SMARTSHIELD: AI-Driven Solution for Automated Cybersecurity Incident Response

Kacem Mathlouthi
INSAT
Tunis, Tunisia
kacem.mathlouthi@ieee.org

Youssef Abid
INSAT
Tunis, Tunisia
youssef.abid.insat@ieee.org

Mohamed Amine Haouas
INSAT
Tunis, Tunisia
med.amine.haouas@ieee.org

Mohamed Masmoudi
INSAT
Tunis, Tunisia
mohamedmasmoudi745@ieee.org

Iyed Mdimegh
INSAT
Tunis, Tunisia
iyed.mdimegh@ieee.org

Raed Addala
INSAT
Tunis, Tunisia
raed.addala@ieee.org

Fayez Zouari
INSAT
Tunis, Tunisia
fayez.zouari@ieee.org

Achraf Benammar
INSAT
Tunis, Tunisia
achraf.benammar@ieee.org

***Abstract*—The increasing frequency and sophistication of cyber-attacks have heightened the need for automated, scalable, and transparent cybersecurity solutions. Traditional threat detection and response mechanisms, often reliant on rule-based systems, are inadequate in dealing with complex, real-time threats. This paper introduces SMARTSHIELD, an AI-driven cybersecurity incident response platform designed to automate threat detection, classification, and reporting. Leveraging machine learning, Explainable AI (XAI), and real-time data pipelines, This platform provides a comprehensive solution for network intrusion detection and rapid response. The system integrates multiple components, including custom log extraction, advanced anomaly detection, and agent-based report generation, offering an efficient, scalable, and interpretable approach to cybersecurity incident management. Our solution aims to enhance situational awareness for cybersecurity teams, improve response times, and ensure robust threat mitigation in dynamic network environments.**

***Index Terms*—cybersecurity, incident response, automation, artificial intelligence, machine learning, natural language processing, explainable AI, RabbitMQ, Pipelines, anomaly detection, threat classification, network intrusion, real-time processing, agentic workflow, catboost**

I. Introduction

The escalating complexity and frequency of cyber-attacks present critical security challenges across industries such as finance, healthcare, and government [1], [2]. Traditional, rule-based defenses struggle to keep pace with sophisticated threats, limiting organizations' ability to respond swiftly to breaches [3].

Advancements in artificial intelligence and machine learning present new opportunities to enhance cybersecurity by enabling anomaly detection across large-scale organizational data [4], [5]. However, these solutions often face limitations in scalability, latency, and interpretability—particularly in real-time enterprise settings where immediate response is crucial [6].

Research in Explainable AI (XAI) addresses the "black box" issue of ML in cyber-security, aiming to provide transparency in model predictions [7], [8]. Yet, integrating XAI for real-time, actionable insights remains challenging, as current implementations often lack scalability and effective integration into live threat-response frameworks [9].

Despite significant progress, several critical gaps remain in existing cyber-security platforms:

- **Scalability and Latency**: High latency in processing large data streams slows threat detection and response [10].
- **Explainability and Transparency**: Black-box models limit analysts' understanding and trust in AI-based detections, reducing decision-making effectiveness [11].
- **Real-Time Reporting Integration**: While many platforms offer real-time detection, few provide fully unified detection, and reporting systems [12], which can delay the delivery of insights necessary for threat response.

To address these challenges, this paper introduces *SMARTSHIELD*, an AI-powered cyber-security incident response platform designed to automate the detection, classification, and reporting of network intrusions. SMARTSHIELD encompasses multiple stages, from log extraction and data preprocessing to advanced ML-driven threat analysis and agentic report generation. The platform processes network-related logs collected through dedicated log extractors, which then pass through a message-oriented pipeline, facilitating high-throughput data handling and low-latency processing.

The key contributions of this paper are as follows:

1) **End-to-End Cyber-Security Platform**: An integrated platform that covers log extraction, ML-based threat analysis, agentic report generation, and data visualization.
2) **Multi-Stage Machine Learning Pipeline**: A dual-model approach utilizing anomaly detection followed by multi-class intrusion classification, designed to maximize threat detection accuracy and efficiency.
3) **Agentic Reporting Workflow**: A report generation system designed as the explainable AI that clarifies model outputs and decisions. It offers transparent, actionable

recommendations, improving analysts' comprehension and enabling swift responses to identified threats.

4) **Data Visualization Interface**: A front-end that presents real-time log data, threat classifications, and reports, improving security team situational awareness and decision-making.

The proposed platform is evaluated based on its ability to detect threats with minimal latency, provide transparent reporting through explainable AI, and offer a scalable infrastructure that supports continuous data handling and analysis. The remainder of this paper is organized as follows: Section II details the proposed platform architecture and the design choices; Section III describes the materials and datasets used in the system; Section IV outlines the methodology and implementation; Section V presents the Deployment Architecture; Section VI presents the evaluation and results; and Section VII concludes with potential future directions.

II. System Architecture

A. Architecture Considerations

The development of an effective threat detection system requires careful consideration of multiple factors including log analysis strategies, data sources, and architectural design choices. These considerations fundamentally shaped our implementation approach.

1) Log Analysis Strategy: Recent research on vulnerability and security log analysis [13] reveals that machine learning has emerged as the primary technique in academic log taxonomy and threat detection approaches. However, examination of widely-adopted Security Information and Event Management (SIEM) tools such as Wazuh[1], Suricata[2], and TheHive[3] indicates a notable absence of machine learning implementations, despite the structured and consistent nature of log data making it particularly suitable for ML applications.

A key consideration in applying machine learning to security is result interpretability [14]. Our hybrid approach combines elements of both, offering explicit reasoning to improve decision transparency—essential in security contexts where false positives require careful investigation and model decisions must be verifiable.

2) Choice of Log Sources: In distributed environments, the selection of appropriate log sources significantly impacts detection capabilities. Analysis of existing solutions indicates a predominant focus on network logs as the primary data source for threat detection. This preference is fundamentally rooted in the network's position as the necessary transit point for external threats and cyber attacks targeting distributed systems. Network logs provide comprehensive coverage of potential attack vectors, as demonstrated by successful implementations such as entropy-based DDoS detection systems [15].

3) Implementation Architecture: The implementation architecture design process evaluated several approaches to log analysis integration. The first considered approach involved a per-machine-based solution, similar to traditional antivirus systems such as Sophos XDR[4]. However, this approach necessitates additional protection mechanisms beyond the scope of our objectives and introduces vulnerabilities to local tampering and log poisoning [16].

Cloud-based detection systems were also evaluated, where logs are transmitted for centralized analysis. Recent research [17] highlights significant challenges with this approach, including increased attack surfaces, dependency issues, and latency concerns. Additionally, achieving robust performance in cloud environments often requires resources beyond reasonable budget constraints.

