Open Access Research Journal of
Science and Technology
Journals home page: https://oarjst.com/
ISSN: 2782-9960 (Online)

OARJ OPEN ACCESS RESEARCH JOURNALS

(RESEARCH ARTICLE)

Adaptive edge-based behavioural anomaly detection and response system for smart home IoT Networks

Nakshatra Rane *, Sanika Mahajan, Satvik Chaudhari and Sai Patil

School of Computing, MIT ADT University, Loni Kalbhor, Pune.

Open Access Research Journal of Science and Technology, 2026, 17(01), 089-103

Publication history: Received on 17 May 2026; revised on 24 June 2026; accepted on 26 June 2026

Article DOI: https://doi.org/10.53022/oarjst.2026.17.1.0065

Abstract

The quick expansion of Internet of Things (IoT) devices in smart home settings has made people much more vulnerable to online threats, especially large-scale IoT botnets and zero-day attacks. Traditional intrusion detection systems based on signatures are useless against new and emerging threats [1]. While many solutions rely on centralized processing, lack device-specific modeling, or do not incorporate adaptive enforcement mechanisms, recent approaches make use of behavioral anomaly detection, such as deep learning-based models for IoT traffic analysis [2]. In this paper, we propose an adaptive edge-based IoT security framework that uses unsupervised anomaly detection to learn device-specific behavioral baselines. The suggested system models typical device activity and instantly identifies deviations using the Isolation Forest algorithm [3]. AEIS uses a hybrid anomaly detection approach combining Isolation Forest (unsupervised) for detecting unknown attacks and Random Forest (supervised) for classifying known attack patterns. The framework ensures privacy preservation and lower latency by only analyzing metadata while operating at the network edge. A closed-loop, self-healing security architecture is made possible by a graduated policy engine that applies proportional restriction, isolation, and automated recovery. In comparison to static threshold-based methods, experimental evaluation using a hybrid dataset that combines publicly available IoT botnet traffic and real network metadata [2] shows increased detection accuracy and decreased false positive rates. For smart home and small-scale IoT deployments, the suggested framework provides a scalable, lightweight, and privacy-conscious solution [4].

Keywords: Internet of Things (IoT); Smart Home Security; Anomaly Detection; Edge Computing; Isolation Forest; Behavioral Profiling; Intrusion Detection System (IDS); Device-Specific Modeling; Self-Healing Security; Network Metadata Analysis.

1. Introduction

Traditional homes are now interconnected smart ecosystems due to the quick spread of Internet of Things (IoT) devices in residential settings. Smart cameras, televisions, plugs, sensors, and voice assistants are just a few examples of the devices that constantly communicate over shared networks, increasing operational convenience while also increasing the attack surface. Malware campaigns and extensive IoT botnets have shown how susceptible poorly secured devices can be to lateral network compromise, data exfiltration, and distributed denial-of-service (DDoS) attacks. Static firewall rules and signature-based intrusion detection systems are examples of traditional security measures that frequently fail to protect against unidentified or changing threats. Additionally, a lot of current solutions depend on cloud-based processing, which raises issues with user privacy, scalability, and latency.

Adaptive IoT security is made possible by recent developments in edge computing and anomaly detection. Systems can simulate typical device activity and spot deviations that point to malicious activity thanks to behavioural anomaly detection. However, the majority of current methods either lack integrated response mechanisms or use uniform models across heterogeneous devices, which leads to increased false positive rates and disruptions in operations.

* Corresponding author: Nakshatra Rane

This study suggests an adaptive edge-based IoT security framework that uses unsupervised anomaly detection techniques to learn behavioural baselines specific to individual devices. The system ensures low-latency detection and privacy preservation by operating at the network edge and analysing only metadata. To create a closed-loop, self-healing security architecture, a graduated policy engine applies restriction, isolation, and automated recovery in addition to anomaly detection.

The main goal of this research is to develop, put into practice, and assess a device-aware, lightweight anomaly detection and response framework that improves smart home security while reducing false positives and maintaining network availability. In practical IoT settings, experimental validation shows the viability and efficiency of the suggested strategy.

2. Literature Review

2.1. Comparative Analysis of Existing Methods

Table 2 puts AEIS in broader perspective of intrusion detection techniques by comparing the techniques used, the learning paradigms they abide by, and the limitations associated with the paradigms. This analogy justifies why AEIS chooses to have a dual-model framework rather than one.

Table 1 Compares existing IoT intrusion detection approaches with the proposed AEIS framework.

Method	Technique	Learning Type	Limitations
Signature-Based IDS	Pattern matching using known attack signatures	Supervised (rule-based)	Cannot detect zero-day attacks; requires frequent updates; high maintenance overhead
Cloud-Based ML IDS	Centralized machine learning processing with traffic sent to cloud	Supervised / Unsupervised	High latency; privacy risks due to data upload; not suitable for edge environments
Autoencoder (AE)	Reconstruction error-based anomaly detection using neural networks	Unsupervised (Deep Learning)	High computational cost; requires GPU; not suitable for resource-constrained edge devices
Isolation Forest (AEIS)	Random partitioning and path length-based anomaly detection	Unsupervised	Precision may reduce when normal data is sparse; depends on contamination parameter
Random Forest (AEIS)	Ensemble of decision trees using labelled features	Supervised	Requires labelled data; retraining needed for new attack patterns

The AEIS dual-model architecture addresses, in particular, the limitations that have already been noted. The Isolation Forest provides the capability of zero-days detection without the requirement of labelled data, and the Random Forest, which is further trained on confirmed attack signatures using SMOTE-lite oversampling to eliminate class imbalance, provides a classification of augmented confidence. Together, the two elements are complementary in that the unsupervised model is used to detect new patterns, and the supervised model is used to support and label the existing patterns.

2.2. Supportive Literature

2.2.1. Meidan et al. (2018) – N-BaIoT Dataset and Deep Autoencoder-Based Detection

Meidan et al. proposed N-BaIoT, a network-based detection scheme against the attack of IoT botnets using deep autoencoders. The study presented a publicly available dataset of both benign and malevolent traffic traces of compromised IoT devices. The primary objectives of their strategy were to learn specific patterns of behaviour of the device and identify the instances of deviations caused by the presence of botnets. The researchers demonstrated that deep learning models have a high level of accuracy and efficiency when detecting IoT botnets. The framework, however, was not based on a real-time enforcing, or a recovery mechanism and was focused on the detection. Additionally, deep autoencoders may require increased computation, and this would restrict their use in the deployment of lightweight edges in small environments.

