Edge-Based Anomaly Detection for IoMT: A Lightweight Unsupervised Model and SIEM Integration

Payton R. Grande

A thesis
Submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington
2026

Committee;
Bryan Goda
Ling-Hong Hung

Program Authorized to Offer Degree:
Engineering and Technology

University of Washington

Abstract

Edge-Based Anomaly Detection for IoMT: A Lightweight Unsupervised Model and SIEM Integration

Payton R. Grande

Chair of the Supervisory Committee:
Bryan Goda
School of Engineering and Technology

The growth of the Internet of Medical Things (IoMT) has increased connectivity across healthcare environments, but it has also reduced visibility into how many medical devices actually behave. Many of these devices run on constrained hardware and provide little to no detailed logging, which makes traditional host-based monitoring difficult in practice. This creates a detection challenge: how to identify abnormal behavior early without adding heavy security tooling or sending large amounts of telemetry to a centralized system.

This thesis presents the design and evaluation of an edge-based, unsupervised anomaly detection pipeline for IoMT telemetry, with a specific focus on how detections integrate into real Security Information and Event Management (SIEM) workflows. An Isolation Forest model is trained using baseline telemetry that represents normal device behavior and then deployed in inference-only mode at the edge, where the model scores new events without retraining or

updating parameters during operation. Instead of giving raw model scores to analysts, anomaly outputs are normalized into SOC-centric severity levels and formatted as structured alerts for ingestion into Elasticsearch and visualization in Kibana.

Evaluation was conducted in two phases. Under controlled injection-based conditions, anomaly scores showed near-perfect separability between normal and abnormal telemetry (ROC AUC = 1.000), indicating that the model can clearly rank strongly defined abnormal deviations. When evaluated on a more realistic dataset where normal and abnormal behavior overlap and anomalies are more subtle, performance decreased (as expected), but was still usable (ROC AUC = 0.783). A threshold sweep was used to explore how detection performance changes as the decision cutoff for an anomaly is adjusted. This analysis showed that the system could be tuned to keep the false positive rate below 1% while still detecting a meaningful portion of anomalous events. Additional analysis confirmed that anomaly scores were normalized consistently, that severity labels were applied predictably, and that alert generation remained stable over longer runs.

Runtime feasibility was measured in both a virtualized environment and on Raspberry Pi–class hardware as a representative constrained edge gateway platform. Mean per-event latency increased from approximately 5 ms in the virtualized baseline to roughly 23 ms when running on edge hardware, which reflects the expected increased overhead of operating with tighter resource limits. Inference still completed without memory exhaustion, swapping, or process failure. These results suggest that lightweight unsupervised scoring can operate within realistic resource limits.

This system is not designed to explain exactly what happened or who carried out an attack. Its purpose is to flag unusual behavior early so it can be investigated further. More

broadly, this work shows how lightweight statistical ranking at the edge can help security teams prioritize abnormal behavior earlier and feed those detections into the centralized monitoring workflows that they already use.

Acknowledgments

I would like to thank my thesis advisors, Dr. Goda and Dr. Hung, for their oversight and support throughout my research process.

I am especially grateful to my colleagues in the SOC who took the time to read early, weaker drafts, challenge my assumptions, and offer their honest feedback on my work. Their perspective as analysts helped shape the practical components of this work and ensured that they reflected how alerts are actually handled in real-world environments. Their input made the project much stronger and more grounded than it would have been without their valuable input.

Finally, I recognize the shoulders upon which I stand. I thank the broader cybersecurity and healthcare communities whose prior research and experience informed and motivated this study.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background and Motivation

Healthcare organizations now depend heavily on networked medical devices such as infusion pumps, bedside monitors, wearable sensors, and remote monitoring systems that constantly produce telemetry. This growth in the Internet of Medical Things (IoMT) improves patient care and enables continuous monitoring, but it also increases the attack surface of hospital networks. Many IoMT devices are hard to patch, run on limited hardware, and offer only basic or no logging, dissimilar to traditional enterprise systems. As a result, security teams, including SOC analysts, often have poor visibility into device behavior despite attacks against healthcare continuing to be frequent and expensive.

These conditions are similar to long-standing challenges in industrial control system (ICS) and operational technology (OT) security. In both settings, systems must be available and safe at all times, but they often run on constrained hardware that can't support heavy, resource-intensive security agents or even frequent updates. In practice, this combination of safety-critical operation, limited visibility, and resource constraints makes it difficult to detect suspicious activity in the timely manner that is required in these types of environments. Organizations need a way to notice abnormal behavior quickly, but traditional host-based monitoring and even signature-driven detection are typically not realistic to have directly on the devices.

One option is lightweight, behavior-based anomaly detection deployed at or near the edge, where telemetry is already being collected and forwarded. Moving detection closer to the devices can reduce latency and avoid sending any unnecessary raw telemetry off the network. It also lets security teams see unusual patterns sooner in the data flow. However, much of the IoMT anomaly detection work so far has focused on model accuracy in controlled experiments and has

not addressed how detections would be used in day-to-day security operations. In real environments, detections only become useful when they are turned into structured security events and fed into centralized monitoring platforms, where analysts can search and correlate them to aid in their investigations.

Security Information and Event Management (SIEM) platforms are central to these security operations. They rely on searchable fields, severity labels, and the ability to connect one alert to other data sources. One gap in the IoMT anomaly detection literature is that many studies stop at the model layer and do not show how edge detections are turned into SIEM-ready alerts that fit the way security operations center (SOC) analysts triage events. This thesis focuses on that gap by building and evaluating an end-to-end pipeline that combines IoMT-style telemetry ingestion with lightweight unsupervised anomaly scoring at the edge. The outputs are then formatted as SIEM-based alerts used to support SOC investigation and correlation.

1.2 Problem Statement

IoMT environments are difficult to monitor because organizations usually have limited visibility into device behavior and cannot easily deploy traditional security tools directly on medical devices. At the same time, hospitals and clinics still need continuous monitoring and real-time detection of abnormal activity to reduce operational risk and to keep patients safe.

Prior work shows that anomaly detection models can be applied to IoMT telemetry, but most studies emphasize model choice and performance instead of how detections would be used end to end in an SOC. Specifically, there is limited guidance on the following:

1. running lightweight unsupervised anomaly detection at the edge under realistic resource constraints, and

2. converting model outputs into structured alerts that are usable within a SIEM for analyst triage and correlation.

This thesis addresses that gap by designing and evaluating an edge-based anomaly detection pipeline. The system generates IoMT-style telemetry, scores events using an unsupervised model, and produces normalized alerts for ingestion into Elasticsearch and Kibana. The goal is not to claim complete accuracy to clinical workflows or full production-scale attack attribution. Instead, the goal is to demonstrate a realistic and reproducible workflow for detecting unusual behavior early and presenting it in a form that matches how a SOC team would monitor an IoMT environment.

1.3 Research Questions

This thesis is guided by three research questions focused on feasibility, detection behavior, and operational integration in IoMT environments:

- RQ1 (Feasibility): Can an unsupervised, lightweight anomaly scoring pipeline run on edge-class hardware with acceptable latency and resource usage when it is restricted to inference-only operation?
- RQ2 (Detection behavior): Do the anomaly scores meaningfully separate baseline telemetry from injected anomalies under controlled conditions? How does that separation change when telemetry becomes more realistic and variable?
- RQ3 (Operationalization): Can raw anomaly scores be transformed into normalized, severity-labeled alerts that integrate into a SIEM workflow without requiring analysts to interpret model-specific scores?

1.4 Approach Overview

To answer these research questions, this thesis implements an end-to-end experimental pipeline that simulates IoMT telemetry flowing from devices, through an edge gateway, and into a centralized SIEM-like platform. IoMT-style messages are generated and published over MQTT, ingested by a lightweight subscriber, and indexed into Elasticsearch, where they can be searched and visualized in Kibana. An Isolation Forest model is trained on baseline telemetry and then used in inference-only mode to assign anomaly scores to incoming events.

Instead of presenting raw scores directly to analysts, the system maps those scores into a normalized SOC scoring range and assigns severity levels that resemble common SOC practices. In many production environments, analysts work with severity labels and filtered alert queues instead of raw statistics, so the pipeline is intended to mirror these operations.

Evaluation is conducted in two phases. The first phase uses controlled anomaly injection to measure how well scores separate normal and abnormal telemetry, using metrics such as receiver operating characteristic (ROC) curves and precision–recall values. The second phase introduces more subtle anomaly behavior to demonstrate more realistic device conditions. Runtime feasibility is measured using operating system–level resource metrics and includes a comparison between a virtualized environment and edge-class hardware.

1.5 Contributions

This thesis makes the following contributions:

- A reproducible IoMT telemetry and alerting pipeline that demonstrates end-to-end flow starting with edge ingestion and ending in SIEM visualization.

- An edge-based implementation of unsupervised anomaly scoring using Isolation Forest in inference-only mode.
- A structured alert schema that normalizes anomaly scores and maps them to severity categories that align with common SOC workflows.
- A two-stage evaluation strategy that contrasts controlled separability with more realistic detection conditions.
- Measurement of inference latency and resource usage in both a virtualized environment and on edge-class hardware to assess deployment feasibility.

1.6 Scope and Boundaries

This work is designed to demonstrate feasibility and operational integration under realistic constraints, not complete detection coverage.

1. Model training is unsupervised. Ground-truth labels are used only for evaluation through controlled anomaly injection.
2. Edge-class hardware is used to validate runtime behavior in inference-only mode. Training and metric computation are performed in a controlled environment to keep experiments reproducible.
3. Telemetry is synthetically generated so that experiments can be structured and repeatable. Results should be interpreted as validation of pipeline behavior and not a prediction of real-world anomaly detection rates.
4. Elasticsearch and Kibana are used to represent SIEM-style ingestion and analysis. The focus is on demonstrating how alerts can be integrated and used, not to imply a full-scale SIEM.

1.7 Thesis Organization

The remainder of this thesis is organized as follows:

- Chapter 2 reviews IoMT and ICS-adjacent security challenges that motivate the use of behavior-based detection.
- Chapter 3 surveys related work in anomaly detection, edge feasibility, and SIEM integration gaps.
- Chapter 4 describes the system architecture, telemetry pipeline, model design, and measurement methodology.
- Chapter 5 presents results from both controlled and realistic evaluation passes and includes detection metrics as well as runtime comparisons.
- Chapter 6 discusses interpretation and limitations of the findings.
- Chapter 7 concludes the thesis and summarizes the contributions.

Chapter 2: IoMT and ICS Security Background

This chapter provides background on IoMT and ICS security with a focus on system architectures, communication protocols, real-world attack trends, and the constraints that influence security monitoring in healthcare environments.

2.1 IoMT System Architecture

The Internet of Medical Things (IoMT) consists of connected medical devices, sensors, and applications designed to collect, transmit, and analyze patient data. IoMT systems are typically described as operating across multiple layers, beginning with physical devices and sensors (the perception layer), moving through the network (the network layer), and ending in cloud or application environments where data is processed (the application layer) (Sindhuja et al., 2023). At the device layer, hardware includes a wide range of devices such as implantable cardiac devices, infusion pumps, bedside monitors, imaging equipment, and consumer wearables, all of which must interact within the same healthcare network. Each of these devices also has different levels of built-in security, adding to the overall complexity of the IoMT ecosystem (Messinis et al., 2024).

Most IoMT systems rely on an edge gateway to collect data from physical sensors and relay it to either hospital networks or cloud platforms, depending on system architecture. In this thesis, an edge gateway refers to an intermediary system that aggregates IoMT device data, performing initial data processing and formatting the data before forwarding telemetry upstream. Data is transmitted by the edge gateway using technologies such as Wi-Fi, Bluetooth Low Energy (BLE), ZigBee, or MQTT, depending on device capabilities and clinical use cases (Koutras et al., 2020). These communication technologies are ideal for low-power and low-cost

environments, but there is often a tradeoff related to security and bandwidth. This tradeoff is most visible at the protocol level, where reducing communication overhead can limit the use of stronger security controls.

