platforms

MDPI

Article

A Scalable Framework for Real-Time Network Security Traffic Analysis and Attack Detection Using Machine and Deep Learning

Zineb Maasaoui [1,2,*], Mheni Merzouki [1,3], Abdella Battou [1] and Ahmed Lbath [2]

[1] Smart Connected Systems Division, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA; m.merzouki@prometheuscomputing.com (M.M.); abdella.battou@nist.gov (A.B.)
[2] LIG/MRIM, CNRS, Grenoble Alpes University, 38400 Saint-Martin-d'Hères, France; ahmed.lbath@univ-grenoble-alpes.fr
[3] Prometheus Computing, Bethesda, MD 20811, USA
* Correspondence: zineb.maasaoui@univ-grenoble-alpes.fr

Abstract: This paper presents an advanced framework for real-time monitoring and analysis of network traffic and endpoint security in large-scale enterprises by addressing the increasing complexity and frequency of cyber-attacks. Our Network Security Traffic Analysis Platform employs a comprehensive technology stack including the Elastic Stack, ZEEK, Osquery, Kafka, and GeoLocation data. By integrating supervised machine learning models trained on the UNSW-NB15 dataset, we evaluate Random Forest (RF), Decision Trees (DT), and Support Vector Machines (SVM), with the Random Forest classifier achieving a notable accuracy of 99.32%. Leveraging Artificial Intelligence and Natural Language Processing, we apply the BERT model with a Byte-level Byte-pair tokenizer to enhance network-based attack detection in IoT systems. Experiments on UNSW-NB15, TON-IoT, and Edge-IIoT datasets demonstrate our platform's superiority over traditional methods in multi-class classification tasks, achieving near-perfect accuracy on the Edge-IIoT dataset. Furthermore, Network Security Traffic Analysis Platform's ability to produce actionable insights through charts, tables, histograms, and other visualizations underscores its capability in static analysis of traffic data. This dual approach of real-time and static analysis provides a robust foundation for developing scalable, efficient, and automated security solutions, essential for managing the evolving threats in modern networks.

Keywords: network security; intrusion detection; artificial intelligence; machine learning; natural language processing

check for updates

Academic Editor: Adel Ben Youssef

Received: 10 December 2024
Revised: 20 February 2025
Accepted: 31 March 2025
Published: 11 April 2025

Citation: Maasaoui, Z.; Merzouki, M.; Battou, A.; Lbath, A. A Scalable Framework for Real-Time Network Security Traffic Analysis and Attack Detection Using Machine and Deep Learning. *Platforms* **2025**, *3*, 7. https://doi.org/10.3390/platforms3020007

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The exponential growth of connected devices, particularly within Internet of Things (IoT) networks, has significantly increased both the complexity and frequency of cyber-attacks, many of which are financially motivated [1]. According to Cisco's 2024 Global Networking Trends Report, IP traffic continues to surge, placing organizations under unprecedented strain as they integrate emerging technologies such as artificial intelligence (AI), cloud computing, and IoT deployments [2]. Red Hat's research further indicates that containerization and cloud-native applications introduce new security challenges; nearly two-thirds of businesses report that these issues have led to delays or slowed deployments due to heightened security concerns [3]. These security challenges are further exacerbated by the rapid advancement of network technologies. The proliferation of 5G networks with ultra-reliable low-latency communication (URLLC) capabilities and the integration of AI

at the network edge [4] pose unprecedented difficulties for traditional intrusion detection systems (IDS) and firewalls, which often prioritize detection accuracy over timeliness. Recent empirical studies have highlighted significant performance limitations in current security systems. Research by Jin et al. [5] reveals that contemporary IDS face substantial performance constraints when processing high-speed network traffic—struggling to maintain real-time detection while preserving accuracy, with detection delays observed in networks exceeding 1.26 Gbps. Moreover, these systems often fail to identify network patterns that mimic normal traffic, raising concerns about the timeliness of IDS updates and the rapid emergence of new attack types [6,7]. As cyber-attacks become increasingly sophisticated, understanding the evolving properties of network flows is imperative which highlights the urgent need for real-time solutions to effectively detect, manage, and control network traffic [8].

Existing IDS are generally signature-based, enabling them to detect known threats effectively. Early studies, such as Denning's work [9], relied on predefined rules to characterize abnormal data and then matched these rules against observed data profiles to identify potential intrusions. Subsequent approaches aimed to improve detection by combining signature-based methods with immune-based intrusion detection to enhance detection rates and reduce false alarms [10]. More recently, machine learning methods have been introduced to detect zero-day intrusions in network traffic [11]. However, these approaches face significant challenges as a consequence of a lack of large, diverse datasets—many of which are outdated or lack traffic diversity [12–14].

The limitations of these traditional methods underscore the urgent need for automated systems that can efficiently process and analyze network traffic [15,16]. This need is particularly acute in large-scale enterprises, where the sheer volume and complexity of network data often overwhelm conventional techniques. High-profile cyber incidents, such as the ransomware attack on Colonial Pipeline in May 2021 [17], illustrate the devastating impact of security breaches and emphasize the necessity for more resilient cybersecurity measures.

The integration of Artificial Intelligence (AI) into network security marks a transformative shift, enhancing threat detection and reducing false positives through advanced machine learning techniques [18,19]. Recent advancements in Natural Language Processing (NLP) have introduced powerful architectures, such as the Transformer model [20] and large language models (LLMs) like BERT (Bidirectional Encoder Representations from Transformers) [21].

The application of NLP to network logs is an emerging research direction. In our approach, we transform raw network log data into a format analogous to natural language, thereby enabling the use of advanced NLP models for threat detection. By contextualizing and structuring network logs to meet the input requirements of these models, we can leverage their ability to capture complex patterns and dependencies—features critical for identifying subtle anomalies and sophisticated cyber-attacks. This innovative cross-domain application not only enhances detection accuracy but also provides a novel framework for analyzing network traffic in real time.

In this context, we present the Network Security Traffic Analysis Platform (NSTAP), which is a comprehensive framework that addresses the need for scalable, efficient, and automated security solutions essential for managing evolving threats in modern networks. NSTAP has evolved through several critical phases. Initially, our research focused on the static analysis of network traffic and endpoint data, which allowed for in-depth monitoring and understanding of network performance and security across large-scale environments. This foundational stage provided crucial insights into network behavior and laid the groundwork for efficient data collection and real-time threat detection.

To meet the growing demands of large-scale networks, we then integrated machine learning models into a scalable platform architecture. By employing technologies such as Elastic Stack [22], Zeek [23], Osquery [24], Kafka [25], and GeoLocation data, NSTAP enables robust, real-time monitoring and analysis. Our platform efficiently ingests, processes, and analyzes massive volumes of data, thereby automating security operations to proactively address evolving threats. NSTAP is also equipped with supervised machine learning models optimized for detecting network anomalies, trained and tested on UNSW-NB15 dataset.

In the final phase of our research, we introduced Large Language Models (LLMs) to further enhance the detection of complex network-based attacks, particularly within IoT environments. By leveraging a BERT model with a Byte-Level Byte-Pair Encoding (BBPE) tokenizer—specially trained on multiple datasets including UNSW-NB15, TON-IoT, and Edge-IIoT—NSTAP demonstrates superior performance in multi-class classification tasks, surpassing traditional machine learning approaches. This novel integration of LLMs allows for a more refined analysis of network logs, improving both the accuracy and scalability of the platform.

Our contributions in this paper are as follows:

- We present a phased approach to network security analysis, progressing from static traffic monitoring to the integration of machine learning and culminating in the deployment of LLMs.
- We offer a scalable, real-time platform that addresses the unique security challenges of large-scale networks by combining big data analytics with advanced endpoint security.
- We evaluate the performance of a BBPE-trained BERT model for detecting network threats across multiple datasets, demonstrating high performance and accuracy.
- Finally, NSTAP serves as a comprehensive solution for proactive threat detection and network management in enterprise environments.

The remainder of this paper is organized as follows: Section 1 provides the introduction and contextual background. Section 2 presents a comprehensive review of related work and current state-of-the-art approaches in real-time intrusion detection systems. Section 3 examines the core architectural components of our platform by detailing the technological infrastructure and data pipeline design principles that enable near real-time monitoring capabilities. Section 4 elaborates on the evolutionary progression of our methodological approach by documenting the integration and optimization of various detection techniques. Section 5 presents our experimental results and provides detailed analysis of our findings, which includes comparative performance metrics and statistical validation. Finally, Section 6 concludes with a summary of our contributions and discusses potential directions for future research in this domain.

2. Related Work

The surge in internet usage has been paralleled by a rise in network attacks, particularly Distributed Denial of Service (DDoS) attacks, which pose significant threats to communication networks [26]. To mitigate these risks, various lightweight algorithms have been proposed for DDoS protection [27]. Traditional intrusion detection systems (IDS) typically employ either signature-based [28] or anomaly-based [29] techniques. While signature-based methods effectively detect known threats, they are inherently limited in identifying novel or zero-day attacks. Conversely, anomaly-based approaches can identify unknown threats but often generate high false-positive rates.

The limitations of traditional IDS have spurred interest in machine learning (ML) and deep learning (DL) methodologies for intrusion detection [30]. A critical component

of ML-based IDS is the availability of robust datasets that reflect contemporary network traffic scenarios. The UNSW-NB15 dataset [31] has become a benchmark for evaluating network intrusion detection systems by combining real-world data with synthetic attack patterns. Other datasets, such as NSL-KDD and KDD, have also been employed to explore various ML techniques in this domain [32]. Feature selection plays a vital role in optimizing model performance; studies have shown that Support Vector Machines (SVM) and Artificial Neural Networks (ANN) can achieve high detection rates when key features are carefully chosen [33]. However, relying on a limited feature set may restrict a model's ability to capture complex patterns, potentially leading to suboptimal performance in diverse conditions.

