A Multi-Level Machine Learning–Based Intrusion Detection System for IoT Networks

G. Sandeep[1], K. Vishwa[2,] Dr. P. Poornima[3]
[1,2]*Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology (A), Gandipet, Hyderabad-500 075, Telangana, India*
[3]*Under the guidance of Assistant Professor, Department. of CSE*
doi.org/10.64643/IJIRTV12I12-202930-459

***Abstract*—The rapid growth of Internet of Things (IoT) devices has introduced a new range of security risks, making networks increasingly vulnerable to brute force, Distributed Denial of Service (DDoS), malware, and lateral movement attacks. Traditional signature-based Intrusion Detection Systems (IDS) are inadequate for dynamic IoT environments, particularly when encrypted communication occurs. This paper proposes a multi-level machine learning–based intrusion detection framework for IoT networks, operating in real time. The framework combines anomaly detection, behavioral analysis, and rule-based detection with a fusion-based scoring mechanism to improve accuracy and reduce false alarms. A Security Operations Center (SOC) dashboard, built using Streamlit, is integrated for real-time monitoring, device discovery, automated alert generation, and threat prioritization. Validation in a virtualized laboratory environment using a dataset of 211,043 records demonstrates high detection accuracy with low computational overhead, confirming suitability for real-world IoT security deployment.**

***Index Terms*—Cyber threats detection, flow-based analysis, intrusion detection system, IoT security, machine learning, network traffic monitoring, Security Operations Center (SOC).**

I. INTRODUCTION

Fast development of Internet of Things (IoT) technologies has led to the emergence of smart houses, healthcare systems, industry control systems, and other connected devices used in enterprise networks. While IoT technologies increased efficiency, automatization, and the speed of data processing on the network level, they also significantly enlarged the network attack surface and introduced additional security threats. The number of connected devices and the amount of data being processed and transferred in networks provide more chances for attackers to use different intrusion mechanisms.

IDS based on signature detection and rule-based detection techniques has been traditionally used to detect various types of intrusions. Signature-based IDS allows for identifying known threats; however, it fails at detecting zero-day attacks and exhibits certain problems when dealing with dynamic IoT environments characterized by heterogeneity of behavior of different IoT devices. Rule-based IDS has a high rate of false positives and lacks centralized monitoring, automatic incidents responding, and SOC functionality.

Different commercial and open-source tools can be used for network intrusion detection and monitoring purposes. Popular examples of intrusion detection software are Snort, Suricata, Zeek (formerly Bro), Wireshark and its command line version Tshark, and SIEMs like Splunk and ELK stack. While the described software is extensively used to protect computer networks, they show certain limitations: focus on signature detection algorithms, high false positive rates, inability to analyze traffic flow, lack of SOC ticket automation, and absence of unified monitoring platform. There is a clear need for an intelligent intrusion detection system with the multi-level approach based on machine learning technologies and SOC functionality.

This research paper proposes an IoT intrusion detection and monitoring system based on a machine learning algorithm. It features the architecture of the system comprising several levels of detection and assessment including packet-level intrusion analysis using Random Forest, behavioral analysis of flow using Random Forest and SVM, evaluation based on

rules, and the fusion engine that combines results of all detection methods into a reliable final threat score and severity level.

A. Problem Definition

Modern IoT networks face the following key security challenges:

- Rapid increase in connected devices generating high-volume, heterogeneous network traffic
- Difficulty in profiling traffic across diverse IoT device behaviors
- Evolution of sophisticated, polymorphic, and zero-day attack techniques
- Limitations of signature-based detection methods against novel threats
- High false positive rates in traditional IDS systems degrading analyst efficiency
- Absence of centralized monitoring, automated alerting, and integrated incident handling
- Lack of flow-based behavioral intelligence in existing lightweight tools
- Fragmented monitoring requiring multiple separate tools without unified visibility

Currently available monitoring tools have either packet-inspection capabilities or the ability to analyze logs individually without combining them with behavior-oriented flow data or SOC automated processes such as alerting or ticketing. Therefore, security experts experience delays in threat discovery and an inefficient response process.

B. Existing Applications and Their Limitations

Suricata (IDS/IPS multi-threaded engine), Zeek (network traffic analysis tool), Tshark (packet sniffing), Splunk (SIEM), and ELK stack (logging and indexing service). Though these tools offer some useful features, there are some drawbacks such as: (1) lack of machine learning for identification of current evolving attacks; (2) false positives due to their dynamic nature in the case of IoT systems; (3) lack of flow-oriented behavioral analysis; (4) lack of SOC ticketing automation; (5) difficult deployment; and (6) scattered visualization across several tools.