2.2 Communication Protocols and Their Security Weaknesses

The most common communication methods in IoMT environments include BLE, ZigBee, Wi-Fi, MQTT, and medical messaging standards such as HL7. These protocols are integral to supporting communication between sensors, wearables, and clinical devices, but they also introduce gaps related to authentication, encryption, and access control.

BLE and ZigBee are commonly used in wearables, patient monitors, and infusion pumps due to their support for low-power operation and short-range communications. However, both protocols are frequently targeted by attacks such as replay attacks, spoofing, and weaknesses in pairing mechanisms (Koutras et al., 2020). These attacks may allow adversaries to spoof devices in order to inject unauthorized commands or intercept transmitted data.

Within the application layer, IoMT systems often rely on MQTT or HL7 to transmit clinical data. MQTT uses a lightweight publish-subscribe model that is well suited for low-power medical sensors but does not provide built-in encryption and instead depends on Transport Layer Security (TLS) configured at the broker. If TLS is misconfigured or if client authentication is weak, attackers may intercept messages or publish malicious data to subscribed systems (Koutras et al., 2020). HL7 is also a widely used medical messaging standard but poses security risks because traditional implementations transmit data in the clear and lack transport-layer protections, exposing sensitive clinical information during transit (Sadhu et al., 2022).

In healthcare environments, these weaknesses carry a more significant risk because of the sensitivity of the data being transmitted. IoMT traffic often includes patient data and device commands, such as those used by smart infusion pumps. When combined with known device limitations like constrained processing power, limited logging, and diverse vendor implementations, these vulnerabilities increase the risk of unauthorized data manipulation and can directly impact patient safety.

2.3 Real-World IoMT Attacks and Data-Breach Trends

Several real-world incidents illustrate how these risks affect healthcare environments. Over the past several years, cybersecurity attacks against healthcare organizations have been persistent and large in scale. Messinis et al. report that in 2023 alone, at least 2,620 healthcare organizations worldwide were affected by data breaches, resulting in the exposure of approximately 77.2 million records. Hacking and ransomware campaigns accounted for roughly 88.5% of reported incidents and were responsible for the vast majority of compromised records. Healthcare also continues to experience the highest average breach cost of any industry, reaching approximately 10.9 million USD per incident in 2023 (Messinis et al., 2024). Although not all of these attacks directly targeted IoMT devices, they demonstrate that healthcare environments continue to be highly attractive targets for cyber threats.

One of the most widely cited examples is the WannaCry ransomware attack, which disrupted the United Kingdom's National Health Service and caused widespread outages across hospital systems connected to medical equipment and clinical networks (Sadhu et al., 2022). While the malware did not specifically target medical devices, hospitals were forced to divert patients and cancel approximately 19,000 appointments. The total estimated cost of mitigation

and recovery was approximately GBP 92 million, and multiple systems interacting with medical devices had to be taken offline during the response (Sadhu et al., 2022).

Beyond large-scale ransomware incidents, research has shown that security weaknesses in specific classes of medical devices can pose direct risks to patient safety. Devices such as infusion pumps, cardiac implants, and remote patient-monitoring systems have been found to contain vulnerabilities, including hardcoded credentials and outdated wireless configurations, as well as communication links without proper protection. While these device-specific weaknesses do not always result in highly public incidents like WannaCry, they represent exploitable technical flaws that can be abused once network access is obtained. Even when medical devices are not the primary target for attacks, they are often indirectly exposed through compromised networks or shared infrastructure. These incidents highlight the importance of early detection of abnormal behavior at the perception and network layers, since failures and misuse often occur there. These incidents show that significant risk in healthcare comes not only from adversary tactics, but also from the difficulty of securing medical devices that operate under tight technical constraints.

2.4 Challenges of Securing Resource-Constrained Medical Devices

Securing IoMT devices is challenging in practice due to inherent hardware limitations, including limited processing power and memory, along with very restricted storage capacity. In this research, resource-constrained devices include medical devices and gateways with less computational and storage resources than traditional enterprise or cloud-based systems. These constraints limit which security controls can be deployed directly on medical devices. Surveys show that many sensors, wearables, and clinical devices were not designed to support endpoint security agents, the overhead that encryption introduces, or continuous monitoring functions

(Messinis et al., 2024). As a result, even basic controls like secure communication protocols, logging, or frequent firmware updates are often difficult to implement.

IoMT deployments also introduce interoperability challenges. Healthcare environments typically consist of devices from multiple vendors, each using their own operating system and firmware, often with different communication standards and data formats. Sadhu et al. note that this can lead to inconsistent security controls and a lack of standardized access control mechanisms across devices (Sadhu et al., 2022). Even devices that perform similar functions may differ in security posture and update mechanisms.

Another significant challenge is the limited visibility into IoMT device behavior. Due to storage and processing constraints, devices often cannot generate detailed audit logs, and some provide minimal telemetry altogether. Nozza demonstrates that these limitations force healthcare organizations to rely on external monitoring rather than device-level data to identify abnormal behavior (Nozza, 2023). This lack of visibility complicates incident investigation as security teams may not have sufficient information to determine the scope of an attack.

These challenges motivate the review of existing anomaly detection research for IoMT and ICS environments in the following chapter. The next chapter looks more closely at machine learning approaches for IoMT, especially whether they can operate at the edge and whether they fit into real SOC environments.

Chapter 3: Literature Review

This chapter reviews prior research on anomaly detection in Internet of Medical Things (IoMT) and ICS-adjacent environments. The focus is on machine learning approaches, deployment constraints at the edge, and gaps in existing work pertaining to operational integration.

3.1 Introduction

3.1.1 Purpose of the Literature Review

The rapid expansion of the Internet of Medical Things (IoMT) has led to the widespread interconnection of medical devices, sensors, and applications that generate continuous telemetry (Sadhu et al., 2022). At the same time, healthcare experiences a disproportionately high rate of cybersecurity incidents compared to other sectors, with some reports estimating rates to be 3.4 times higher. This statistic highlights that IoMT devices remain a significant attack vector for threat actors today (Sindhuja et al., 2023).

Prior work in IoMT threat detection shows that medical device traffic can be analyzed using machine learning approaches to distinguish normal behavior from anomalies. In this thesis, anomaly detection refers to identifying deviations from normal device or network behavior instead of relying on predefined attack signatures. This is relevant in healthcare environments where labeled attack data is hard to find and device behavior varies significantly across vendors and clinical environments.

High-level frameworks like the National Institute of Standards and Technology Cybersecurity Framework (CSF) 2.0 emphasize detection outcomes like continuous monitoring, adverse event analysis, and event correlation (NIST, 2024). However, these frameworks are intentionally outcome-focused and do not prescribe specific implementations for embedded or resource-

constrained medical devices. As a result, several questions remain pertaining to implementation: which ML models are realistic for edge-class hardware, what telemetry can be collected, and how should anomaly outputs be formatted so that security teams can act on them. This review focuses on prior work that informs those questions.

3.1.2 Scope and Boundaries of the Review

This literature review focuses on four key areas that support the goals of this thesis. First, it builds on the IoMT security landscape introduced in the Chapter 2 and synthesizes work that describes vulnerabilities related to device diversity, protocol limitations, and architectural complexity (Sindhuja et al., 2023).

Second, it examines machine learning approaches for anomaly detection, with attention to both supervised and unsupervised methods. Emphasis is placed on approaches that can function without large, labeled datasets, since labeled attack data if often unavailable in real clinical environments.

Third, the review evaluates edge-computing constraints. Many IoMT and ICS devices operate with limited computational and memory resources, which directly influences which machine learning models are feasible for local deployments and how quickly they can operate (Nozza, 2023).

Finally, this review includes literature on log monitoring and SIEM integration, exploring how device telemetry and anomaly detection results can be fed directly into SIEM tools for monitoring and alerting in real-world healthcare environments.

Some areas are intentionally excluded from this review. Hospital workflows and non-medical IoT deployments are not included unless they directly affect device telemetry collection or threat detection. Frameworks such as NIST CSF 2.0 and NIST SP 800-82 are referenced as needed for

context. These boundaries keep the review focused on work that addresses how lightweight anomaly-detection machine learning models can be deployed at the edge and integrated into SIEMs in IoMT environments.

3.1.3 Why IoMT and ICS Both Require Specialized Detection

IoMT devices were not originally designed to withstand modern cybersecurity threats. Many operate with limited power, memory, and storage, which makes it challenging to use traditional security controls or host-based monitoring agents directly on the devices themselves. Existing surveys note that IoMT devices rely heavily on lightweight wireless protocols such as Bluetooth Low Energy (BLE), ZigBee, Wi-Fi, and MQTT, which are all vulnerable to attacks like replay, spoofing, and unauthorized access due to weak authentication or encryption mechanisms (Koutras et al., 2020). These constraints limit visibility into device behavior and make it challenging to obtain a complete picture of telemetry for threat detection purposes (Sadhu et al., 2022).

Logging is another constraint. Logging functionality is often either limited or disabled to avoid service disruption or to maintain regulatory compliance. As a result, modern endpoint security solutions that depend on host-level telemetry are often impractical. Monitoring of IoMT traffic is therefore typically performed externally through an edge gateway, network tap, or centralized logging platform, rather than on the devices themselves (Nozza, 2023).

Industrial control systems (ICS) share many of these same characteristics. NIST SP 800-82 discusses constraints in operational systems like legacy hardware, minimal computational resources, strict availability requirements, and real-time performance constraints (Stouffer et al., 2023). Since IoMT environments share many of the same constraints, techniques used in ICS monitoring are directly relevant.

3.2 Machine Learning Approaches to IoMT/ICS Anomaly Detection

3.2.1 Traditional Supervised Machine Learning Approaches

Supervised machine learning has played a major role in earlier IoMT intrusion detection research because the models are straightforward to train when labeled datasets are available. Studies apply a wide range of models from tree-based to deep-learning architectures to identify abnormal traffic patterns.

Many of the studies rely on synthetic or benchmark IoMT datasets, which limit generalizability but still reveal clear performance patterns

Tree-based methods such as Random Forest and XGBoost frequently achieve high classification accuracy. For example, in a study using the CICIoMT2024 dataset, Random Forest achieved 99% accuracy, outperforming XGBoost (98%) and Decision Trees (97%) when detecting attacks like DoS, spoofing, malformed packets, and MQTT-based threats (Lipsa et al., 2025). Another IoMT-based study using simulated data found similar results. Random Forest reached 99.13% accuracy and 99.77% sensitivity, higher than both KNN (99.05% accuracy and 92.15% sensitivity) and Logistic Regression (89.73% accuracy and 99.70% sensitivity) in detecting DoS and delay attacks created in an OMNeT++ testbed (Al-Abadi et al., 2023).

Ensemble methods combine the strengths of multiple models to improve performance. One study from Goumidi and Pierre using a stacking model evaluated on an IoMT dataset achieved 98.02% accuracy and correctly predicted 97 out of 100 real-time messages, outperforming Random Forest, XGBoost, ANN, and KNN in both pre-training and detection phases (Goumidi & Pierre, 2025). Other work shows that combining tree-based models with feature selection improves detection while keeping the model lightweight enough for resource-

constrained deployments. Their Random Forest variant reached 99.89% accuracy on the CICIDS2017 IoMT dataset (Balhareth & Ilyas, 2024).

Deep-learning approaches, including LSTM, CNN-LSTM, and GRU, have also been evaluated in IoMT contexts. In one study, the LSTM and GRU models were able to detect shifts in DoS and DDoS patterns effectively, with validation accuracy nearing 1.0 in later training iterations (Chandekar et al., 2024). CNN-LSTM models trained more efficiently due to the CNN's feature-extraction stage. The main drawback is that these models usually need more computing power than many IoMT devices can support.