Integrating ML into real-time detection systems within large-scale network environments presents significant challenges. Handling big data streams requires architectures capable of processing substantial volumes of information efficiently. Several studies have addressed these challenges by leveraging big data technologies. For instance, an ML-driven IDS designed for Big Data Analytics in Vehicular Ad-Hoc Networks (VANET) utilized Apache Kafka for message queuing and the Elastic Stack (comprising Elasticsearch, Logstash, and Kibana) for data storage, indexing, and visualization [34]. Similarly, the integration of the Elastic Stack with IDS tools such as Zeek and Slips has been explored to enhance cyber threat detection by combining IDS, ML, and Security Information and Event Management (SIEM) systems [35]. Additionally, proactive threat detection mechanisms that combine the Elastic Stack with open-source threat intelligence platforms have demonstrated effectiveness against advanced persistent threats (APTs) [36].

While these solutions represent significant progress, they often lack scalability when dealing with large volumes of network flows and may not provide access to historical traffic data. Moreover, alternatives like OpenSearch exist for large data processing, but Elasticsearch has advanced more rapidly in key innovations and features hence resulting in improved performance.

Recent advancements have seen the application of Large Language Models (LLMs) in intrusion detection, which offer novel approaches for processing unstructured data such as raw log messages. BERT-Log [37] leverages the pre-trained BERT model to automate anomaly detection in log data and significantly outperforms traditional methods in capturing semantic information. NeuralLog [38] addresses issues such as log parsing errors and out-of-vocabulary words by eliminating the need for log parsing and directly utilizing raw log messages. Similarly, LogBERT [39] introduces a self-supervised framework for detecting log anomalies without conventional parsing by incorporating tasks such as masked log key prediction and reducing hypersphere volume.

In contrast to prior studies, our research develops a scalable platform—NSTAP—for real-time network traffic analysis and intrusion detection. NSTAP integrates supervised ML models and LLMs with the Elastic Stack, Osquery, and Kafka to efficiently process massive data streams in large-scale enterprise networks. Beyond achieving superior detection accuracy, NSTAP provides access to historical data for in-depth analysis and long-term threat assessment, thereby setting a new benchmark for comprehensive, automated cybersecurity solutions in enterprise environments.

To evaluate the contributions of existing frameworks systematically, Table 1 presents a comprehensive comparison of notable security frameworks. This comparative analysis examines key dimensions, including framework capabilities (e.g., HIDS/NIDS integration, dataset collection abilities, and large-scale data processing), detection techniques (ML, DL, and NLP approaches), classification methodologies, and the datasets employed for validation. As illustrated in Table 1, while existing frameworks have made significant strides in specific aspects of network security, most solutions tend to focus on particular capabilities

rather than offering a holistic approach. Our NSTAP framework distinguishes itself through the integration of multiple security components and advanced analytical techniques.

Table 1. Comprehensive Analysis of Security Frameworks: Capabilities, Implementation Characteristics, and Performance Metrics.

Framework	Framework Capabilities				Detection Techniques			Classification		Datasets	Detection Accuracy (%)
	HIDS	NIDS	Dataset Collection	Large Scale Data Flow	ML	DL	NLP	Binary	Multi-Class		
NSTAP [6,8,13]	✓	✓	✓	✓	✓	✓	✓	✓	✓	• CICIDS 2017 • ToN-IoT • Edge-IIoT • UNSW-NB15 • FDIA	100 (Edge-IIoT)
ELK-ML [36]	✓	✓	✓	✗	✓	✗	✗	✓	✓	• Custom APT • DNS/HTTP Exfiltration	-
AB-TRAP [40]	✗	✓	✓	✓	✓	✗	✗	✓	✗	• KDD 99 • NSL-KDD	-
Catboost-IDS [41]	✓	✓	✓	✓	✓	✓	✗	✓	✓	• NSL-KDD • IoT-23 • BoT-IoT • Edge-IIoT	100 (Edge-IIoT)
MalBERT [42]	✗	✗	✓	✓	✓	✓	✓	✓	✓	• Androzoo (22,000 apps)	97.61
MADAM ID [43]	✓	✓	✓	✓	✓	✗	✗	✓	✓	• DARPA 1998 • BSM audit data	-
Feed-forward NN [44]	✗	✓	✗	✓	✓	✗	✗	✓	✓	• NSL-KDD	82.41
Kitsune [45]	✗	✓	✓	✓	✓	✗	✗	✓	✓	• Custom collected Data	-
LSTM [46]	✗	✓	✗	✓	✓	✓	✗	✓	✓	• KDD99	99.49
IDBN + RNN [47]	✓	✓	✓	✓	✓	✓	✗	✓	✓	• Bot-IoT • UNSW-NB15	92.6 (UNSW)

3. Platform Architecture

To tackle the complexity of real-time network security analysis effectively, the Network Security Traffic Analysis Platform (NSTAP) is built upon a robust and scalable platform architecture. In this section we will delve into the design principles and core components of the platform. We introduce and define the key technologies integrated. Comparative analyses with similar technologies are provided to justify our choices and demonstrate the advantages of our architecture.

3.1. Design Principles and Data Pipeline

The platform for collecting network logs and traffic follows a structured data pipeline model to ensure efficient log collection, transformation, storage, and analysis. The pipeline is designed to handle large-scale network traffic efficiently, ensuring minimal data loss and real-time processing capabilities. this is achieved by leveraging the Elastic Stack's key components: Elastic agents, Logstash, Elasticsearch, and Kibana, each playing a critical role in the data pipeline. Beats agents, which are lightweight data shippers installed on network devices and endpoints, are responsible for collecting various types of network logs and traffic, including system logs, application logs, and packet capture data. In addition, the elastic agents Zeek and OSQuery are used to collect network flows and end system logs, as shown in Figure 1. These logs are then forwarded to Logstash, which acts as the platform's data processing engine. Logstash performs transformations, filtering, and enrichment of the incoming data, ensuring that it is normalized and structured for further analysis.

In this manuscript, we use the term 'pipeline' to describe a structured, multi-stage data flow architecture designed for high-throughput and real-time processing. Unlike hard-

ware pipelining, which focuses on overlapping execution stages for improved clock cycle efficiency, the data pipeline model optimizes concurrent data ingestion, transformation, and storage. This approach ensures continuous processing of application logs and network events without bottlenecks, leveraging streaming and asynchronous communication mechanisms for efficiency.

Kafka enables real-time data streaming by decoupling log producers from consumers, ensuring continuous ingestion without waiting for downstream processing. Logstash processes and transforms logs concurrently as they arrive, allowing ingestion and enrichment to occur in parallel. Additionally, Elasticsearch's indexing mechanism facilitates rapid querying and retrieval without impacting live data ingestion. This architecture reduces end-to-end log processing latency, ensuring that security insights are available in near real-time.

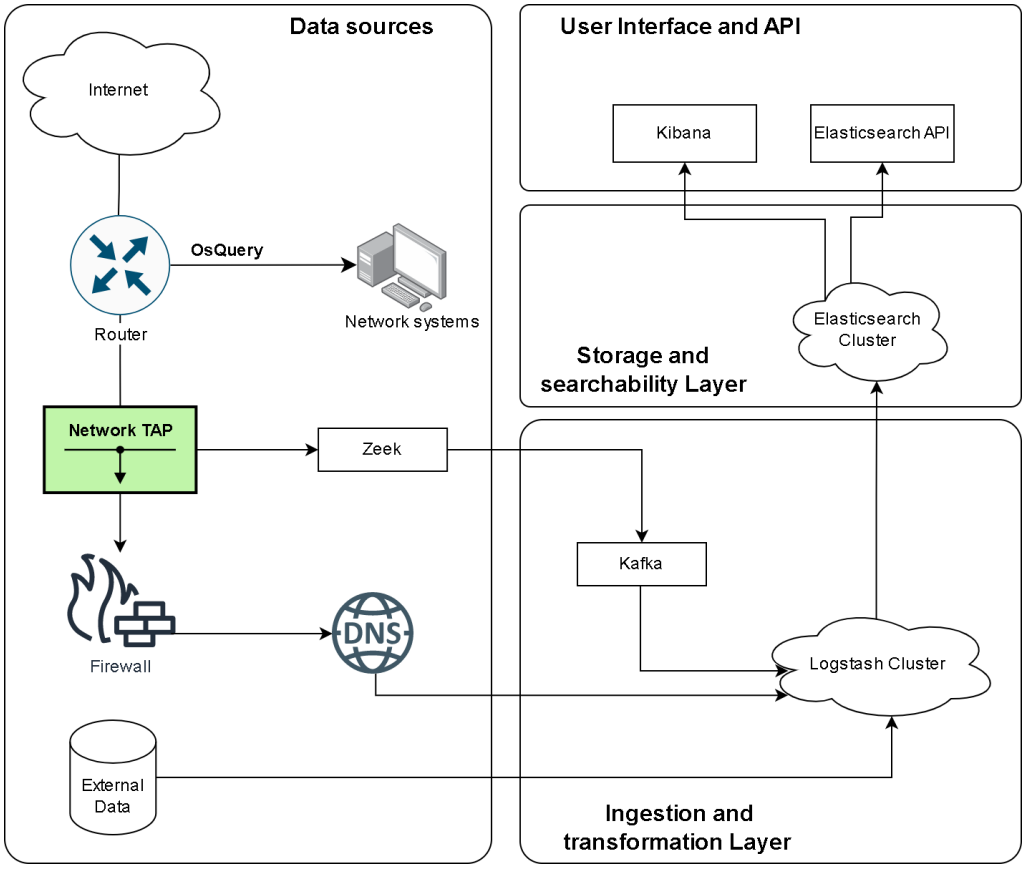


Figure 1. Data Pipeline Architecture, Reprinted with permission from "Design and Implementation of an Automated Network Traffic Analysis System using Elastic Stack [8]", IEEE Proceedings, 2023. © 2023 IEEE.

Once processed, the data is indexed by Elasticsearch, a distributed search and analytics engine designed to store large volumes of data while allowing for near-real-time querying and retrieval. The data stored in Elasticsearch is indexed, enabling rapid search capabilities that are crucial for detecting anomalies and potential intrusions. Kibana, the platform's visualization layer, allows for the creation of interactive dashboards and data exploration tools, providing security teams with insights into network behavior and facilitating the monitoring of security events.