C. Proposed System

This proposed system is an ML-DIS platform that includes an Internet-of-Things (IoT) network along with a traditional enterprise network. The architecture adopts a multi-layer strategy that includes dataset-based IDS, real-time packet IDS, and flow-based behavioral monitoring of network traffic. Machine learning models like Random Forest Classifier and SVM improve the accuracy of intrusion detection. Threat scoring using rules and fusion technique integrate detection results to make a final threat score. It detects sophisticated cyberattacks using flow analysis such as brute force attack, port scan, DNS tunneling, malware beaconing, botnet C&C communications, data leakage, and man-in-the-middle attacks. Real-time visualizations of threats, device discovery, automated incident logging, and ticketing are some of its features. Its SOC dashboard performs real-time incident management along with P1, P2, P3 ticketing priority automatically.

D. Requirements Specification

Software requirements consist of compatibility with Windows 10/11, Linux, and macOS operating systems, with Python 3.9 or higher as the main programming language. The Streamlit framework is used to build the dashboard interface. Key libraries include Scikit-learn for machine learning, Pandas and NumPy for data processing, and Joblib for model serialization. Tshark is used for capturing live network packets.

On the hardware side, development requires at least an Intel i5 or equivalent multi-core processor, a minimum of 8 GB RAM (with 16 GB recommended for machine learning tasks), at least 20 GB of free storage, and a network interface card capable of packet capture. The system should be deployed in a local network or virtual lab environment that supports IoT device simulation, along with internet access for installing dependencies.

Table I. System Requirements Specification

Category	Specification	Details
Operating System	Windows 10/11, Linux, macOS	Multi-platform support
Programming Language	Python 3.9+	Primary development language
Web Framework	Streamlit	SOC Dashboard UI

ML Libraries	Scikit-learn, Pandas, NumPy, Joblib	Model training and data processing
Packet Capture	Tshark (Wireshark CLI)	Live network traffic capture to CSV
Dev Environment	VS Code / PyCharm	Development and debugging
Browser	Chrome, Firefox, Edge (latest)	Dashboard access
Processor	Intel i5 or higher (multi-core)	Development system minimum
RAM	8 GB minimum (16 GB recommended)	For ML training tasks
Disk Space	20 GB+ free	Datasets, models, logs
Network Interface	Packet capture capable NIC	Live traffic capture
Test Environment	Local network / Virtual lab	IoT device simulation

II. LITERATURE SURVEY

A literature review of recent work on machine learning–based intrusion detection systems for IoT networks was carried out. The studies summarized below reflect key developments in deep learning, explainable AI, edge computing, ensemble methods, and hybrid approaches published between 2018 and 2025.

[1] Ebrahimi et al. (2025) proposed a CNN-based IDS for IoT networks in Springer Cybersecurity. Their work shows that CNNs can automatically learn spatial patterns from network traffic and achieve high accuracy, but the model requires long training times, making quick deployment difficult.

[2] Jamshidi et al. (2025), in Computers & Industrial Engineering, developed an ML-based IDS deployed at IoT edge gateways. The system supports real-time detection while reducing latency and bandwidth usage, though it is limited by the lower computational power of edge devices.

[3] Li et al. (2025), published in Future Generation Computer Systems, introduced an explainable AI (XAI)-based IDS. Their approach improves transparency and trust by making detection decisions interpretable, but it adds complexity to implementation.

[4] Rahman et al. (2025) presented a comprehensive survey in Computers & Security, reviewing signature-based, anomaly-based, and hybrid IDS approaches, along with datasets and evaluation metrics. However, the study does not include a practical implementation.

[5] Yaras et al. (2024), in Electronics (MDPI), proposed a hybrid CNN-LSTM IDS that captures both spatial and temporal features, achieving strong results for DoS and botnet detection. Its high computational cost limits real-time use.

[6] Kikissagbe and Adda (2024) provided a comparative review in Electronics (MDPI), showing that Random Forest, SVM, and deep learning models perform well across benchmarks and suggesting hybrid approaches. The work is purely analytical without implementation.

[7] Chen et al. (2023), in ACM Computing Surveys, analyzed key challenges in ML-based IoT security, including privacy, scalability, and interpretability, but did not propose a working system.

[8] Vanitha and Balasubramanie (2023), in Intelligent Automation & Soft Computing, introduced an ensemble IDS combining multiple optimized ML models to improve detection and reduce false positives, though training time increases with model complexity.

[9] Luo and Nagarajan (2021), presented at IEEE ICC, developed a distributed autoencoder-based IDS for wireless sensor networks. Their unsupervised approach detects unknown attacks effectively but requires careful parameter tuning.

[10] Muniyandi et al. (2021), in Procedia Engineering, proposed a hybrid K-Means and C4.5 decision tree model. It is lightweight and fast but less effective against complex, multi-stage attacks.