Although supervised models achieve strong results in experimental settings, much of the existing work focuses on either centralized or cloud-based processing rather than on edge deployments. Additionally, very few studies report operational metrics such as sustained CPU usage, memory footprint, or latency under a continuous load. This makes it challenging to assess whether these supervised approaches would work in a resource-constrined IoMT environment.

3.2.2 Unsupervised Machine Learning Approaches

Unsupervised learning is now widespread in IoMT and ICS security research because it does not require labeled attack data. Instead, unsupervised models learn a baseline of typical behavior and flag deviations as anomalous.

In practice, most anomaly detection research applies novelty detection, where models are trained primarily on normal traffic. Common unsupervised methods include Isolation Forest (IF), Local Outlier Factor (LOF), One-Class SVM (OCSVM), and density-based approaches like Gaussian or kernel density estimation.

Reported results vary widely depending on the characteristics of the dataset used as well as class balance. Ea et al. evaluate IF, LOF, OCSVM, and Elliptic Envelope models on a 50,000-sample IoMT telemetry dataset that includes heart rate, SpO2, blood pressure, temperature, and fall-detection data. This dataset included about 22,000 normal and 27,000 anomalous records. Reported performance varied significantly across models. IF reached an accuracy of 55.0%, a precision of 75.4%, a recall of 55.0%, an F1-score of 40.0%, and ROC-AUC of 75.5%. LOF (86.8% accuracy, 89.8% precision) and OCSVM (76.4% accuracy and 80.6% precision) did achieve better detection metrics but required greater computational and memory resources (Ea et al., 2024). These results show that LOF and OCSVM are more sensitive to subtle anomalies.

Other studies highlight challenges related class imbalance and changing behavior within the dataset over time. Work from Idowu (2025) used an edge-based Isolation Forest experiment and introduced three types of anomalies (outliers, drift, and seasonal changes) into a synthetic IoT dataset. The overall accuracy was 58%, but the model behaved very differently across classes. For the normal class, the precision was 92%, recall was 60%, and F1-score was 72% on a dataset of 183 samples. However, for the attack class, precision dropped to 10%, recall was 47%, and F1-score was 16% on a set of just 17 samples (Idowu, 2025).

Ea et al. examine a similar challenge where a change in data distribution over time degrades model performance. To combat this, they retrain their clustering model every 5,000 samples to handle concept drift before updating their IF, LOF, OCSVM, and EE models (Ea et al., 2024). While this approach improves model adaptability, it does increase computational load. Altogether, these results suggest that unless class imbalance and drift are explicitly addressed, unsupervised methods tend to struggle with minority attack classes and often fail to adapt to changing traffic patterns.

Finally, many studies and surveys show that while unsupervised models are gaining popularity in IoMT security, their effectiveness still varies. Tabassum et al. combine clustering with IF and LOF models on over two million electronic health record (EHR) audit-log entries and achieved a sensitivity of 90.1% and specificity of 96.5% when detecting unauthorized EHR access. The IF-based models generally outperformed LOF in accuracy, sensitivity, and F1-score (Tabassum et al., 2024). Vafadoost Sabzevar et al. reviewed current IoMT anomaly detection work and reported that an IF model used on the NSL-KDD network dataset achieved a 90.54% accuracy across both normal and attack traffic (Vafadoost Sabzevar et al., 2023). Conversely, edge-based systems like LightESD show a lightweight model that can perform well with an average F1-score of about 87%, whereas an Isolation Forest baseline in the same experiments averaged about 58% F1-score (Das & Luo, 2023).

Overall, unsupervised methods offer flexibility and avoid reliance on labeled attack data, but their performance depends heavily on dataset characteristics, managing drift, and resource constraints.

3.2.3 Isolation Forest: Mechanism and Advantages

Isolation Forest (IF) is one of the most widely used unsupervised anomaly-detection methods in both IoMT and ICS research due to its computational efficiency. Instead of explicitly modeling normal behavior, IF isolates data points with random partitioning. Points that require fewer splits to isolate are assigned higher anomaly scores (Liu et al., 2008).

This isolation-based scoring mechanism makes IF ideal for IoMT systems where data often comes from different device types or uses different numeric formats. Because it requires minimal parameter tuning and has a near-linear time complexity, it’s well-suited to edge-based environments (Liu et al., 2008).

Despite these advantages, IF also has limitations. IF depends on the assumption that anomalies are easier to isolate than normal data. When anomalies are subtle or when baseline distributions shift over time, performance tends to degrade (Idowu, 2025; Das & Luo, 2023). These tradeoffs must be considered when deploying IF in IoMT environments.

3.2.4 Ensemble, Deep Learning, and Hybrid Methods

A growing number of studies explore ensemble, deep-learning, and hybrid approaches to improve detection performance. While these approaches often perform well in controlled experimental settings, their computational and memory requirements often exceed what most IoMT devices and edge gateways can support.

Several studies illustrate this tradeoff. Li and Zhang evaluate a CNN-LSTM autoencoder (named ConvAE-ALSTM) on two public health datasets and report accuracies of 95.37% and 95.56%, outperforming their best baseline model by 8.6% and 12.3%, respectively (Li & Zhang, 2025). These results demonstrate the strength of deep learning models when evaluated on complex behavior. but also show why these types of architectures are difficult to deploy on resource-constrained devices. Each component (CNN, LSTM, multi-head attention) adds computational and memory overhead.

Zachos et al. report similar findings when evaluating models such as OCSVM, LOF, and KDE on resource-constrained hardware. Although all evaluated models maintained CPU usage below 1% during runtime on a Raspberry Pi, training and tuning required significantly greater resources, limiting feasibility for sustained edge learning (Zachos et al., 2025).

As a result, while complex models can improve performance in a centralized environment, their feasibility in resource-constrained IoMT environments is limited.

3.3 Edge Computing for Anomaly Detection

3.3.1 Why Edge Computing in IoMT

IoMT devices generate a continuous stream of physiological and operational data that often requires real-time interpretation for security purposes. While cloud-based processing can support large-scale analytics, it also introduces additional latency and increases risk of exposure of protected health information (PHI) by transmitting it off site. As a result, researches have explored edge-based anomaly detection as a method to process data closer to where it's generated.

Latency is one of the main drivers behind the shift from cloud-based to edge-based detection. Transmitting telemetry to a centralized cloud platform introduces a round-trip delay, which can slow down the identification of abnormal device behavior. Studies have shown that moving anomaly detection to the edge can significantly reduce response times. For example, one study showed an approximately 80% reduction in detection latency when processing was performed at the edge (Ngo et al., 2020). Lightweight frameworks such as LightESD also point out that much of the delay in cloud-based systems comes from transmitting the raw telemetry to the cloud, not the detection computation itself (Das & Luo, 2023).

Privacy and compliance considerations further motivate edge-based designs. IoMT devices routinely handle sensitive health information and transmitting medical data off-site increases exposure risks. Reviews emphasize that IoMT systems should minimize the amount of patient data that leaves the local environment, especially when devices have limited built-in security controls. (Messinis et al., 2024).

Bandwidth is another practical constraint. Hospitals may use hundreds of connected devices simultaneously, generating a continuous stream of data. Processing data locally reduces

bandwidth consumption by transmitting only alerts or summarized outputs instead of complete telemetry (Das & Luo, 2023). At the same time, in IoMT environments, even small delays in identifying suspicious activity can affect patient health and safety. Detection is most effective when processing occurs locally, avoiding the latency introduced by multi-hop transmission to centralized systems (Nozza, 2023).

3.3.2 Hardware Constraints

Though edge computing is appealing in theory, IoMT gateways operate under much stricter hardware constraints than traditional servers. Even relatively capable single-board computers like the Raspberry Pi 4 have limited processing power (typically a quad-core 1.5 GHz ARM Cortex-A72 CPU) and limited memory (often 2-8 GB of LPDDR4 RAM). Storage is also constrained by microSD-based storage with lower throughput compared to solid state drives.

Studies like Alwaisi et al. highlight the impact of these constraints. Alwaisi et al. found that while Raspberry Pi devices can support traditional machine learning models, performance degrades as the workload complexity increases. In their evaluation, one dataset containing around 4,000 samples caused the Arduino Nano to disconnect during processing, while detection times on a Raspberry Pi were noticeably higher than cloud-based baselines (30.4s versus 23.7s for a k-NN model) (Alwaisi et al, 2024).

Ni et al. also compare the Raspberry Pi and Jetson Nano. While the Jetson Nano provided higher computational throughput, it consumes more energy. Their results also showed that only compact models like Edge Isolation Forest and quantized One-Class SVM achieved both acceptable latency and energy usage. The reported memory footprints for these models were in the range of tens of kilobytes (26.7 KB and 14.2 KB, respectively). (Ni et al., 2025).

These findings indicate that IoMT anomaly detection must prioritize algorithms with minimal memory footprints, low computational overhead, and stable behavior under a sustained load.

3.3.3 Gaps in Existing Edge Methods

Even with clear progress in lightweight detection, current edge-based IoMT research still has several gaps. The most significant gap is the lack of integration with Security Information and Event Management (SIEM) platforms. Even studies that successfully evaluate detection completely at the edge rarely address how alerts generated at edge devices should be correlated and triaged within enterprise security environments. This gap makes edge detection difficult to apply in real-world clinical environments where Security Operations Center (SOC) work depends on central log ingestion pipelines.

A second limitation is that current studies heavily rely on synthetic or partially synthetic datasets (e.g., Zachos et al., 2025; Liu et al., 2021). While this approach simplifies experimentation, it limits the generalizability of these studies to real-world attack behavior.

Finally, current studies provide limited insight into how edge-based detection methods perform under real-world operations. Most evaluations focus on accuracy in controlled environments but do not assess device behavior under sustained stress or degraded network conditions. For example, few studies examine how lightweight models behave when IoMT devices operate at a high CPU load for extended periods of time, when data must be processed continuously, or when connectivity is intermittent, which is a common problem in hospitals and homecare. Many evaluations assume stable networks and ideal operating conditions, which is rarely the case in real clinical environments.

Overall, while lightweight edge detection is technically feasible, many aspects of its practical deployment in IoMT deployments remain unexplored.

3.4 Frameworks, Standards, and Operational Integration

3.4.1 Regulatory Context for IoMT

IoMT environments operate within multiple cybersecurity frameworks and regulatory guidelines that emphasize both continuous monitoring as well as timely detection. The National Institute of Standards and Technology Cybersecurity Framework (CSF) 2.0 lists Detect as a core security function, requiring organizations to recognize when cybersecurity events occur. However, the framework does not prescribe specific technical implementations for embedded medical devices.

Additional guidance is provided by NIST SP 800-82, which outlines security considerations for industrial control systems (ICS). Although focused on operational technology, many of its core principles also apply directly to IoMT devices as medical devices often function similar to smaller OT components. In both cases, systems are safety-critical, resource-constrained, and must operate continuously, which makes behavioral monitoring essential for identifying anomalies (Stouffer et al., 2023).

The MITRE ATT&CK for ICS framework also reinforces the need for behavior-based detection. Many adversary techniques, such as manipulation of device logic, unexpected command sequences, or timing irregularities, do not produce clear signatures, necessitating anomaly detection that can pick up on subtle behaviors (Alexander et al., 2020). Other regulatory guidance from the U.S. Food and Drug Administration emphasize resilience through the medical device lifecycle, requiring manufacturers to implement mechanisms that can identify unsafe conditions as they occur (U.S. Food and Drug Administration [FDA], 2016; FDA, 2023). Across

these frameworks, behavioral monitoring is consistently emphasized, motivating the integration of anomaly detection outputs into centralized security platforms like SIEMs.

