iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** XI **Month of publication:** November 2026

DOI: https://doi.org/10.22214/ijraset.2026.85596

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue XI Nov 2026- Available at www.ijraset.com

Dynamic Cybersecurity Sentinel: AI-Based Network Anomaly Detection and Threat Response System

Tanmay Wanjari[1], Vaishali Ghodichor[2], Sarthak Thakre[3], Snehal Urkude[4], Sangam Bopche[5], Shravani Varsale[6]
Department of Artificial Intelligence, G H Raisoni College of Engineering and Management, Nagpur, India

Abstract: ***Network intrusion detection requires the identification of malicious traffic while maintaining the ability to observe unusual network behavior. This paper presents Dynamic Cybersecurity Sentinel, a network monitoring and intrusion detection system that combines supervised machine learning with behavioral anomaly detection. The system uses the UNSW-NB15 dataset for the development and evaluation of its classification component. An XGBoost-based classifier is used to categorize network traffic into Normal, Reconnaissance, Exploits, and Generic classes. In addition to classification, the system incorporates Isolation Forest for behavioral anomaly detection. The live monitoring part uses the Scapy library to catch network packets, which are then turned into network flows before extracting and analyzing features. The information from this process goes through baseline calibration, risk and severity evaluation, recommendation for responses, storing alerts, and showing results on a dashboard. The system was built in two steps, starting with a structured CSV-based setup and moving on to real-time network traffic monitoring. Testing the final XGBoost classifier showed an accuracy of 75.00%, weighted precision of 0.83, weighted recall of 0.75, and weighted F1-score of 0.77.The live version was tested using traffic from several connected computers. The results show how classification, behavior analysis, risk assessment, alert creation, and visualization work together in one network security monitoring system***

Keywords: Network Intrusion Detection System, UNSW-NB15, XGBoost, Isolation Forest, Network Traffic Analysis, Anomaly Detection, Scapy, Cybersecurity.

I. INTRODUCTION

As more systems rely on connected computers, network monitoring has become a key part of keeping cybersecurity strong. The traffic moving across a network shows how systems communicate, and strange traffic patterns may signal harmful or unwanted actions. A Network Intrusion Detection System (NIDS) watches over network activity and spots signs of unauthorized access or odd behavior. How well a machine-learning-based NIDS works depends a lot on the quality and fairness of the data used to build and test it [9], [10].

Machine-learning methods have been investigated extensively for intrusion detection because network traffic can be represented through measurable flow or packet characteristics. Reviews of intrusion detection research show the use of supervised, unsupervised, and hybrid learning methods for identifying network attacks [10]–[12]. At the same time, studies of network intrusion datasets emphasize that the evaluation dataset and its recording conditions strongly affect the interpretation of reported results [9], [15].

The UNSW-NB15 dataset was introduced to provide a network intrusion dataset containing modern normal activity together with contemporary synthesized attack activity [1]. It has subsequently been used in studies evaluating machine-learning approaches for intrusion detection, including XGBoost and other classical classifiers [6], [17], [18]. These studies provide a basis for evaluating supervised learning approaches on structured network-flow data.

Supervised classification and anomaly detection address different aspects of network monitoring. A classifier can assign traffic to known categories, whereas an anomaly detector can identify observations that differ from the behavior represented during its learning process. Research on unsupervised network intrusion detection and anomaly-based methods has therefore investigated approaches that do not depend entirely on predefined attack labels [2], [13]. Isolation Forest is another method used for detecting anomalies, and it was first introduced to identify observations that are simpler to separate from the rest of the data [8].

The proposed Dynamic Cybersecurity Sentinel combines these two analysis approaches into a single operational process. XGBoost is used for attack classification, while Isolation Forest is used for detecting behavioral anomalies. The live component uses Scapy for packet acquisition and a flow builder that produces a 42-feature network-flow representation.

©IJRASET: All Rights are Reserved

Both the CSV-based Version 1 and the live Version 2 use the same machine-learning engine. The Version 2 process also includes baseline calibration with a threshold of 3.0, then risk scoring on a 0100 scale, assigning severity, suggesting response actions, storing alerts in alerts.json, and using Flask for live monitoring.

The system was built step by step. The first version set up the process for handling CSV input, doing some preliminary work, and then using machine learning for analysis. Version 2 extended the same project toward live traffic by capturing packets, building flows, and passing the resulting features through the common analysis engine. The objective of this work is not to introduce a new learning algorithm, but to integrate the implemented classification, anomaly detection, risk assessment, alerting, and visualization components into a single working network-security monitoring system.

The main contributions of the implemented system are: (1) a common processing path that supports both structured CSV input and live network traffic; (2) multi-class traffic classification using XGBoost for categories like Normal, Reconnaissance, Exploits, and Generic; (3) behavioral anomaly detection using Isolation Forest; (4) baseline calibration and 0–100 risk scoring with severity assessment in the live processing path; and (5) alert storage and Flask dashboard monitoring for the live network experiment.