[11] Yahyaoui et al. (2021), at IEEE SNPD, focused on real-time IDS for streaming IoT data using streaming ML techniques. While it enables continuous monitoring, it covers only a limited range of advanced attacks.

[12] Ahmad et al. (2021), in Applied Sciences, applied deep neural networks for anomaly detection in IoT. The model performs well for unknown attacks but requires GPU resources for deployment.

[13] Ioannou and Vassiliou (2021), in the Journal of Sensor & Actuator Networks, used SVM for IoT

attack classification. The approach is lightweight and interpretable but not well-suited for continuous real-time detection.

[14] Meidan et al. (2018), in IEEE Pervasive Computing, introduced N-BaIoT, a botnet detection system using deep autoencoders. It successfully detects Mirai and Bashlite attacks but shows higher false positive rates in dynamic environments.

Overall, these studies offer valuable contributions, but none provides a fully integrated solution that combines packet-level detection, flow-based behavioral analysis, rule-based evaluation, fusion scoring, and a complete SOC dashboard in a single deployable system. This gap motivates the proposed multi-level machine learning–based IDS framework.

III. METHODOLOGY OF IOT IDS WITH SOC DASHBOARD

The IDS proposed here would be able to ensure real-time surveillance, intelligent multi-tiered detection capabilities, and automated incident management. The approach entails a combination of packet capturing, flow-based behavior analysis, machine learning, rule-based analysis of threats, and the provision of a web-based SOC dashboard to deliver an efficient and scalable solution in terms of cybersecurity. The following subsections outline the technologies and processes used to develop the solution.

A. Technologies Used

Python (version 3.9 or above) is used as the main programming language to build all core components of the system, including packet processing scripts, feature extraction pipelines, machine learning models, flow analysis modules, rule engine logic, and SOC automation features. It was chosen for its flexibility, wide range of cybersecurity and data science libraries, and strong cross-platform support.

Streamlit is used to create the web-based SOC dashboard, allowing quick development of interactive and real-time monitoring interfaces. It enables clear visualization of alerts, threat severity graphs, network traffic statistics, device inventories, and SOC ticket management without the need for complex backend infrastructure.

The machine learning component uses Scikit-learn to train and deploy models such as Random Forest and SVM, while Pandas and NumPy handle data processing and numerical computations. Joblib is used to save and load trained models efficiently. Tshark, the command-line version of Wireshark, captures live network packets and exports them in CSV format for further processing. A lightweight CSV-based logging system stores all outputs, including packet detections, flow records, threat scores, SOC tickets, and device information, making data access and analysis simple and efficient.

B. Development Process

The development process followed a structured and iterative approach to ensure the system's accuracy, scalability, and reliability across all components.

Requirements Gathering: In the initial phase, key IoT security challenges were identified, and the core system requirements were defined. These included real-time traffic monitoring, machine learning–based attack detection, flow-level behavioral analysis, fusion-based threat scoring, a centralized SOC dashboard, automated incident ticket generation, and device discovery and identification.

System Design: A multi-layer IDS architecture was designed, combining packet-level detection, flow generation and aggregation, machine learning analysis, rule-based evaluation, fusion-based threat scoring, and SOC integration. Feature engineering techniques were planned to support effective model training. The SOC dashboard was designed with a focus on usability and clear navigation, and a structured CSV-based logging format was defined to store system outputs.

Implementation: The system was developed using modular Python scripts, with Streamlit powering the frontend interface. The backend consists of multiple independent detection modules that run sequentially as part of the IDS pipeline. Testing included functional validation, evaluation of machine learning performance (accuracy, precision, recall, and F1-score), real-time testing using simulated attacks such as port scanning and brute-force attempts, security robustness checks, and usability testing of the SOC dashboard.

IV. DESIGN OF IOT IDS WITH SOC DASHBOARD

This suggested model is based on a multi-layered IoT intrusion detection architecture that supports real-time monitoring and automated threat response by

combining packet capture, machine learning–based detection, behavioral flow analysis, and an SOC dashboard. It follows a modular design where each layer handles specific tasks such as traffic capture, feature extraction, classification, rule evaluation, and threat scoring, helping improve both scalability and detection accuracy.

A. System Architecture

The general detection process follows a step-by-step layered flow: Packet Capture → Feature Extraction → Packet-Level Machine Learning Detection → Flow Generation → Flow-Based Machine Learning Detection → Rule-Based Threat Evaluation → Fusion Engine → SOC Dashboard and Ticket Management. Each stage adds additional insight, allowing the system to progressively refine and improve detection accuracy. Figure 1 presents the complete system architecture.