These considerations led to our implementation of a pseudo-spectator detection system that is not directly involved within the system it's surveilling, but rather monitoring its status in an independent infrastructure. This makes the solution less prone to attacks and more reliable and easily integrated into most systems.

B. System Architecture Overview

The SMARTSHIELD platform employs a modular, three-tier architecture comprising Network, Pipeline, and Backend layers, each designed to handle specific aspects of cybersecurity incident detection and response as shown in Figure 1

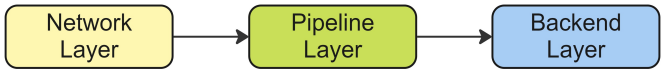

Fig. 1. System Architecture Overview

C. Network Layer

The network infrastructure usually consists of a heterogeneous environment of servers that can run different operating systems and machines. This enables our system to comprehensively monitor various system types and enhances the platform's capability to detect threats across different operating environments. Each server is configured to capture network packets, which are then forwarded to the pipeline component for processing as shown in Figure 2.

D. Pipeline Layer

The pipeline layer serves as the central nervous system of SMARTSHIELD, implementing a sophisticated data processing workflow:

- **RabbitMQ Integration**: Network packet captures are initially queued through a RabbitMQ broker, which provides reliable message queuing and ensures no packet data is lost during processing.
- **Dual Processing Path**: The pipeline implements two parallel processing paths:

[1]https://wazuh.com
[2]https://suricata.io
[3]https://github.com/TheHive-Project/TheHive
[4]https://www.sophos.com/en-us/products/extended-detection-and-response

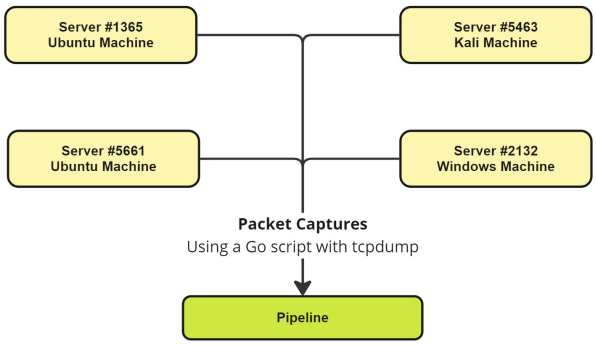

Fig. 2. Network Layer Overview

  - **Packet Capture (PCAP) Processing**: Captured packets are processed through a dedicated PCAP files processor, which prepares the data for classification.
  - **Packet Analysis**: A separate packet analyzer performs deep packet inspection and feature extraction.
- **Data Classification**: Processed data is passed through a classification module that interfaces with the ML model pipeline via a FastAPI endpoint, enabling real-time threat detection and classification.

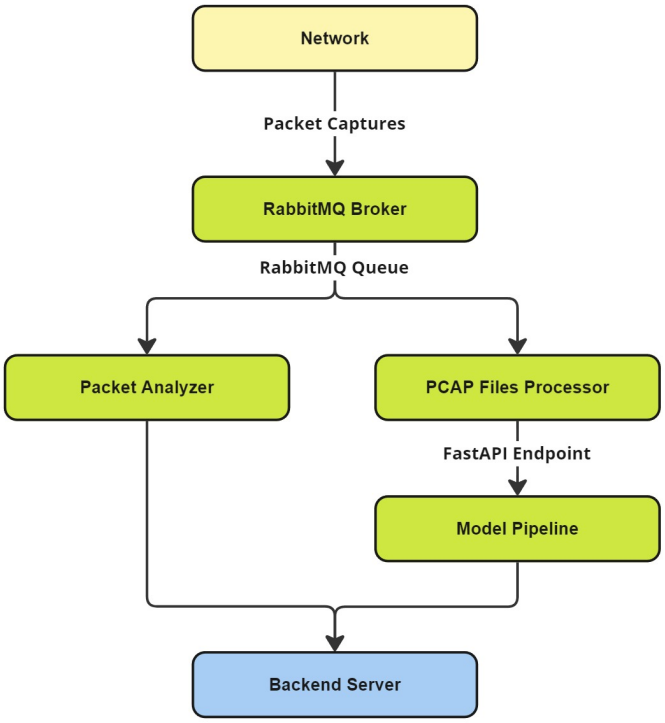

Fig. 3. Pipeline Layer Overview

E. Backend Layer

The backend architecture is built on modern web technologies and consists of several key components:

- **NestJS Endpoints**: The system utilizes NestJS as the primary backend framework, providing RESTful API endpoints for communication between components.
- **PostgreSQL Database**: A PostgreSQL database is the persistent storage solution, maintaining historical data of network activities, detected threats, and generated reports.
- **Report Generator**: This component synthesizes threat detection results and Network logs into comprehensive security reports.
- **Dashboard**: A React-based frontend dashboard implements two complementary interfaces: a real-time network monitor for immediate threat detection and response, and an analytics overview providing broader insights into system security patterns and trends. The dashboard receives data directly from the NestJS endpoints and processed reports from the Report Generator.

This three-tier architecture ensures efficient data flow from network monitoring through analysis to visualization while maintaining modularity and scalability. The use of message queuing (RabbitMQ) and modern web technologies (NestJS, React) enables real-time processing and responsive user interaction, while the PostgreSQL database provides robust data persistence and retrieval capabilities. The system's modular design allows for independent scaling of components and easy integration of new features or additional security tools. The separation of concerns between network monitoring, data processing, and presentation layers ensures robust operation and maintainable code structure.

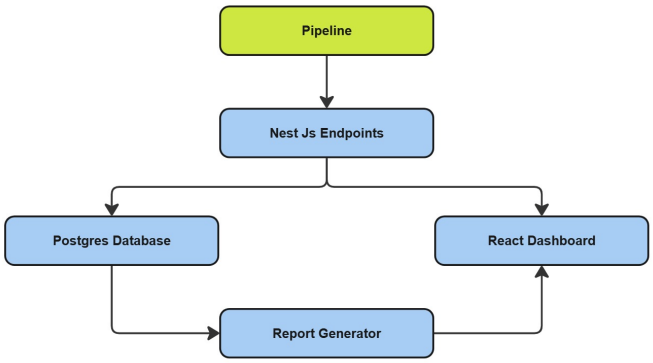

Fig. 4. Backend Overview

III. Materials

In our research into network intrusion detection systems, we conducted a thorough evaluation of available datasets. While several options exist in the field, two datasets emerged as leading candidates due to their comprehensive nature and recent relevance: the UNSW-NB15 dataset and the CIC-IDS 2017 dataset. After extensive evaluation of both, we selected UNSW-NB15 as our primary dataset due to its superior data quality, validated labeling strategy, and robust representation of real-world network conditions. Our evaluation of the CIC-IDS 2017 dataset revealed several limitations that made it

less suitable for our specific research goals, despite its initial promising results.