3.4.2 SIEM Integration in Existing Literature

Although anomaly detection research is extensive, fewer studies address how model outputs can be integrated into real operational environments. Most studies stop at evaluating models in controlled settings and reporting performance metrics like accuracy, latency, or resource usage. What's missing is the operational layer where detections move from an edge device into a security workflow where analysts can act on them.

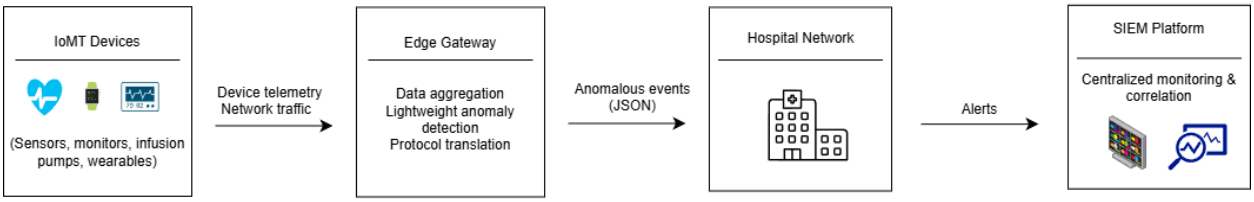
Only a small number of studies extend beyond local experimentation. For example, Zachos et al. include a lightweight Grafana dashboard for visualization purposes, but it is not designed to support security operations such as alert triage and correlation. Moreover, the literature does not describe a method for forwarding anomaly alerts from IoMT edge gateways into SIEM platforms, normalizing the alerts, or correlating them with other security events, despite these steps being standard practice in enterprise security environments. Therefore, anomaly detection in the current literature continues to be disconnected from day-to-day security operations.

This gap matters because anomaly scores by themselves are not actionable. In practice, SOC analysts rely on normalized severity levels, indexable fields that can be queried, and correlation with other data sources like authentication events and network traffic. Without structured integration, anomaly detection continues to be isolated from incident response and long-term monitoring.

In this thesis, SIEM integration refers to the process of converting anomaly scores into structured and normalized alerts that can be indexed and correlated within a centralized monitoring platform. Figure 3.1 illustrates the high-level system architecture, starting with IoMT telemetry to SIEM ingestion.

Figure 3.1

System Architecture Overview for Edge-Based Anomaly Detection and SIEM Integration in an IoMT Environment



3.5 Synthesis

3.5.1 Consensus Findings

The reviewed literature reveals several consistent themes and unresolved challenges in IoMT anomaly detection research.

IoMT and ICS environments share constraints related to resource constraints and safety-critical operation. These characteristics make traditional host-based security unfeasible. As a result, external monitoring and behavior-based detection are frequently used as a solution.

Supervised models often achieve high accuracy in controlled datasets, but they require labeled data and often depend on centralized infrastructure. Unsupervised methods, including Isolation Forest, are considered an alternative solution when labeled attack data is limited.

Edge computing is generally viewed as feasible for lightweight anomaly detection, especially when using models with low computational overhead. Feasibility ultimately depends on hardware constraints and workload intensity.

3.5.2 Identified Research Gap

Although the literature demonstrates that lightweight anomaly detection at the edge is possible, fewer studies address how to integrate outputs into real-world security operations. Most studies do not address how detections transition beyond the edge device itself. In particular, the literature does not describe how anomaly alerts should be formatted, transmitted, or interpreted by security teams once they leave the edge environment. Even studies that incorporate some type of visualization or cloud dashboard do not implement a complete security pipeline capable of supporting alert correlation and triage. This limitation is especially significant in IoMT environments where security decisions require centralized visibility across devices and networks.

This thesis addresses these gaps by implementing an unsupervised, edge-based model such as Isolation Forest directly generating SIEM-ready alerts designed for real-time correlation and triage in an IoMT setting. Addressing this gap is the focus of the work completed in this thesis.

Chapter 4: Methodology

This chapter explains how the system was built, tested, and evaluated. It covers the experimental environment, overall architecture, and the implementation of an edge-based anomaly detection pipeline designed to send alerts into a SIEM-style workflow.

4.1 Experimental Environment

All experiments were conducted in a dedicated virtual machine (VM) to keep the test environment isolated. The VM ran Ubuntu 24.04.3 LTS and was used only for this project to avoid interference from the host system. The VM was provisioned with 4 vCPUs, 16 GB of RAM, and 60 GB of disk space.

The environment included Docker (v29.1.3) for service orchestration and Elasticsearch/Kibana (v8.12.0) to represent the centralized monitoring and analysis layer. Core services, including the MQTT broker and Elastic Stack components, were deployed as containers to keep networking and restarts consistent across sessions. All configuration files and scripts were version-controlled using Git and stored in a private repository so changes could be tracked across experiments.

4.2 System Architecture Overview

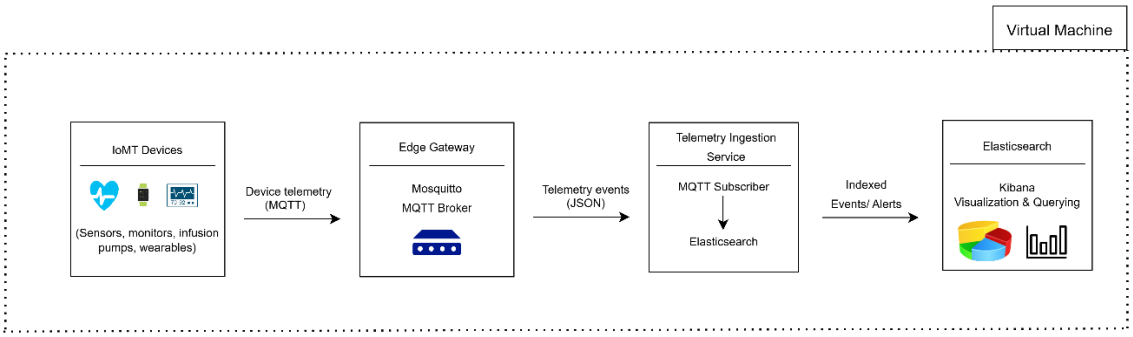
This architecture follows a typical healthcare monitoring flow. Devices send telemetry to a gateway, the gateway forwards structured data for centralized monitoring, and analysts review events in a SIEM-like interface. The design allows anomaly detection logic to be added at the edge without changing the ingestion path or SIEM layer.

As shown in Figure 4.1, simulated IoMT devices publish telemetry over MQTT to a Mosquitto broker running on the edge gateway. A lightweight ingestion service subscribes to

these topics, normalizes message payloads, and forwards structured events into Elasticsearch. Kibana is then used to confirm that events are indexed correctly and can be queried. Long-term storage and analysis occur in the SIEM, while the edge gateway can flag unusual behavior quickly without putting excessive strain on the device.

Figure 4.1

Implemented Edge Ingestion and SIEM Pipeline Architecture (MQTT →Elasticsearch → Kibana)



4.3 Pipeline Design and Validation

The ingestion pipeline was designed to reflect common IoMT and SOC monitoring processes, while remaining simple enough to reproduce. Components were selected based on typical IoMT telemetry transport and standard SOC-style log ingestion workflows.

MQTT was chosen as the telemetry transport protocol due to its widespread use in IoT and IoMT deployments, particularly on low-power devices. Mosquitto was selected as the broker because it is lightweight and easy to containerize. Elasticsearch was used as the backend storage and analysis platform to represent a centralized SIEM. While Elasticsearch alone is not a complete SIEM, it does provide the core capabilities required for security monitoring like structured event indexing and support for correlation and visualization. Kibana was used to

validate ingestion and provide visual confirmation that telemetry events were successfully indexed and able to be queried.

Docker Compose orchestrated all services in one environment with consistent networking. Services communicated over an internal Docker bridge network to minimize host exposure.

Before adding anomaly detection, the pipeline was validated step by step. First, container startup, Elasticsearch cluster status, and Kibana connectivity were verified. Next, synthetic IoMT-style telemetry messages were published to MQTT topics and confirmed at the broker. Finally, full end-to-end ingestion was validated by forwarding telemetry into Elasticsearch and verifying indexing through index counts and Kibana's discovery view. This ensured that any anomalies detected in later experiments could be narrowed down to model behavior instead of pipeline issues.

4.3.1 Baseline Telemetry Definition and Experimental Assumptions

Once the ingestion path was stable, baseline telemetry was defined before introducing machine learning. The baseline dataset was synthetically generated to represent "normal" IoMT device behavior under controlled conditions. All messages were intentionally well-formed, contained consistent fields, and had no missing values. More complex scenarios like malformed payloads or behavioral drift were excluded at this stage so that model behavior could be evaluated on a stable pipeline first.

For anomaly detection, only numeric measurements were used as model inputs. Identifiers and timestamps were kept for context in the SIEM but excluded from the model to prevent it from learning static identifiers instead of behavioral patterns.

Telemetry follows a simple metric–value structure, allowing multiple measurement types to be handled with a single ingestion pipeline. This design also allows new numeric measurements to be added later without restructuring the system.

The baseline telemetry document schema used in the ingestion pipeline is summarized in Table 4.1.

Table 4.1

Baseline Telemetry Document Schema

Field Name	Type	Example Value	Role In Analysis
@timestamp	date	2025-12-10T15:32:45.123Z	Event time; used for ordering and correlation (not a feature)
device_id	text (and keyword subfield)	pump-03	Grouping and analysis context
metric	text (and keyword subfield)	heart_rate	Signal type identifier
mqtt_topic	text (and keyword subfield)	iomt/pump-03/telemetry	Source topic
value	number	78	Numeric feature for anomaly detection

Elasticsearch automatically adds system metadata fields such as _id and _version at index time. These fields are not part of the telemetry schema itself and are excluded from the baseline definition.

Identifiers such as patient IDs, device IDs, and routing metadata are excluded from model inputs so the detector learns behavior rather than identity. If these fields were included, the model could separate records based on static tags instead of temporal patterns in the measurements. Leaving identifiers out helps ensure anomalies are driven by changes in telemetry values over time.

Instead of analyzing raw JSON data, the model works with a simplified numeric representation of each telemetry message. The baseline schema and feature eligibility criteria established in this section are kept only for the initial phase of experimentation. Additional feature engineering is deferred until baseline statistics and model behavior are set.

4.4 Model Selection and Training Procedure

4.4.1 Model Selection and Experimental Assumptions

Isolation Forest was selected because it supports unsupervised anomaly detection without labeled attack data and is lightweight enough for edge-class deployments. Instead of matching known attack signatures, it learns what normal behavior looks like and flags deviations. This is useful in IoMT conditions where labeled attack data is limited and device behavior varies widely across environments.

In this architecture, Isolation Forest is not intended to replace a full intrusion detection system. Its role is limited to flagging potential anomalies early and producing a lightweight anomaly score that can be forwarded to a centralized SIEM platform for correlation and analyst interpretation in the context of other data sources.

Each MQTT message is treated as a single observation. In the initial implementation, the model operates on one numeric feature: the telemetry measurement value itself. Categorical fields, identifiers, and timestamps are excluded from model inputs, and are reserved for analyst context and correlation at the SIEM layer. This keeps the design minimal and computationally efficient.

The model is trained on a fixed window of the first 2,000 baseline telemetry messages. It is trained once and remains static during evaluation to ensure that changes in detection results

reflect telemetry behavior instead of model updates. This specific window was chosen to provide enough observations to stabilize the baseline distribution but minimize training time.

Because the baseline training data is synthetically generated to represent normal behavior, the contamination parameter is set to 0.01, indicating that approximately 1% of the training data will be treated as anomalous by the model. This is intended to limit false positives during initial testing.

In this study, anomalous behavior is defined as numeric deviations from baseline behavior instead of specific attack signatures. Examples include out-of-range values, abrupt spikes or drops, sustained deviation, or unexpected variability. These categories are intentionally broad and focus on behavioral change instead of predefined attack scenarios.