II. RELATED WORK

A. *Network Intrusion Detection and Datasets*

Research on network intrusion detection has developed from signature-oriented methods toward approaches that use statistical analysis and machine learning to identify suspicious traffic [10], [11]. Signature-based methods are useful when previously known attack patterns can be represented explicitly, but they are less suited to behaviors that do not match an available signature. Machine-learning-based NIDS approaches instead represent traffic using measurable characteristics and learn relationships between those characteristics and security events. The choice of traffic representation is therefore an important part of NIDS development.

Dataset selection is particularly important because intrusion-detection models are evaluated on traffic collected under specific network conditions, feature definitions, attack scenarios, and labeling procedures [9], [15]. Ring et al. surveyed network-based intrusion-detection datasets and highlighted differences in traffic generation, attack coverage, feature representation, and evaluation conditions [9]. Shiravi et al. also discussed the importance of systematic benchmark-dataset construction for meaningful intrusion-detection evaluation [15]. These observations are relevant to the present project because the supervised component is developed and evaluated using a fixed structured dataset before being connected to live traffic.

The UNSW-NB15 dataset was introduced by Moustafa and Slay to provide a network-intrusion benchmark containing normal activity together with contemporary synthesized attack activities [1]. The dataset has been used in various machine learning-based network intrusion detection system (NIDS) studies, including those that assess traditional classifiers and feature processing methods [6], [17], [18].In this current system, the UNSW-NB15 dataset is specifically used for the supervised classification part. The project organizes the classification output into the four implemented classes Normal, Reconnaissance, Exploits, and Generic, rather than claiming classification of every original UNSW-NB15 category.

B. *Supervised Machine Learning for NIDS*

Supervised machine learning is widely used in NIDS when labeled traffic is available because the model can learn a mapping between traffic features and predefined attack categories [10], [11]. In this context, how the data is prepared, how features are represented, how classes are defined, and how the model is tested all have a direct impact on how well the model performs. Therefore, the reported performance of a classifier should be interpreted together with the dataset configuration and experimental procedure rather than as an independent property of the algorithm.

Clottey, Yaokumah, and Appati evaluated several machine-learning algorithms, including XGBoost, using the UNSW-NB15 dataset [6]. Their work provides a direct reference for the use of supervised learning on the same benchmark used by the present project. Other studies have also examined machine-learning approaches on UNSW-NB15 and reported that performance varies with preprocessing, feature handling, model configuration, and the formulation of the classification task [17], [18]. These studies support the use of supervised learning for structured network-traffic classification while also showing why results from different implementations should not be compared without considering their experimental conditions.

In Dynamic Cybersecurity Sentinel, XGBoost is used as the supervised classification component. Its role is limited to assigning the implemented traffic category based on the structured feature representation. The classifier is therefore just one part of the overall detection pipeline, rather than the complete security decision mechanism. This design allows the project to retain the predicted class while separately obtaining a behavioral anomaly signal from Isolation Forest.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

C. Anomaly Detection and Unsupervised Approaches

Anomaly-based intrusion detection tackles a different issue compared to supervised attack classification. Rather than depending only on predefined categories, this method tries to spot activities that don't match a known or learned pattern of normal behavior. Surveys on network anomaly detection and intrusion detection techniques highlight that unsupervised and anomaly based methods are helpful for analyzing network traffic that might not fit into existing labeled attack categories [10], [12], [19].

Casas, Mazel, and Owezarski studied an unsupervised network intrusion detection method designed to find unknown attacks without fully relying on prior knowledge of attack signatures or labeled traffic [13]. Carrera et al. looked at combining different unsupervised methods for detecting network traffic anomalies in near real-time and tested how these methods worked on various intrusion detection datasets [2]. Their research supports the idea of having a separate step for analyzing anomalies in systems were classifying known types and identifying unusual behavior are handled as separate tasks.

This project follows that approach. XGBoost is used for supervised classification, while Isolation Forest is used to detect behavioral anomalies. These two outputs are not seen as the same thing: an anomaly signal doesn't automatically mean an attack is happening, and a classification result doesn't automatically mean the traffic is behaving unusually. Keeping these signals, separate aligns with how the project is designed to function.

D. Isolation Forest and Related Research

Isolation Forest was introduced by Liu, Ting, and Zhou as an anomaly detection method that focuses on isolating observations that are easier to separate from the rest of the data [8].The method works well for anomaly analysis because it tackles the problem by focusing on how easily unusual observations can be isolated, rather than requiring a conventional multi-class attack label for each observation. Moomtaheen et al. investigated Extended Isolation Forest for intrusion detection using Zeek connection data and examined the effect of hyperparameters on detection performance[5]. Their work is related to the present project because both use an isolation-forest-based approach for network intrusion analysis, but the implemented algorithms are not the same. The present system uses the original Isolation Forest method described by Liu et al.[8], not the Extended Isolation Forest variant.

Within Dynamic Cybersecurity Sentinel, Isolation Forest is integrated after the common machine-learning input stage and its output is used as a behavioral signal. This allows the project to add a separate anomaly signal to the XGBoost class prediction. The anomaly component is therefore an implemented part of the system architecture, not a replacement for the supervised classifier.