A. Primary Dataset: UNSW-NB15

The UNSW-NB15 [18] dataset, developed by the Australian Centre for Cyber Security (ACCS), consists of over 2.5 million records and serves as a comprehensive benchmark for intrusion detection, addressing limitations of earlier datasets like KDD99 [19] and NSL-KDD [20]. This dataset is highly imbalanced, with attack instances comprising only about 5% of the total data. Training a model directly on this full dataset could lead to poor performance, as the model might struggle to recognize the minority class (attacks). To manage this, we used the pre-split training and testing files provided by the dataset's creators. These files were specifically designed to support intrusion detection model development, ensuring a representative distribution of attack and benign instances across both sets.

The UNSW-NB15 dataset includes 47 features, categorized into various types that capture distinct aspects of network traffic. These categories are as follows:

- **Flow Features**: Characteristics related to network flow, which help in understanding the nature of data flow.
- **Basic Features**: Basic attributes, which provide foundational information on each network connection such as *state*, *source to destination bytes (sbytes)*, *Source packets retransmitted or dropped (sloss)*.
- **Content Features**: Features derived from packet payload content, useful for detecting suspicious content patterns.
- **Time Features**: Time-related characteristics, which are critical for identifying time-based anomalies and patterns in network behavior.
- **Additional Features**: Includes general-purpose attributes and connection-specific details, offering extra insights into connection dynamics and overall traffic characteristics.

The dataset comprises two main categories: normal network traffic and malicious activities. The malicious traffic is further categorized into nine distinct attack families based on their characteristics and attack patterns as shown in Table I:

TABLE I
CATEGORY DATA AND DESCRIPTIONS

Category	Description
Normal	Natural transaction data.
Generic	Block-cipher attack technique, structure-independent.
Exploits	Attack on known software vulnerabilities.
Fuzzers	Random data input to disrupt programs.
DoS	Attempts to disrupt network/service availability.
Reconnaissance	Information-gathering attacks on targets.
Analysis	Includes scans, spam, and HTML vulnerabilities.
Backdoors	Hidden access bypassing security controls.
Shellcode	Code payload to exploit software flaws.
Worms	Self-replicating code spreading across systems.

The dataset includes two predefined subsets:

- **Training Set**: Contains **175,341 records** (with approximately 68% malicious and 32% benign labels).
- **Testing Set**: Contains **82,332 records** (with approximately 55% malicious and 45% benign labels).

Figure 5 shows the distribution of attack categories within the UNSW-NB15 dataset across both the training and testing sets.

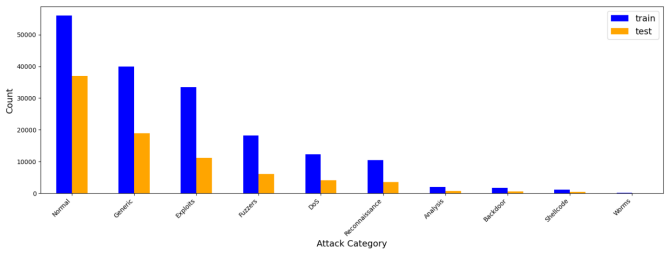

Fig. 5. Attack Category Value Count (Test vs. Train)

B. Alternative Dataset Consideration: CIC-IDS 2017

During our dataset selection process, we also extensively evaluated the CIC-IDS 2017 dataset [27]. This dataset initially appeared promising, offering diverse attack categories and features extracted through CICFlowMeter [28]. However, several critical limitations emerged during our thorough evaluation:

1) **Data Quality Issues**
   - Significant simulation errors were found, particularly in specific attack types like DoS Hulk, compromising their real-world representativeness.
   - Over 25% of flows lack identifying characteristics, with some attack categories containing up to 50% artifacts, introducing substantial noise.
2) **Labeling Deficiencies**
   - The dataset employs an overly generalized and unvalidated labeling strategy, resulting in numerous mislabeled flows that can adversely affect model training.
3) **Environmental Limitations**
   - Data generation in a controlled laboratory environment over a brief period inadequately captures real-world network traffic complexity and variability [26].

Our subsequent real-world testing using CICFlowMeter with simulated attacks revealed severe underperformance in detection and classification. Further validation against the CIC-IDS 2018 dataset [29] confirmed this poor generalization. These findings led us to select UNSW-NB15 as our primary dataset, as it better addresses these limitations and provides a more robust foundation for practical intrusion detection research.

IV. METHODOLOGY AND IMPLEMENTATION

The methodology of SMARTSHIELD begins with *Log Collection and Extraction*, where network logs are gathered, processed, and filtered through custom extractors to produce datasets optimized for anomaly detection and threat classification. This is followed by *Machine Learning and Pipelines*,

utilizing a dual-stage machine learning pipeline to identify anomalies and classify threats. Next, the *Agentic Workflow for Report Generation* engages specialized agents to generate structured incident reports detailing threat analysis, impact assessment, and recommended mitigations. All are built on a scalable Infrastructure and Backend system that ensures seamless data flow, storage, and component interaction. The *Data Visualization* module provides an interactive front-end for displaying insights, detection metrics, and response recommendations to cybersecurity analysts.

A. Log Collection and Extraction

The SMARTSHIELD platform's custom log extractors are specifically designed to capture critical network-related data for real-time threat detection. This section provides a detailed overview of the data collection and feature extraction mechanisms, which are essential for the timely detection and accurate classification of network anomalies.

To support real-time threat detection, a custom Go script, wrapped in an executable, utilizes `tcpdump` to capture network traffic from each server. Each data capture is tagged with a unique server ID and timestamp to identify the source and timing as shown in Figure 6.