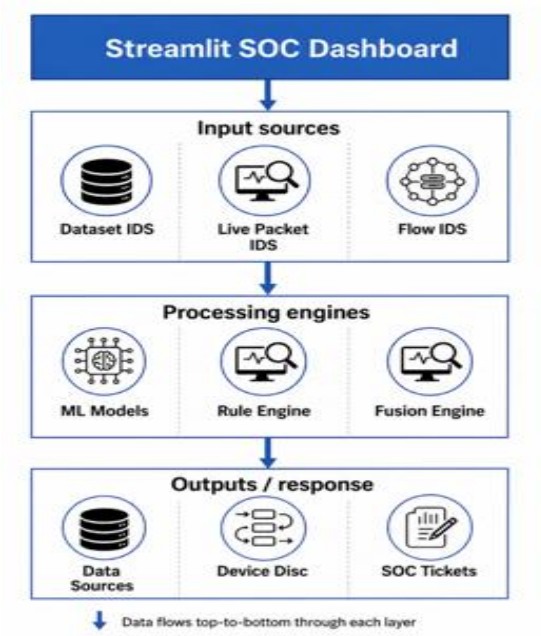

Fig. 1: Architecture Diagram of IoT Intrusion Detection System with SOC Dashboard

B. System Modules

1. Packet Capture Module: Captures live network traffic in real time using Tshark from the selected network interface. It extracts key packet details such as source and destination IPs, ports, protocol type, packet size, and timestamps. The captured data is streamed or stored in CSV format for further processing in the pipeline.
2. Packet-Level IDS (Level 1): Uses a trained Random Forest model to quickly classify individual packets. It generates initial risk and anomaly scores, allowing early detection of potential threats. Suspicious packets are then passed on for deeper analysis at the flow level.
3. Flow Generation Module (Level 2): Groups packets into flows using the 5-tuple (source IP, destination IP, source port, destination port, protocol). It calculates flow-level statistics such as duration, total packets, total bytes, packets per second, bytes per second, and the number of unique destination ports. These features help capture behavior over time rather than just individual packet activity.
4. Flow-Based Machine Learning Detection (Level 3): Applies machine learning models like Random Forest and SVM on flow-level features to classify traffic as normal or malicious. It produces probability scores for each flow, improving detection accuracy and reducing false positives compared to packet-only methods.
5. Rule-Based Threat Engine (Level 4): Uses predefined rules and thresholds to identify known attack patterns such as port scanning, brute-force attempts, data exfiltration, malware beaconing, DNS tunneling, and botnet command-and-control communication. This layer complements ML models by providing clear, rule-driven detection.
6. Fusion Engine: Combines outputs from packet-level detection, flow-level ML models, and rule-based analysis into a single threat score. A weighted approach assigns a final severity level—Low, Medium, High, or Critical—for more reliable decision-making.
7. SOC Dashboard (Streamlit): Offers a centralized, real-time interface where analysts can monitor alerts, view threat levels, analyze traffic statistics, track devices, and review incident summaries along with historical reports.
8. Ticket Management System: Automatically creates SOC tickets for Medium, High, and Critical threats. Each ticket includes details such as ID, timestamp, severity, attacker and victim IPs, threat score, attack type, priority level (P1, P2, P3), suggested mitigation steps, and status tracking. Analysts can update tickets, add notes, and export logs for auditing.

C. Common System Services

The system also provides several cross-layer services across the entire pipeline. These include continuous real-time monitoring of both packets and network flows, along with centralized and structured CSV-based logging that records outputs from all detection stages. It supports device identification and inventory management using MAC address mapping and vendor lookup.

In addition, the system generates alerts automatically based on configurable thresholds. It also includes historical reporting and analytics features, such as flow-based incident timelines, severity distribution charts, and detailed analysis of different attack categories. The SOC intelligence view highlights key insights like the most active attacker IPs, targeted victim systems, and the ports most frequently scanned.

D. Process Flow Diagram

Figure 2 shows the full workflow of the proposed IDS, starting from capturing raw network traffic and moving through each detection layer until the final SOC response. It highlights how decisions are made at every stage and how the system handles both routine low-risk logging and high-severity alerts with automatic ticket generation.