Throughout this pipeline, the data flows seamlessly from collection (Beats), to processing (Logstash), to storage (Elasticsearch), and finally to visualization (Kibana). This modular architecture ensures that new data sources can be easily added or modified, while the platform's scalability allows it to adapt to growing data volumes. Machine learning

models and AI-driven analysis can be integrated directly within Elasticsearch, enhancing the platform's capabilities for real-time intrusion detection and threat analysis.

3.2. Core Components

This section outlines the foundational tools and frameworks that power NSTAP's capabilities in real-time data processing, monitoring, and analysis. Each component (Elastic Stack, Kafka, Zeek, and Osquery ...etc.) plays a unique role, from handling high-throughput data streams to monitoring network traffic and endpoints. This section discusses how these technologies are integrated into NSTAP to provide a scalable and efficient architecture for data collection, storage, and analysis, with comparisons to alternative solutions where relevant.

3.2.1. Elastic Stack

Elastic Stack, formerly known as ELK Stack, is a collection of open-source software tools for collecting, parsing, storing, and visualizing large volumes of data in real-time [22]. The stack comprises:

- Elasticsearch: A distributed, RESTful search and analytics engine capable of storing and indexing data.
- Logstash: A data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a stash like Elasticsearch.
- Kibana: A data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases.
- Beats: Lightweight data shippers that send data from hundreds or thousands of machines to Logstash or Elasticsearch.

In NSTAP, Elastic Stack serves as the backbone for real-time data ingestion and storage. It enables the platform to collect logs and metrics from various sources, process and analyze data in real-time, and provide dashboards for monitoring and alerting.

3.2.2. Kafka

Apache Kafka is a distributed streaming platform designed for high-throughput, low-latency data processing [25]. It enables the publishing and subscribing to streams of records, similar to a message queue or enterprise messaging system. Kafka is scalable, fault-tolerant, and optimized for real-time data feeds.

In NSTAP, Kafka is utilized to handle large-scale data streams efficiently. It acts as a buffer between data producers (Beats agents, Zeek, Osquery) and data consumers (Logstash, machine and deep learning modules), ensuring that data ingestion is smooth and that the system can handle high volumes of data without loss or delay.

To determine the suitability of Apache Kafka for NSTAP, we compare it with other popular message brokers such as RabbitMQ and ActiveMQ. Table 2 summarizes the key features and capabilities of these technologies.

Kafka meets the high throughput and scalability, which are critical for NSTAP's requirement to handle large volumes of data in real-time. Its ability to retain data and replay messages also provides robustness in data processing workflows.

Table 2. Comparative Analysis of Message Brokers.

Feature	Message Broker Systems		
	Apache Kafka [25]	**RabbitMQ [48]**	**ActiveMQ [49]**
Messaging Model	Pub/Sub, Streams	Queue-Based	Queue-Based
Throughput	Very High	Moderate	Moderate
Latency	Low	Low	Moderate
Scalability	Horizontal	Vertical	Horizontal
Data Retention	Yes	Limited	Limited
Fault Tolerance	High	High	Moderate
Use Cases	Real-Time Analytics	Messaging, Asynchronous Tasks	Messaging, Integration
Community Support	Strong	Strong	Moderate

3.2.3. Zeek

Zeek, formerly known as Bro, is a powerful open-source network analysis framework focused on security monitoring [23]. It passively monitors network traffic and records detailed logs of activity, providing a rich set of data for analysis.

In NSTAP, Zeek is deployed to perform deep inspection of network traffic. It generates comprehensive logs that include details about protocols, connections, and potential security events. Additionally, Zeek offers a flexible scripting language, which allows for the creation of custom filters to tailor the collected network data flow. These logs are then forwarded to the Elastic Stack for storage and visualization, and to the machine learning modules for anomaly detection and intrusion detection. In our approach, we used a custom Zeek script to store network logs in an SQLite database and to integrate geolocation capabilities. Storing Zeek logs in an SQLite database facilitates both data storage and querying. Moreover, incorporating geolocation information into Zeek enables us to focus on specific regions and better highlight the origins of potential attacks.

To assess how Zeek meets the requirements for network monitoring, we compare it with other tools such as Snort and Suricata in Table 3. This comparison is conducted to analyze the capabilities and features of these tools relative to the specified needs.

Table 3. Comparative Analysis of Network Monitoring Tools.

Feature	Network Monitoring Tools		
	Zeek [23]	**Snort [50]**	**Suricata [51]**
Type	NTA (Network Traffic Analysis)	IDS/IPS	IDS/IPS
Detection Method	Signature and Anomaly-Based	Signature-Based	Signature and Anomaly-Based
Protocol Analysis	Deep	Shallow	Deep
Performance	High	Moderate	High
Custom Scripting	Yes (Zeek Script)	Limited	Yes (Lua Scripting)
Use Cases	Monitoring, Forensics	Intrusion Detection	Intrusion Detection
Community Support	Strong	Strong	Growing

Zeek offers deep protocol analysis and flexible scripting capabilities, allowing for extensive customization. Its focus on detailed logs and analysis makes it more suitable for comprehensive network monitoring and forensic analysis within NSTAP.

3.2.4. Osquery

Osquery is an operating system instrumentation framework for Windows, macOS, and Linux [24]. It exposes an operating system as a high performance relational database, allowing users to write SQL-based queries to explore the system data.

In NSTAP, Osquery is used to collect detailed endpoint data, such as running processes, loaded kernel modules, open network connections, and file integrity monitoring. This information enhances NSTAP's visibility into endpoint activities, contributing to effective intrusion detection and incident response.

We compare Osquery with other endpoint security tools such as Sysmon and Auditd in Table 4.

Table 4. Comparative Analysis of Endpoint Security Tools.

Feature	**Endpoint Security Tools**		
	Osquery [24]	**Sysmon [52]**	**Auditd [53]**
Platform Support	Cross-Platform	Windows Only	Linux Only
Data Collection	System Info, Processes, File Integrity	Process Creation, Network Connections	System Calls, File Access
Query Language	SQL-Based Queries	None	Rules-Based
Real-Time Monitoring	Yes	Yes	Yes
Customization	High	Moderate	Moderate
Integration	Easy (via SQL Queries)	Requires Event Forwarding	Complex Configuration
Community Support	Strong	Moderate	Moderate

The cross-platform support of Osquery and the SQL-based query language make it a versatile and powerful tool for endpoint monitoring in NSTAP. Its ability to standardize data collection across different operating systems simplifies the integration process and enhances overall security visibility.

3.3. Deployment

The system architecture is deployed using virtual machines (VMs). To ensure a repeatable and automated infrastructure setup, we employed Infrastructure as Code (IaC) with Terraform. This approach streamlines the deployment of complex systems like the Elastic Stack cluster, making it efficient and scalable. In addition, the use of virtualization technologies, such as QEMU and libvirt, provides a secure and isolated environment for running virtual machines.

The deployment includes several specialized nodes tailored to specific tasks: five storage nodes for Elasticsearch, utilizing the Hot, Warm, and Cold data storage strategy; two nodes for data processing using Logstash; two messaging queue nodes powered by Kafka; and a single dashboard node running Kibana. Each of these nodes operates on VMs created using QEMU and libvirt, running atop Linux (Ubuntu 20.04) bare-metal hosts.

Before initiating the deployment, the host machines undergo specific configurations, including the installation of required virtualization packages, disabling of AppArmor or SELinux to accommodate QEMU, and the setup of network traffic rules to allow seamless communication to and from the VMs.

The following Figure 2 provides a visual representation of the Elastic Stack deployment architecture, which outlines the allocation of VM and hardware resources across host machines.

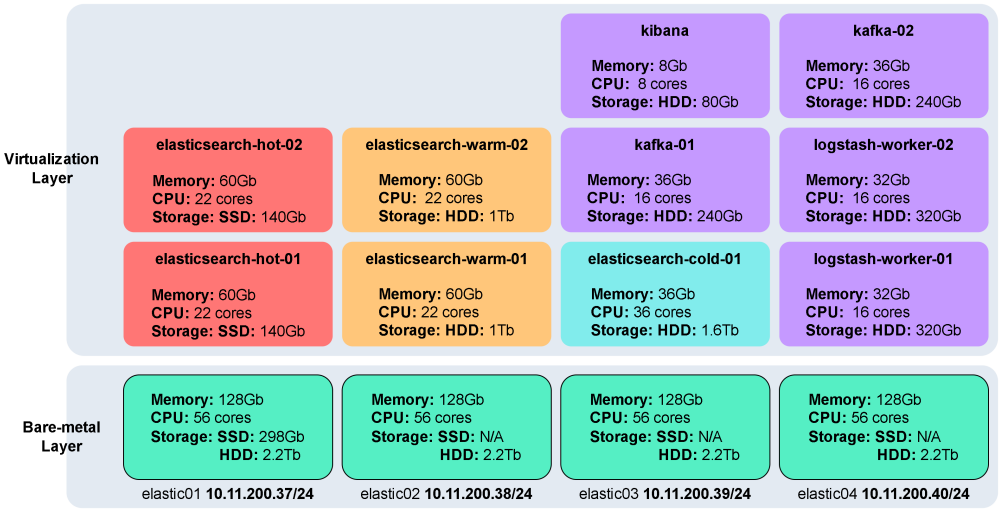


Figure 2. Elastic Stack deployment. This figure illustrates the architecture for log ingestion, processing, and visualization within the NSTAP framework. Reprinted with permissio n from "Design and Implementation of an Automated Network Traffic Analysis System using Elastic Stack [8]", IEEE Proceedings, 2023. © 2023 IEEE.

User Role and Access Control

In modern network security platforms, managing user access and permissions is crucial to maintain the integrity and confidentiality of sensitive data. NSTAP incorporates a robust role-based access control (RBAC) system that allows administrators to define specific roles and assign appropriate permissions to users. The admin user has the authority to create custom roles tailored to organizational needs, granting or restricting access to various components of the platform.