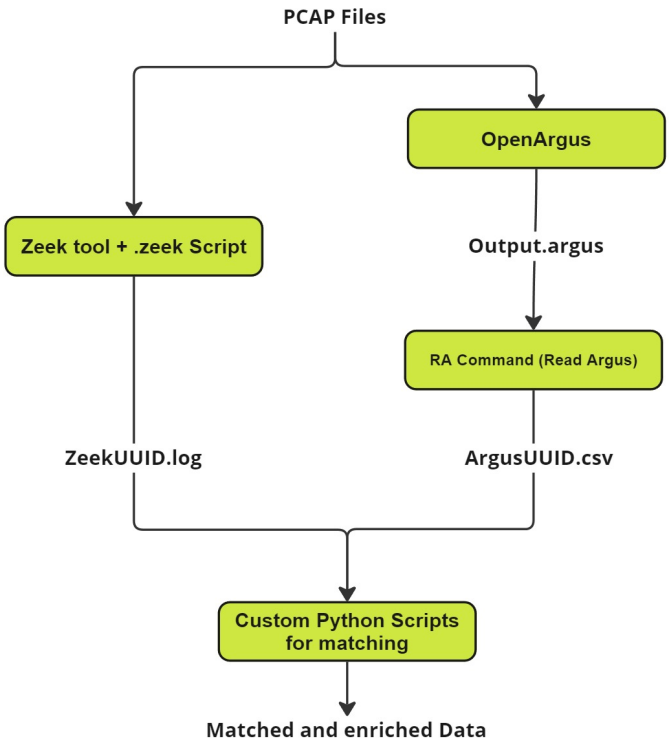

Fig. 6. Log Extraction and Collection Lifecycle

The detection model relies on input in the `UNSW-NB15` dataset format, which necessitates feature extraction methods tailored to match this data structure. To achieve this, SMARTSHIELD leverages two open-source intrusion detection systems (IDS): `openArgus` and `Zeek`, each responsible for extracting critical network features.

- **openArgus (formerly Argus)**: An open-source tool for flow-level network analysis, `openArgus` extracts essential features, including connection duration, protocol types, and packet loss rates, from packet capture (`pcap`) files. Processed files are then parsed by a custom Argus-Client Agent into CSV format, containing relevant features required for model input. as summarized in Table II.

TABLE II
NETWORK CONNECTION FEATURES EXTRACTED USING OPENARGUS

Feature	Description
`dur`	Duration of the connection
`proto`	Network protocol
`state`	Connection state
`spkts`	Source packets
`dpkts`	Destination packets
`sbytes`	Source bytes
`dbytes`	Destination bytes
`rate`	Flow rate
`sload`	Source load
`dload`	Destination load
`sloss`	Source packet loss
`dloss`	Destination packet loss
`sinpkt`	Source inter-packet time
`dinpkt`	Destination inter-packet time
`sjit`	Source jitter
`djit`	Destination jitter
`tcprtt`	TCP connection setup time

- **Zeek (formerly Bro)**: A robust network security monitor, `Zeek` provides event-based data on network activity. Custom scripts written in the Zeek scripting language extract specific features matching the `UNSW-NB15` dataset. Logs generated by Zeek contain the necessary information for the classification model, as shown in Table III.

TABLE III
CONNECTION-LEVEL FEATURES FROM ZEEK, OPENARGUS, AND CUSTOM SCRIPTS

Feature	Tool	Description
`smean`	openArgus	Source mean packet size
`trans_depth`	Zeek	HTTP transaction depth
`ct_src_dport_ltm`	Custom	Src-dest port connection count
`ct_dst_sport_ltm`	Custom	Dest-src port connection count
`is_ftp_login`	Zeek	Detects FTP login
`ct_flw_http_mthd`	Zeek	HTTP method flow count
`is_sm_ips_ports`	Zeek	Src and dest IPs/ports match check

Both tools are optimized for real-time performance, and configurations are further tuned to meet SMARTSHIELD's operational requirements.

Data outputs from `openArgus` and `Zeek` are stored in designated directories, identified by unique IDs associated with the analyzed `pcap` files. Custom scripts further enhance the tool outputs by calculating additional features and consolidating all extracted data into a single CSV file, which is subsequently used by the classification model.

B. Machine learning and Pipelines

1) Data Preprocessing and Feature Engineering: In our model training pipeline, we began by addressing data contamination issues in the UNSW-NB15 dataset [21], as outlined

TABLE IV
FEATURES USED TO MATCH TOOL OUTPUTS

Feature	Description
`src_ip`	Source IP address
`dst_ip`	Destination IP address
`src_port`	Source port
`dst_port`	Destination port
`proto`	Network protocol

by previous research. According to the study, "contaminated" features are those that inadvertently reveal information about the labels, often due to metadata artifacts that can expose attack types or the presence of intrusions. This contamination can lead to overfitting, as the model may learn patterns stemming from dataset construction artifacts rather than authentic network traffic characteristics. To mitigate this issue, we removed seven contaminated features identified in the paper: three *Time to Live Features* (*sttl*, *dttl*, *ct_state_ttl*) and four *Connection Count Features* (*ct_srv_src*, *ct_srv_dst*, *ct_dst_ltm*, *ct_src_ltm*). This adjustment ensures a more realistic evaluation of the model's performance.

In addition to removing contaminated features, we conducted a thorough analysis of feature correlations to identify and address any redundant or highly correlated attributes. We calculated the Pearson correlation coefficient between all pairs of numerical features and removed one feature from any highly correlated pair ($|r| > 0.95$) to minimize multicollinearity. This step helped us streamline the feature set and prevent the model from being overly influenced by correlated variables.

After establishing a baseline model, we analyzed feature importance to identify the most critical attributes for intrusion detection. Based on this insight, we engineered additional features aligned with the top contributing factors. This iterative process of feature engineering, informed by model feedback, allowed us to create a more informative and discriminative set of features for training the final detection and classification models.

Furthermore, we observed significant positive skewness in several numerical features, which could negatively impact model performance. To address this, we applied the $\log 1p$ ($\log(1 + x)$) transformation to these features (see Figure 7). This transformation helped to normalize the distributions and improved the model's ability to learn underlying patterns in the data.

2) Model Selection: Our model selection process began with a comprehensive benchmarking of both traditional Machine Learning algorithms and Deep Learning approaches. Given the relatively small dataset size, we conducted comparative experiments across multiple algorithms to identify the most effective approach, testing each baseline model without hyperparameter tuning and with minimal preprocessing to account for each model's specific requirements.

a) Evaluation Metrics: For evaluating the performance, we focused on several key metrics beyond just overall accuracy, as the dataset exhibited significant class imbalance.

The primary metrics we used were F1-score, precision,

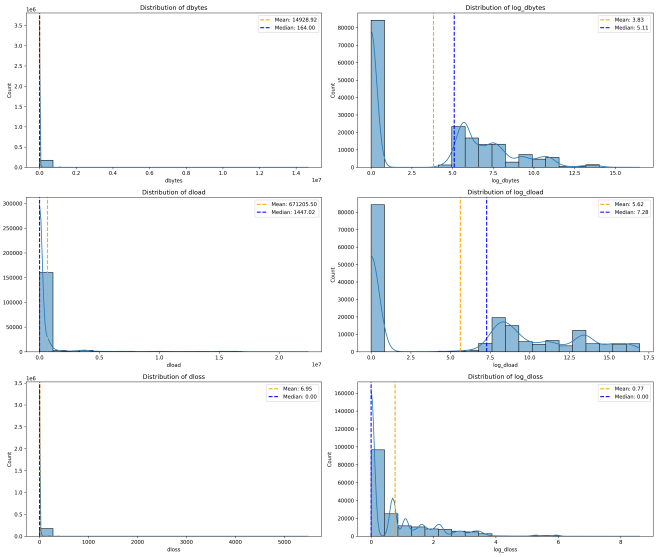

Fig. 7. Normalized distributions of selected numerical features after $\log 1p$ transformation.

and recall. These metrics provide a more nuanced assessment of model performance, as they capture the balance between correctly identifying positive (malicious) instances while minimizing both false positives and false negatives.