E. Deep Learning and Hybrid Detection Research

Recent NIDS research has also examined deep-learning and representation-learning approaches for feature extraction, anomaly detection, and hybrid intrusion analysis. Yoshimura et al. proposed DOC-IDS, a deep-learning-based method that combines learned feature representations with anomaly detection in network traffic [3]. Almeida et al. investigated a semi-supervised autoencoder approach for intrusion detection, illustrating another way of using learned representations when fully labeled traffic is not available [4]. Surveys of deep-learning-based anomaly detection further describe the use of neural models for learning complex traffic representations [12], [20].

These studies are relevant as related work because they demonstrate alternative approaches to combining traffic representation and anomaly analysis. However, they are not components of the present implementation. Dynamic Cybersecurity Sentinel does not use DOC-IDS, autoencoders, or a deep neural network as part of its implemented detection pipeline. The current project instead uses XGBoost for supervised classification and Isolation Forest for anomaly detection, with the emphasis placed on integrating these components with live packet acquisition and operational monitoring.

F. Position of the Present Work

The reviewed literature includes studies on supervised classification, unsupervised anomaly detection, flow-based analysis, benchmark datasets, and near real-time monitoring [2], [9], [13], [14], [19], [20].The current work brings these themes together at the system integration level instead of introducing a new machine learning algorithm. The main contribution is the creation of a working pipeline that links offline dataset-based classification with real-time network traffic processing.

Specifically, the project integrates an XGBoost classifier, an Isolation Forest anomaly detector, Scapy for packet capture, a flow builder that produces a 42-feature representation, and a common machine learning engine used by both the V1 CSV path and the V2 live path. The live path also applies a baseline calibration threshold of 3.0, generates a 0–100 risk score and severity level, recommends response actions, stores alert information in alerts.json, and displays the monitoring status through a Flask dashboard.

This scope sets the project apart from studies that focus mainly on the performance of a single learning algorithm or one anomaly detection technique. The paper therefore evaluates the implemented system as an integrated network monitoring workflow. The goal is not to claim that the integration itself is a new learning algorithm, but to show how the selected components work together within the project.

III. SYSTEM DESIGN AND METHODOLOGY

A. Proposed Architecture

The system has two ways for input. The Version 1 (V1) path takes a CSV file as an offline input and does some preprocessing, which includes one-hot encoding and aligning features. The Version 2 (V2) path gets live network packets and uses Scapy along with the flow builder to create a 42-feature network flow representation. Both paths come together at a shared machine-learning engine used by both V1 and V2.

The overall structure is sequential, where traffic enters through one of the two input paths and is transformed into a format that works with the common machine-learning engine. The V1 path handles structured CSV records, while the V2 path captures live packets using Scapy and turns them into network flows. Both paths meet before the analytical parts, so that later stages like classification and anomaly detection use the same processing structure at the project level.

After the shared machine-learning engine, the traffic data is analyzed by two separate parts. Isolation Forest is used to detect behavioral anomalies. XGBoost supplies the supervised traffic-class prediction. The outputs are kept as separate pieces of information and are then sent to the live-path calibration and risk-assessment steps. This setup allows the system to report both the class the classifier predicts and whether the observed feature representation is considered unusual by the anomaly detector.

For V2, the downstream processing continues with baseline calibration using a threshold of 3.0.The resulting detection information is turned into a 0–100 risk score and a severity level, followed by suggested response actions and storage of alerts in alerts.json. The Flask dashboard serves as the presentation layer for live monitoring. Thus, the architecture connects packet acquisition, flow creation, machine learning analysis, behavioral evaluation, risk interpretation , alert storage, and visualization all within a single implemented pipeline.

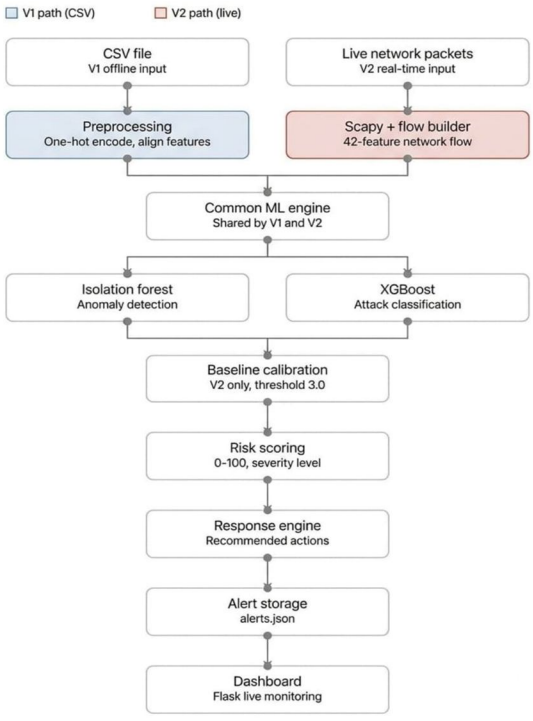


Figure 1: The implemented architecture of the Dynamic Cybersecurity Sentinel, which includes the V1 CSV path, V2 live path, common machine learning engine, stages for anomaly and attack classification, calibration, risk scoring, response recommendation, alert storage, and dashboard.

After the common machine-learning engine, the system splits into two separate analytical parts: Isolation Forest for detecting anomalies and XGBoost for classifying attacks. Their results are combined before the V2 baseline calibration stage. The live path then applies a threshold of 3.0 for baseline calibration, calculates a 0–100 risk score and severity level, suggests recommended response actions, saves alerts in alerts.json, and shares the monitoring results via the Flask dashboard.