Isolation Forest may perform poorly with subtle deviations or when the baseline distribution shifts significantly over time. The model remains static and does not automatically retrain or adjust for shifts in baseline behavior. These limitations are addressed later in the chapter.

For each event, the model produces an anomaly score and a binary anomaly flag. These outputs are packaged with basic telemetry context (device, metric, timestamp) and are forwarded to the SIEM. The score simply acts as an indicator and is not meant to be a definitive security judgement.

4.4.2 Isolation Forest Training Procedure

The Isolation Forest model was trained in the same VM environment used for pipeline validation. Training was completed using a minimal standalone Python script built around the scikit-learn library to keep dependencies limited and reduce the implementation complexity.

The trained model was saved using joblib as a reusable file for runtime scoring. A separate metadata file recorded the dataset boundaries, parameter configuration, and total training duration to support reproducibility. After saving the model, it was reloaded to confirm persistence and correct configuration.

Training completed in approximately 1.2 seconds for 2,000 records. No additional retraining or parameter tuning was performed.

4.5 Deployment and Ingestion Implementation

All system components were deployed as containerized services using Docker Compose. Docker Compose allowed the stack to be created and managed with a single configuration file, which helps keep the setup reproducible and simplifies the experiment setup. A single docker-compose.yml file defined service configuration, port exposure, persistent volumes, and required environment variables.

Figure 4.2 shows the compose file defining the Mosquitto broker, Elasticsearch, and Kibana services. Persistent volumes and restart policies were included to preserve data across restarts and prevent manual reconfiguration. Centralizing service configuration in one place made the pipeline easier to maintain and reproduce.

Figure 4.2

Docker Compose Configuration for Core Pipeline Services

```
(venv) payton@payton-VirtualBox:~/thesis-lab/docker$ cat  docker-compose.yml
version: "3.8"

services:
  mosquitto:
    image: eclipse-mosquitto:2
    restart: unless-stopped
    container_name: mosquitto
    ports:
      - "1883:1883"
    volumes:
      - ../configs/mosquitto.conf:/mosquitto/config/mosquitto.conf
      - mosquitto-data:/mosquitto/data

  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:8.12.0
    restart: unless-stopped
    container_name: elasticsearch
    environment:
      - discovery.type=single-node
      - xpack.security.enabled=false
    ports:
      - "9200:9200"
    volumes:
      - es-data:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:8.12.0
    restart: unless-stopped
    container_name: kibana
    environment:
      - ELASTICSEARCH_HOSTS=http://elasticsearch:9200
    ports:
      - "5601:5601"
    depends_on:
      - elasticsearch

volumes:
  es-data:
  mosquitto-data:
```

4.5.1 Environment Persistence and Reproducibility

To ensure consistency after VM or container restarts, persistence mechanisms were built into the deployment.

Elasticsearch was configured with a persistent data volume so indexed telemetry survived container recreation. This prevented the need to recreate indices or re-ingest test data manually

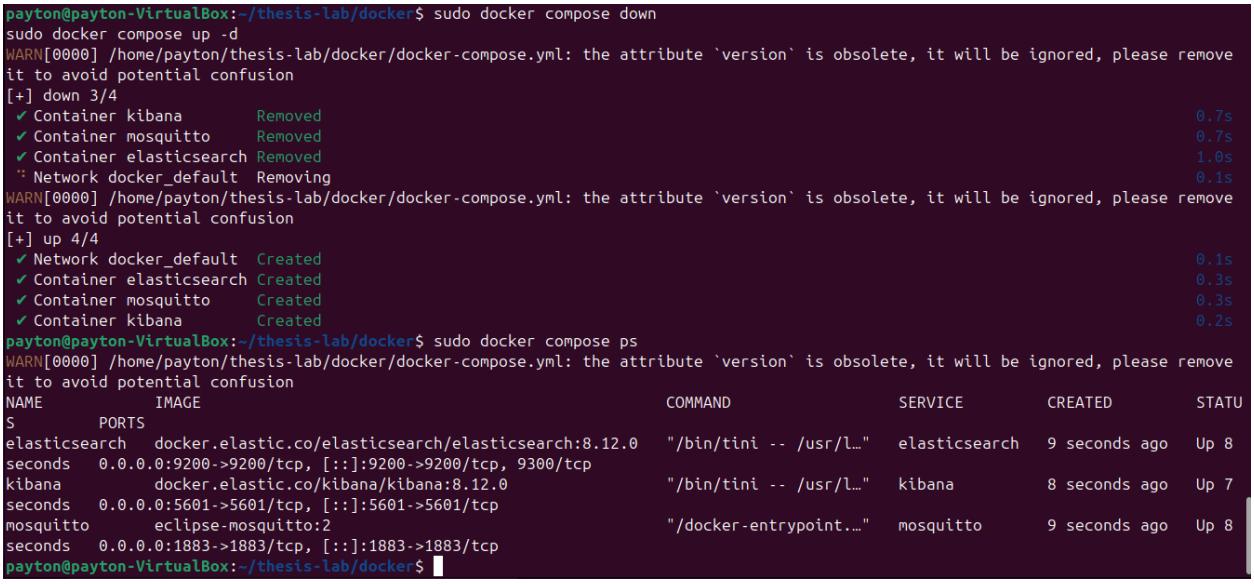
after restarts. The Mosquitto MQTT broker was also configured with a persistent data directory to preserve broker state and queued messages, avoiding reliance on temporary container storage.

Persistence was tested through repeated VM and container restarts. After each restart, service health, MQTT publish-subscribe functionality, and Elasticsearch index availability were verified. Previously ingested telemetry remained accessible in Kibana, confirming that the environment could recover without any manual intervention. All core services (Mosquitto, Elasticsearch, Kibana, and the ingestion subscriber) were configured to automatically restart after host reboots so ingestion could resume without user action.

This allows the full stack to restart cleanly without manual reconfiguration and ensures that telemetry ingestion stays consistent across VM sessions. Figure 4.3 shows the running Docker containers used in the experimental environment, confirming that all services are active and accessible after restart validation.

Figure 4.3

Full Docker Restart Testing



4.5.2 Service Health and Inter-Container Communication

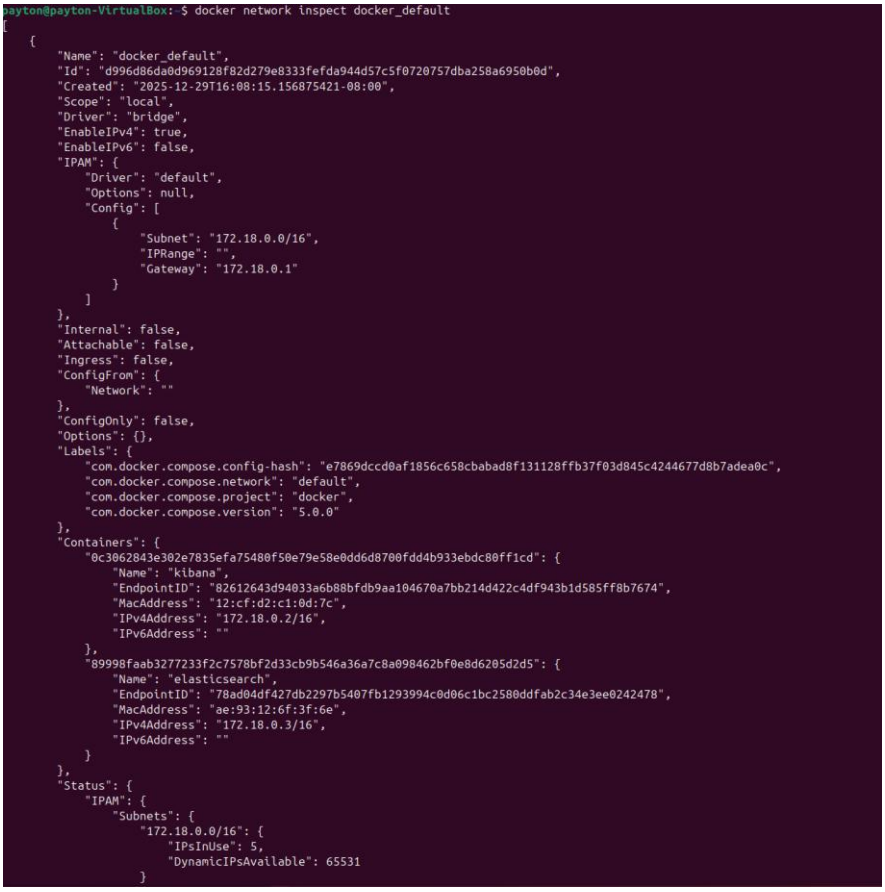
Service health and networking were validated before ingestion testing began. Elasticsearch cluster health and Kibana connectivity were confirmed to ensure the backend was stable.

Docker networking was inspected to verify that all services communicated through the internal bridge network without relying on host-level routing. Verifying inter-container communication early helped isolate any later issues to application behavior instead of Docker networking.

Figure 4.4 shows the Docker network inspection output confirming that all pipeline components operate within an isolated internal network.

Figure 4.4

Docker Network Configuration for Pipeline Services



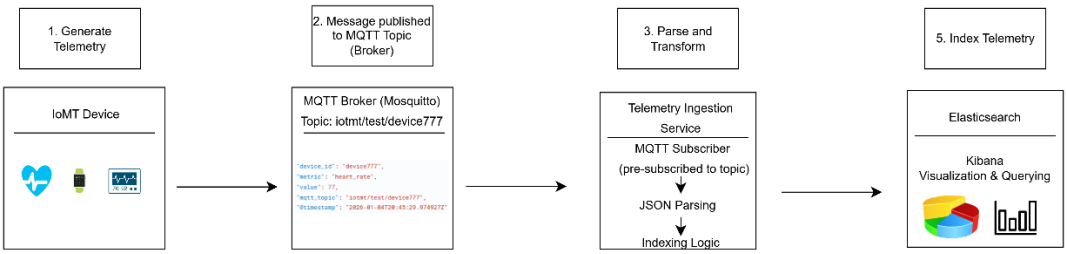
4.6 Telemetry Ingestion via MQTT

MQTT was used to simulate telemetry transport in an IoMT environment. Its lightweight publish-subscribe model fits the constraints typically seen in medical devices and gateways.

Figure 4.5 shows the overall telemetry ingestion workflow for this research. Simulated IoMT devices publish telemetry messages to MQTT topics. The Mosquitto broker receives and queues these messages. A lightweight ingestion service subscribes to relevant topics, parses and normalizes the payloads, and forwards structured JSON documents to Elasticsearch for indexing and analysis in Kibana.

Figure 4.5

Telemetry Ingestion Workflow from IoMT Device to SIEM



Initial validation involved manually publishing test telemetry messages to MQTT topics and confirming receipt through local subscribers. This confirmed that the broker was functioning correctly and capable of receiving device-generated data.

Figure 4.6 shows manual publication and subscription output confirming that well-formed telemetry messages were successfully transmitted and received.

Figure 4.6

Manual MQTT Telemetry Publication for Pipeline Validation

To remove manual interaction, an automated MQTT-to-Elasticsearch ingestion service was implemented. A lightweight Python script subscribes to configured topics, transforms message payloads into structured JSON, and forwards them to Elasticsearch using its REST API.

A systemd service was created to run the ingestion script at boot and automatically restart it if dependencies were temporarily unavailable. During startup, the service may restart briefly while Elasticsearch becomes available. Systemd's retry behavior allows the ingestion service to recover automatically once Elasticsearch becomes reachable.

Figure 4.7 shows the service definition specifying restart behavior, execution context, and required environment variables to keep the ingestion service active across reboots and failures.