Additionally, we calculated the Area Under the Receiver Operating Characteristic (AUC-ROC) curve [22], which is a widely used metric for binary classification tasks. The AUC-ROC curve plots the true positive rate (TPR) against the false positive rate (FPR) at various decision threshold settings. The AUC-ROC score ranges from 0 to 1, with 1 indicating perfect discrimination between the positive and negative classes, and 0.5 representing random guessing.

The AUC-ROC score can be calculated with the following formula:

$$\text{AUC-ROC} = \int \text{TPR}(\text{FPR})\, d\text{FPR} \tag{1}$$

The benchmarking results (Table V) clearly demonstrated that traditional ML approaches, particularly gradient boosting frameworks, significantly outperformed deep learning models for our specific use case. Both XGBoost [23] and CatBoost [24] emerged as top performers, with nearly identical performance metrics (F1 score of 86.88% and 86.84% respectively).

TABLE V
BENCHMARKING RESULTS OF VARIOUS MODELS

Model Name	F1 Score	Precision	Recall	AUC
XGBoost	0.868845	0.884861	0.871520	0.980165
LightGBM	0.858948	0.880255	0.862508	0.980667
CatBoost	**0.868412**	**0.885045**	**0.871168**	**0.980447**
Shallow NN	0.779047	0.840607	0.792292	0.770125
TabNet	0.810352	0.856883	0.819268	0.972947

Based on our benchmarking results, we selected CatBoost as the final model architecture for our network intrusion

detection and attack classification system. This decision was primarily driven by CatBoost's exceptional ability to handle categorical features. Its advanced algorithm automatically processes categorical variables, eliminating the need for explicit preprocessing and delivering more robust and efficient encoding. Additionally, CatBoost offers built-in handling of missing values, removing the need for separate imputation steps. Its ordered boosting technique also helps reduce the risk of overfitting, ensuring strong generalization to unseen data [24]. Furthermore, the model's minimal hyperparameter tuning requirements make it an attractive choice, achieving high performance with minimal manual optimization.

The minimal performance difference between XGBoost and CatBoost, along with CatBoost's advantages, made it the optimal choice for our network intrusion detection system.

3) Model Training:

a) Detection Model: For the intrusion detection stage of our pipeline, our primary focus was on minimizing false negative rates. Given the critical importance of accurately identifying all malicious traffic, we placed a greater emphasis on maximizing the recall metric over other performance indicators like precision.

To achieve this, we carefully tuned the hyperparameters of our CatBoost detection model using the Optuna framework [25]. Rather than optimizing for overall accuracy, we defined a custom objective function that heavily weighted the recall score. This ensured that the model was optimized to catch as many intrusion attempts as possible, while still maintaining strong performance on other key metrics like F1-score and AUC.

After extensive experimentation, the best-performing hyperparameter set was the one shown in Table VI:

TABLE VI
DETECTION MODEL HYPERPARAMETERS

Hyperparameter	Value
iterations	4545
learning_rate	0.034781065245863746
depth	9
l2_leaf_reg	4.974504622139129
bootstrap_type	Bayesian
bagging_temperature	0.17707455985311188

b) Classification Model: For the attack classification stage of our pipeline, we made the strategic decision to only train the model on the malicious traffic identified by the detection stage. Given the significant class imbalance in the overall dataset, with normal traffic vastly outnumbering the various attack categories, we wanted to ensure the classification model could focus solely on distinguishing between the different types of intrusions.

After extensive analysis and testing, we consolidated the attack categories into 5 main groups: **Generic**, **Fuzzers**, **DoS**, **Reconnaissance**, and **Exploits**. This simplification helped us address the challenges posed by the low representation of certain attack types, such as Worms and Shellcode, in the dataset.

Similar to the detection model, we used the Optuna framework to systematically tune the hyperparameters of our CatBoost classification model, this time focusing on multi-class metrics like precision, recall, and F1-score for each attack category.

After extensive hyperparameter search, the best performing hyperparameter set was the one shown in Table VII:

TABLE VII
CLASSIFICATION MODEL HYPERPARAMETERS

Hyperparameter	Value
iterations	1117
learning_rate	0.09100148551636808
depth	7
l2_leaf_reg	0.8791036277340738

4) Detection and Classification Pipeline: The dual-stage intrusion detection architecture begins with raw input data undergoing an initial preprocessing phase, which extracts standardized essential features. This preprocessed data then enters a *Detection Model* that performs a binary classification to label traffic as either *Benign* or *Malicious*. Malicious traffic proceeds to a pipeline where it is further processed and then analyzed by a *Classifier Model* to determine the specific *Attack Category*. (see Figure 8).

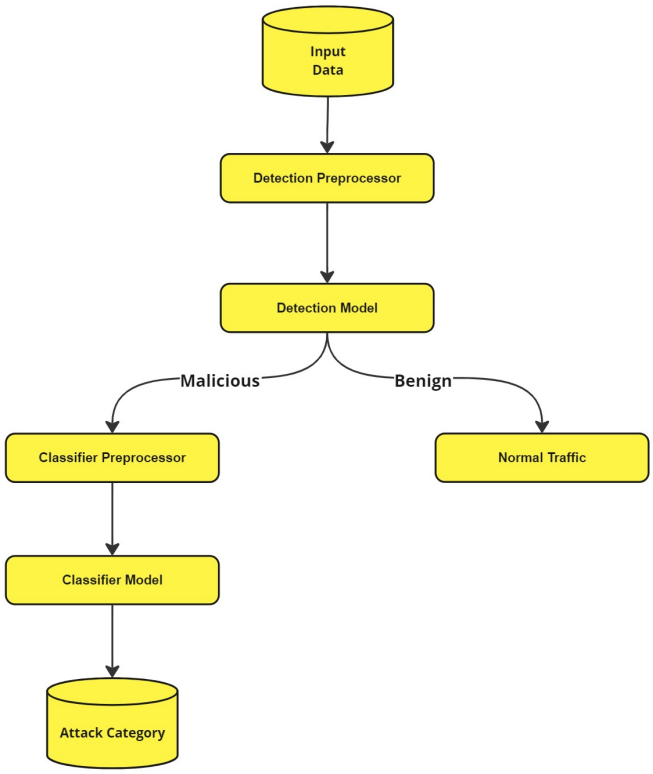

Fig. 8. Intrusion and detection pipeline architecture

C. Agentic Workflow for Report Generation

The agentic workflow within the SMARTSHIELD platform is designed to facilitate the automated generation of

comprehensive incident reports following the detection and classification of a cyber threat. The core idea behind this workflow is to delegate distinct tasks to specialized agents, each capable of performing specific functions required for accurate threat analysis, mitigation strategy development, and report generation. This approach ensures that each task is handled by an agent with the appropriate expertise, promoting efficiency and high-quality outputs. The workflow is orchestrated using CrewAI[5], which allows for seamless coordination between multiple agents.

