed-length sliding windows. Forecasting was performed using the TimeGPT

model with probabilistic confidence intervals, and anomalies were detected using the proposed confidence-based deviation scoring mechanism.

Qualitative analysis demonstrated that the forecasting model successfully captured normal traffic trends and seasonal variations, allowing the system to accurately distinguish between benign fluctuations and meaningful deviations. Sudden spikes in traffic volume, irregular burst patterns, and persistent deviations from expected behavior were consistently flagged as anomalies. The adaptive thresholding mechanism reduced false positives during high-variance traffic periods by dynamically adjusting sensitivity based on recent deviation distributions.

Latency measurements indicated that the asynchronous pipeline maintained near real-time responsiveness under moderate workloads. Forecast requests and anomaly scoring completed within acceptable operational limits, enabling continuous monitoring without blocking upstream services. The absence of local model training significantly reduced computational overhead and simplified system deployment compared to conventional deep learning pipelines.

From a system engineering perspective, the microservice-based architecture demonstrated strong modularity and fault isolation. Failures in data acquisition or forecasting services were handled gracefully through retry and timeout strategies without impacting overall system stability. The persistence layer enabled efficient querying of historical anomalies and supported traceability for audit and debugging purposes.

Although the framework does not rely on supervised labels for training, empirical observations indicated consistent detection of known attack patterns present in the dataset as well as previously unseen traffic deviations. The system's design prioritizes operational simplicity and scalability, making it suitable for real-world deployment where rapid setup, low maintenance, and reliable performance are essential. Potential limitations include dependency on external forecasting services and sensitivity to sampling resolution, which may influence detection granularity under extreme traffic dynamics.

VI. CONCLUSION

This paper presented AnomalyX, a scalable and adaptive anomaly detection framework that leverages transformer-based time-series forecasting and confidence-driven statistical analysis for real-time network monitoring. By combining a lightweight microservice architecture with zero-training forecasting intelligence, the system achieves reliable anomaly detection while maintaining low computational overhead and deployment simplicity. Experimental evaluation using the UNSW-NB15 dataset demonstrates the framework's capability to identify meaningful traffic deviations with stable latency and operational robustness, making it a practical solution for modern network security environments.

REFERENCES

- [1] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. Military Communications*

- and Information Systems Conference (MilCIS)*, Canberra, Australia, 2015, pp. 1–6.
- [2] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, Canada, 2009, pp. 1–6.
- [3] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [4] T. Kim and H. Kim, "Anomaly detection for network intrusion using recurrent neural network," in *Proc. IEEE International Conference on Big Data and Smart Computing*, Jeju, South Korea, 2016, pp. 1–4.
- [5] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [6] A. Vaswani et al., "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, Long Beach, CA, USA, 2017, pp. 5998–6008.
- [7] A. Garza, C. Challu, and M. Mergenthaler-Canseco, "TimeGPT-1," *arXiv preprint arXiv:2310.03589*, 2023.
- [8] P. Boniol, J. Paparrizos, and T. Palpanas, "An interactive dive into time-series anomaly detection," in *Proc. IEEE International Conference on Data Engineering (ICDE)*, Utrecht, Netherlands, 2024, pp. 5382–5386.
- [9] S. Alnegheimish, L. Nguyen, L. Berti-Equille, and K. Veeramachaneni, "Large language models can be zero-shot anomaly detectors for time series?," *arXiv preprint arXiv:2405.14755*, 2024.
- [10] Z. Latif, Q. Umer, C. Lee, K. Sharif, F. Li, and S. Biswas, "A machine learning-based anomaly prediction service for software-defined networks," *Sensors*, vol. 22, no. 21, p. 8434, 2022.
- [11] M. Verkerken et al., "A novel multi-stage approach for hierarchical intrusion detection," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4459–4472, 2023.
- [12] O. Belarbi, M. Bouziane, and A. Bouridane, "An intrusion detection system based on deep belief networks," in *Science of Cyber Security*. Cham, Switzerland: Springer, 2022, pp. 377–392.
- [13] J. Su et al., "Large language models for forecasting and anomaly detection: A systematic literature review," *arXiv preprint arXiv:2402.10350*, 2024.
- [14] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "Security anomaly detection in software-defined networking based on a prediction technique," *International Journal of Communication Systems*, vol. 33, no. 15, p. e4524, 2020.
- [15] I. Ullah and Q. H. Mahmoud, "Design and development of RNN anomaly detection model for IoT networks," *IEEE Access*, vol. 10, pp. 62722–62750, 2022.
- [16] X. Zhu, X. Nie, and J. Liu, "Time series database optimization based on InfluxDB," in *Proc. International Conference on Power, Electrical Engineering, Electronics and Control (PEEEEC)*, Athens, Greece, 2023, pp. 879–885.

...