TED UNIVERSITY

SEENG 484 - Ethical Hacking and Countermeasures

Project Progress Report 2

Ateş Öztürk

Mehmet Alp Demiral

Ozan Alp Sarıdoğan

1. Introduction

Following the initial data discovery and preliminary preparation phase outlined in the previous report, the IoTGuard project is now in the model training and system integration phase. The first phase focused on analyzing the CIC-IoT-2023 dataset and defining the architecture, this phase focused on the execution of the machine learning pipelines and the initial integration with the Suricata environment.

2. Completed Activities

2.1. Machine Learning Implementation Building

Upon the design specifications established in the initial report, the following models have now been trained:

- Isolation Forest: The unsupervised model has been trained on "Benign Final" traffic to establish a baseline of normal behavior. The contamination rate is being fine-tuned to minimize false positives.

- LightGBM: The supervised model has been trained using the 33 distinct attack categories identified including DDoS, Mirai, and Reconnaissance types. The class balancing strategy using SMOTE was applied to handle the disparity between high-volume attacks like DDoS-HTTP Flood and lower-volume attacks such as XSS.

2.2. Data Pipeline Integration

The preprocessing pipeline is now operational. Raw flow data, originally including features such as bytes_in, duration, and inter-arrival times, is now automatically passed through the cleaning module (median imputation and removal of zero-variance features) before ingestion by the models.

3. Current Challenges & Solutions

Real-time Traffic Simulation: As noted in the previous report, the team faced difficulties capturing real attack packets due to Suricata's high-security configuration. To bridge the gap between the offline UNB dataset and the live environment, we are currently utilizing traffic replay tools to simulate the CIC-IoT-2023 flows through the Suricata interface.

4. Next Steps

Suricata Integration: Parsing eve.json logs (Part B of Team Organization) to feed live flow data into the trained LightGBM model.

Response System: Implementing the blocking policy on the Raspberry Pi based on model predictions.

5. Strategic Analysis & Suggested Improvements

As mentioned in the initial report, here are specific technical improvements and critiques to enhance the success of the project.

1. Addressing the "Real-Time" Gap

The Issue: The report explicitly states that "the system struggled to detect real attack packets" and we decided to work on "primarily with recorded (offline) data". This creates a major risk. An offline model often fails in production because it cannot handle the latency or format of live packet capture.

Improvement: Do not rely solely on offline CSVs. We must validate the pipeline with Traffic Replay. Using tools like tcpreplay to blast the CIC-IoT-2023 PCAP files through the network interface where Suricata is listening. This proves that your feature extractor works on live wire data, not just static CSV rows.

2. Refine the Hybrid Model Architecture

The Issue: The report lists Isolation Forest and LightGBM as separate components.

Improvement: Implement a cascaded pipeline.
Stage 1 (Isolation Forest): Acts as a high speed gatekeeper to filter "Obvious Normal" vs. "Anomalous" traffic. This reduces the necessity of computational power.

Stage 2 (LightGBM): Only triggers when Stage 1 detects an anomaly, classifying the type of attack (e.g., distinguishing Mirai from SQL Injection).

Why: LightGBM is fast, but running it on every single packet in a high-throughput IoT network might cause latency.

3. Feature Engineering for IoT Specificity

The Issue: The extracted features listed are generic flow stats (bytes, packets, duration, TCP flags).

Improvement: Since the project is an IoT project, we need features specific to IoT protocols mentioned in the dataset (like Mirai attacks which often target Telnet/IoT ports).

Add Payload Analysis: Check for specific IoT headers (MQTT topics, CoAP codes) if available in the flow data.

Add Ratio Features: The report mentions "flag ratios". Ensure you calculate the ratio of incoming vs. outgoing bytes (IoT devices usually have a specific producer/consumer ratio compared to normal PCs).

4. Response Mechanism Optimization

The Issue: The "System & Response Engineer" is tasked with "Decision engine" and "integration with OS-level firewalls".

Improvement: If the Python script (Flask API/Dashboard) is responsible for triggering the block, it may be too slow to stop a DDoS attack.

Recommendation: Use IPS Mode (Intrusion Prevention System) in Suricata directly, or use eBPF/XDP for packet dropping at the kernel level. The Python script should update the rules, not process the blocking of individual packets.

5. Dataset Clarification (CIC-IoT-2023)

The Issue: The report lists attacks like "XSS" and "SqlInjection". These are Application Layer (Layer 7) attacks.

Improvement: Ensure Suricata is configured to decode HTTP traffic properly. Flow-based features (Layer 3/4) (like bytes/duration) are often poor predictors for XSS or SQLi, which reside in the payload. You may need to verify if your feature set includes payload content or http-specific fields (e.g., URI length, query parameters).

6. Conclusion

The IoTGuard project has successfully transitioned from the initial data discovery and design phase into the technical implementation stage. By leveraging the CIC-IoT-2023 dataset, we completed the training and validation of the hybrid machine learning architecture, consisting of the Isolation Forest model for anomaly detection and LightGBM for multi-class attack classification.

While the initial phase identified challenges in capturing high-volume real-time attack data via Suricata, the project has mitigated this by implementing a rigorous data preprocessing pipeline and adopting traffic replay methodologies to validate the models against the "Benign Final" and 33 distinct attack categories.

Moving forward, the primary focus will shift to the integration of these trained models with the Suricata eve.json log stream and the finalization of the automated response mechanism on the Raspberry Pi. This next phase represents the critical step in realizing the project's ultimate goal: delivering a dynamic, AI-driven security layer that overcomes the limitations of traditional signature-based systems in IoT environments.