the agentic workflow also serves as XAI. By assigning tasks to agents that analyze and interpret threat data, the workflow aims to provide insights into the underlying logic of the model's decision-making process. Each agent's analysis contributes to understanding the threat patterns, providing context and rationale for decisions, and thus enhancing the transparency and explainability of the system's threat assessments.

1) **Threat Analysis**: The first step involves analyzing the detected threat. A *Threat Analyzer Agent* is responsible for this task. It examines the raw threat data to define the nature of the threat, classify it within a predefined set of intrusion types, and evaluate its potential impact on the system's confidentiality, integrity, and availability (CIA). Additionally, the agent identifies key indicators within the data that provide further context or highlight specific attack vectors. The outcome of this task is a detailed analysis, including the classification of the threat and its impact assessment.
2) **Mitigation Strategy Development**: Once the threat has been analyzed, the next step is to develop a mitigation plan. A *Mitigation Strategist Agent* is tasked with this responsibility. Using insights from the threat analysis, this agent proposes both short-term and long-term mitigation actions. These actions are tailored to the specific characteristics of the threat and aim to minimize its impact while reinforcing the organization's security posture against future attacks. The Mitigation Strategist Agent also creates an incident response playbook, which provides step-by-step instructions for handling similar incidents in the future. The mitigation plan is driven by data-informed strategies specific to the detected threat.
3) **Report Generation**: After the threat analysis and mitigation strategies have been formulated, the final step is to compile all gathered information into a comprehensive incident report. The *Report Generator Agent* is responsible for this task. This agent synthesizes the information from the previous two stages and organizes it into a structured report. The report typically includes the following sections:
    - **Executive Summary**: A high-level overview of the incident and its key findings.
    - **Threat Analysis**: A detailed breakdown of the nature, classification, and impact of the detected threat.
    - **Impact Assessment**: An evaluation of the effects of the attack on confidentiality, integrity, and availability.
    - **Mitigation Strategy**: A set of actionable steps to address the detected threat, including short-term and long-term actions.
    - **Conclusions**: A summary of the incident and recommendations for preventing future occurrences.

The workflow is summarized in Figure 9 as the chain of the following key stages:

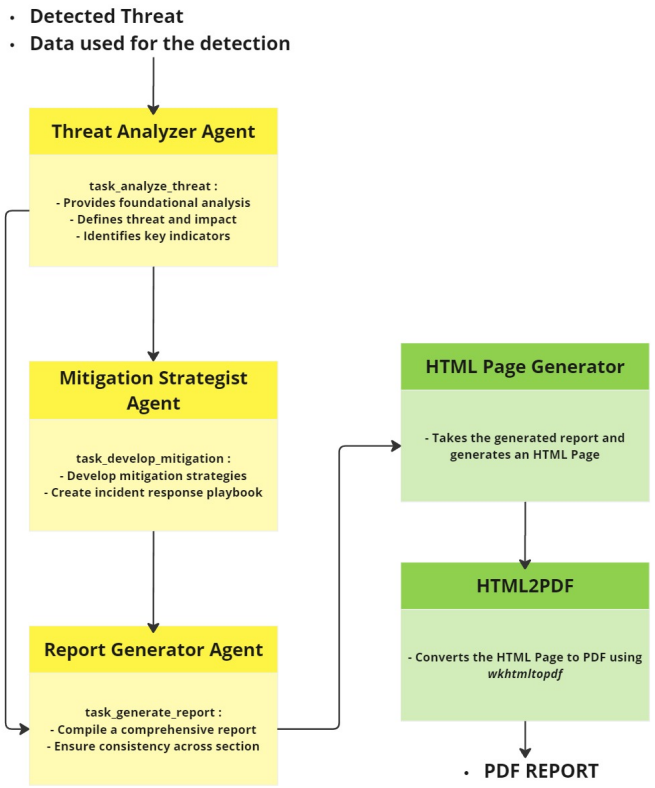

Fig. 9. Design of the Agentic Workflow for Report Generation

Throughout the process, each agent operates autonomously but works in close coordination with the others to ensure that the report is accurate, complete, and actionable. The agentic workflow is orchestrated in such a way that each agent's output serves as the input for the next, creating a seamless pipeline that ultimately delivers a comprehensive report ready for review by security analysts and decision-makers.

This methodology leverages the strengths of specialized agents, each focused on a specific task, to create a highly efficient and automated incident response process. CrewAI ensures that tasks are dynamically assigned based on the available agents and their capabilities, allowing the system to scale and adapt to varying types of cyber threats. The resulting report provides actionable intelligence that can be used to mitigate the current threat and prevent future incidents.

The design of this agentic workflow guarantees that the

[5]https://github.com/crewAIInc/crewAI

incident response process is not only automated but also structured in a way that provides clear, accurate, and comprehensive information for security teams. By dividing the work among specialized agents, SMARTSHIELD can deliver high-quality results quickly and consistently, thereby enhancing the overall efficiency and effectiveness of the cybersecurity incident response process.

D. Data Visualization

Our data visualization strategy offers enterprise users a streamlined, interactive dashboard for real-time security monitoring and analysis.

Primary data visualizations include:

- **Line Chart:** Displays log volume every 5 minutes, allowing users to identify unusual spikes in activity.
- **Pie Chart:** Categorizes machine learning-detected threats to prioritize response based on risk distribution.

A *Recent Security Offenses* section provides details on current incidents, including user, description, severity, and time since last update, supporting rapid threat response.

To enhance data access and analysis, we integrated the ELK stack[6] (Elasticsearch, Logstash, Kibana) for advanced data search, ingestion, and visualization. Kibana's custom query and filter options enable in-depth exploration of historical and real-time data within the dashboard, facilitating granular threat analysis.