2.2.2. Liu et al. (2008) – Isolation Forest for Anomaly Detection

Liu et al. first proposed the unsupervised anomaly detector, the Isolation Forest (iForest) algorithm, an unsupervised data classification algorithm that uses separation of outliers as its mechanism to detect anomalies rather than examining normal data distributions. Large volumes of data or streaming data are suitable to the technique because it is computationally efficient, linear time, and low-memory. Isolation Forest has been well applied since then in both anomaly and network intrusion detection applications. Nevertheless, the original work does not address issues associated with IoT, such as heterogeneity of devices, behavioral drift and adaptive policy enforcement. This study builds on Isolation Forest by integrating response mechanisms based on edges and device-specific baseline modelling.

2.2.3. Antonakakis et al. (2017) – Understanding the Mirai Botnet

Antonakakis et al. studied the Mirai botnet which played a major role by performing distributed denial-of-service (DDoS) attacks using weak IoT devices. The research has illuminated command-and-control communication patterns, scanning behavior, and malware propagation strategy. The findings are pointers to the weaknesses of traditional perimeter-based defense and also the vulnerability of IoT systems. The study emphasizes the volume and importance of IoT botnets, although it does not offer any suggestions about behavior-driven and adaptive detection techniques that are unique to home networks. This paper bridges this gap through the focus on early detection of behaviour deviation at the network edge.

2.2.4. Ameen et al. (2019) – IoT Security Challenges and Detection Techniques

Ameen et al. conducted a comprehensive discussion of the IoT technologies, security concerns, and mitigation strategies to avert threats. The paper discussed encryption systems, intrusion detection systems as well as strategies of anomaly detection which are applicable in the Internet of Things environment. It illustrated critical challenges of IoT devices such as computational capacity, high false positive, privacy protection, and scalability. The survey also notes that much of the recommendations are still centralized and do not possess the ability to respond adaptively.

3. Proposed System

3.1. System Overview

Autonomous Edge Immune System (AEIS) is a community-based, decentralized method of detecting and combating intrusions in the case of IoT networks. It is not your traditional signature-based arrangement, but rather AEIS observes abnormal behaviour at the boundary and educates itself on what is normal behaviour of each device. The concept is based on behavioural botnet detection such as N -BaIoT, but AEIS relies on lightweight models, which can be deployed on edge hardware rather than massive central autoencoders. It is all about fast, privacy-conscious detection in small-scale IoT areas, e.g. smart home. The entire process is carried out in six steps, which are passive traffic sniffing, feature extraction, baseline modelling, anomaly detection, risk-based decision making, and autonomous response.

3.2. Hybrid Dataset Construction and Data Processing

We created a hybrid dataset combining real network traffic collected using Tshark framework and publicly available IoT attack datasets which can be used for the proposed Autonomous Edge-based Intrusion Detection System (AEIS), The hybrid approach allows us to improve the robustness of the anomaly detection model, since we captured already collected attack patterns and the real-world behaviour.

Firstly, packet capture tools were used to collect the real network metadata from a single node (pc/laptop). Network metadata such as timestamps, packet length, and communication endpoints were extracted, instead of storing heavy data like fully packet payloads. Due to this we were able to ensure privacy along with having enough behavioural feature data for anomaly detection,

TShark was used for packet capture, it extracts the preset packet fields directly into a structured data format.

```
tshark -i 1 -T fields -E header=y \

-e frame.time \

-e ip.src \
```

```
-e ip.dst \

-e frame.len \

-e _ws.col.Protocol > data/raw/local/normal_capture.csv
```

Then the pandas library was used to standardize and clean the collected network traffic. It involved removal of irrelevant fields and handling of missing values.

```
df = pd.read_csv("data/raw/local/normal_capture.csv")

df["ip.src"] = df["ip.src"].fillna("unknown")

df["ip.dst"] = df["ip.dst"].fillna("unknown")

df["_ws.col.protocol"] = df["_ws.col.protocol"].fillna("unknown")
```

The N-BaIoT dataset containing Mirai and Gafgyt botnet traffic was integrated in the dataset in order to strengthen the dataset with real attack scenarios. All CSV files were loaded and merged into a singular dataset using automation techniques.

```
for file in os.listdir(dataset_path):

  if file.endswith(".csv"):

    df = pd.read_csv(os.path.join(dataset_path, file))

    df["label"] = 0 if "benign" in file else 1

    dataframes.append(df)

combined_dataset = pd.concat(dataframes, ignore_index=True)
```

This created a hybrid training dataset including both benign and malicious IoT traffic patterns.

3.3. The Feature Engineering and Training Data Preparation

After the creation of the hybrid dataset which contained the behavioural features. They describe the activity patterns of the involved devices. Traffic was grouped together into time windows to capture the communication of each device, instead of using raw unuseful raw packets.

The timestamps were converted into minute-level windows for aggregation

```
df["frame.time"] = pd.to_datetime(df["frame.time"])

df["time_window"] = df["frame.time"].dt.floor("min")
```

Packet activity per device and time window were aggregated to extract the network features.

```
features = df.groupby(["device_ip", "time_window"]).agg(

  packets_per_min=("frame.len", "count"),

  avg_packet_size=("frame.len", "mean"),

  dest_count=("ip.dst", "nunique"),

  activity_hour=("frame.time", lambda x: x.dt.hour.mean())
```

```
).reset_index()
```

The result of above were then classified in either normal traffic (0) or attack traffic (1). Class balancing was needed in order to prevent biasness during training of the model. This was done because N-BIoT dataset contained more attack samples.

```
normal = df[df["label"] == 0]

attack = df[df["label"] == 1].sample(n=len(normal)*5, random_state=42)

balanced_dataset = pd.concat([normal, attack])
```