Figure 4.7

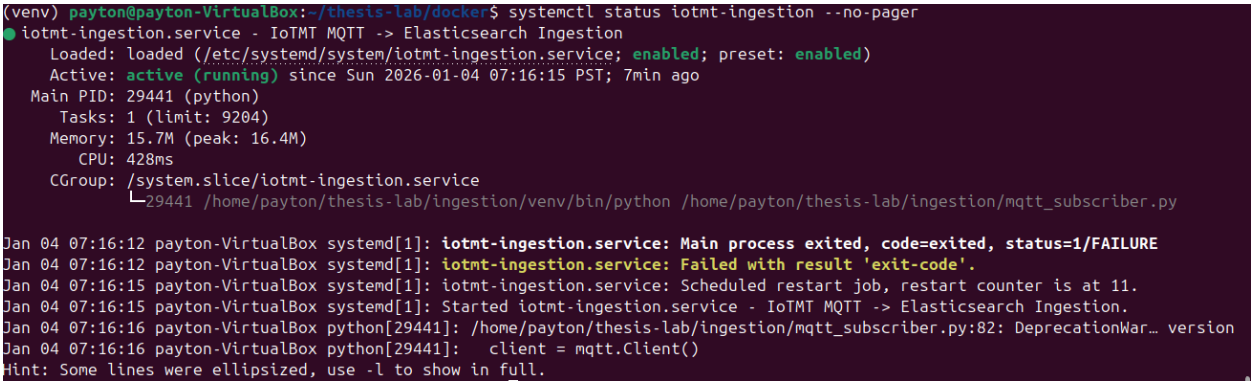
Systemd Service Configuration for MQTT-to-Elasticsearch Ingestion

```
(venv) payton@payton-VirtualBox:~/thesis-lab/docker$ cat /etc/systemd/system/iotmt-ingestion.service
[Unit]
Description=IoTMT MQTT -> Elasticsearch Ingestion
After=network-online.target docker.service
Wants=network-online.target

[Service]
User=payton
WorkingDirectory=/home/payton/thesis-lab/ingestion
Environment=MQTT_HOST=127.0.0.1
Environment=MQTT_PORT=1883
Environment=MQTT_TOPIC=iotmt/test/#
Environment=ES_HOST=http://127.0.0.1:9200
Environment=ES_INDEX=iotmt-test
ExecStart=/home/payton/thesis-lab/ingestion/venv/bin/python /home/payton/thesis-lab/ingestion/mqtt_subscriber.py
Restart=always
RestartSec=3

[Install]
WantedBy=multi-user.target
```

Figure 4.8 shows system status output confirming that ingestion runs as a persistent background service and does not require an interactive session, supporting continuous telemetry collection.

Figure 4.8

Active Status of IoMT Ingestion System Service



4.7 Thresholding and Scoring Design

4.7.1 Runtime Edge-Based Scoring and Thresholding

After training the Isolation Forest model, a lightweight runtime scoring component was implemented to evaluate incoming IoMT telemetry at the edge gateway.

The runtime scoring logic was implemented as a standalone Python utility that loads the Isolation Forest model once at startup and scores each telemetry messages individually. Each message is converted into the same numeric input used during training to keep training and inference consistent. In the initial implementation, the model operates on a single numeric feature representing the telemetry value. All identifiers and metadata are excluded from the model input and are kept only for context.

Instead of choosing a cutoff manually, the anomaly threshold was calculated from the baseline training distribution and stored alongside the trained model. Saving the threshold with the model ensures that runtime scoring uses the same decision boundary established during training.

At runtime, each telemetry event is scored and flagged if it falls below this threshold. Inference latency was measured per event to evaluate edge feasibility. After the initial model load, scoring stabilized between approximately 6–20 milliseconds per event in the virtualized environment, indicating low overhead.

Each scored event produces a structured JSON record containing:

- anomaly score
- anomaly flag
- applied threshold
- scoring latency
- device metadata

These enriched telemetry records are designed to be forwarded to the SIEM for centralized correlation and investigation. At the edge, the model's role is intentionally limited to flagging events that look unusual, rather than issuing definitive alerts or deciding on root causes. Final interpretation and response decisions are deferred to the SIEM layer, where additional context and other data sources become available.

4.8 Edge Deployment Constraints

4.8.1 Edge Gateway Role and Design Constraints

The edge gateway is designed as a lightweight intermediary between IoMT devices and centralized monitoring systems. While development occurred in a virtualized environment, design decisions reflect constraints typically found in healthcare edge deployments.

A Raspberry Pi–class device is used to represent an example of edge hardware. It is not meant to be treated as a certified medical device, but instead it is used as a realistic reference for limited CPU and memory capacity.

Edge functionality is intentionally constrained. The gateway performs:

- telemetry preprocessing
- anomaly scoring
- alert forwarding

It does not perform:

- long-term storage
- advanced analytics
- correlation across devices

4.8.2 Edge Hardware Baseline Validation

A Raspberry Pi running a 64-bit Debian-based Linux distribution was configured with a minimal operating system installation to reduce background overhead. Network connectivity was validated using basic ICMP tests to external hosts, confirming stable outbound communication.

A dedicated Python virtual environment was created on the device using Python 3.13 to isolate project dependencies. Only the runtime libraries required for inference and alert forwarding were installed.

The project repository was cloned directly onto the Raspberry Pi. The trained Isolation forest model and threshold artifact were transferred without modification, reinforcing the separation between training and deployment phases.

System-level metrics (available memory, storage, processor architecture, and kernel information) were recorded to document hardware constraints.

The runtime script was tested using safe execution options (such as argument checks and dry-run execution) to confirm correct behavior without modifying data. No memory exhaustion, swapping, or instability occurred during testing. This validated that the architecture could operate on edge hardware without modification, supporting deployment beyond just the virtualized test environment.

4.8.3 Behavioral Sanity Checking

Before integrating scored events into the SIEM workflow, behavioral sanity checks were performed. Telemetry values within the expected operating range consistently produced scores above the anomaly threshold. Clearly abnormal values (such as extreme numeric spikes) produced lower scores and were flagged as anomalous. This suggested that detection was driven by patterns learned during training, not fixed thresholds or predefined rules.

4.9 SIEM Integration and Alert Schema

To confirm that the SIEM layer functioned independently of the ingestion pipeline, test events were indexed directly into Elasticsearch using its REST API. This verified that the cluster could accept documents without relying on MQTT or the subscriber service.

Following manual ingestion, Kibana was used to confirm that indexed documents were searchable and visible. This step ensured that telemetry originating at the edge could be made accessible to security analysts for use in the investigation and monitoring tasks typically performed in a security operations center.

Figure 4.9 shows cluster health output indicating a stable single-node Elasticsearch deployment (green status).

Figure 4.9

Elasticsearch Cluster Health Status

```
payton@payton-VirtualBox:~/thesis-lab/docker$ curl http://localhost:9200/_cluster/health?pretty
{
  "cluster_name" : "docker-cluster",
  "status" : "green",
  "timed_out" : false,
  "number_of_nodes" : 1,
  "number_of_data_nodes" : 1,
  "active_primary_shards" : 0,
  "active_shards" : 0,
  "relocating_shards" : 0,
  "initializing_shards" : 0,
  "unassigned_shards" : 0,
  "delayed_unassigned_shards" : 0,
  "number_of_pending_tasks" : 0,
  "number_of_in_flight_fetch" : 0,
  "task_max_waiting_in_queue_millis" : 0,
  "active_shards_percent_as_number" : 100.0
}
```

Figure 4.10 shows manual document indexing performed using the Elasticsearch REST API to confirm index accessibility and document storage independent of upstream services.

Figure 4.10

Direct Telemetry Indexing Test via Elasticsearch REST API

```
payton@payton-VirtualBox:~/thesis-lab/docker$ curl -X POST "http://localhost:9200/iotmt-test/_doc" \
-H "Content-Type: application/json" \
-d '{
  "@timestamp": "2025-12-30T18:00:00Z",
  "device_id": "device004",
  "metric": "heart_rate",
  "value": 72
}'
{"_index":"iotmt-test","_id":"BiCehZsBnLC2OIXpPEV6","_version":1,"result":"created","_shards":{"total":2,"successful":1,"failed":0},"
_seq_no":0,"_primary_term":1}payton@payton-Vircurl http://localhost:9200/_cat/indices?vlocalhost:9200/_cat/indices?v
health status index      uuid                   pri rep docs.count docs.deleted store.size pri.store.size dataset.size
yellow open   iotmt-test A7TAiLpnT0O4iZa5_6jMbQ   1   1          1            0      5.9kb          5.9kb        5.9kb
```

Figure 4.11 shows Elasticsearch index summary showing successfully stored baseline telemetry documents, confirming functional indexing.

Figure 4.11

Indexed Baseline Telemetry Documents

```
(venv) payton@payton-VirtualBox:~/thesis-lab/docker$ curl -s "http://localhost:9200/_cat/indices/iotmt-test*?v"
health status index      uuid                   pri rep docs.count docs.deleted store.size pri.store.size dataset.size
yellow open   iotmt-test KJdDqK-nRm2ThnEdP7vUYw   1   1          2            0     13.3kb         13.3kb       13.3kb
```

Figure 4.12 shows the Kibana discovery view, confirming that indexed telemetry documents are searchable and filterable.

Figure 4.12

Kibana Index View

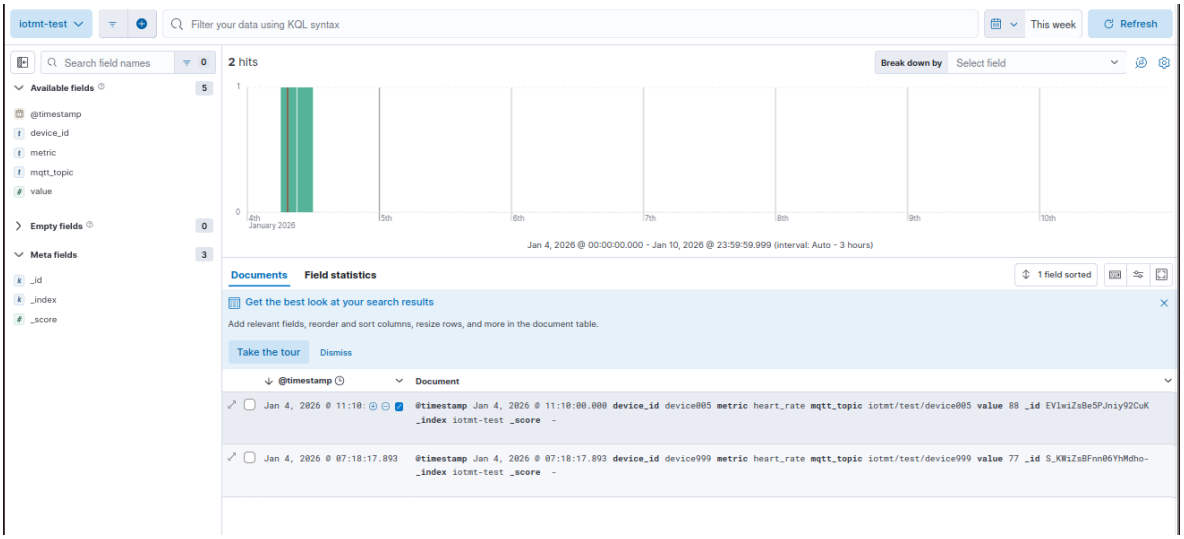


Figure 4.13 shows the structure of an individual stored telemetry document.