B. Version 1: CSV-Based Processing

Version 1 was created as a way to process data offline. The input is a structured CSV file representing network-flow records. The preprocessing stage performs one-hot encoding and aligns the resulting features with the feature structure expected by the machine-learning engine. This version provides the controlled environment used for the supervised classification evaluation.

Version 1 was developed first as an offline processing path to establish the data-processing and machine-learning workflow before live traffic was introduced. Its input is a structured CSV file representing network-flow records. The preprocessing stage performs one-hot encoding and feature alignment so that the resulting input follows the feature structure expected by the machine-learning engine.

The prepared records are then sent to the common machine learning engine. XGBoost handles the four-class classification, while Isolation Forest gives the results for anomaly analysis. Since V1 works with an existing structured file, it creates a controlled environment for testing the supervised classifier and checking how the combined analytical parts work. It doesn't need packet capture.

The V1 path is therefore the offline base of the project. It sets up the way data is shown and processed, which is later used by the live path. This step-by-step design makes the difference between using datasets for testing and real-time inference smaller at the architecture level, while keeping packet collection specific to V2.

The V1 path also allows the project to apply the anomaly-detection component to structured data without requiring live packet acquisition. After preprocessing, the data enter the common machine-learning engine and are analyzed by the implemented Isolation Forest and XGBoost components.

C. Version 2: Live Network Processing

Version 2 extends the system to live network traffic. Scapy is used to capture packets from the monitored network. The captured packets are processed by the flow builder, which constructs a 42-feature network-flow representation. This representation is then aligned with the common machine-learning input so that the live traffic can be analyzed by the same downstream detection components.

Version 2 extends the established processing path to live network traffic. Scapy is used to capture packets from the monitored network interface. The captured packets are not passed directly to the classifier as individual raw packets; instead, the flow builder groups the observed traffic into flow records and constructs the 42-feature representation used by the V2 input path.

After flow formation, the resulting feature representation is aligned with the common machine-learning input and passed to the same downstream analytical components used in V1. XGBoost produces the implemented traffic class and Isolation Forest supplies the behavioral anomaly signal. The live path goes through the baseline calibration stage, risk and severity processing, response recommendations, alert storage, and dashboard presentation.

The live implementation was tested using traffic generated across multiple connected computers in a controlled environment. This offers an operational demonstration of the entire process from network activity to packet capture, flow formation, machine learning analysis, alert generation, and dashboard monitoring. The live implementation is therefore an extension of the V1 processing workflow rather than a separate machine learning system.

The live path is intended to show how the complete system operates using traffic from connected computers. It therefore provides an operational stage where packet acquisition, flow construction, machine learning analysis, risk assessment, alert generation, and dashboard visualization can all be seen together.

D. Flow Formation and Feature Representation

The project handles live packets by grouping them into network flows instead of treating each packet as a separate classification. Flow-based intrusion detection has been studied as an alternative to checking every packet individually , especially because flow records give a summary of communication data [14].In the current setup, the flow builder creates 42 features for the V2 input path. These features make up the organized data sent to the common machine learning engine. The live system groups packets into network flows instead of treating each packet as a separate classification.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

Flow-based intrusion detection has been studied as a useful method because a flow shows the overall behavior of communication across a series of packets and can provide summarized information for analysis [14].Using flow records also gives a structured input format that can be matched with the features expected by the machine learning stage.

In the current implementation, the V2 flow builder creates 42 features for each network flow record. These features form the structured data passed to the common machine learning engine. The paper does not claim that all 42 features are newly designed; they are the features used by the project's live flow-building process.

After creating the features, the live record is processed using the same downstream XGBoost and Isolation Forest components used by the project. This link between flow formation and the common machine learning engine is important because it lets live traffic follow the same analytical structure used during the CSV-based development stage.

E. XGBoost Attack Classification

XGBoost is the supervised classification part of the system. Chen and Guestrin introduced XGBoost as a scalable tree-boosting system for structured learning problems [7].In this project, the trained XGBoost model classifies traffic into four categories: Normal, Reconnaissance, Exploits, and Generic.

XGBoost is the supervised classification component of the system. Chen and Guestrin introduced XGBoost as a scalable tree-boosting system for structured learning problems [7].In this project, XGBoost is trained and evaluated as the classification component using the UNSW-NB15-based processing configuration .

The implemented classifier produces one of four traffic categories: Normal, Reconnaissance, Exploits, and Generic. The class output represents the supervised prediction produced from the structured feature representation. The reported evaluation metrics in this paper therefore describe the performance of this particular project configuration and its selected class definitions.

During integrated operation, the XGBoost result is not used as the only security signal. It is retained alongside the Isolation Forest anomaly result so that the system can distinguish between a predicted traffic category and a behavioral anomaly indication. This separation is part of the implemented architecture and is maintained in both the offline and live processing paths.

The classifier output is one part of the downstream security assessment. It is not treated as the only detection signal because the project also includes Isolation Forest-based anomaly analysis. This separation allows the system to retain both the predicted attack category and the behavioral anomaly result.