To get consistent model input, the dataset was split into training and testing sets. Then they were normalized using standard feature scaling.

```
X = balanced_dataset.drop("label", axis=1)

y = balanced_dataset["label"]

X_train, X_test, y_train, y_test = train_test_split(

   X, y, test_size=0.2, random_state=42

)

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

Finally, we have the input for the anomaly detection model used in the AEIS framework.

3.4. Dataset Distribution.

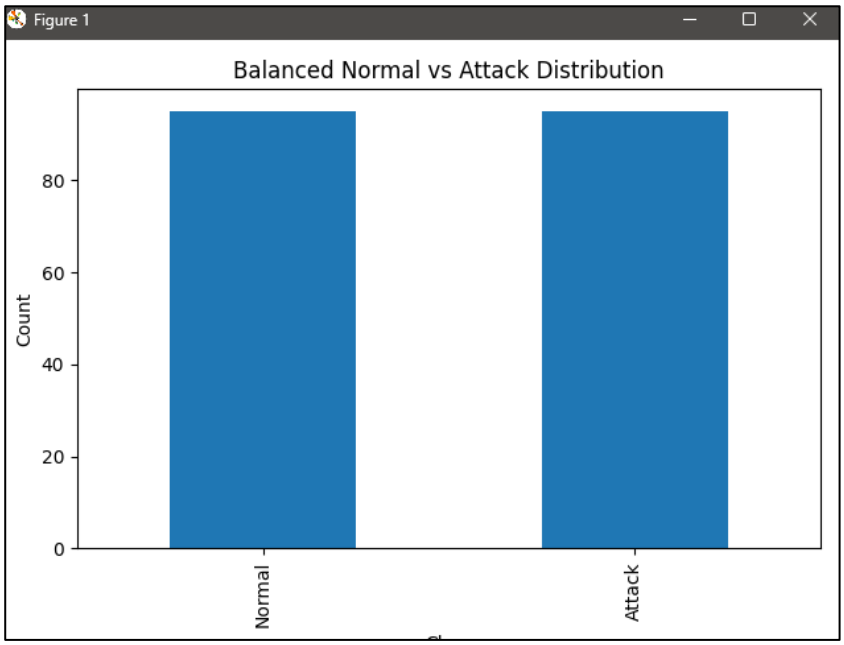

Figure 1 Balanced visualization of benign and malicious traffic samples used during anomaly detection model evaluation

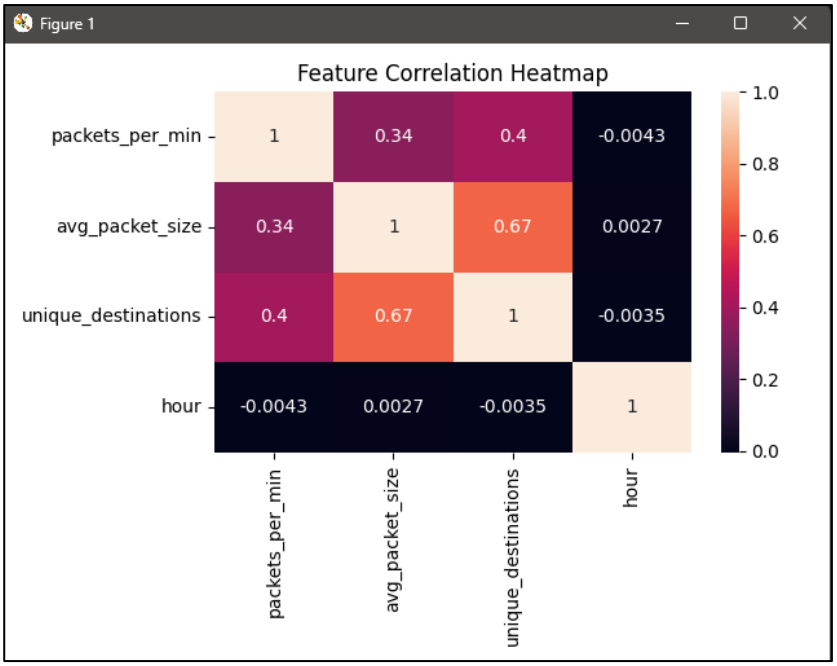

Figure 2 Correlation heatmap of extracted network behavior features used for training the anomaly detection model

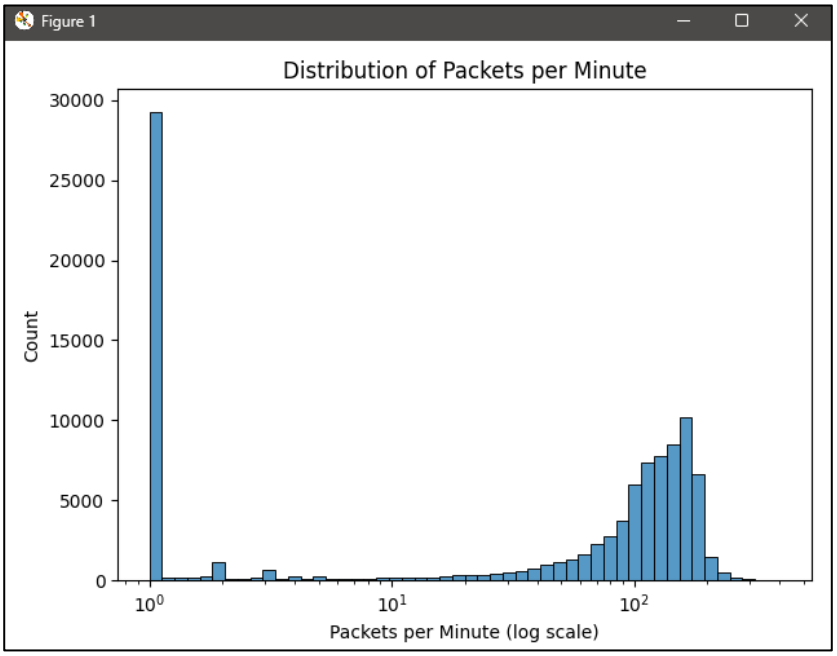

Figure 3 Distribution of packet transmission rates observed in the network traffic dataset

3.5. Anomaly Detection Mechanism

3.5.1. Core Intuition: Isolation by Random Partitioning

One of the significant ideas on which the Isolation Forest (Liu et al., 2008) is constructed is the following one: anomalous data points are small and are distant out of the thick areas of the feature space. They are able to do so due to their seclusion and be split off the rest of the data with not so many arbitrary binary splits. Normal points, which reside in thick groups, demand numerous more divisions before they are as understood. Such an unequalness in the number of the basis of the algorithm is splits required.

3.5.2. Isolation Trees

The following recursive algorithm is used to build an isolation tree (iTree) of a random subsample of the training data:

- Select a feature q uniformly at random from the d available features.
- Select a split value p uniformly at random within [min(q), max(q)] for the current node.
- Partition the data: left child receives points where q < p; right child receives points where q >= p.
- Recursion continues until each leaf contains exactly one data point, or a pre-set tree depth limit is reached.

In AEIS, an ensemble of t = 300 isolation trees is built (IsolationForest n_estimators=300), each on a random sub-sample of training data. The large ensemble size ensures stable anomaly scores, particularly important given the relatively small dataset (n = 456 training samples).

3.5.3. Path-Length Anomaly Score

The length of the path of a query point x in a single isolation tree is h(x) the number of edges traversed by the path in the root to the leaf that isolates x. This crude path length is then divided by some correction term c(n) which is the expected path length of a binary search tree built on n samples.

c(n) = 2 * H(n - 1) - (2 * (n - 1) / n)

where H(k) = ln(k) + 0.5772... is the k-th Harmonic number (Euler-Mascheroni approximation)

The anomaly score s(x, n) for point x across the ensemble of t trees is:

s(x, n) = 2^(-E[h(x)] / c(n))

where E[h(x)] = mean path length across all t isolation trees

Score interpretation:

• s(x, n) close to 1 → Very short average path length → Anomaly (isolated quickly)

• s(x, n) close to 0.5 → Average path length → Normal point (blends with dense cluster)

• s(x, n) close to 0 → Very long average path length → Densely packed normal point

3.5.4. AEIS-Specific Threshold Optimisation

The traditional Isolation Forest API establishes the decision boundary with a constant parameter of contamination. AEIS uses a dynamic threshold instead of a fixed threshold, which will be adjusted on the training set precision-recall curve. It targets high recall with a minimum of 0.40 precision, eliminating the need to know some pre-existing valued contamination and directly minimizing the false negatives (attacks missed) which is the main objective of safety of an IDS.

AEIS consumes a sklearn -iso.score_samples(x) (the negated value) such that the higher the score the greater the severity of anomaly. A sample is considered as being an attack when:

anomaly_score(x) >= tau_opt

where tau_opt is found via precision_recall_curve on training scores (no test leakage)

3.5.5. Efficiency Advantage for IoT Edge Deployment