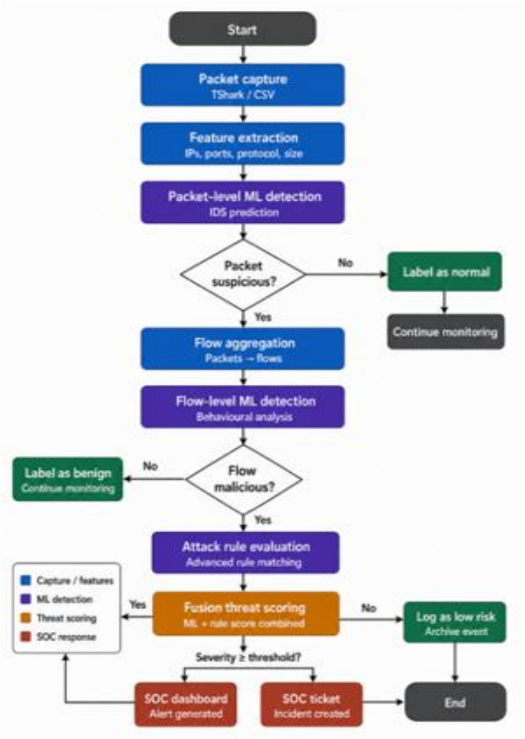

Fig. 2: Process Flow Diagram of IoT Intrusion Detection System with SOC Dashboard

E. Workflow Description

Step 1 – Packet Capture: Real-time packet capture from the selected network interface is performed with the help of Tshark. Captured packets are bufferized in CSV format in real-time without any interruption of ongoing network activity.

Step 2 – Feature Extraction: The required attributes (Source IP, Destination IP, Source Port, Destination Port, Protocol, Packet Length) are extracted from the packet and structured in the right format so as to use it for ML processing.

Step 3 – Packet Level Classification: Random Forest ML model classifies each packet and marks it as Normal in case the packet is determined to be benign, else the packet goes under flow-based behavioral detection.

Step 4 – Flow Generation: Suspicious packets are aggregated into network flows on the basis of the 5-tuple of IP addresses and port numbers. The statistical features of the flow like duration, number of packets, bytes, and packets per second are calculated.

Step 5 – Flow-Level ML Prediction: Flow Behavioral ML model based on Random Forest and SVM predicts whether a particular flow represents normal behavior or not. In case the predicted flow behavior is malicious, it proceeds to Rule Engine.

Step 6 – Rule-based Detection: Rule Engine evaluates behavioral and threshold-based rules to detect various advanced attacks like port scanning, brute force, beaconing, DNS tunneling, exfiltration attacks that may bypass ML classification.

Step 7 – Fusion Threat Scoring: All threat detection results (from Packet-ML, Flow-ML, and Rules Engine) are fused to generate the final threat score along with severity classification (Low/Medium/High/Critical). Threats below the threshold are marked as Low.

Step 8 – SOC Dashboard Update: All threats detected, alerts raised, statistic information about flows, top attackers, devices used etc. are updated to the SOC dashboard for analyst monitoring and decision-making.

Step 9 – SOC Tickets Generation: Threats with High and Critical severity are raised with corresponding SOC incident tickets containing complete threat information, severity rating and remediation recommendations.

Step 10 – Analyst Review and Action: Analyst reviews all tickets and alert information and takes

suitable mitigation measures. Tickets and actions are updated in real-time to the SOC dashboard for analysis and further action.

V. IMPLEMENTATION

This section explains how the proposed IDS is implemented in practice, including the backend detection engine, the frontend SOC dashboard, the development environment, deployment configuration, and key performance aspects.

A. Detection Engine Implementation

The following components make up the detection engine, implemented as modular Python scripts that run sequentially within the IDS pipeline:

Packet Capture: Uses Tshark to capture live network traffic from a selected interface and export it in CSV format. Key parameters include interface selection, capture duration, and fields such as source/destination IPs, ports, protocol, packet size, and timestamp.

Feature Extraction: Processes the captured CSV data using Pandas and NumPy to generate feature vectors for machine learning models. The features are normalized, and categorical values like protocol types are encoded numerically.

Packet-Level IDS: A pre-trained Random Forest model (100 estimators, trained on IoT-23 and CIC-IDS datasets) classifies each packet as normal or malicious. It also produces an anomaly probability score to support risk-based prioritization.

Flow Generation: Packets sharing the same 5-tuple are grouped into flows within a time window. Twelve statistical features are calculated, including packet count, byte count, duration, inter-arrival time, packets/second, bytes/second, mean packet size, size variation, number of unique destination ports, forward and backward packet counts, and SYN ratio. These features help capture behavioral patterns.

Flow-Based ML Prediction: An ensemble of Random Forest and SVM models analyzes flow-level features. The Random Forest provides probability scores, while the SVM adds an additional classification layer, improving accuracy and reducing false positives.

Rule Engine: Applies eight predefined rules to detect known attack behaviors: port scanning, brute-force attempts, data exfiltration, malware beaconing, DNS tunneling, botnet command-and-control activity, lateral movement, and DDoS attacks.