This visualization approach ensures efficient data-driven decision-making by combining high-level insights with detailed exploration options.

V. Deployment Architecture

The complete system architecture, illustrated in Figure 10, demonstrates our containerized microservices approach to log-based threat detection. This design philosophy yields several key advantages for both operational efficiency and security

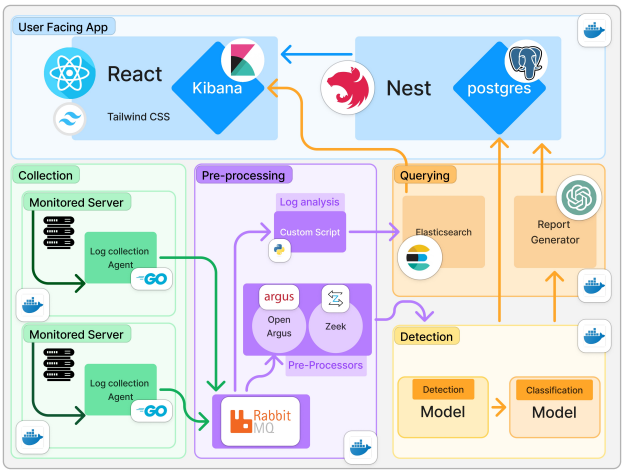

Fig. 10. Overall System Architecture

[6]https://elastic.co

A. Loose Coupling and Modularity

Each component operates independently in its own container, minimizing dependencies and enabling updates or replacements without disrupting the entire system. This design follows the single responsibility principle, enhancing both maintainability and testing.

B. Horizontal Scalability

The collection layer can easily scale to include more monitored servers without architectural changes, while preprocessing and detection components scale independently based on demand. RabbitMQ load balancing ensures efficient log processing distribution.

C. Resilience and Fault Tolerance

Component isolation prevents cascade failures, with failed containers automatically restarting without impacting other services. The message queuing system preserves log data during downtime.

D. Enhanced Security

Containerization ensures boundary isolation, limiting the impact of security breaches. Each component runs with minimal privileges, and network segmentation enables granular access control. Compromised components can be swiftly isolated and replaced.

E. Extensibility

The modular design allows easy integration of new detection models, preprocessing steps, and analysis tools as separate containers. It also supports parallel deployment of multiple model versions for testing and validation.

VI. Evaluation and Results

A. Detection Model

Upon evaluating the optimized detection model on the held-out test set, we obtained a set of performance metrics that illustrate its effectiveness.

TABLE VIII
Performance Metrics for Detection Model

Metric	Value
Precision	88.97%
Recall	94.26%
F1-Score	91.5%
AUC	98.04%

These results, presented in Table VIII, demonstrate that the tuned CatBoost intrusion detection model achieved exceptional performance. Of particular note is the high recall score of 0.94, which indicates the model's ability to accurately identify the vast majority of malicious traffic instances.

Figure 11 shows the Receiver Operating Characteristic (ROC) curve and the corresponding Area Under the Curve (AUC) metric of 0.98. This AUC score indicates that the model has excellent discriminative power in distinguishing between malicious and benign network traffic.

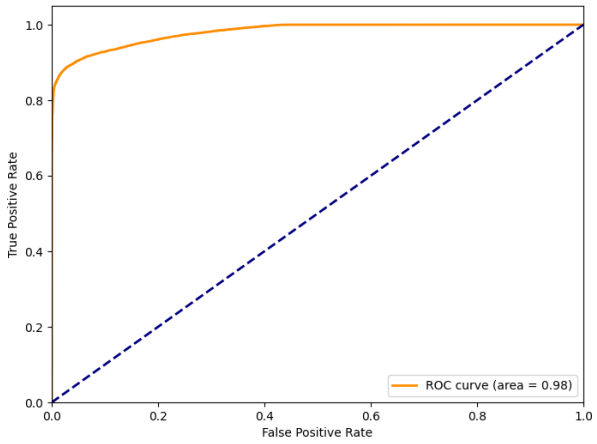

Fig. 11. ROC Curve of Detection Model with AUC of 0.97

B. Classification Model

The multi-class classification model's performance was thoroughly evaluated using the held-out test data, providing a detailed assessment of its capabilities across different attack categories. The results are shown in Table IX.

TABLE IX
PERFORMANCE METRICS FOR CLASSIFICATION MODEL BY ATTACK TYPE

Type	Precision	Recall	F1-Score
Generic	1.00	0.97	0.98
Reconnaissance	0.84	0.84	0.85
Fuzzers	0.84	0.77	0.80
DoS	0.39	0.78	0.52
Exploits	0.84	0.62	0.71

The classification model demonstrated strong performance across several attack categories. The high F1 scores for the Generic, Reconnaissance, and Fuzzers categories, ranging from 0.80 to 0.98, indicate the model's ability to accurately predict the correct attack type in these areas.

However, the model's performance was less balanced across all categories. The DoS category exhibited a relatively low F1-score of 0.52, primarily due to a precision of 0.39. This suggests that the model may have difficulty distinguishing DoS attacks from other types of malicious traffic. One possible reason for this is that DoS attacks and Fuzzer attacks can share similar characteristics from a network traffic perspective, particularly in terms of high-volume, repetitive requests, which may lead to misclassification between these two categories. Additionally, the lower number of DoS attack samples in the training data could also contribute to the model's reduced ability to generalize effectively in this category, resulting in poorer performance compared to others with more abundant data.

C. API

Through extensive stress testing, we successfully optimized the API's performance, achieving an average response time that consistently falls between 0.4 and 0.8 seconds per request, even under high load. By using a moving average to analyze response times, we gained a clearer view of the API's performance over time, smoothing out fluctuations to identify stable trends (see Figure 12). This approach allowed us to confirm that our optimizations provided consistent results throughout various stages of the test.

The key optimizations implemented include enhanced concurrency management, efficient request handling, and refined data processing techniques. Together, these improvements ensured that response times remained within the desired range across diverse and high-traffic scenarios.

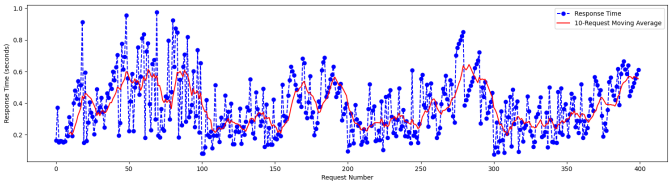

Fig. 12. API Response Time Trends using Moving Average Analysis

VII. CONCLUSION AND FUTURE WORK

In this work, we presented SMARTSHIELD, an AI-powered cybersecurity incident response platform designed to detect, classify, and mitigate network threats in real time. Leveraging a combination of custom-built log extractors, pipelines, and machine learning models, SMARTSHIELD provides an integrated solution for anomaly detection and network intrusion classification. Our approach incorporates both data-driven detection techniques and agentic workflows, enabling the automated generation of comprehensive incident reports with actionable recommendations. This ensures a structured response process, enhancing the platform's ability to react to complex and evolving threats.