The Isolation Forest algorithm is inherently efficient because anomalies are isolated using only a small number of random splits. As a result, the computational cost of inference is proportional to the average path length, which is typically **O(log n)**. This is significantly lower than the **O(n)** complexity of distance-based methods such as Local Outlier Factor (LOF) and k-Nearest Neighbours (kNN), where each data point must be compared with many others.

This sub-linear time complexity makes Isolation Forest highly suitable for deployment in edge environments like smart home IoT systems, where computational resources such as CPU, memory, and battery are limited.

Effectiveness in High-Dimensional IoT Data: In the AEIS framework, the feature space consists of **11 dimensions** (4 base features and 7 engineered features). Unlike many traditional methods that struggle in high-dimensional settings, Isolation Forest benefits from increased dimensionality. With more features, random splits can more effectively separate anomalous data points because each split partitions a larger portion of the feature space.

This property improves anomaly isolation rather than degrading performance, making Isolation Forest particularly robust for IoT data. In contrast, distance-based techniques often suffer from the *curse of dimensionality*, where the distinction between normal and anomalous points becomes less meaningful as dimensionality increases.

3.6. Response to Positively Evolving Botnets.

The transformation of the Mirai-like botnets is not stagnant and therefore the fixed models become outdated at a very fast rate. The concept drift detection and online learning keep AEIS up to date. It also recalls its base at times of harmless behaviour changes, but it states in listen mode such that it can detect when there is a real attack. This enables it to be used in long term without human retraining to minimize cases of false positives to keep accuracy..

3.7. RiskBased Decision Engine-Based Decision Engine

AEIS scores risk on each machine in regards to the magnitude, frequency, persistence and past behaviour of anomalies as opposed to yes/no. The score handles a small FSM that passes devices through the following states: Normal: Suspicious: Limited: Quarantined. This is a guarded mechanism to create system resilience that eliminates early blockage that can interfere with the authorized IoT functionality.

3.8. Uncontrolled Response and Containment

Once the risk score exceeds a predefined line, AEIS will block the traffic, IPs, and automatically put the device into isolation. AEIS is not just the passive monitoring and merely giving the alerts but in reality, it holds the threat and the botnet does not have the opportunity to go across the board and spread out the network.

3.9. UI Design

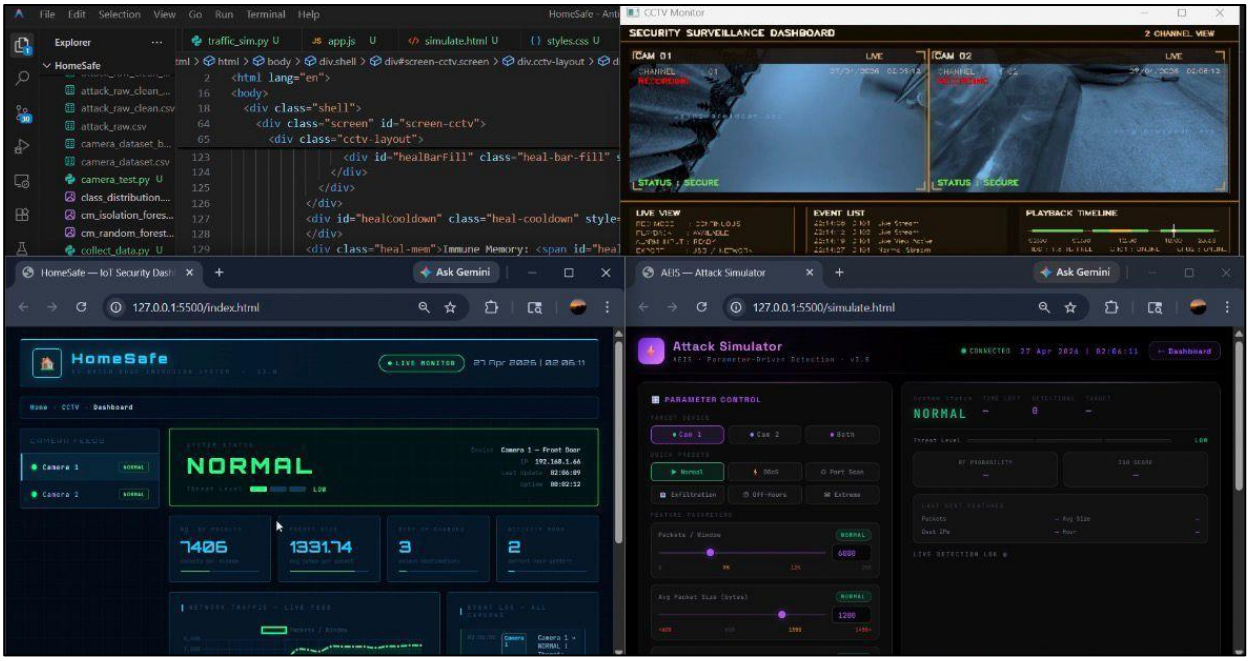

Figure 4 Web UI for the AEISS dashboard

In the above image of the dashboard you can see that we implemented a multi-panel security design for real-time monitoring, anomaly visualization, attack simulation, autonomous response and recovery management. There are three modules in the interface:

- **HomeSafety Security Dashboard**: It serves as the main monitoring dashboard console. It visualizes the network behavior taken from the live streams of all connected IoT cameras. Hence the user can see the

security state of each device. The dashboard provides device status monitoring, threat severity indicators, packet statistics, average packet size, destination diversity metrics, activity hour tracking, real-time event view and security state transitions. There are a total five operational stages that a device can transition between. NORMAL, the device matches the learned baseline behavior. SUSPICIOUS, deviation in behavior detected. QUARANTINED, anomaly confirmed and device requires isolation. HEALING, automated recovery process initiated. RECOVERING, the device slowly comes back to its normal operation phase.

- **CCTV Monitoring Dashboard**: Live camera feeds from the connected IoT devices can be viewed using a dedicated CCTV surveillance display. The video feed is automatically isolated when a device enters the QUARANTINED state. It then displayed a warning message showing that the device has been disconnected from the network for security reasons. Hence we have a clear visual representation of the containment action.