F. Isolation Forest Anomaly Detection

The anomaly-detection component uses Isolation Forest. The original method identifies anomalies by isolating observations rather than just creating a profile of normal observations [8]. In the present system, the Isolation Forest output is used as a behavioral signal along with the XGBoost classification result.

The anomaly-detection component uses the original Isolation Forest method. Liu, Ting, and Zhou introduced Isolation Forest as an approach that identifies anomalies by isolating observations that are easier to separate from the rest [8].This method provides an anomaly-oriented signal that complements the supervised class prediction.

In Dynamic Cybersecurity Sentinel, the Isolation Forest stage receives the structured representation produced by the project's processing pipeline. Its output is used as behavioral information rather than as a replacement for the XGBoost class prediction. Consequently, the system can retain a classification result and an anomaly result as separate outputs for downstream interpretation .

The implementation uses Isolation Forest itself and does not implement the Extended Isolation Forest variant discussed by Moomtaheen et al. [5].This distinction is maintained throughout the methodology because the purpose of the component in the present project is to provide the implemented behavioral anomaly signal within the common machine-learning pipeline.

The project therefore uses supervised and unsupervised analysis for different purposes. XGBoost provides a class prediction for the traffic categories represented during model development, while Isolation Forest provides anomaly information for the observed feature representation. The implementation does not replace Isolation Forest with the Extended Isolation Forest approach described by Moomtaheen et al. [5].

G. Baseline Calibration

The live V2 path includes a baseline calibration stage after the outputs from Isolation Forest and XGBoost. The implemented diagram sets a threshold value of 3.0 for this calibration stage. The purpose of this stage in the project is to compare the live behavioral results with the calibrated baseline before moving on to the risk-scoring stage. This threshold is part of the implemented V2 pipeline and is not intended to be a universal threshold for intrusion detection.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

The live V2 path also includes a baseline calibration stage after the outputs from XGBoost and Isolation Forest. The implemented configuration specifies a threshold value of 3.0 for this stage. The calibration step is used to interpret the live behavioral result against the project's configured baseline before the result is passed to risk scoring.

The threshold of 3.0 is an implementation parameter of the current project configuration. It is not derived in this paper as a universal network-security threshold, and it should not be assumed to have the same meaning for other networks or datasets. Its role is to support the interpretation of the live behavioral signal within the tested V2 pipeline.

H. Risk Scoring and Severity Assignment

The system converts the detection information into a risk score on a 0–100 scale and assigns a severity level. This stage provides an interpretation of the detection output for monitoring rather than changing the underlying machine-learning prediction. The resulting risk and severity information is passed to the response and alerting components.

Following the live detection and baseline-calibration stages, the system converts the available detection information into a risk score on a 0–100 scale and assigns a corresponding severity level. This stage offers an operational explanation of the machine learning results, so the dashboard and alerting parts can show the detection status in a way that's easy to monitor.

The risk score and severity levels don't change or retrain the original machine learning models. Instead, they work after the detection results are ready. The information from this stage is then sent to the response suggestions and alert storage parts. This way, the project keeps both the original detection signals and their operational meaning.

I. Response Engine and Alert Storage

The response engine creates suggested actions based on the security assessment made by the system. These are recommendations from the implemented response component; the paper does not consider them as automatic network blocking or remediation because those functions are not part of the confirmed project scope.

The response engine works after the risk and severity assessment and generates suggested actions based on the security status created by the pipeline. These actions are recommendations from the implemented response component. The current project does not treat this stage as an automatic network-blocking or remediation method because such functionality is outside the confirmed implementation.

The alert-storage component saves generated alert information in alerts.json. The stored information provides a lasting record of the detection status that can be used by the monitoring layer. This stage connects the analytical output to the dashboard and also keeps the generated alert information beyond the immediate model-inference step.

The alert-storage component saves generated alert information in alerts.json. This creates a lasting record that can be used by the monitoring interface and displayed to show the current detection status.

J. Dashboard

The final stage of the live pipeline is the Flask dashboard. It offers live monitoring of the information created by the detection and risk-assessment stages. So, it acts as the presentation layer through which you can see packet, flow, classification, anomaly, risk, severity, and alert information during a live experiment.

The Flask dashboard is the presentation layer of the live monitoring pipeline. It makes available the information generated by the packet-capture, flow-formation, machine-learning, anomaly, risk, severity, and alerting stages so that you can observe how the implemented system behaves during live testing.

The dashboard itself doesn’t do the classification or anomaly detection. Instead, it shows the results from the processing pipeline. This separation keeps the analytical parts and the visualization layer separate, while still allowing the live experiment to display the full sequence from captured network activity to interpreted security information.

IV. EXPERIMENTAL SETUP

A. Dataset

The supervised classification part uses the UNSW-NB15 dataset. Moustafa and Slay introduced UNSW-NB15 as a comprehensive network intrusion dataset that includes modern normal traffic and contemporary synthesized attack activities [1].The dataset is widely used in NIDS research, and its suitability and characteristics have been discussed in dataset surveys [9].