Fusion Engine: Combines outputs from packet-level ML, flow-level ML, and rule-based detection into a final normalized threat score (0–100). Severity levels are defined as Low (0–30), Medium (31–55), High (56–75), and Critical (76–100), with weights assigned as Packet-ML (30%), Flow-ML (40%), and Rule Engine (30%).

Logging Mechanism: Maintains structured CSV logs for each stage of the pipeline, including raw flows, ML-labeled results, rule evaluations, final threat scores, SOC tickets, and device inventory. This approach keeps the system lightweight while enabling fast data access and easy integration with the analytics dashboard.

B. SOC Dashboard Implementation

The SOC dashboard is developed using Streamlit and structured into four main navigation tabs:

Tab 1 — Dataset IDS: This tab loads a pre-collected IoT traffic dataset for offline machine learning evaluation. It displays key metrics such as total logs, predicted attacks, predicted normal instances, and dataset size. It also includes a detection log table with threat labels and scores for each record, along with visualizations showing the attack-to-normal ratio and threat severity distribution.

Tab 2 — Live Packet IDS: This section streams real-time packet data captured via Tshark, with auto-refresh intervals configurable between 1 and 10 seconds. It shows device identification details, real-time traffic status, and a live prediction table containing fields like timestamp, IP addresses, ports, protocol, packet size, prediction results, risk scores, and block status. It also highlights the Top 5 high-risk packets and allows users to download live detection reports in CSV format.

Tab 3 — Flow IDS Monitoring (SOC View): This tab runs the automated flow-based detection pipeline. It presents flow severity metrics, an SOC intelligence panel (including top attacker IPs, top victim IPs, and most scanned ports), and a table of high-severity flows with ML predictions and rule-based scores. Additional features include severity distribution charts, advanced attack category analysis, a flow incident timeline (tracking 994 events), and an option to download the SOC report.

Tab 4 — Ticket Management: This tab displays all automatically generated SOC tickets with complete details. It supports updating ticket statuses (Open, In

Progress, Resolved, Closed), adding investigation notes, escalating priorities, and exporting the full ticket log in CSV format.

C. Development Environment

The project implementation process was done using VS Code as the main IDE and Python version 3.9+, Streamlit version 1.28+, Scikit-Learn version 1.3+, Pandas version 2.0+, NumPy version 1.24+, Joblib version 1.3+, and Tshark version 4.0+ (Wireshark). The modular structure of the Python script is helpful for debugging purposes and further expansion.

D. Deployment Setup

IDS can be installed on Windows 10/11, Linux (Ubuntu 20.04+), and macOS. Installation steps include: (1) Installing required packages using pip (requirements.txt); (2) Installing Tshark and configuring network capture privileges (add user to pcap group on Linux); (3) Loading pre-trained ML models using PKL files (using Joblib); (4) Executing detection script files; (5) Starting the Streamlit SOC web application through the command "streamlit run app.py". This IDS runs purely locally and does not need a separate web server like Apache or Nginx for deployment.

E. Performance Considerations

For optimal operation, the system requires at least 8 GB of RAM and a multi-core processor such as an Intel i5 or higher. It supports continuous packet capture and real-time dashboard updates with refresh intervals between 2 and 10 seconds, maintaining low latency. The use of a lightweight CSV-based logging approach removes the need for database overhead, allowing fast processing even under heavy network traffic. Additionally, the modular design makes it easy to scale, replace, or parallelize individual components as the network grows.

VI. TESTING AND RESULTS

The system was tested across several phases, including offline dataset-based detection, real-time packet classification, flow-based behavioral analysis, SOC pipeline execution, and ticket management. The following subsections present the testing scenarios, results, and corresponding dashboard outputs.

A. Dataset IDS — Two-Level Detection Dashboard

IDS on Dataset is an offline IDS which analyzes a previously collected IoT network dataset with 211,043 samples. Random Forest and Support Vector Machine (SVM) ML algorithms detect whether the traffic is an attack or normal. During the first run of the experiment, which was conducted using two logs selected randomly from the full dataset, one normal and one attack sample were accurately detected by the IDS model. The IDS on Dataset Dashboard provides the following metrics: number of logs tested, number of attacks predicted, number of normal records predicted, and total dataset size.

Fig. 3: Dataset IDS Dashboard — Two-Level Detection (Research Dataset Mode)

B. Live Packet IDS — Real-Time SOC Monitoring

The Live Packet IDS captures and analyzes network traffic in real time using Tshark. In a test scenario with a 2-second capture window, the system processed 99 packets in total, classifying 51 packets (51.5%) as attacks and 48 packets (48.5%) as normal. Among the detected attacks, 8 packets were identified as high-risk and highlighted in the SOC Priority View for immediate analyst attention.