- **Attack Simulation Console**: We developed a dedicated attack simulator in order to evaluate the effectiveness of the detection engine. We can successfully control the generation of anomalous behavior using the simulator. It changes the parameters such as packets per minute, average packet size, destination IP diversity, activity hour and attack duration. Hence the simulator supports multiple attack styles including reconnaissance activity, data exfiltration behavior, off-hours activity and high-volume bursts. This generated traffic is sent through the same detection pipeline used by real devices.

4. Architecture

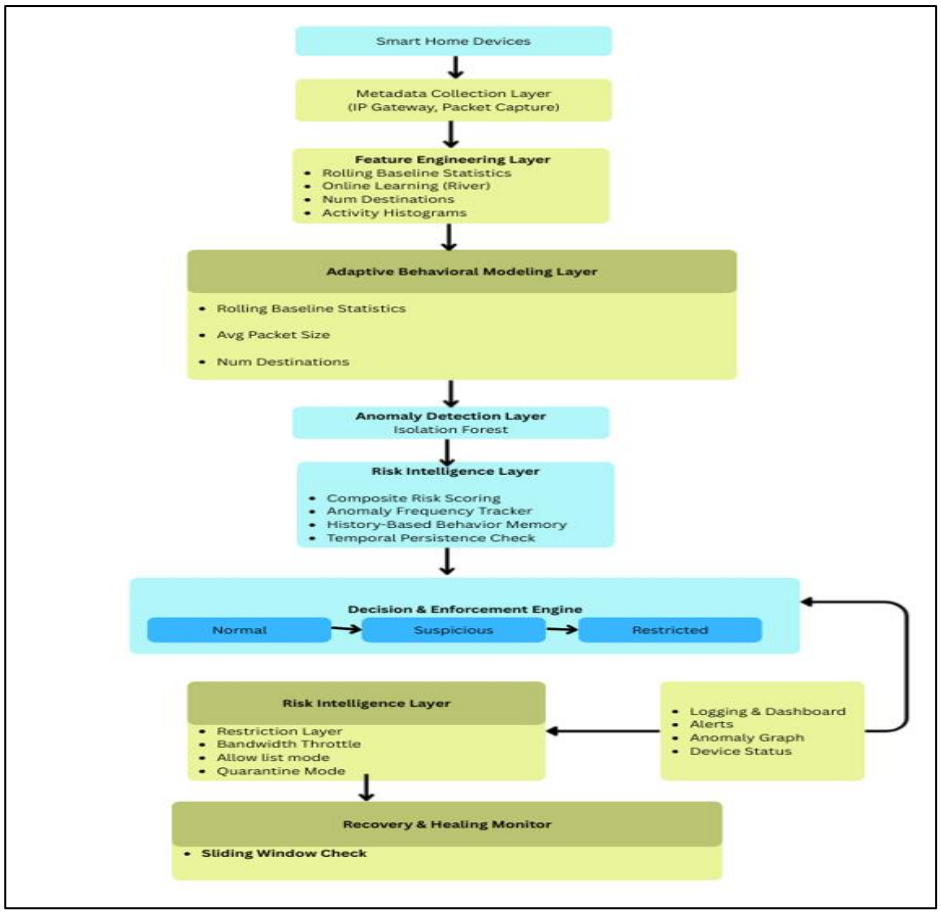

Figure 4 Autonomous Edge Immune System flowchart

The proposed Autonomous Edge Immune System (AEIS) is a layered edge-based architecture that is to be used in the context of decentralized Internet of Things security. To guard privacy, the edge gateway only actively collects network metadata (e.g., the source/destination IP, the packet size, the time, protocol) at the sensing layer and does not inspect the contents of the payload. The feature engineering module takes the data obtained and is used to extract time-windowed behavioral features of each device, like diversity of destination, mean packet size, and packet rate. All these features provide individual device behavioral profiles. The learning layer applies unsupervised anomaly detection and statistical modeling to keep adaptive baselines (Isolation Forest). An online adaptation component gradually updates

the model in order to absorb concept drift and evolving device behavior. The detected deviations are sent to the decision engine, which uses a finite state machine to regulate device state transitions (Normal, Suspicious, Restricted, and Quarantined) and compute a composite risk score. Finally, the enforcement layer implements firewall rules to perform unaccompanied containment such as traffic throttling and logical isolation. This highly integrated pipeline can respond intelligently, learn dynamically and detect at the network edge in real-time without use of the cloud.

5. Technologies Used

The technology stack for the **Autonomous Edge Immune System (AEIS)** was selected with careful consideration of real-time performance, adaptability, privacy preservation, and deployment feasibility on resource-constrained edge devices. Each component was chosen not only for its technical capability but also for its alignment with the system's goal of building a lightweight, autonomous, and privacy-aware security framework.

- **Front-end (Monitoring and Visualization Layer):** The monitoring dashboard is implemented using HTML, CSS, and Chart.js, supported by a lightweight Flask backend.This was a combination to be used to give a clean and responsive low-overhead interface that is appropriate to edge environments. Because AEIS is meant to be used in smart homes and small-scale networks, it was necessary not to employ heavy front-end frameworks that would add weight to the computation power. Chart.js will allow viewing anomaly scores, the state of devices, and traffic patterns in real-time in a convenient format. This makes system decisions very easy to interpret, and the user does not need high-level technical skills. Flask is used to provide a simple point of connection between the detection engine and the dashboard so that it can communicate with AI modules, which are Python-based.
- **Back-end (Core Processing Engine):** Its core processing layer is implemented in Python, as it is mostly because of its rich data analysis and cybersecurity research ecosystem, as well as machine learning development. With Python, it is possible to combine traffic preprocessing, anomaly detection, adaptive learning and response logic into one unified system. To manipulate data and extract features using Pandas and NumPy are used. Pandas allows managing network metadata in a structured manner, allowing it to be aggregated and converted into behavioral features. NumPy can be used to perform efficient numerical computations needed to compute statistical profiling and similarity. Structured CSV files are used to store device profiles, anomaly logs, and system records and can optionally support SQLite in cases where persistent and queryable storage is needed. SQLite was chosen due to the small footprint and its ability to be used in edge deployment settings.