Within this project, the classification output is organized into four classes: Normal, Reconnaissance, Exploits, and Generic. The paper reports the results obtained from the implemented project configuration and does not generalize those results to all original UNSWNB15 attack categories.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

B. Live Network Environment

The live implementation was tested using multiple connected computers. Network traffic was created between the computers while the monitoring system captured packets using Scapy. The captured traffic was turned into flows and processed by the V2 path. This experiment was meant to show the shift from offline dataset processing to live traffic analysis.

The live experiment was purposely done in a controlled multi-computer environment. The measured live results therefore describe the tested configuration and traffic-generation procedure rather than all possible production network conditions.

C. Evaluation Metrics

The supervised classifier was evaluated using accuracy, weighted precision, weighted recall, and weighted F1-score. These metrics provide complementary information about the correctness of the classification results. Using several evaluation methods aligns with the general ideas in NIDS research, as relying only on accuracy might not be enough when dealing with multi-class or imbalanced traffic classification [6], [10], [17].

V. RESULTS AND DISCUSSION

A. XGBoost Classification Results

The final XGBoost classifier generated these evaluation results:

Metric	Result
Accuracy	75.00%
Weighted Precision	0.83
Weighted Recall	0.75
Weighted F1-score	0.77

TABLE 1: Final Xgboost Classification Results

The accuracy of 75.00% means that three quarters of the records that were tested were correctly classified using the final setup of the project. The weighted precision of 0.83 shows how precise the model is when considering the importance of each class based on how many examples there are. The weighted recall of 0.75 and the weighted F1-score of 0.77 give more insight into how well the model is balancing correctly identifying instances with the precision of its predictions. These results are part of this project's outcome and should not be directly compared to results from other studies unless the data split, class definitions, preprocessing steps, and evaluation methods are the same.

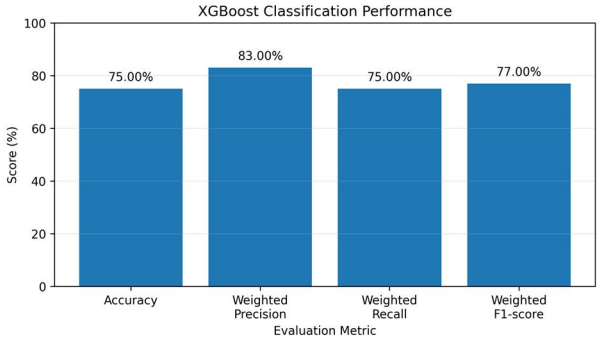


Figure 2: XGBoost classification performance on the test set.

B. Integrated Detection Behavior

The main system-level outcome is combining the supervised and anomaly detection steps. XGBoost is used to classify the traffic, while Isolation Forest gives a separate signal for anomalies. The downstream baseline and risk. Severity and alert stages use these detection outputs. This structure follows the project's architecture shown in Fig.1 and does not treat an anomaly flag as though it were an attack prediction.

The live path also shows that the detection pipeline can work with traffic captured from connected computers. The packet acquisition and flow-building stages link the physical network activity to the same machine learning engine used by the project. The dashboard then shows the resulting state for monitoring.

C. Live Monitoring

During the live test, the system captured packets, formed flows, analyzed the resulting feature records, and displayed the processed information through the Flask dashboard. The live experiment also tested the anomaly detection and alerting path. Because the project uses a controlled multi-computer environment, the live observations are best interpreted as an operational demonstration of the implemented pipeline rather than as a statistical benchmark over a large production network.

D. Discussion

The literature shows that supervised classification, unsupervised anomaly detection, and flow based analysis each handle different parts of the intrusion detection problem [2], [8], [13], [14].The architecture of Dynamic Cybersecurity Sentinel combines these components into one unified pipeline. Using a shared machine-learning engine for V1 and V2 also helps make the difference between offline model evaluation and real-time inference smaller at the system level.

The 75.00% classification result shows that the current supervised model doesn't correctly identify every tested instance. Because of this, the system is presented as a combined monitoring and detection framework, not as a tool that ensures every network event is correctly identified. The anomaly detection stage adds extra behavioral information, while the risk and severity stages turn the detection results into a format suitable for operational monitoring.

VI. COMPARISON WITH RELATED STUDIES

The studies mentioned offer various viewpoints on the development of NIDS. Clottey et al. looked at XGBoost and other traditional machine learning models on the UNSW-NB15 dataset [6], while More et al. compared several machine learning models using the same dataset [18].Carrera et al. focused on combining unsupervised methods for near real-time anomaly detection [2].Yoshimura et al. and Almeida et al. explored deep learning and autoencoder-based approaches [3], [4]. Moomtaheen et al. studied the Extended Isolation Forest using Zeek data [5].

The current work is different in scope rather than introducing a new learning algorithm. It focuses on integrating XGBoost classification with Isolation Forest anomaly detection, using live Scapy packet capture, 42-feature flow construction, V1/V2 processing paths, baseline calibration, risk scoring, recommended responses, JSON alert storage, and Flask monitoring. It's important to note the difference when comparing this paper with studies that look at just a model or just an anomaly detection method.

VII. LIMITATIONS

The supervised classification performance is limited by the final model and data setup used in the project. The reported accuracy of 75.00% shows that some of the evaluated records were not correctly classified.

The project uses four implemented traffic classes instead of reporting classification performance across every original UNSW-NB15 attack category. The results should therefore be understood within the class definitions used by the system.