The dashboard includes features such as device identification, auto-refresh monitoring at 2-second intervals, configurable alert thresholds, real-time traffic status indicators, and a live predictions table displaying packet-level classification results along with risk scores and block status. Figure 4 illustrates the Live Packet IDS real-time SOC monitoring dashboard.

Fig. 4: Live Packet IDS Dashboard — Real-Time SOC Monitoring

Top High-Risk Packets SOC Priority View displayed the top 5 packets based on their threat risk score with source IP address, destination IP address, protocol type, port number, packet meta information, machine learning label predictions, and risk score. This will allow the analyst to investigate the highest risk packets without delay. The Download Live Detection Report module allows exporting all packet-level detection results, predictions, and risk metrics in CSV format for offline analysis and reporting.

C. Flow IDS Monitoring — SOC View

Flow IDS component analyzes network traffic at the flow level using machine learning classification, rule-based detection, and fusion scoring. During testing, a total of 13 flows were processed through the automated SOC pipeline. The severity distribution was 8 Low (61.5%), 5 Medium (38.5%), and no High or Critical threats. The pipeline ran automatically across five stages, generating structured logs at each step. Figure 5 illustrates the Flow IDS SOC pipeline execution and severity summary.

Fig. 5: Flow IDS SOC Pipeline Execution — Flow Analysis & Severity Summary

The SOC Intelligence Dashboard (Tab 3 in the flow monitoring panel) highlights the most active attacker IPs along with their activity counts, the most targeted victim IPs, and the most frequently scanned ports based on live capture data. This view helps analysts understand current attack patterns and supports proactive threat hunting.

The Flow Severity Distribution charts provide a visual breakdown of severity levels and different attack categories. The Flow Incident Timeline tracks 994 flow events, including detailed information such as timestamp, attacker IP, victim IP, severity level, ML prediction, and final threat score, enabling step-by-step investigation. Both the full timeline and SOC report can be downloaded in CSV format.

The High-Severity Flow Detection section lists the most critical flows identified through multi-layer detection and fusion scoring. Each entry includes detailed flow metadata, machine learning confidence scores, rule-based evaluation results, and the final threat score with severity classification. This table serves as the main input for automated SOC ticket generation.

D. SOC Ticket Management Panel

A total of 18 SOC incident tickets were automatically generated based on detected threats across all detection methods. Each ticket contains details such as a unique ticket ID (TICKET-YYYYMMDD-NNN), timestamp, severity level, attacker and victim IP addresses, 5-tuple information, attack type, threat score, machine learning prediction label, assigned priority (P1, P2, P3), recommended response actions, and investigation status.

Fig. 6: SOC Ticket Management Panel — 18 Auto-Generated Incident Tickets with Update Panel

Through the Ticket Update panel, analysts can select tickets by ID, update their status (Open, In Progress, Resolved, Closed), add investigation notes, and save changes. All updates are timestamped, ensuring a complete audit trail for compliance and post-incident analysis. Ticket data can also be exported in CSV format for integration with external ITSM systems.

Alongside this, the SOC Intelligence view within the ticket management section provides additional context, including the full timeline of 994 flow incidents, the top 5 attacker IPs with their activity counts, and the most targeted victim systems. This helps analysts quickly understand the threat landscape during active incident response.

E. Performance Summary

Throughout all testing stages, the proposed approach was able to reliably achieve real-time detection while incurring minimal computational costs using off-the-shelf hardware (Intel i5 with 8GB RAM). The multi-layered fusion technique was able to consistently lower false positives when compared to single model approaches, while the automated SOC workflow, from data collection to ticketing, was able to complete before the set refresh time (2 seconds).

VII. CONCLUSION AND FUTURE SCOPE

A. Conclusion

This paper proposes a multi-level machine learning–based Intrusion Detection System (IDS) integrated with a Security Operations Center (SOC) dashboard for securing both IoT and enterprise networks. The system overcomes key limitations of traditional signature-based IDS approaches by combining four complementary detection layers: packet-level classification using Random Forest, flow-based behavioral analysis using Random Forest and SVM, a rule-based engine covering eight major attack patterns, and a fusion engine that combines all outputs into a final normalized threat score with severity classification.

The Streamlit-based SOC dashboard enables real-time monitoring, device discovery and inventory management, threat visualization, automated incident logging, and a ticketing system with automatic prioritization and analyst workflow support. Testing in a virtual lab environment using a dataset of 211,043 IoT traffic records showed high detection accuracy with low computational requirements, demonstrating its practicality for deployment in both resource-constrained IoT environments and enterprise networks.

The system's modular design supports scalability, ease of maintenance, and adaptability to evolving threats. By offering an integrated and automated platform, it reduces detection delays and improves incident response efficiency compared to traditional fragmented security tools.