Authorized users can create teams, enabling collaborative efforts among security analysts and IT personnel. Teams can be granted access to specific information, such as logs from certain network segments or data from particular hosts. This granular control ensures that users have access only to the data necessary for their roles, adhering to the principle of least privilege.

The RBAC system is integrated with NSTAP's core components, leveraging Elastic Stack's security features for seamless authentication and authorization. By providing a user-friendly interface for role management, NSTAP simplifies user administration while enhancing security.

4. Intrusion Detection Techniques Integrated into the Platform

In this paper we investigate several aspects of intrusion detetion techniques for network logs. This section describes all the considered techniques.

4.1. Static Analysis in NSTAP

In the early stages of platform development, we deployed multiple Zeek and Osquery agents to collect and forward data in near-real time. For the Osquery agents, each was directly deployed on the host machines, running in the background as osqueryd. This setup allows for querying individual hosts, scheduling queries for specific hosts, and adding relevant hosts to the platform for continuous monitoring.

The data collected by both Zeek and Osquery agents serve as input for static analysis within the NSTAP platform. To illustrate this analysis, we focused on a segment of the substation's energy network [54], as shown in Figure 3. The one-line diagram represents a power system used to generate network traces for cybersecurity analysis. This system

consists of buses and 18 Intelligent Electronic Devices (IEDs), each identified by a unique MAC address. These IEDs communicate using the IEC 61850 Generic Object Oriented Substation Event (GOOSE) protocol, which facilitates information exchange between devices in electrical substations. The generated GOOSE trace files can be analyzed within Zeek, producing goose.log files. These logs capture a subset of the GOOSE data fields, specifically those relevant to cybersecurity analysis, enabling further examination of communication patterns and potential threats.

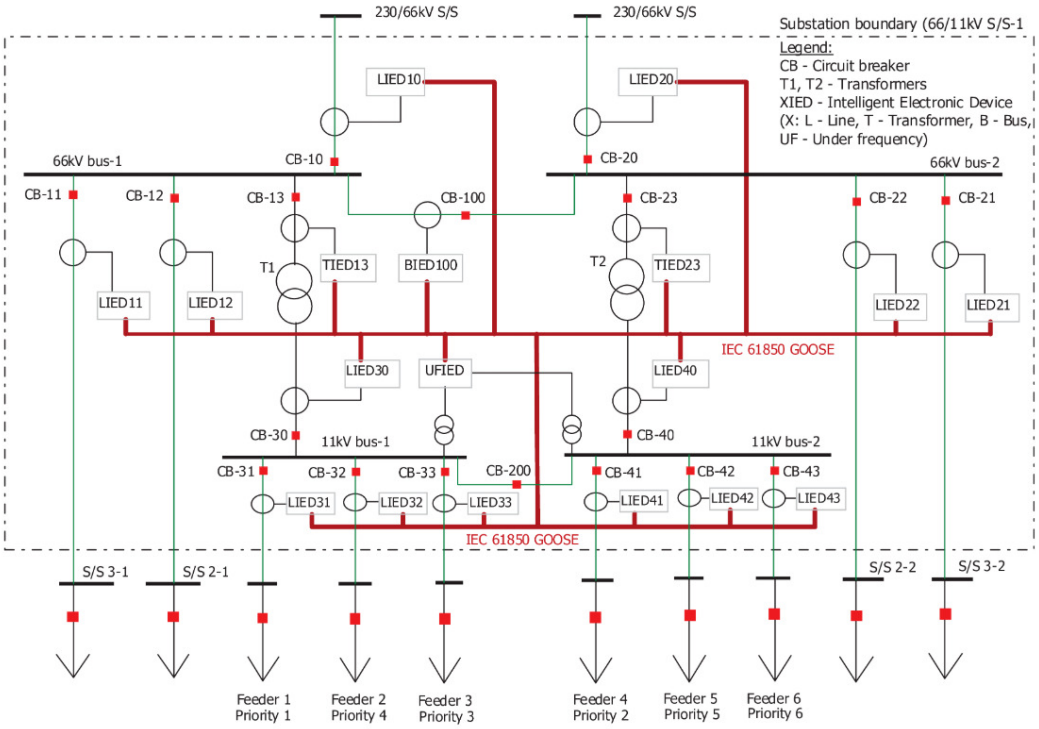


Figure 3. Substation one-line diagram depicting the fundamental power system components and their interconnections [54]. The diagram illustrates the essential electrical configuration and protection schemes typical of modern substation architecture.

We utilized the IEC-61850 dataset [54] to feed the NSTAP platform and simulated various scenarios including attack-free, attack-free with disturbance, and attack trace with message suppression. The analysis within NSTAP is conducted by: (1) Searching the logs through the NSTAP logs interface, and (2) Generating visualizations to illustrate the network traffic behavior.

For the normal scenario, where all IEDs operate normally, the trace generator is set to 1 GOOSE frame per second, with no event changes in the dataset. In Figure 4, the SqNum and StNum are correlated with each GOOSE frame and event, respectively, where StNum and SqNum represent counters indicating changes within the GOOSE frames and the dataset [54].

In one key scenario, a breaker failure was simulated as a type of disturbance in the power system. A line fault triggered the breaker failure protection scheme, prompting LIED11 to send a GOOSE frame to its neighboring IEDs, informing them of the malfunction in CB-11. The traffic analysis, visualized in Figure 4, shows how the status number (StNum) increases progressively from 1 to 3 as the fault evolves, while the sequence number (SqNum) resets to 0 when the disturbance occurs. This analysis highlights NSTAP's capability to track changes in network traffic in real time and detect critical disturbances.

Additionally, custom traffic filters, allow for more refined analysis, enhancing the detection of anomalies and improving overall traffic monitoring.

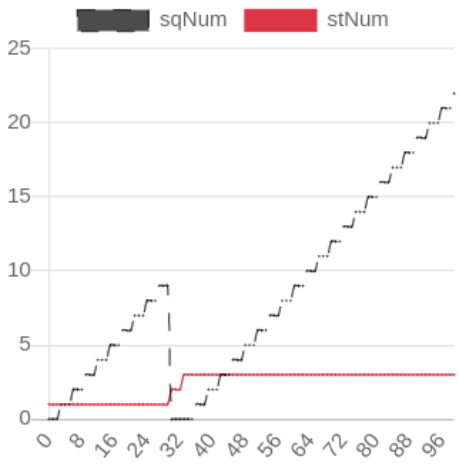


Figure 4. NSTAP Visualization of Network Traffic Disturbance. The x-axis represents temporal progression in discrete intervals, while the y-axis quantifies the evolution of State Number (StNum) and Sequence Number (SqNum) metrics during the observation period. Reprinted with permission from "Network Security Traffic Analysis Platform—Design and Validation [6]", IEEE Proceedings, 2022. © 2022 IEEE.

4.2. AI Integration in NSTAP

With the expansion of remote access in recent years, organizations' attack surfaces have grown significantly, presenting new and complex security challenges. Addressing these vulnerabilities requires innovative, robust solutions that extend beyond traditional methods. Artificial intelligence (AI) and machine learning (ML) offer predictive capabilities, enabling proactive incident management by anticipating threats before they occur. By helping organizations prevent downtime and mitigate the impact of outages or cyberattacks, these technologies add substantial value to network security.

Building on the static analysis capabilities in NSTAP, we now integrate AI-driven techniques to detect patterns within network traffic flows. This approach enhances detection capabilities and strengthens our strategies for early response, making NSTAP a more resilient platform for addressing modern cybersecurity threats.

4.2.1. Datasets

This study utilizes three foundational datasets: UNSW-NB15, ToN-IoT, and Edge-IIoTset. The UNSW-NB15 dataset, produced by the Cyber Range Lab at UNSW Canberra, includes approximately 100 GB of raw network traffic data with nine distinct attack types such as DoS, Fuzzers, and Exploits. This dataset contains over 2.5 million records, organized into training and testing sets and characterized by 49 features. ToN-IoT is a dataset specifically designed for Industry 4.0, covering IoT and IIoT environments. It combines telemetry data from IoT sensors, operating system logs, and network traffic. Collected from UNSW Canberra's Cyber Range and IoT Labs, this dataset captures multiple attack types, including DoS, DDoS, and ransomware, to facilitate AI-based cybersecurity research. The Edge-IIoTset dataset was generated using a custom IoT/IIoT testbed and includes data from more than ten types of IoT devices. It encompasses fourteen distinct attack types grouped into five categories: DoS/DDoS, information gathering, man-in-the-middle, injection, and malware. With 61 correlated features extracted from system logs, network traffic, alerts, and resource usage, it offers valuable insights for intrusion detection research.

These datasets were specifically selected for their complementary nature and contemporary relevance. By combining them for training, we expose our model to a diverse range of attack patterns and network behaviors. Testing each dataset separately allows us to evaluate our model's ability to generalize across different network environments-

from traditional IT networks (UNSW-NB15) to IoT (ToN-IoT) and industrial IoT systems (Edge-IIoTset). This approach ensures our framework's effectiveness across various modern cybersecurity contexts.

Table 5 presents the detailed attack categories and distribution of records across datasets. By leveraging the diversity in these datasets, the study achieves more robust and generalizable model performance across various attack scenarios.

Table 5. Statistical Distribution of Attack Categories Across Multiple Datasets [13].

Attack Category	ToN-IoT	UNSW-NB15	Edge-IIoT	Total
Normal	50,000	93,000	1,615,643	1,758,643
XSS	20,000	-	15,915	35,915
Scanning	20,000	-	22,564	42,564
Ransomware	20,000	-	10,925	30,925
Password	20,000	-	50,153	70,153
Injection	20,000	-	51,203	71,203
DoS	20,000	16,353	50,062	86,415
DDoS	20,000	-	171,630	191,630
Backdoor	20,000	2329	24,862	47,191
MiTM	1043	-	1214	2257
Generic	-	58,871	-	58,871
Exploits	-	44,525	-	44,525
Fuzzers	-	24,246	-	24,246
Reconnaissance	-	13,987	-	13,987
Analysis	-	2677	-	2677
Shellcode	-	1511	-	1511
Worms	-	174	-	174
Vul Scanner	-	-	50,110	50,110
DDoS_ICMP	-	-	116,436	116,436
DDoS_HTTP	-	-	49,911	49,911
Uploading	-	-	37,634	37,634
Port Scanning	-	-	22,564	22,564
Fingerprinting	-	-	1001	1001
Attack types	9	9	14	22

4.2.2. Evaluation Metrics

To comprehensively assess our model's effectiveness in classifying network logs, we employ a suite of robust metrics that provide detailed insights into its accuracy, precision, and ability to generalize across different network threat types. These metrics are crucial for verifying the model's performance and pinpointing potential areas for enhancement.