Figure 4.13

Sample Telemetry Document


```
{
  "_index": "iomt_anomaly_alerts",
  "_id": "kA1s3pwBppdCffplNhFY",
  "_version": 1,
  "_score": 0,
  "_source": {
    "@timestamp": "2026-01-08T17:52:15.481548+00:00",
    "device_id": "device_05",
    "metric": "temp_f",
    "value": 200,
    "score": -0.05869286718267919,
    "soc_score": 95,
    "threshold": -2.558039500707987e-18,
    "latency_ms": 2550.4977703094482,
    "is_anomaly": true,
    "alert_type": "iomt_anomaly_score",
    "severity": "critical",
    "message": "IoMT anomaly detected: device=device_05 metric=temp_f
     value=200.0 score=-0.05869286718267919 threshold=-2.558039500707987e-18
     severity=critical",
    "pipeline": {}
  },
```


4.9.1 Alert Schema and Enrichment

Runtime anomaly alerts follow a structured schema optimized for SIEM ingestion and analyst triage. These alerts preserve original telemetry context while adding detection-specific fields like anomaly score, threshold, severity, and inference latency.

Baseline telemetry and anomaly alerts are written to separate indices. The original telemetry data remains unchanged. When an anomaly is detected, a separate alert record is created that includes additional fields such as the anomaly score, severity level, and model

details. This separation keeps ingestion independent from detection logic and allows alerts to be filtered and correlated separately within the SIEM.

The structured anomaly alerts are persisted within a dedicated Elasticsearch index designed for SIEM ingestion. Figure 4.14 illustrates the document mapping used to define alert attributes, anomaly scores, and associated device metadata.

Figure 4.14

Anomaly Alert Document Schema (SIEM Index)

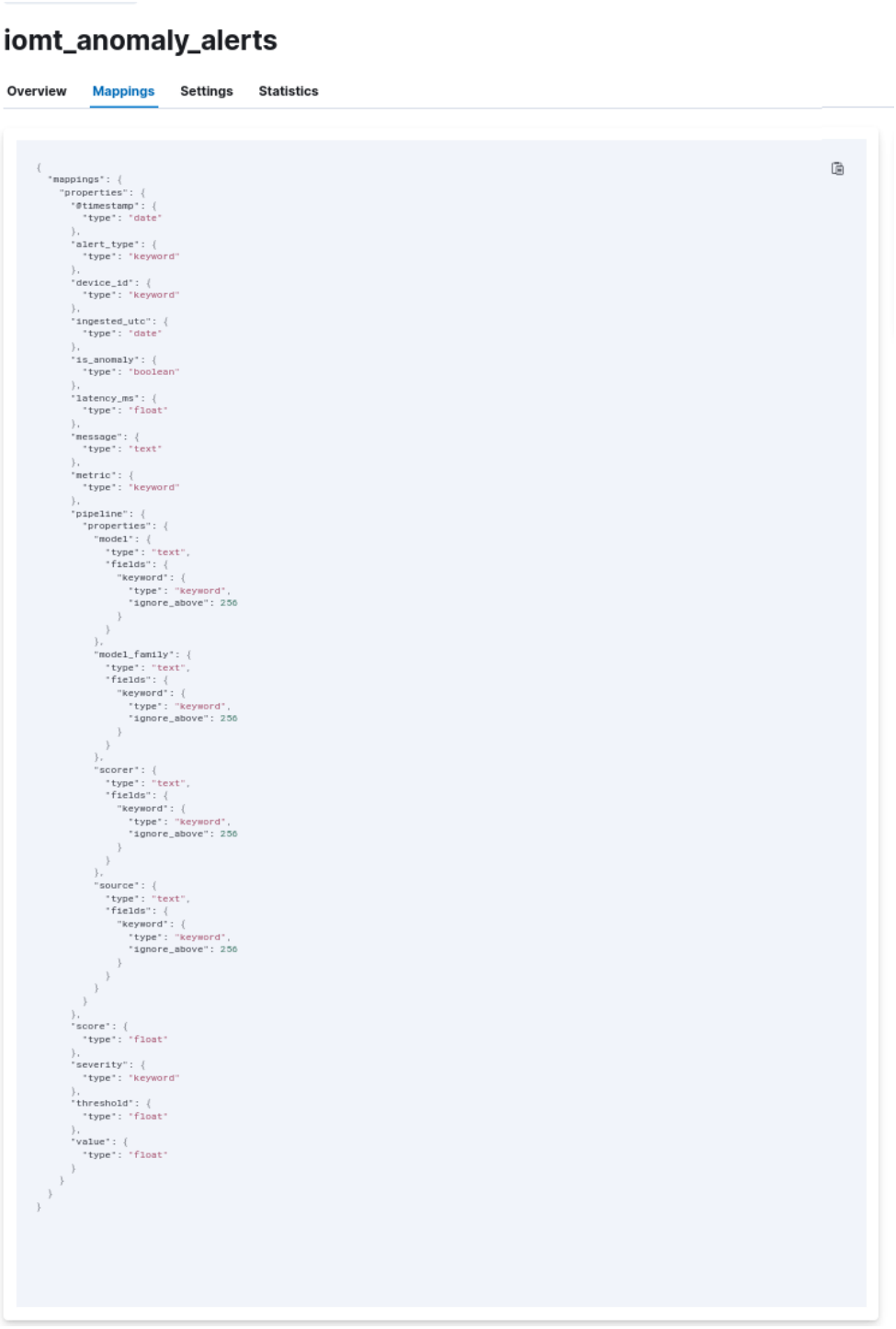


Table 4.2 summarizes the schema used to represent anomaly alerts, including device identifiers, telemetry metadata, model scores, and threshold parameters.

Table 4.2

Anomaly Alert Document Schema for SIEM Integration

Field Name	Data Type	Description
@timestamp	date	Event timestamp derived from the original telemetry message
device_id	keyword	Unique identifier of the IoMT device generating telemetry
metric	keyword	Name of the telemetry metric (e.g., heart_rate_bpm)
value	float	Observed metric value from the device
score	float	Isolation Forest decision_function score (lower = more anomalous)
threshold	float	Static anomaly threshold derived from baseline telemetry
is_anomaly	boolean	Indicates whether the event is anomalous (score < threshold)
latency_ms	float	Per-event inference latency measured at the edge (milliseconds)
alert_type	keyword	Alert classification (e.g., iomt_anomaly_score)
severity	keyword	Categorical severity label derived from anomaly characteristics
message	text	Human-readable alert description for analyst consumption
ingested_utc	date	Timestamp when the alert was ingested into the SIEM
model_family	text (+ keyword subfield)	Model family/version identifier (e.g., isoforest_v1)
pipeline.model	text (+ keyword subfield)	Path or identifier of the model artifact used for scoring

pipeline.scorer	text (+ keyword subfield)	Scoring component responsible for inference
pipeline.source	text (+ keyword subfield)	Source script or pipeline stage that generated the alert

Some fields are stored as text with an additional keyword subfield (for example, pipeline.source.keyword), which allows both free-text search and exact-match filtering and aggregations in Kibana. Fields prefixed with pipeline. are stored as nested properties under the pipeline object in the Elasticsearch mapping.

4.10 Runtime Measurement and Evaluation Design

4.10.1 Runtime Performance Measurement Methodology

Runtime performance was measured using the Linux time -v utility to capture execution time, CPU utilization, and memory consumption at the operating system level.

Baseline measurements were collected in the virtualized environment before testing on edge hardware (the Raspberry Pi). This provides a stable reference point for interpreting edge deployment performance.

Both environments processed the same fixed evaluation dataset of 9,999 telemetry records using a pre-trained Isolation Forest model in inference-only mode.

4.10.2 Evaluation Design, Ground Truth, and Metrics

Normal behavior is represented by synthetic baseline telemetry generated within expected value ranges. Anomalies were introduced by injecting abnormal values and patterns into the same telemetry schema. These injected events were labeled for evaluation only and were not used during training.

Per-event inference latency was measured by timing how long it took to score each telemetry record during execution. CPU utilization, memory usage, and related statistics were collected using operating system–level monitoring tools during execution.

Evaluation metrics include:

- ROC curve and AUC to measure how well anomaly scores separate baseline events from injected anomalies across decision thresholds.
- Precision to estimate the proportion of flagged events that correspond to injected anomalies under controlled conditions.
- Recall, to estimate the proportion of injected anomalies that are successfully detected
- F1 score, to provide a combined summary of precision and recall.

Evaluation was conducted in two distinct phases. The first phase used clearly defined injected anomalies to confirm that the model could separate normal and abnormal telemetry under controlled conditions. The second phase introduced more subtle and overlapping deviations to simulate behavioral drift. This staged design allows performance to be interpreted across both ideal and more challenging conditions.

Both evaluation phases were conducted in the virtualized environment to ensure repeatability of quantitative metrics. The Raspberry Pi deployment was used only to validate runtime feasibility (e.g., latency and resource usage), not for recomputing ROC-based metrics.

This study does not explore extensive model tuning, comparisons across multiple datasets, or supervised baseline models. The goal is to evaluate a single lightweight configuration under controlled conditions rather than optimize performance across many variations.

4.11 Analyst Triage Framework

Anomaly scores and binary flags indicate deviation, but they do not directly determine response actions. To support practical use within a SOC context, severity levels were mapped to defined score ranges.

Severity classifications are derived from normalized SOC score ranges and represent increasing levels of anomaly confidence. The model produces indicators for review, but the final remediation decision is taken by the analyst.

4.11.1 Structured Severity Mapping

To translate anomaly scores into actionable response tiers, a structured severity mapping framework was defined as shown in Table 4.3.

Table 4.3

Structured Analyst Triage Framework

Severity	SOC Score Range	Interpretation	Required Action	Escalation Path
Informational	0–49	Within expected behavioral variance	Keep for trend analysis	None
Low	50–69	Minor deviation from baseline	Review if recurring	SOC Tier 1
Medium	70–84	Moderate anomaly confidence	Investigate device telemetry	SOC Tier 2
High	85–94	Significant deviation	Deep triage, correlate logs	SOC Tier 2 + Notify stakeholder
Critical	95–100	Sustained, high-confidence anomaly	Escalate, isolate (if safe), notify customer	Incident Response

4.11.2 Example Analyst Triage Notes

Example 1 – Low Severity (Score: 62)

- Device: infusion_pump_07
- Behavior: minor deviation in transmission interval
- Context: No correlated anomalies in last 24 hours
- Assessment: likely baseline fluctuation
- Action: monitor, no escalation

Example 2 – High Severity (Score: 88)

- Device: patient_monitor_03
- Behavior: significant deviation in network communication pattern
- Context: Sustained deviation across multiple intervals, latency spike observed
- Assessment: Suspicious behavioral anomaly
- Action: escalate to Tier 2 investigation

Example 3 – Critical Severity (Score: 96)

- Device: infusion_pump_02
- Behavior: Extreme deviation from learned baseline in outbound network activity
- Context: Sustained and multi-metric deviation
- Assessment: High-confidence anomaly
- Action: Immediately escalate to Incident Response and isolate device from network segment (if clinically safe to do so)

4.12 Pipeline Limitations

The ingestion pipeline was designed for reproducibility rather than full production hardening. Telemetry messages are assumed to be well-formed JSON, and the pipeline does not include advanced mechanisms like buffering or message replay in the event of downstream outages.

During startup, the ingestion service briefly checks whether Elasticsearch is available before beginning to process messages. This prevents the pipeline from failing if services start in a slightly different order. The ingestion service performs a brief availability check against Elasticsearch before initiating MQTT subscriptions. However, the system does not implement fault-tolerant message queues or guaranteed delivery functionality.

Because both the baseline data and the injected anomalies are synthetic, the evaluation reflects how well the model separates clearly defined normal and abnormal behavior under controlled conditions. Precision and recall values depend on how anomalies are injected and should not be interpreted as production-level adversary detection rates.

4.13 Summary

Chapter 4 establishes the experimental environment and pipeline used throughout the remainder of the thesis. Telemetry transport, SIEM ingestion, runtime scoring, and service stability were validated before conducting anomaly detection work. Experimental assumptions defined in this chapter are fixed during initial evaluation to isolate model behavior from infrastructure variability. Future work could extend this design by introducing durable message queues, multi-node deployments, and more realistic failure scenarios to evaluate system robustness.