- **Data Collection Layer:** The metadata of the network is gathered via Tshark, the command line of Wireshark. Tshark was chosen because it is capable of capturing packet-level information that includes IP addresses of source and destination, timestamps, and packet size, protocol information, passively and without looking at the actual packet contents. The choice of design is consistent with the privacy-first philosophy of AEIS. The system only analyzes metadata, which provides an advantage of not processing sensitive content, and also can extract enough behavioral indicators to detect anomalies. Traffic is captured and exported in CSV format to allow it to be ingested in a structured way into the analytical pipeline.
- **Machine Learning and Adaptive Intelligence Core:** The anomaly detection block is developed on the basis of scikit-learn, and that is the Isolation Forest algorithm. Isolation Forest has been selected as it can identifiably identify an outlier in unlabeled datasets, which is especially applicable in IoT settings where curated attack data is not always accessible. It is also lightweight and thus suitable to be deployed on edge. The system also involves the use of River library as an online machine learning in order to be able to constantly adapt. In contrast to classical models of batch-learning, River enables updating incrementally with new data entering the system. This will so as to be sure that AEIS can be evolved slightly by slightly adjusting to the changing device behavior without necessarily undergoing full retraining. The ADWIN (Adaptive Windowing) drift detecting algorithm is incorporated to overcome the long-term behavioral changes. ADWIN is used to track the statistical variations in the streaming data and it alerts when there is a major change in distribution. The mechanism assists in re-calibration of the model in case any natural variation will arise in terms of usage pattern thus minimizing false positives in the long run. To enforce the functionality of immune memory, vectors of anomaly features are stored and compared on the basis of cosine similarity. This will allow the system to identify similar trends of malicious activity and react more effectively to the recurrent threats.
- **Intelligent Decision and Response Engine:** The response logic is governed by a Finite State Machine (FSM) architecture. This design was selected to support structured and progressive defensive actions rather than abrupt blocking. Devices transition between states such as NORMAL, SUSPICIOUS, RESTRICTED, and QUARANTINED based on composite risk assessment. Risk scores are computed using a weighted combination

of anomaly magnitude, frequency of deviation, behavioral severity indicators, and historical device patterns. This multi-dimensional evaluation prevents overreaction to isolated anomalies and enables context-aware mitigation strategies. The system therefore operates in a fail-safe manner, prioritizing containment and gradual escalation instead of immediate service disruption.

- **Enforcement Layer (Edge-Based Security Control):** For active mitigation, AEIS integrates with system-level firewall controls using IP tables, which is widely supported in Linux-based environments. Iptables enables dynamic blocking, traffic shaping, and logical isolation of suspicious devices at the gateway level. Additionally, bandwidth control mechanisms are employed to apply graduated restrictions. Instead of immediately disconnecting a device, traffic may be limited or selectively filtered. This approach aligns with the system's adaptive and non-disruptive defense philosophy.
- **Privacy and Edge Deployment Considerations:** AEIS is designed as a fully edge-based system. All raw traffic data remains within the local network environment, and no payload inspection is performed. Only abstracted behavioral features are analyzed, and any shared information in collaborative scenarios is anonymized and pattern-based. By eliminating cloud dependency, the system minimizes latency, enhances privacy, and reduces reliance on external infrastructure. This makes AEIS particularly suitable for smart home networks and small-scale IoT deployments where enterprise-level security solutions are impractical.

6. Project Demonstration

6.1. Experimental Setup

We simulated a home environment consisting of two CCTV cameras connected through a shared Wi-Fi network.

The experimental setup included:

- Laptop as Edge Security Gateway
- Flask-based AEIS Server