Validation Loss

Validation loss quantifies the model's error on the validation dataset, providing a direct measure of performance:

$$L(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

where y is the vector of true values, and $\hat{y}$ is the vector of predicted values by the model. Lower validation loss indicates better generalization.

Weighted F1 Score

The $F1$ score is the harmonic mean of precision and recall, particularly important in the presence of class imbalance. It is calculated as:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

where Precision and Recall are defined as:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN}$$

Weighted F1 considers each class's support:

$$F1_{\text{weighted}} = \sum_{j=1}^{k} \left(\frac{n_j}{N} \cdot F1_j \right)$$

where n_j is the number of samples in class j, N is the total number of samples, and $F1_j$ is the F1 score of class j.

Accuracy

Overall accuracy is defined as the proportion of true results (both true positives and true negatives) among the total number of cases examined:

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} 1(y_i = \hat{y}_i)}{N}$$

where 1 is the indicator function.

Class-Specific Accuracy

Accuracy for each class is critical for understanding model performance on individual threat types, calculated for each class as:

$$\text{Accuracy}_{\text{class}} = \frac{TP + TN}{TP + TN + FP + FN}$$

4.2.3. Machine Learning Techniques

Machine learning (ML), a subset of artificial intelligence (AI), involves training algorithms to recognize patterns from existing data, enabling them to make predictions on new, unseen data. In the realm of cybersecurity, machine learning methods have proven to be significantly more effective than traditional static approaches, making them crucial to explore.

Among the three common categories of machine learning, this study focuses on supervised learning. Supervised machine learning entails training a model on labeled inputs and their corresponding desired outputs, with the goal of enabling it to perform specific tasks when presented with new or unfamiliar data. In cybersecurity applications, a typical use of supervised learning is training models on datasets of benign and malicious samples to enable the prediction of whether new samples are malicious.

In NSTAP, one of the primary applications of ML is to enhance anomaly detection. By identifying irregularities in data, ML models can inform risk scoring and guide

threat investigations, thereby strengthening our ability to detect and respond to security incidents promptly.

To enhance the classification efficacy for intrusion detection within NSTAP, we evaluated the following established supervised machine learning algorithms:

Decision Trees (DT)

The Decision Tree algorithm is a supervised learning technique commonly used for classification tasks, known for its interpretability, ease of implementation, and effectiveness in handling both categorical and numerical data. In the context of network log classification, Decision Trees are particularly useful due to their ability to capture complex decision boundaries and handle large amounts of structured network data efficiently [55,56].

Random Forest (RF)

Random Forest is a powerful ensemble learning technique that builds upon the foundation of Decision Trees, known for its robustness and high accuracy in complex classification tasks such as network log analysis [57]. In this domain, RF excels at distinguishing between normal and malicious traffic patterns by combining the predictions of multiple Decision Trees, each trained on random subsets of data and features.

The Random Forest algorithm works by constructing multiple Decision Trees during the training phase through a process called bootstrapping, where each tree is trained on a random sample of the dataset [58]. At each node, a random subset of features is considered, ensuring that the trees are diverse. This diversity mitigates overfitting and enhances the model's ability to generalize to unseen data—a critical requirement when handling diverse network logs that may include both common and rare attack types. By aggregating the outputs of individual trees through majority voting, RF achieves more robust and stable predictions compared to a single Decision Tree model. Additionally, RF naturally provides an estimate of model performance through out-of-bag error calculation and offers flexibility in tuning hyperparameters, which is essential for optimizing detection performance in real-world scenarios.

After selecting suitable machine learning algorithms for classification, the next step in the workflow involves optimizing the dataset through feature selection. This ensures that each classifier operates on the most relevant attributes, improving overall accuracy and efficiency.

4.2.4. Feature Selection Techniques

Optimizing machine learning performance hinges on effective feature selection. To improve classification results, we focused on reducing the number of features in the dataset. We employed two primary techniques:

Pearson Correlation Coefficient

The Pearson correlation coefficient (r) quantifies the linear relationship between two variables, ranging from -1 (perfect negative correlation) to 1 (perfect positive correlation), with 0 indicating no linear correlation. It is calculated using the formula:

$$r = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

where x_i and y_i are individual data points, and $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively. In our study, we analyzed the Pearson correlation matrix to retain only the attributes that exhibited a high correlation with the target label, thereby eliminating redundant or irrelevant features.

SVM-Based Feature Reduction Algorithm

Inspired by the work of Chowdhury et al. [33], we developed a customized feature selection algorithm based on SVM to address limitations such as reliance on a fixed number of features and computational burden. Our algorithm aims to achieve near-optimal accuracy through an iterative process that evaluates feature subsets ranging from three to ten features.

Algorithm Overview

Our SVM-based feature reduction algorithm follows these steps:

1. Initialization: Set initial parameters, including the range of feature subset sizes and the accuracy threshold for identifying "bad features".
2. Feature Combination Generation: Iteratively generate combinations of features, starting from subsets of three features and incrementally adding more.
3. Evaluation and Exclusion:
    - Evaluate each feature subset using the SVM classifier.
    - Measure the classification accuracy.
    - If the inclusion of a feature results in an accuracy decrease equal to or greater than the threshold, label it as a "bad feature".
    - Exclude subsets containing "bad features" from further evaluation.
4. Selection of Optimal Features: Identify the feature subset that achieves the highest accuracy without including any "bad features".

4.2.5. Deep Learning Techniques in NSTAP

Building upon the machine learning approaches discussed previously, we extend our investigation into deep learning methods for intrusion detection. Deep learning models, particularly transformer-based architectures like BERT, have demonstrated remarkable performance in handling large-scale data and capturing complex patterns, making them highly suitable for network security applications.

Proposed Approach

In this study, we address the challenge of adapting network logs for classification using a BERT model by transforming these logs into a structured textual format conducive for Byte-Level Byte-Pair Encoding (BBPE) and subsequent BERT processing. Our approach involves several key steps: data preprocessing, tokenizer configuration and training, and fine-tuning the pre-trained BERT model.

Data Preprocessing

The raw network logs from the datasets were initially unsuitable for processing by the BBPE tokenizer and the BERT model due to their non-linguistic nature and the presence of irrelevant or missing values. To address this, we performed several data cleaning steps:

- Removal of Null Values: Entries with missing data were eliminated to ensure the integrity of the dataset.
- Feature Selection: Unnecessary features that do not contribute to intrusion detection were discarded to reduce complexity and improve model efficiency.
- Normalization: Applied normalization techniques to standardize data formats and scales across different features.

To enable the use of natural language processing techniques, we transformed the structured network logs into a textual format resembling natural language. This transformation involved:

- Annotating Data Elements: Each feature-value pair in a log entry was concatenated into a string, with feature names and their corresponding values separated by a delimiter (e.g., '$'), and spaces between pairs. This method preserves the semantic relationship between features and their values.
- Consolidation: All feature-value pairs were merged into a single text attribute for each entry in the dataset, resulting in a unified textual representation suitable for tokenization.

The transformation process ensures the integrity and completeness of the original data while making it compatible with the BBPE tokenizer and the BERT model. Mathematically, the transformation can be represented as:

$$\begin{aligned} M_{\text{text}} &= s(i,j) \\ &= \text{feature}_1 \,\$\, \text{value}(\text{feature}_1) \;+\; \text{feature}_2 \,\$\, \text{value}(\text{feature}_2) \\ &\quad +\; \ldots \;+\; \text{feature}_n \,\$\, \text{value}(\text{feature}_n) \end{aligned}$$

where M_{text} is the transformed textual representation, $s(i,j)$ denotes the string constructed from the i-th sample and j-th feature, and n is the total number of features.

An illustration of this transformation is provided in Figure 5.

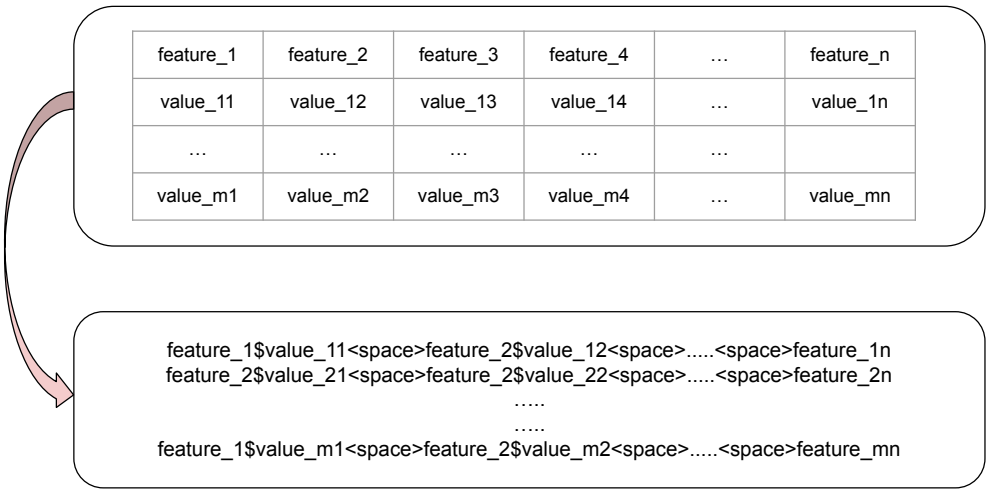


Figure 5. Transformation of Network Logs into Contextual Text Representation [13].