The live evaluation was done in a controlled multi-computer environment. It shows how the implemented pipeline works, but it doesn't represent all possible real-world network conditions.

The risk score, severity thresholds, and V2 baseline threshold are specific to the implementation. The threshold value of 3.0 should not be taken as a general rule for other network environments.

The current project combines classification and anomaly detection, but the two outputs mean different things. An anomaly indication does not automatically mean an attack class, and a classification result does not necessarily mean the traffic is behaviorally unusual.

VIII. FUTURE SCOPE

Future improvements could focus on enhancing the supervised classification process by using more training data, analyzing features better, and testing under a wider variety of traffic conditions. The live system can also be tested on more network setups and a broader range of traffic situations.

Additional work might look into adjusting the live behavioral baseline using larger normal traffic samples and checking how anomaly scores, classification outcomes, and actual security incidents relate to each other. The risk and severity levels can also be assessed through more controlled scenarios to ensure the scoring system works consistently across different traffic patterns.

These are suggestions for future work and are not part of the current system's capabilities.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

IX. CONCLUSION

This paper introduces Dynamic Cybersecurity Sentinel, a system that detects network intrusions and analyzes behavioral anomalies. It combines supervised classification with anomaly detection and real-time network monitoring. The system uses the UNSW-NB15 dataset for classification, XGBoost for identifying attacks, and Isolation Forest for detecting unusual behavior. The live V2 version captures packets using Scapy and builds a network flow with 42 features.

The system's architecture supports both a CSV-based V1 path and a live V2 path through a shared machine learning engine. The V2 path includes baseline calibration with a threshold of 3.0, generates a risk score from 0 to 100 and a severity level, suggests response actions, stores alerts in alerts.json, and displays the monitoring status via a Flask dashboard.

The final XGBoost evaluation achieved 75.00% accuracy, 0.83 weighted precision, 0.75 weighted recall, and 0.77 weighted F1-score.A live multi-computer experiment demonstrated the workflow from packet capture to flow construction, analysis, alerting, and dashboard monitoring. The results show that the implemented system provides an integrated framework for the project's intended network monitoring workflow and also highlights areas where further evaluation and model improvement can be done.

REFERENCES

[1] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in Proc. Military Communications and Information Systems Conf. (MilCIS), Canberra, ACT, Australia, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.

[2] F. Carrera, V. Dentamaro, S. Galantucci, A. Iannacone, D. Impedovo, and G. Pirlo, “Combining unsupervised approaches for near real-time network traffic anomaly detection,” Applied Sciences, vol. 12, no. 3, Art. no. 1759, 2022, doi: 10.3390/app12031759.

[3] N. Yoshimura, H. Kuzuno, Y. Shiraishi, and M. Morii, “DOC-IDS: A deep learning-based method for feature extraction and anomaly detection in network traffic,” Sensors, vol. 22, no. 12, Art. no. 4405, 2022, doi: 10.3390/s22124405.

[4] A. Almeida, L. Gonçalves, C. Zanchettin, and B. L. D. Bezerra, “A semi-supervised autoencoder approach for efficient intrusion detection in network traffic,” in Proc. IEEE Latin American Conf. Computational Intelligence (LA-CCI), Recife, Brazil, 2023, pp. 1–6, doi: 10.1109/LA-CCI58595.2023.10409491.

[5] F. Moomtaheen, S. S. Bagui, S. C. Bagui, and D. Mink, “Extended Isolation Forest for intrusion detection in Zeek data,” Information, vol. 15, no. 7, Art. no. 404, 2024, doi: 10.3390/info15070404.

[6] R. N. Clottey, W. Yaokumah, and J. K. Appati, “Modelling and evaluation of network intrusion detection systems using machine learning techniques,” International Journal of Intelligent Information Technologies, vol. 17, no. 4, pp. 1–19, 2021, doi: 10.4018/IJIIT.289971.

[7] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining, San Francisco, CA, USA, 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation Forest,” in Proc. 8th IEEE Int. Conf. Data Mining, Pisa, Italy, 2008, pp. 413–422, doi: 10.1109/ICDM.2008.17.

[9] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” Computers & Security, vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.

[10] A. L. Buczak and E. Guven, “A survey of data mining and machine learning methods for cyber security intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016, doi: 10.1109/COMST.2015.2494502.

[11] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: Techniques, datasets and challenges,” Cybersecurity, vol. 2, Art. no. 20, 2019, doi: 10.1186/s42400-019-0038-7.

[12] S. Gamage and J. Samarabandu, “Deep learning methods in network intrusion detection: A survey and an objective comparison,” Journal of Network and Computer Applications, vol. 169, Art. no. 102767, 2020, doi: 10.1016/j.jnca.2020.102767.

[13] P. Casas, J. Mazel, and P. Owezarski, “Unsupervised network intrusion detection systems: Detecting the unknown without knowledge,” Computer Communications, vol. 35, no. 7, pp. 772–783, 2012, doi: 10.1016/j.comcom.2012.01.016.

[14] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, “An overview of IP flow-based intrusion detection,” IEEE Communications Surveys & Tutorials, vol. 12, no. 3, pp. 343–356, 2010, doi: 10.1109/SURV.2010.032210.00054.