Evaluation results demonstrate the effectiveness of our model in identifying and classifying intrusions with high accuracy and low latency, meeting real-time processing requirements critical in a dynamic network environment. The UNSW-NB15 datasets proved to be robust resource for training the anomaly detection and classification models, while the use of tools like openArgus and Zeek enabled efficient feature extraction tailored to each model's requirements. Additionally, our custom optimizations for feature extraction provided a lightweight solution that maintained accuracy and minimized resource consumption.

Despite its strengths, SMARTSHIELD has several areas that could benefit from further development. One limitation is the dependency on specific datasets, which may restrict model generalization to certain types of attacks. To address this, future work could involve training new models with more diverse, real-world datasets to improve adaptability to emerging threats. Another area for enhancement is the integration of adaptive learning mechanisms that allow the models to continuously evolve based on new data, thereby improving threat classification accuracy over time.

Moreover, while the current agentic workflow for report generation provides structured insights, expanding this functionality to allow for dynamic interaction with human analysts could increase the platform's utility in high-stakes incident response scenarios. Future iterations of SMARTSHIELD could also incorporate Explainable AI (XAI) components for model interpretability, giving security teams greater visibility into the factors driving detection and classification decisions. This would enhance transparency and trust in the system, especially in scenarios where model outputs significantly impact decision-making.

In conclusion, SMARTSHIELD represents a comprehensive solution for automated cybersecurity incident response, combining advanced machine learning techniques with a modular, real-time architecture. By addressing its current limitations and expanding on its capabilities, SMARTSHIELD has the potential to become an adaptable and valuable asset in the ongoing battle against sophisticated cyber threats.

ACKNOWLEDGMENTS

The authors would like to thank the IEEE INSAT Student Branch Computer Society for providing resources and support throughout this project. Special appreciation is extended to INSAT for its invaluable assistance in creating the SMARTSHIELD platform.

Our gratitude also goes to Dr. Lilia Sfaxi for her guidance and constructive feedback during the development and review process.

REFERENCES

[1] Liu, H., et al. (2019). "Machine learning and deep learning methods for intrusion detection systems: A survey." Applied Sciences, 9(20), 4396.

[2] Sarker, I. H., et al. (2020). "Cybersecurity data science: an overview from machine learning perspective." Journal of Big Data, 7(1), 1-29.

[3] Dongare, P., et al. (2021). "A comprehensive survey on ML-based IDS and challenges." Security and Communication Networks.

[4] Ahmad, Z., et al. (2021). "Network intrusion detection system: A systematic study of machine learning and deep learning approaches." Transactions on Emerging Telecommunications Technologies.

[5] Wu, K., et al. (2020). "An interpretable deep learning framework for the intrusion detection system." IEEE Access, 8, 3014-3027.

[6] Hindy, H., et al. (2020). "A taxonomy of network threats and the effect of current datasets on intrusion detection systems." IEEE Access.

[7] Roscher, R., et al. (2020). "Explainable machine learning for scientific insights and discoveries." IEEE Access.

[8] Nascimento, G., et al. (2021). "A systematic literature review on using XAI for cybersecurity." Future Generation Computer Systems.

[9] Samek, W., et al. (2019). "Towards explainable artificial intelligence in cybersecurity." Lecture Notes in Computer Science.

[10] Zhou, Y., et al. (2020). "Deep learning-based real-time network intrusion detection architecture." IEEE Access.

[11] Guo, W., et al. (2021). "An interpretable deep learning framework for intrusion detection systems." Expert Systems with Applications.

[12] Yang, K., et al. (2019). "Real-time network anomaly detection with reduced latency." IEEE Transactions on Information Forensics and Security.

[13] Svacina, J., et al. (2020, October). On vulnerability and security log analysis: A systematic literature review on recent trends. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems (pp. 175-180).

[14] Zhang, Z., et al. (2022). Explainable artificial intelligence applications in cyber security: State-of-the-art in research. IEEE Access, 10, 93104-93139.

[15] Singh, J., et al. (2013). Detection of DDOS attacks using source IP-based entropy. vol, 3, 201-210.

[16] Chen, S., et al. (2018). Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. computers & security, 73, 326-344.

[17] Mehmood, Y., et al. (2013, December). Intrusion detection system in cloud computing: Challenges and opportunities. In 2013 2nd National Conference on information assurance (NCIA) (pp. 59-66). IEEE.

[18] Moustafa, N., et al. (2015, November). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In 2015 military communications and information systems conference (MilCIS) (pp. 1-6). IEEE.

[19] Olusola, A. A., et al. (2010, October). Analysis of KDD'99 intrusion detection dataset for selection of relevance features. In Proceedings of the world congress on engineering and computer science (Vol. 1, pp. 20-22). WCECS.

[20] Revathi, S., et al. (2013). A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. International Journal of Engineering Research & Technology (IJERT), 2(12), 1848-1853.

[21] D'hooge, L., et al. (2022, August). Discovering non-metadata contaminant features in intrusion detection datasets. In 2022 19th Annual International Conference on Privacy, Security & Trust (PST) (pp. 1-11). IEEE.

[22] Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern recognition, 30(7), 1145-1159.

[23] Chen, T., et al. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[24] Prokhorenkova, L., et al. (2018). CatBoost: unbiased boosting with categorical features. Advances in neural information processing systems, 31.

[25] Akiba, T., et al. (2019, July). Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 2623-2631).

[26] Rodríguez M, Alesanco Á, Mehavilla L, García J. Evaluation of Machine Learning Techniques for Traffic Flow-Based Intrusion Detection. Sensors (Basel). 2022 Nov 30;22(23):9326. doi: 10.3390/s22239326. PMID: 36502028; PMCID: PMC9740321.

[27] Panigrahi, Ranjit & Borah, Samarjeet. (2018). A detailed analysis of CICIDS2017 dataset for designing Intrusion Detection Systems. International Journal of Engineering & Technology. 7. 479-482.

[28] Habibi Lashkari, Arash. (2018). CICFlowmeter-V4.0 (formerly known as ISCXFlowMeter) is a network traffic Bi-flow generator and analyser for anomaly detection. https://github.com/ISCX/CICFlowMeter. 10.13140/RG.2.2.13827.20003.

[29] Leevy, J. L., & Khoshgoftaar, T. M. (2020). A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data. J Big Data 7 (1).