Dataset Standardization

A unified preprocessing approach was implemented to harmonize three distinct datasets: UNSW-NB15 [31], ToN-IoT [59]), and Edge-IIoTset [60]). Despite their heterogeneous attributes and feature sets, each dataset was restructured to conform to a standardized schema with consistent field nomenclature, including `text`, `label`, and `category`. This standardization facilitates model generalization across diverse network traffic patterns while mitigating dataset-specific variations. The integration and standardization of these datasets yielded a consolidated dataset of 3.4 GB, comprising 2,687,917 records, where each record represents a feature vector with its corresponding values, as illustrated in Figure 5. To ensure robust model evaluation, we employed a stratified sampling approach to split the consolidated dataset into training and validation sets, with an 85:15 ratio. The stratification was performed based on the attack type distribution, maintaining the proportional representation of each attack category in both sets. This approach mitigates potential sampling bias and ensures that both training and validation sets contain representative samples of all attack patterns, particularly for less frequent attack types.

Tokenizer Configuration and Training

After transforming the network logs into a textual format, we employed the BBPE tokenizer to encode the data for input into the BERT model. The tokenizer was trained from scratch to effectively capture the unique characteristics of the network log data. The configuration and training involved:

- Vocabulary Size: We experimented with three different vocabulary sizes: 5000; 10,000; and 20,000 tokens. Adjusting the vocabulary size allows us to balance the granularity of token representation with computational efficiency.
- Special Tokens: Specific tokens relevant to the datasets were incorporated to handle special cases and delimiters within the data.
- Minimum Frequency Threshold: A minimum frequency threshold of 2 was set to include tokens in the vocabulary, ensuring that rare tokens do not adversely affect the model performance.

Consistent with the methodology in [61], we explored different training scenarios to assess the impact on the tokenizer's performance:

1. Joint Training on All Datasets: The tokenizer was trained on the combined data from all three datasets to create a unified vocabulary that captures the diversity of the network logs.
2. Individual Training on Each Dataset: The tokenizer was trained separately on each dataset to capture dataset-specific terminology and patterns.

For each scenario, we varied the vocabulary sizes to evaluate their effect on the model's performance. This comparative analysis aimed to determine whether a unified or individual dataset training approach, coupled with different vocabulary sizes, yields superior results.

Fine-Tuning the Pre-Trained BERT Model

We fine-tuned the pre-trained `bert-base-uncased` model for the multi-class classification of network logs. The fine-tuning process involved:

- Model Adaptation: Modified the final classification layer of BERT to match the number of classes in our dataset, as specified in the `label_dict`. This ensures the model outputs are compatible with the classification task.
- Optimization Settings: Employed the AdamW optimizer with a learning rate of 1×10^{-5} and an epsilon of 1×10^{-8} to facilitate efficient training while preventing overfitting.

Prior to training, the tokenized data was prepared.

- Embedding Generation: The BBPE tokenizer was used to convert the textual data into input IDs and attention masks, which are the required inputs for the BERT model.
- Data Batching: The data was organized into batches, each containing input IDs, attention masks, and labels, to enable efficient training.

We fine-tuned the pre-trained `bert-base-uncased` model for multi-class classification across several log datasets to cover various scenarios. For instance, using the Edge-IIoTset dataset—which required the longest training time—we ran the fine-tuning on an NVIDIA server with four Tesla V100-DGXS-32GB GPUs. Over four epochs, the training took approximately 20 hours in total. During the fine-tuning process, the model learned to capture the complex patterns and contextual information within the network logs, enhancing its ability to accurately classify various network threat categories.

Integration with NSTAP

By integrating the BBPE tokenizer and the fine-tuned BERT model into NSTAP, we enhanced the platform's capability to detect sophisticated network intrusions in real-time. The deep learning approach leverages the strengths of transformer models by capturing intricate patterns and contextual relationships within the data, leading to improved detection performance.

The integration process involved deploying the fine-tuned model within NSTAP's real-time data processing pipeline. Incoming network logs are preprocessed and transformed into the required textual format, tokenized using the trained BBPE tokenizer, and then fed into the BERT model for classification. The classification results are used to generate alerts and inform risk scoring, enabling prompt response to detected threats.

5. Results and Discussion

For the machine learning component, we evaluated the proposed approaches detailed in Section 4.2.3, using the UNSW-NB15 dataset as a benchmark. The results, summarized in Figure 6, highlight the effectiveness of our feature selection strategies. In this Figure, the blue bars represent classification accuracy achieved with various feature selection methods, while the orange bars denote the number of feature combinations tested (scaled down by a factor of 10,000 for clarity).

Our SVM-based feature selection algorithm achieved a substantial reduction in the number of evaluated feature combinations while maintaining high classification accuracy, making it highly suitable for real-time intrusion detection systems where computational efficiency is crucial.

Effective feature selection is essential to optimizing machine learning models by enhancing accuracy and reducing dataset dimensionality. Our process began with the Pearson Correlation Coefficient to identify attributes highly correlated with the target variable. Following this, we developed an SVM-based feature reduction algorithm, inspired by [33], which we validated on a smaller dataset. This approach was instrumental in refining the model by focusing on the most relevant features, minimizing unnecessary computations.

To evaluate the model further, we assessed our fine-tuned BERT model on three datasets—UNSW-NB15, TON-IoT, and Edge-IIoT—for multi-class classification tasks. Using validation loss and weighted F1 scores as primary evaluation metrics, we gained comprehensive insights into the model's performance across various training epochs.

Dataset-Specific Performance

- TON-IoT: Exhibited high accuracy, with F1 scores improved progressively across four epochs. The lowest recorded validation loss was 0.0191, reflecting effective model learning and generalization capabilities. A notable observation from the results is the impact of vocabulary size on classification performance. As shown in Figure 7, while all vocabulary sizes led to strong performance, the models trained with larger vocabulary size 20,000 and on all datasets exhibited marginally better F1 scores compared to the 5000-vocabulary model, particularly in later epochs. This suggests that increasing the vocabulary size provides the model with richer token representations, which is particularly advantageous for complex multi-class classification tasks, such as those in the TON-IoT dataset.
- UNSW-NB15: As shown in Figure 7 the tokenizer trained on all datasets with a vocabulary size of 20,000 demonstrates significant improvements in both training and validation metrics over the first four epochs. The F1 score steadily increases, indicating optimal model performance. Both training and validation F1 scores showed clear progression, reflecting the model's ability to generalize effectively with a larger and

more diverse vocabulary. The decrease in validation loss suggests that the model learns meaningful patterns and avoids overfitting during these early epochs. This highlights the importance of using a larger vocabulary size to capture complex network traffic patterns.

- Edge-IIoT: The model achieved near-perfect accuracy across all classes, with an F1 score of 0.99999 and a validation loss as low as 0.0000441 by the third epoch. Even for minority classes like Ransomware (97% accuracy) and MITM (99% accuracy), the model maintained strong performance, demonstrating its robustness in handling class imbalance. Despite potential concerns of overfitting due to the high accuracy and low validation loss, future work will test the generalizability of the fine-tuned model by applying it to other datasets to ensure its broader applicability. This approach will help confirm whether the model overfits to the EDGE-IIoT dataset or generalizes effectively to unseen data from different network environments.

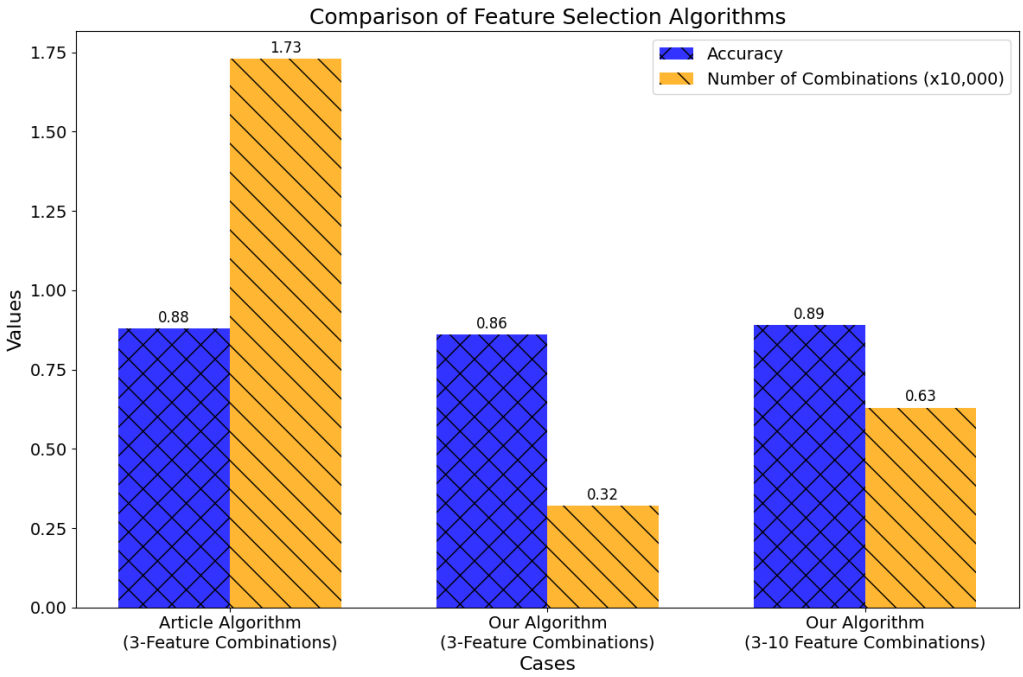


Figure 6. Comparison of Feature Selection Algorithms. Reprinted with permission from "Design and Implementation of an Automated Network Traffic Analysis System using Elastic Stack" [8], IEEE Proceedings, 2023. © 2023 IEEE.