[15] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Toward developing a systematic approach to generate benchmark datasets for intrusion detection,” Computers & Security, vol. 31, no. 3, pp. 357–374, 2012, doi: 10.1016/j.cose.2011.12.012.

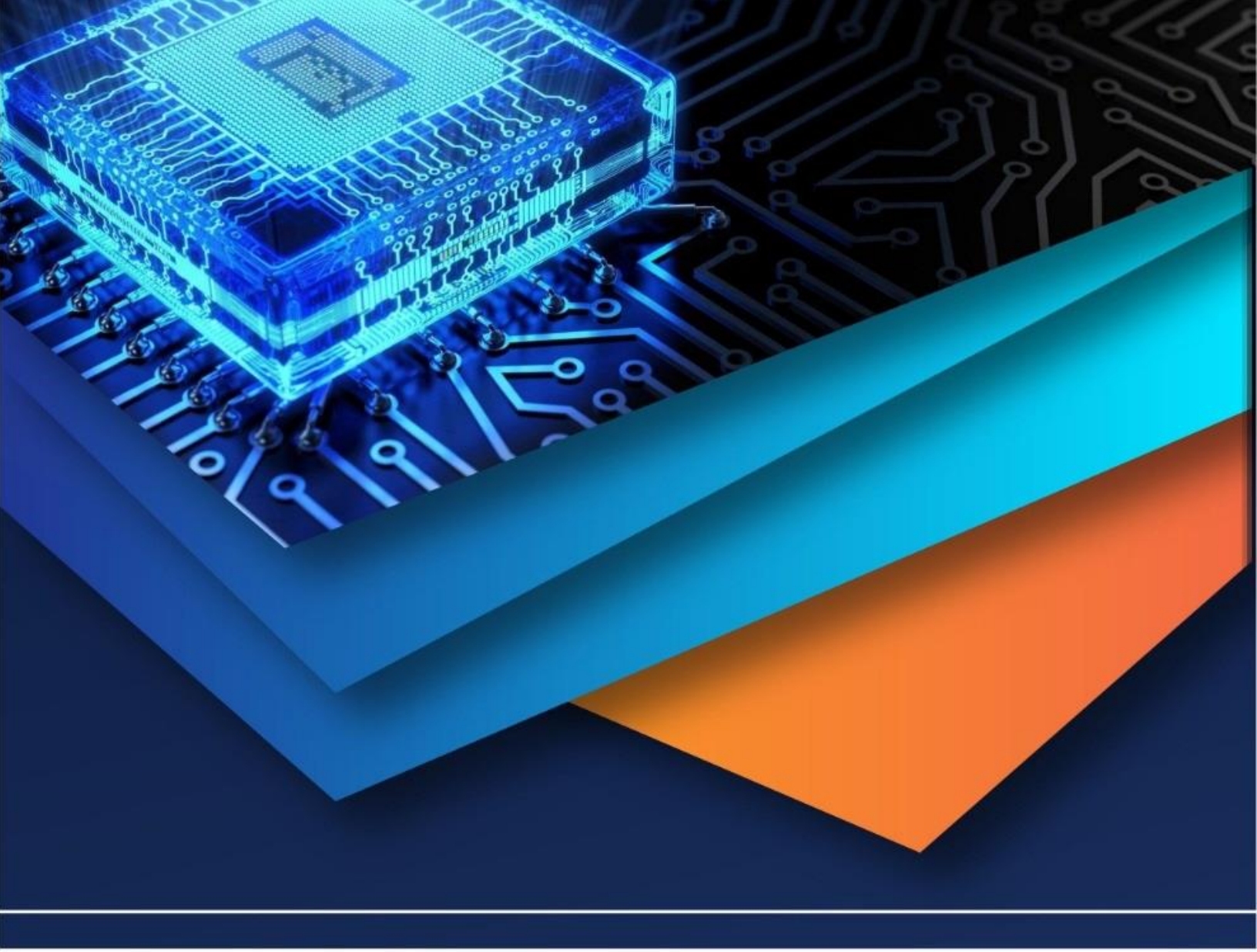
[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, “Toward generating a new intrusion detection dataset and intrusion traffic characterization,” in Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP), Funchal, Portugal, 2018, pp. 108–116, doi: 10.5220/0006639801080116.

[17] S. Moualla, K. Khorzom, and A. Jafar, “Improving the performance of machine learning-based network intrusion detection systems on the UNSW-NB15 dataset,” Computational Intelligence and Neuroscience, vol. 2021, Art. no. 5557577, 2021, doi: 10.1155/2021/5557577.

[18] S. More, M. Idrissi, H. Mahmoud, and A. T. Asyhari, “Enhanced intrusion detection systems performance with UNSW-NB15 data analysis,” Algorithms, vol. 17, no. 2, Art. no. 64, 2024, doi: 10.3390/a17020064.

[19] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, “Network anomaly detection: Methods, systems and tools,” IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303–336, 2014, doi: 10.1109/SURV.2013.052213.00046.

[20] A. Aldweesh, A. Derhab, and A. Z. Emam, “Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues,” Knowledge-Based Systems, vol. 189, Art. no. 105124, 2020, doi: 10.1016/j.knosys.2019.105124.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)