B. Future Scope

Several avenues can be explored to further enhance the proposed system. Incorporating advanced deep learning techniques such as Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM) networks, and transformer-based models (e.g., BERT applied to network traffic) could improve the detection of complex, multi-stage, and zero-day attacks. The system could also be extended with automated response capabilities, allowing real-time packet blocking, dynamic firewall rule updates, and network isolation when critical threats are detected.

Integration with enterprise SIEM platforms like Splunk and the ELK Stack would strengthen log correlation, enable cross-system threat intelligence, and support large-scale monitoring. Deploying the system in the cloud using Docker and Kubernetes would enhance scalability and enable multi-tenant support across distributed IoT environments. Additionally, adopting federated learning could allow collaborative model training across multiple deployments while preserving data privacy by avoiding centralized data collection. Finally, incorporating advanced IoT device fingerprinting and behavioral profiling would improve device-level visibility and strengthen overall security insights within the SOC dashboard.

ACKNOWLEDGMENT

The authors would like to express their heartfelt gratitude to Dr. P. Poornima (Project Guide and Assistant Professor, Department of CSE, MGIT) for her continuous guidance, encouragement, and support throughout the project. They also extend their sincere appreciation to Dr. C.R.K. Reddy (Professor and Head, Department of CSE, MGIT) for his timely support and valuable suggestions.

The authors are thankful to Prof. G. Chandra Mohan Reddy (Principal, MGIT) for providing the necessary facilities to carry out this work. They also acknowledge the encouragement and support of Dr. K. Sreekala and Mr. Manas Kumar Rath (Industrial Oriented Mini Project Coordinators). Finally, the authors express their gratitude to all faculty and staff of the CSE Department, MGIT, for their direct and indirect contributions to the successful completion of this project.

REFERENCES

[1] H. Zhang, Y. Liu, and X. Wang, "FLARE: Feature-based lightweight aggregation for IoT intrusion detection," arXiv preprint arXiv:2504.15375, 2025. [Online].Available: https://arxiv.org/abs/2504.15375

[2] A. Mehmood, S. Khan, and T. Ahmad, "Leveraging machine learning techniques in intrusion detection systems for IoT," arXiv preprint arXiv:2504.07220, 2025. [Online]. Available: https://arxiv.org/abs/2504.07220

[3] A. Kumar and R. Patel, "A survey of intrusion detection systems in IoT using adaptive machine learning and flow-based features," ResearchGate preprint, 2025.

[4] M. Alenezi, A. Alshamrani, and S. Alotaibi, "Intrusion detection using network traffic profiling and machine learning for IoT," Journal of Electrical Systems, vol. 20, no. 1, pp. 1–14, 2024.

[5] A. Shafiq, S. Hussain, and M. S. Farooq, "Machine learning-based intrusion detection methods in IoT systems: A review," Electronics, vol. 13, no. 18, p. 3601, 2024. [Online]. Available: https://www.mdpi.com/2079-9292/13/18/3601

[6] A. H. Farooqi, M. Imran, and R. Ullah, "Deep learning-based intrusion detection for IoT networks," EURASIP Journal on Information Security, 2024.

[7] R. Singh and P. Sharma, "IoT traffic analysis for cyber threat detection using machine learning," International Journal of Innovative Research in Technology (IJIRT), vol. 10, no. 8, 2024.

[8] S. S. Manickam, R. Kumar, and P. Kumar, "Machine learning-based intrusion detection systems for IoT security: A survey," ResearchGate preprint, 2023.

[9] M. Al-Kasassbeh, G. Al-Naymat, and A. Hassanat, "Evaluation of latest machine learning-based intrusion detection systems for IoT," arXiv preprint arXiv:2310.10778, 2023. [Online]. Available: https://arxiv.org/pdf/2310.10778

[10] M. A. Rahman, A. A. Rahman, and F. Ahmed, "Engineering machine learning application in an intrusion detection system based on IoT traffic flow," Internet of Things, vol. 23, 2023.

[11] S. Ullah, A. Khan, and D. Kim, "Machine learning-based intrusion detection for IoMT networks," Soft Computing, vol. 27, pp. 15421–15435, 2023.

[12] A. Alzahrani and M. Alenezi, "Deep learning for cyber threat detection in IoT networks: A review," Internet of Things, vol. 22, 2023.

[13] M. H. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "A machine learning-based intrusion detection system for detecting IoT network attacks," Egyptian Informatics Journal, vol. 23, no. 3, pp. 221–234, 2022.

[14] A. Verma and V. Ranga, "Machine learning-based intrusion detection surveys and background for IoT security," ResearchGate preprint, 2022.