Figure 7 highlights the class-wise F1 scores across TON-IoT, UNSW-NB15, and Edge-IIoT datasets, comparing tokenizers trained individually (Single) versus jointly (All). For UNSW-NB15, the combined tokenizer utilized a 20,000-word vocabulary and 4 training epochs, contrasting with the 5000-word vocabulary when trained exclusively on UNSW-NB15. Despite this expanded vocabulary and additional training, specific attack classes like DoS and Backdoor in UNSW-NB15 continue to show suboptimal performance, indicating persistent dataset-specific challenges. Conversely, attack classes with substantial representation such as Ransomware and Injection demonstrate robust performance consistently across all datasets, suggesting these classes particularly benefit from the broader vocabulary and multi-dataset training approach. Misconception #1: Machine learning is better than conventional analytical or statistical methods. Although machine learning can be a highly effective tool, it may not be suitable for use across every problem space. Other analytics or statistical methods may produce highly accurate and effective results or may be less

resource-intensive than a machine learning approach, and be the more suited approach for a given problem space.

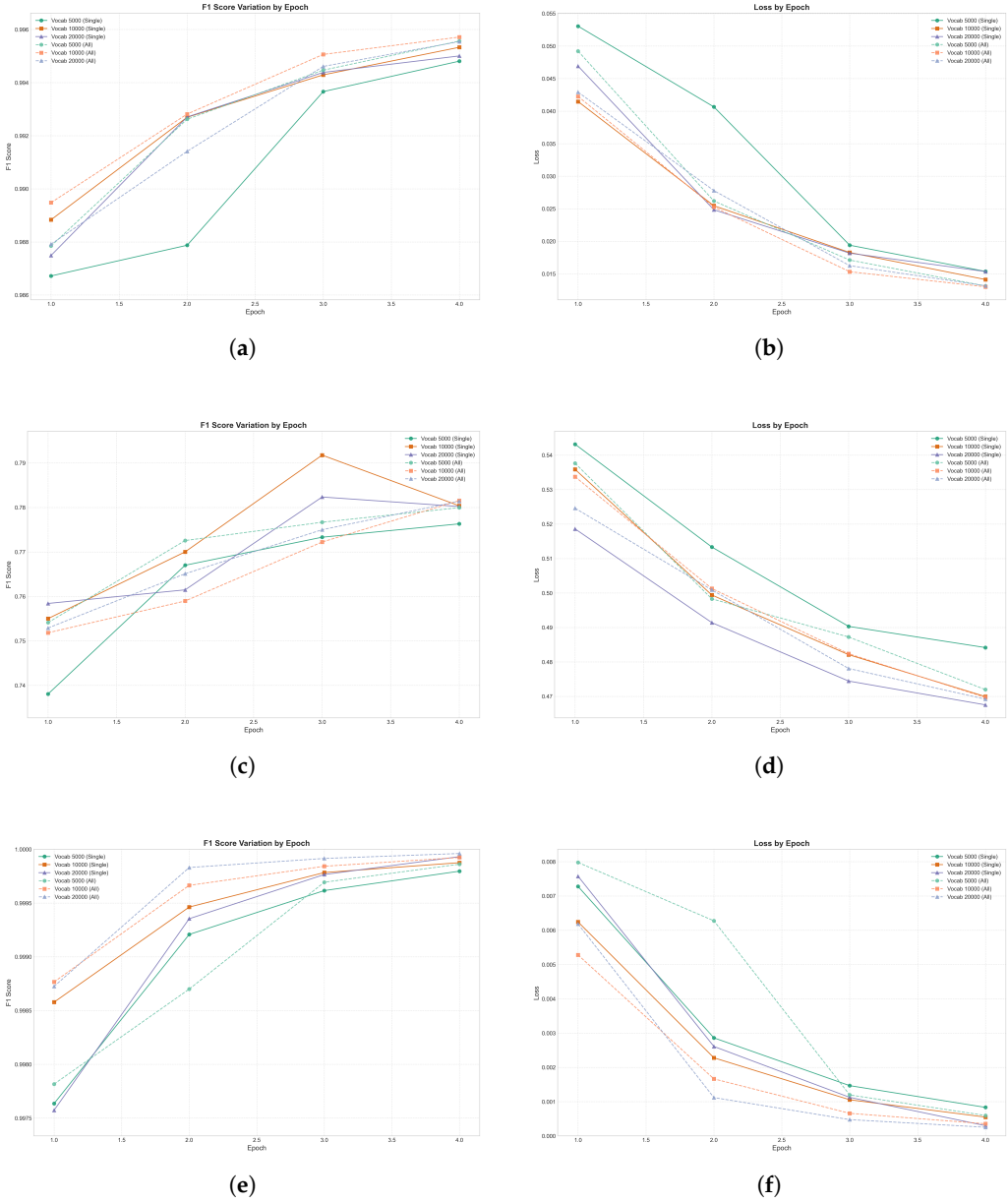


Figure 7. Comparative analysis of training performance metrics across three distinct network security datasets: ToN-IoT (**top row**), UNSW-NB15 (**middle row**), and Edge-IIoT (**bottom row**). The left column illustrates F1 score progression, while the right column depicts the corresponding loss function minimization. (**a**) F1 Score Variation (ToN-IoT Dataset). (**b**) Training Loss Progression (ToN-IoT Dataset). (**c**) F1 Score Variation (UNSW-NB15 Dataset). (**d**) Training Loss Progression (UNSW-NB15 Dataset). (**e**) F1 Score Variation (Edge-IIoT Dataset). (**f**) Training Loss Progression (Edge-IIoT Dataset).

6. Conclusions and Future Work

In this research, we introduced the Network Security Traffic Analysis Platform (NSTAP), designed to address the demands of real-time monitoring and intrusion detection within large-scale, complex network environments. NSTAP's architecture—integrating Elastic Stack, Zeek, Osquery, Kafka, and GeoLocation data—offers robust and scalable solutions tailored for comprehensive security analytics. By unifying disparate network

and IoT datasets, we improved data quality and volume for anomaly detection, thereby enhancing model performance across multiple attack categories.

A critical aspect of this work involved leveraging Large Language Models (LLMs) and supervised machine learning techniques to address network security challenges. Our findings demonstrate the importance of effective data transformation and diverse feature integration for accurate detection in varied scenarios.

The platform's architectural design emphasizes scalability and usability through containerized services, facilitating potential deployment in industrial settings where real-time analysis and traffic visualization are crucial. NSTAP's integration of advanced visualization tools, real-time filters, and stream processing capabilities provides a foundation for network administrators to gain actionable insights and respond promptly to security incidents.

While our experimental evaluation demonstrates promising results using benchmark datasets, the transition to operational environments presents both opportunities and challenges that warrant further investigation. The theoretical framework established in this research suggests that NSTAP's architecture could accommodate live network traffic analysis, though additional research is needed to validate this hypothesis in production environments.

In future work, we plan to address several fundamental challenges to enhance NSTAP's practical applicability. First, we will focus on developing theoretical models for resource allocation and infrastructure requirements, establishing baseline parameters for enterprise-scale deployments. Second, we aim to investigate the theoretical implications of deploying LLMs in operational settings, particularly focusing on the trade-offs between computational overhead, latency requirements, and detection accuracy. Furthermore, we intend to explore the integration of Retrieval Augmented Generation (RAG) techniques, hypothesizing that the combination of external knowledge retrieval with generative capabilities could enhance both interpretability and detection accuracy for sophisticated attack patterns.

Future research directions will also include the automation and refinement of system components to improve efficiency across diverse datasets, positioning NSTAP as a foundation for robust security solutions in dynamic, large-scale corporate environments. Additionally, we will explore the theoretical foundations for integrating advanced deep learning and transfer learning models, with the objective of optimizing detection accuracy while minimizing computational overhead. These investigations will contribute to the broader understanding of AI-driven network security systems and their potential real-world applications.

Author Contributions: A.B. and A.L.; methodology, M.M.; software, Z.M. and M.M.; validation, Z.M., M.M. and A.B.; formal analysis, Z.M.; investigation, Z.M. and M.M.; resources, A.B.; data curation, Z.M.; writing—original draft preparation, Z.M. and M.M.; writing—review and editing, A.B. and A.L.; visualization, Z.M.; supervision, A.B. and A.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in public repositories. The UNSW-NB15 dataset can be accessed at https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 9 December 2024), the TON-IoT dataset at https://research.unsw.edu.au/projects/toniot-datasets (accessed on 9 December 2024), and the Edge-IIoT dataset at https://doi.org/10.21227/mbc1-1h68.

Conflicts of Interest: Author Mheni Merzouki was employed by the company Prometheus Computing. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NSTAP	Network Security Traffic Analysis Platform
IDS	Intrusion Detection Systems
NIDS	Network-based Intrusion Detection System
HIDS	Host-based Intrusion Detection System
IPS	Intrusion Prevention Systems
AI	Artificial Intelligence
LLMs	Large Language Models
NLP	Natural Language Processing
DL	Deep Learning
ML	Machine Learning
RF	Random Forest
DT	Decision Tree
SVM	Support Vector Machine
BERT	Bidirectional encoder representations from transformers
BBPE	Byte-Level Byte-Pair Encoding
IoT	Internet of Things
GOOSE	Generic Object Oriented Substation Event

References

1. Goldman, Z.K.; McCoy, D. Deterring financially motivated cybercrime. *J. Natl. Secur. Law Policy* **2015**, *8*, 595.
2. Global Networking Trends. Available online: https://www.cisco.com/c/en/us/solutions/enterprise-networks/global-networking-trends.html (accessed on 23 January 2025).
3. Red Hat-Open Source Solutions. Available online: https://www.redhat.com/en (accessed on 23 January 2025).
4. Technology Trends 2024. Available online: https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/technology-trends-2024 (accessed on 23 January 2025).
5. Jin, D.; Lu, Y.; Qin, J.; Cheng, Z.; Mao, Z. SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism. *Comput. Secur.* **2020**, *97*, 101984. [CrossRef]
6. Maasaoui, Z.; Hathah, A.; Bilil, H.; Mai, V.S.; Battou, A.; Lbath, A. Network Security Traffic Analysis Platform—Design and Validation. In Proceedings of the 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA), Abu Dhabi, United Arab Emirates, 5–8 December 2022.
7. Mehmood, Y.; Shibli, M.A.; Habiba, U.; Masood, R. Intrusion Detection System in Cloud Computing: Challenges and opportunities. In Proceedings of the 2013 2nd National Conference on Information Assurance (NCIA), Rawalpindi, Pakistan, 11–12 December 2013; pp. 59–66. [CrossRef]
8. Maasaoui, Z.; Merzouki, M.; Bekri, A.; Abane, A.; Battou, A.; Lbath, A. Design and Implementation of an Automated Network Traffic Analysis System using Elastic Stack. In Proceedings of the 2023 20th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA), Giza, Egypt, 4–7 December 2023; pp. 1–6. [CrossRef]
9. Denning, D.E. An Intrusion-Detection Model. *IEEE Trans. Softw. Eng.* **1987**, *SE-13*, 222–232. [CrossRef]
10. Liu, C.; Zhang, Y. An Intrusion Detection Model Combining Signature-Based Recognition and Two-Round Immune-Based Recognition. In Proceedings of the 2021 17th International Conference on Computational Intelligence and Security (CIS), Chengdu, China, 19–22 November 2021; pp. 497–501. [CrossRef]
11. Almseidin, M.; Alzubi, M.; Kovacs, S.; Alkasassbeh, M. Evaluation of machine learning algorithms for intrusion detection system. In Proceedings of the 2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 14–16 September 2017; pp. 277–282. [CrossRef]
12. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]