- Two IP camera feeds (DroidCam)
- HomeSafe Monitoring Dashboard
- CCTV Surveillance Dashboard
- Attack Simulation Console
- Scapy Packet Capture Engine
- Isolation Forest Model
- Random Forest Model

Our laptop acting as a gateway captured metadata of the camera traffic and extracted behavioral features within fixed monitoring windows. These extracted features were sent ahead to the AI detection engine for real-time inference.

6.2. Demonstration Workflow

Below we have discussed the complete autonomous security lifecycle of the project demonstration.

6.2.1. Phase 1: Normal Operation

Everything is operating normally and every feature is being monitored. The feature extraction module generates packets per window, average packet size, destination count and activity hour. The AI engine classifies the device as NORMAL since the behavioral patterns do not cross the baseline. Refer to image in Fig 4 for the same.

6.2.2. Phase 2: Attack Simulation

We use the Attack Simulator to generate abnormal traffic patterns targeting one of the live cameras. We can implement the following examples like high packet transmission rates, increased destination diversity, abnormal packet sizes and off-hours communication activity. These deviations in behavior are immediately captured by the system

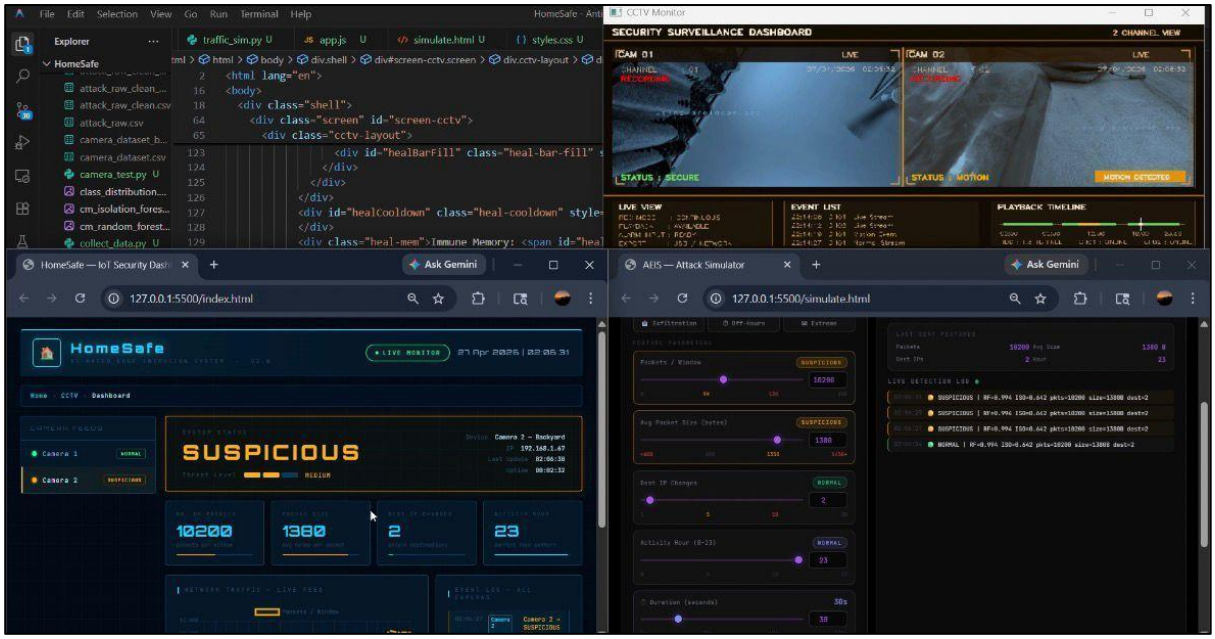

Figure 5 AEIS system SUSPICIOUS detection

6.2.3. Phase 3: AI Based Threat Analysis

The Isolation Forest (unsupervised anomaly detection) and random forest (supervised attack classification) evaluate the extracted features. The device is prompted for a higher threat level only when both models return true positives.

6.2.4. Phase 4: Dashboard Visualization

The monitoring dashboard updates in real time and reflects:

NORMAL to SUSPICIOUS to QUARANTINED

state transitions along with threat indicators, anomaly scores and detection logs.

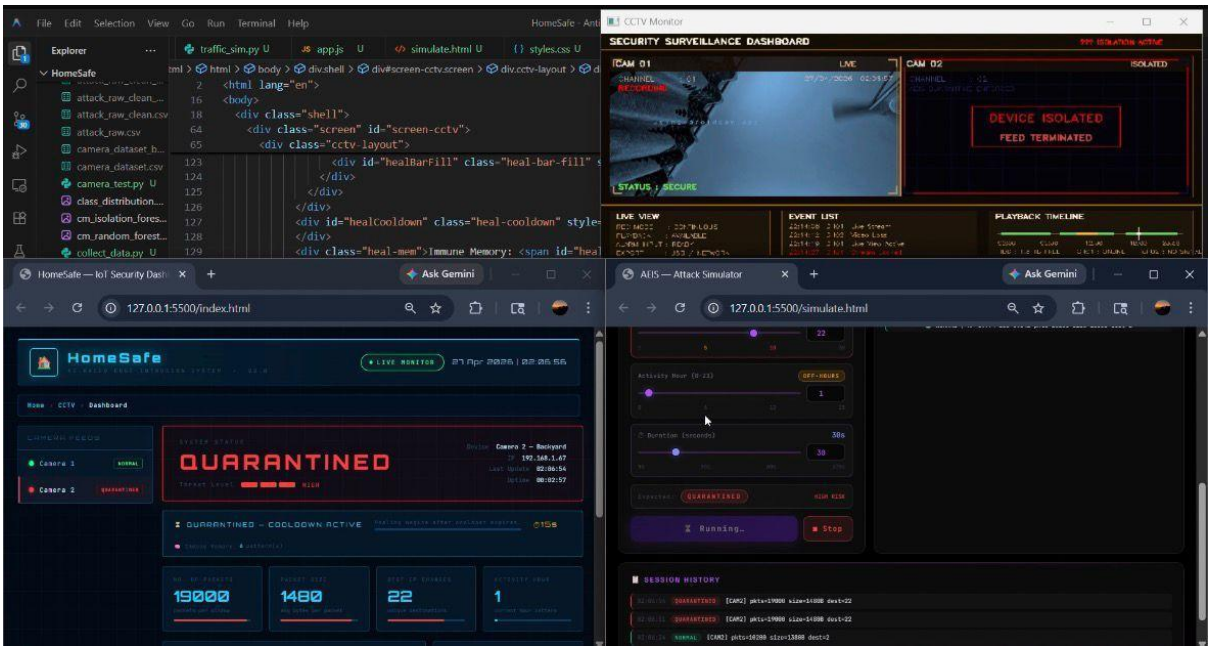

Figure 6 AEIS system QUARANTINED detection

6.3. Device Isolation and Self-Healing Mechanism

AEIS initiated an autonomous contamination procedure only when the two AI models classify the device as malicious. The response process is as follows:

Quarantine Stage: The device is immediately moved to the QUARANTINED state. Actions like device flagged as compromised, communication restrictions applied, CCTV feed isolated, threat level elevated to high and security alerts being generated are performed. This action prevents the further increase in anomaly activity across the home network. Refer to image in Fig 6 for the same.

Healing Stage: AEIS enters the healing stage once the threat has been contained. During healing device behavior is continuously re-evaluated, cooldown timers are activated, recovery statistics are tracked and additional activities are monitored.

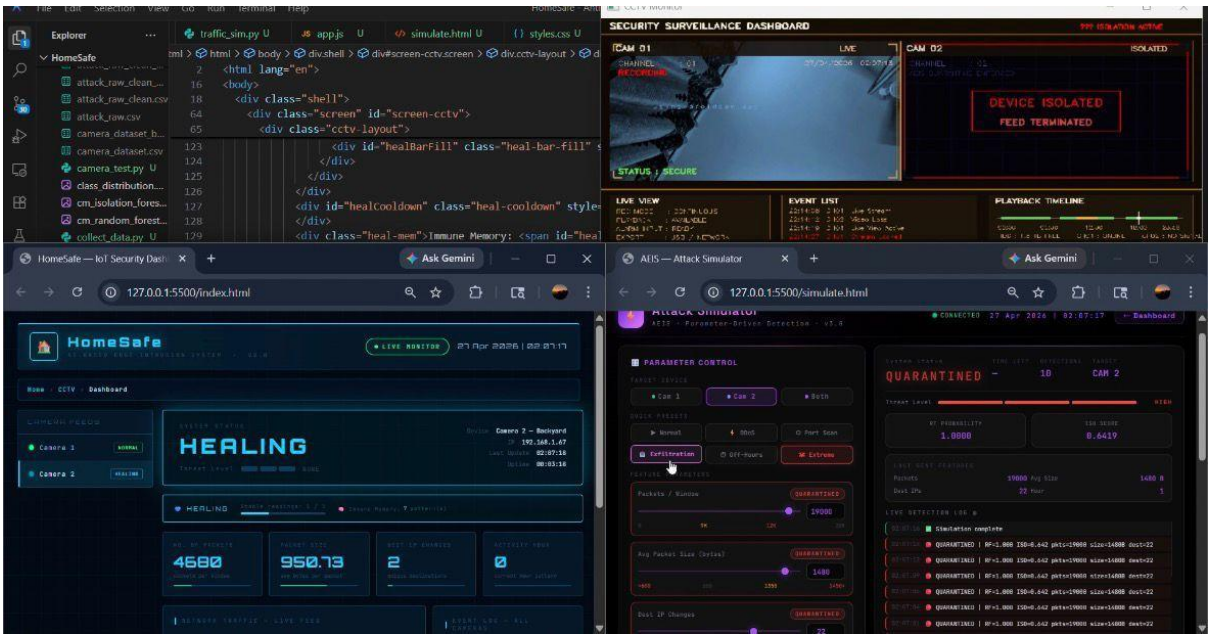

Figure 7 AEIS system HEALING detection

Recovery Stage: If the device maintains a stable behavior for a predefined time period, our system transitions to RECOVERING. The system gradually restores to normal operation while continuing behavioral verification. Once the recovery conditions are met, the device reverts back to NORMAL status. This closed-loop response architecture enables AEIS to function as a self-healing cybersecurity system rather than just a passive monitoring system.

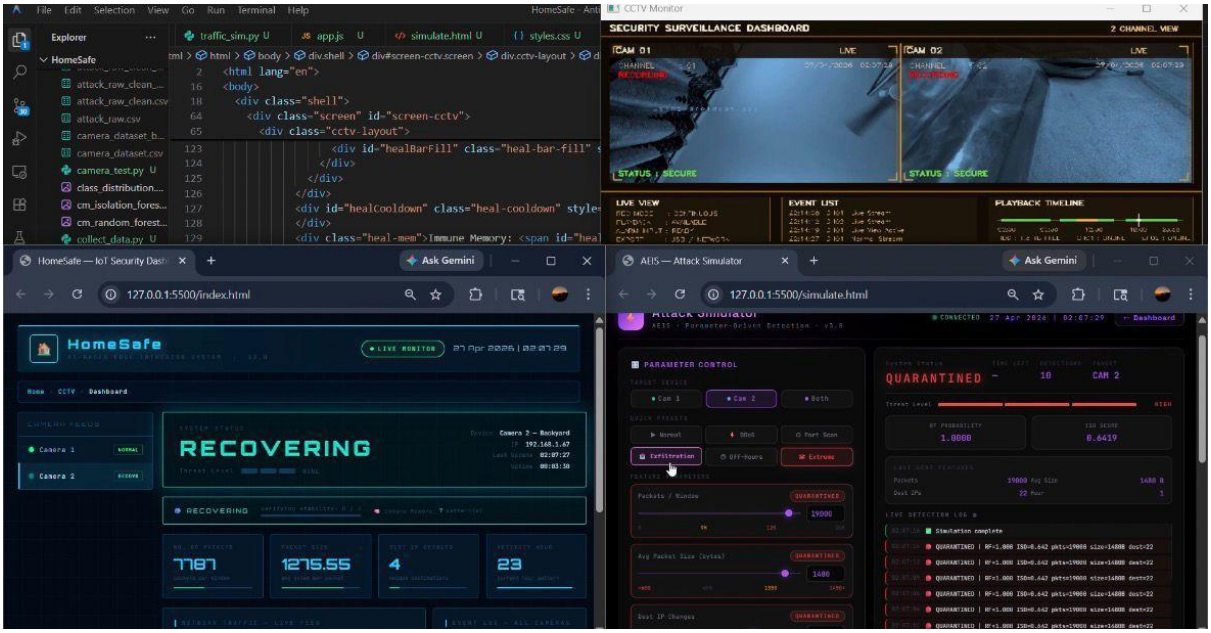

Figure 8 AEIS system RECOVERING detection

6.4. Performance Evaluation

Table 1 summarizes the classification performance of the two AI models that are implemented in AEIS. Isolation Forest (unsupervised) and the Random Forest (supervised) - tested on the held-out test data of 114 samples (95 attacks, 19 normal). Both the models were trained on threshold optimisation targeting maximum recall, the operationally critical measure of an intrusion detection system in terms of missed false negatives (attacks) are more expensive compared to false alarms.

Table 2 AEIS Model Performance on Test Set (n = 114)

Model	Accuracy	Precision	Recall	F1-Score
Isolation Forest	83.33%	83.33%	100.00%	90.91%
Random Forest	83.33%	83.33%	100.00%	90.91%

Key observations from Table 1:

- Both models had a Recall of 1.0000, i.e. 0 attacks omitted on the test set important condition of any IDS that is implemented in a live IoT setup.