13. Maasaoui, Z.; Merzouki, M.; Battou, A.; Lbath, A. Anomaly Based Intrusion Detection using Large Language Models. In Proceedings of the 2024 IEEE/ACS 21st International Conference on Computer Systems and Applications (AICCSA), Sousse, Tunisia, 22–26 October 2024; pp. 1–8.
14. Gharib, A.; Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. An Evaluation Framework for Intrusion Detection Dataset. In Proceedings of the 2016 International Conference on Information Science and Security (ICISS), Pattaya, Thailand, 19–22 December 2016; pp. 1–6. [CrossRef]
15. Abbasi, M.; Shahraki, A.; Taherkordi, A. Deep learning for network traffic monitoring and analysis (NTMA): A survey. *Comput. Commun.* **2021**, *170*, 19–41. [CrossRef]
16. D'Alconzo, A.; Drago, I.; Morichetta, A.; Mellia, M.; Casas, P. A Survey on Big Data for Network Traffic Monitoring and Analysis. *IEEE Trans. Netw. Serv. Manag.* **2019**, *16*, 800–813. [CrossRef]
17. Easterly, J.; Fanning, T. The Attack on Colonial Pipeline: What We've Learned & What We've Done Over the Past Two Years. 2023. Available online: https://www.cisa.gov/news-events/news/attack-colonial-pipeline-what-weve-learned-what-weve-done-over-past-two-years (accessed on 1 September 2023).
18. Ahmad, R.; Alsmadi, I. Machine learning approaches to IoT security: A systematic literature review. *Internet Things* **2021**, *14*, 100365. [CrossRef]
19. Abdullahi, M.; Baashar, Y.; Alhussian, H.; Alwadain, A.; Aziz, N.; Capretz, L.F.; Abdulkadir, S.J. Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review. *Electronics* **2022**, *11*, 198. [CrossRef]
20. Vaswani, A. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
21. Devlin, J. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
22. Elastic Stack. The Leading Platform for Search-Powered Solutions. Available online: https://www.elastic.co/elastic-stack (accessed on 3 October 2024).
23. Zeek-Network Security Monitoring. Available online: https://zeek.org/ (accessed on 3 October 2024).
24. Osquery-SQL Powered Operating System Instrumentation. Available online: https://www.osquery.io/ (accessed on 3 October 2024).
25. Apache Kafka. A Distributed Event Streaming Platform. Available online: https://kafka.apache.org/ (accessed on 3 October 2024).
26. Borkar, A.; Donode, A.; Kumari, A. A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS). In Proceedings of the 2017 International Conference on Inventive Computing and Informatics (ICICI), Coimbatore, India, 23–24 November 2017; pp. 949–953. [CrossRef]
27. Gkountis, C.; Taha, M.; Lloret, J.; Kambourakis, G. Lightweight algorithm for protecting SDN controller against DDoS attacks. In Proceedings of the 2017 10th IFIP Wireless and Mobile Networking Conference (WMNC), Valencia, Spain, 25–27 September 2017; pp. 1–6. [CrossRef]
28. Wu, H.; Schwab, S.; Peckham, R.L. Signature Based Network Intrusion Detection System and Method. U.S. Patent 7,424,744, 9 September 2008.
29. Garcia-Teodoro, P.; Diaz-Verdejo, J.; Maciá-Fernández, G.; Vázquez, E. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Comput. Secur.* **2009**, *28*, 18–28. [CrossRef]
30. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 581–585. [CrossRef]
31. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
32. Patgiri, R.; Varshney, U.; Akutota, T.; Kunde, R. An investigation on intrusion detection system using machine learning. In Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence (SSCI), Bangalore, India, 18–21 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1684–1691.
33. Chowdhury, M.N.; Ferens, K.; Ferens, M. Network Intrusion Detection Using Machine Learning. In Proceedings of the International Conference on Security and Management (SAM), Las Vegas, NV, USA, 25–28 July 2016; p. 30.
34. Zang, M.; Yan, Y. Machine Learning-Based Intrusion Detection System for Big Data Analytics in VANET. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021; pp. 1–5. [CrossRef]
35. Moustafa, N.; Slay, J. The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set. *Inf. Secur. J. Glob. Perspect.* **2016**, *25*, 18–31. [CrossRef]
36. Stoleriu, R.; Puncioiu, A.; Bica, I. Cyber Attacks Detection Using Open Source ELK Stack. In Proceedings of the 2021 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 1–3 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.

37. Chen, S.; Liao, H. BERT-Log: Anomaly Detection for System Logs Based on Pre-Trained Language Model. *Appl. Artif. Intell.* **2022**, *36*, 2145642. [CrossRef]
38. Le, V.-H.; Zhang, H. Log-Based Anomaly Detection without Log Parsing. In Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, Australia, 15–19 November 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 492–504.
39. Guo, H.; Yuan, S.; Wu, X. LogBERT: Log Anomaly Detection via BERT. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
40. Bertoli, G.D.C.; Júnior, L.A.P.; Saotome, O.; Dos Santos, A.L.; Verri, F.A.N.; Marcondes, C.A.C.; Barbieri, S.; Rodrigues, M.S.; De Oliveira, J.M.P. An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access* **2021**, *9*, 106790–106805. [CrossRef]
41. Douiba, M.; Benkirane, S.; Guezzaz, A.; Azrour, M. An improved anomaly detection model for IoT security using decision tree and gradient boosting. *J. Supercomput.* **2023**, *79*, 3392–3411. [CrossRef]
42. Rahali, A.; Akhloufi, M.A. Malbert: Malware detection using bidirectional encoder representations from transformers. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 3226–3231.
43. Lee, W.; Stolfo, S.J. A framework for constructing features and models for intrusion detection systems. *ACM Trans. Inf. Syst. Secur. TiSSEC* **2000**, *3*, 227–261. [CrossRef]
44. Ashfaq, R.A.R.; Wang, X.-Z.; Huang, J.Z.; Abbas, H.; He, Y.-L. Fuzziness based semi-supervised learning approach for intrusion detection system. *Inf. Sci.* **2017**, *378*, 484–497. [CrossRef]
45. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. *arXiv* **2018**, arXiv:1802.09089.
46. Laghrissi, F.; Douzi, S.; Douzi, K.; Hssina, B. Intrusion detection systems using long short-term memory (LSTM). *J. Big Data* **2021**, *8*, 65. [CrossRef]
47. Bobade, S.Y.; Apare, R.S.; Borhade, R.H.; Mahalle, P.N. Intelligent detection framework for IoT-botnet detection: DBN-RNN with improved feature set. *J. Inf. Secur. Appl.* **2025**, *89*, 103961. [CrossRef]
48. RabbitMQ. Messaging That Just Works. Available online: https://www.rabbitmq.com/ (accessed on 23 January 2025).
49. Apache ActiveMQ. Message Broker. Available online: https://activemq.apache.org/ (accessed on 23 January 2025).
50. Snort. The Open Source Network Intrusion Detection System. Available online: https://www.snort.org/ (accessed on 23 January 2025).
51. Suricata. The Open Source Network Threat Detection Engine. Available online: https://suricata.io/ (accessed on 23 January 2025).
52. Sysmon. System Monitor. Available online: https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (accessed on 23 January 2025).
53. auditd. The Linux Auditing System. Available online: https://man7.org/linux/man-pages/man8/auditd.8.html (accessed on 23 January 2025).
54. Biswas, P.P.; Tan, H.C.; Zhu, Q.; Li, Y.; Mashima, D.; Chen, B. A Synthesized Dataset for Cybersecurity Study of IEC 61850 Based Substation. In Proceedings of the 2019 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm), Beijing, China, 21–24 October 2019; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019. [CrossRef]
55. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
56. Decision Tree Algorithm. Available online: https://www.towardsanalytic.com/decision-tree-algorithm/ (accessed on 3 October 2024).
57. Restack. Decision Making Models: Answer Decision Trees vs. Random Forests. Restack, 2025. Available online: https://www.restack.io/p/decision-making-models-answer-decision-trees-vs-random-forests-cat-ai (accessed on 3 October 2024)
58. Towards Data Science, Random Forest Explained: A Visual Guide with Code Examples. Medium-Towards Data Science, 2025. Available online: https://medium.com/towards-data-science/random-forest-explained-a-visual-guide-with-code-examples-9f736a6e1b3c (accessed on 2 April 2025).
59. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Den Hartog, F.T. ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. *IEEE Internet Things J.* **2022**, *9*, 485–496. [CrossRef]

60. Ferrag, M.A.; Friha, O.; Hamouda, D.; Maglaras, L.; Janicke, H. Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications: Centralized and Federated Learning. *IEEE Dataport* **2022**. [CrossRef]
61. Ferrag, M.A.; Ndhlovu, M.; Tihanyi, N.; Cordeiro, L.C.; Debbah, M.; Lestable, T. Revolutionizing Cyber Threat Detection with Large Language Models. *arXiv* **2023**, arXiv:2306.14263.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.