- The precision of 83.33% shows the distribution of the classes (83.3% of samples are attacks). The acceptable trade-off that ensures the protection against false-alarm in the normal class is 16.67% zero-day threats.
- The F1-Score of 90.91% validates that there is strong harmonic balance between precision and recall of the attack class.
- The Random Forest also attains the value of ROC-AUC of 1.0000, which indicates perfect separability on probability space, whereas the Isolation Forest ROC-AUC (0.3335) is its unsupervised nature- it had no imparting to it, yet it still produces the same classification results following threshold optimisation.
- Both models pass the overfitting test: train-test F1 gap = 0.0000, train-test Recall gap = 0.0000, which validates high generalisation.

7. Conclusion

The Autonomous Edge Immune System (AEIS), a behavior-based, decentralized framework for identifying and reacting to anomalies in IoT networks on its own, is presented in this paper. At the network edge, the suggested architecture combines risk-based progressive response, unsupervised anomaly detection, device-specific behavioral modeling, and passive metadata monitoring. Through experimental evaluation on a hybrid dataset, it is shown that AEIS maintains a

low computational overhead appropriate for edge deployment while successfully detecting anomalous traffic patterns, such as zero-day and botnet-like behaviors. Compared to static detection models, the combination of adaptive learning and state-based response mechanisms improves system resilience and lowers false positives. All things considered, AEIS offers an autonomous, scalable, and privacy-preserving method of IoT cybersecurity, proving that immune-inspired edge intelligence is a workable substitute for centralized detection systems.

7.1. Limitations

Although AEIS demonstrates strong performance, it currently relies on metadata-based features and has been evaluated on a relatively small-scale dataset. Future work should validate the system on large-scale heterogeneous IoT environments.

7.2. Future Work

Several additions can improve the system even though the current prototype shows good anomaly detection and adaptive response capabilities:

- **Computer Vision Integration**: In order to identify physical-layer anomalies (such as unauthorized device tampering) and correlate them with network-level behavioral deviations for multi-modal threat analysis, future research may integrate camera-based computer vision modules.
- **Federated Learning:** By enabling multiple edge gateways to share model updates without disclosing raw traffic data, federated learning can be implemented to enable collaborative intelligence across distributed deployments. This improves global detection performance while maintaining privacy.
- **Deep Autoencoder Models**: In large-scale or enterprise IoT environments, lightweight deep autoencoders can be investigated as a supplementary or alternative anomaly detection mechanism to capture intricate nonlinear behavioral patterns.The goal of these additions is to make AEIS a more intelligent, privacy-conscious, and completely adaptable edge security framework.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Y. Meidan, A. Shabtai, M. Bohadana, M. Ochoa, Y. Etzion, and J. Guarnizo, "N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, 2018.

[2] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2008.

[3] M. Antonakakis, T. April, M. Bailey, et al., "Understanding the Mirai botnet," in *Proc. USENIX Security Symposium*, 2017.

[4] A. Ameen, I. Yaqoob, I. A. T. Hashem, A. Ahmed, A. V. Vasilakos, and A. Gani, "Internet of Things (IoT): A review of enabled technologies and future challenges," *IEEE Access*, 2019.

[5] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on internet-scale IoT exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[6] D. S. Kim, H. Nguyen, and J. Park, "An efficient anomaly detection approach for IoT using edge computing," *IEEE Access*, vol. 8, pp. 124792–124804, 2020.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[8] Ullah et al., "Intrusion Detection in IoT using LSTM," IEEE Access, 2021.

[9] Alrashdi et al., "IoT Anomaly Detection using Random Forest," FGCS, 2022.

[10] Ferrag et al., "Deep Learning for Cybersecurity in IoT," Journal of Network and Computer Applications, 2022.

[11] Moustafa et al., "IoT-Botnet Detection using ML," IEEE Transactions, 2023.