Chapter 5: Results

This chapter presents a two-stage evaluation of the proposed anomaly detection pipeline, first assessing anomaly score separability under controlled, injection-based conditions and then examining the operational usability of model outputs when transformed into SOC-facing alerts.

5.1 Ground Truth Construction for Evaluation

Because the anomaly detection model is trained using unsupervised learning, labeled attack data is not required during training. However, labeled data is necessary to quantitatively evaluate detection performance using standard metrics such as precision, recall, and ROC-AUC.

To enable controlled evaluation without modifying training logic, a separate labeled evaluation dataset was generated from baseline telemetry using an injection-based labeling approach. Baseline telemetry was left unchanged to represent normal device behavior, and synthetic anomalies were inserted into a subset of records.

Each record in the evaluation dataset includes a binary label:

- 0 represents baseline behavior
- 1 represents an injected anomaly.

These labels are used exclusively for evaluation and are not exposed to the anomaly detection model during training or inference.

5.2 Evaluation Dataset Generation

The evaluation dataset was generated using a standalone script that operates separate from the training and runtime pipeline. This separation helps ensure that the reported metrics reflect model behavior instead of artifacts of the ingestion or scoring pipeline.

A total of 10,000 total telemetry records were created for evaluation:

- 9,000 represent baseline telemetry (label = 0)
- 1,000 represent injected anomalies (label = 1)

This results in a 10% anomaly rate.

Anomalies were injected across multiple telemetry metrics, including heart rate, temperature, battery percentage, and infusion rate. Two types of anomalies were introduced:

- Moderate anomalies, representing plausible but abnormal deviations
- Obvious anomalies, representing clear out-of-range or implausible values

Injection locations were selected randomly using a fixed random seed to ensure reproducibility. All injected records were explicitly labeled, while baseline records kept their original values and were labeled as normal.

Anomalies were evenly distributed across monitored telemetry metrics to prevent bias toward any single signal type. This ensures that evaluation results reflect general anomaly detection performance rather than metric-specific behavior.

The final evaluation dataset keeps the original telemetry structure intact, with additional fields added for anomaly labels and injection type.

5.3 Phase 1: Controlled Injection Evaluation

To evaluate anomaly score behavior under clearly defined conditions, the labeled evaluation dataset described in Section 5.2 was scored using the runtime inference pipeline. Model parameters, training configuration, and thresholds were not modified during this phase.

The objective was to assess whether anomaly scores meaningfully separate baseline telemetry from injected anomalies under controlled conditions.

All metrics were computed using anomaly scores generated during batch inference. Ground-truth labels were assigned solely based on the injection process. The model itself remained fully unsupervised, with labels used only for evaluation.

5.3.1 ROC and Precision-Recall Analysis

Across the full evaluation dataset, the model achieved:

- ROC-AUC: 1.000
- Average Precision: 0.994
- Precision at operational threshold: 0.908
- Recall: 1.000
- F1 score: 0.952

These results indicate near-perfect separation between baseline telemetry and injected anomalies under controlled injection conditions.

Figure 5.1 shows the overall ROC curve for the labeled evaluation dataset. The curve trends toward the upper-left region, reflecting strong anomaly score separability.

Figure 5.1

Overall ROC Curve for Labeled Evaluation Dataset

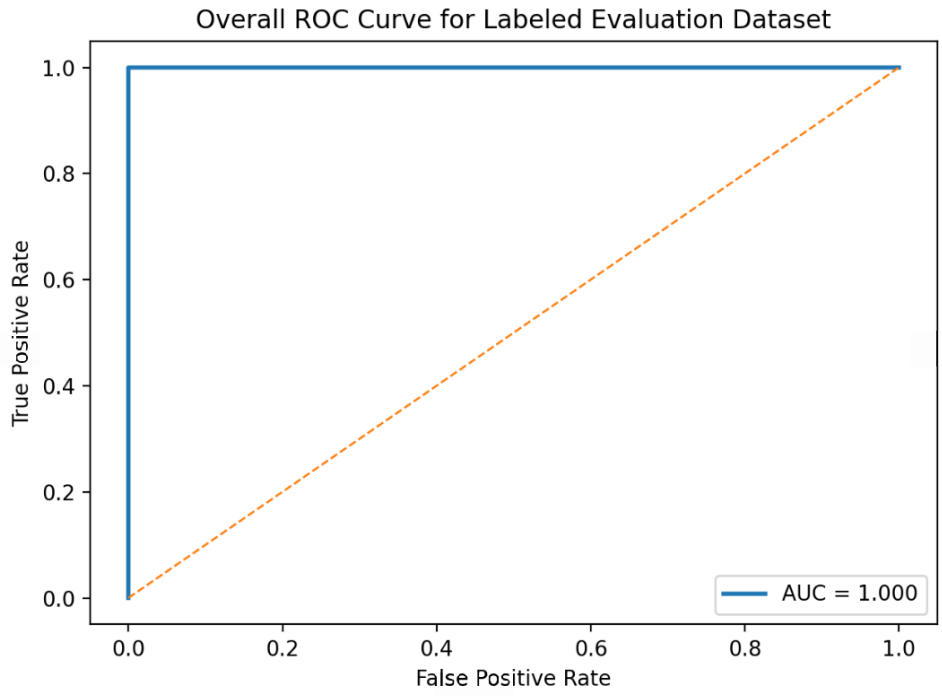


Figure 5.2 presents ROC curves stratified by telemetry metric. Although minor differences appear across individual signal types, all monitored metrics show strong separability, indicating that the model's ranking behavior is consistent across telemetry types.

Figure 5.2

ROC Curves by Telemetry Metric

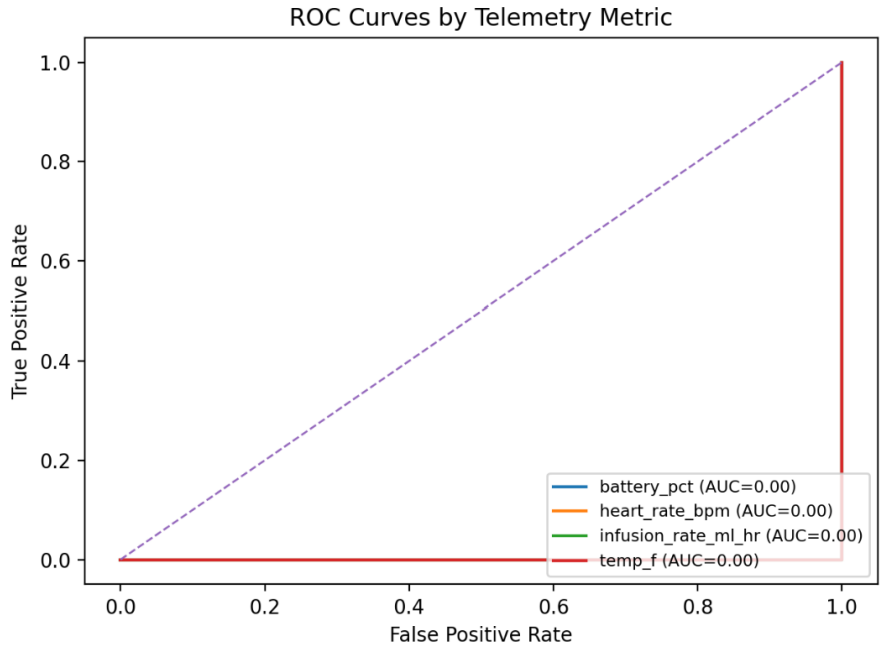


Figure 5.3 shows the overall Precision–Recall curve for the evaluation dataset. This view is useful given the imbalance between baseline telemetry and injected anomalies.

Figure 5.3

Overall Precision–Recall Curve

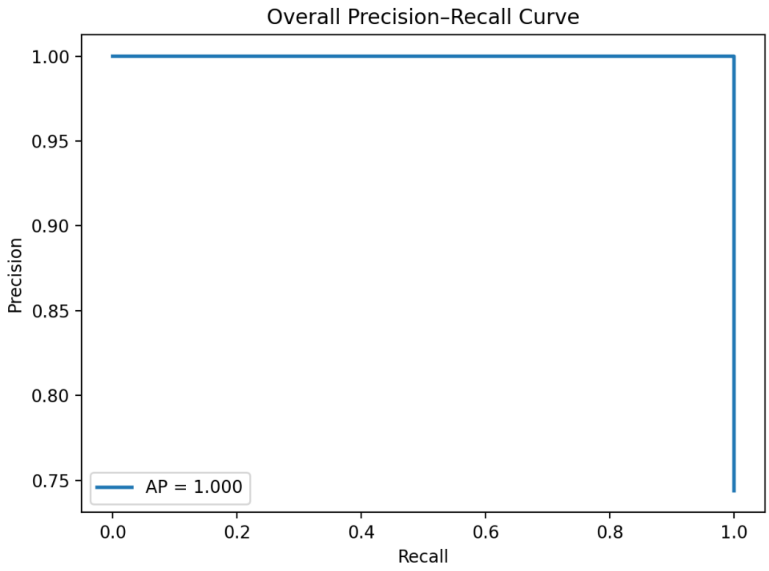
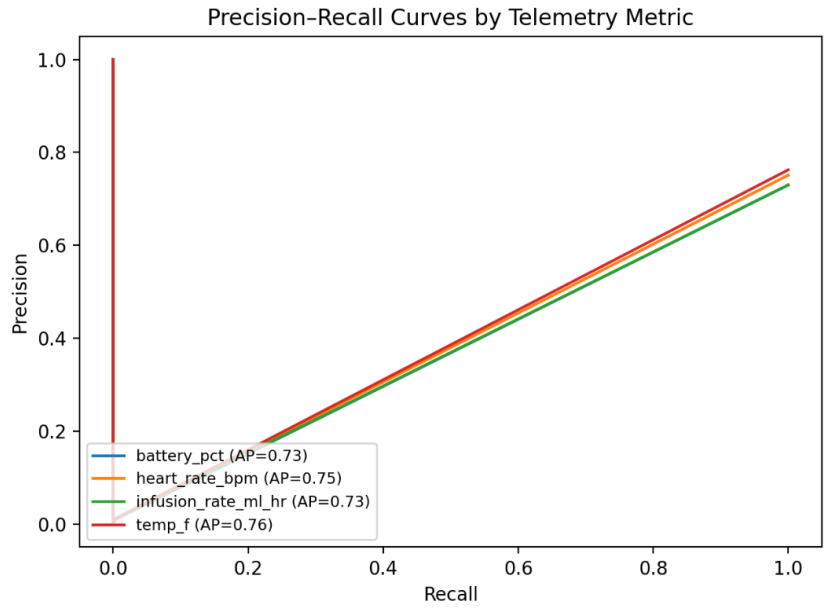


Figure 5.4 displays Precision–Recall curves by telemetry metric, showing how detection performance varies across individual signals.

Figure 5.4

Precision–Recall Curves by Telemetry Metric



5.3.2 Interpretation of Score Separability

The ROC curves confirm that the model effectively ranks injected anomalies above baseline observations across a wide range of thresholds. This validates the anomaly scoring pipeline under controlled conditions.

These results confirm score structure and separability, but do not represent real-world detection performance. The next sections examine runtime feasibility and operational usability under more realistic conditions.

5.4 Runtime Performance Baseline (Virtualized Environment)

Inference over 9,999 telemetry records completed in 58.76 seconds, corresponding to an average per-event inference latency of approximately 5.9 ms.

CPU utilization reached approximately 101%, indicating full saturation of a single virtual CPU core during sustained batch inference. Maximum resident memory usage was approximately 165 MB, with no observed swapping or memory growth over time.

These results establish a stable baseline for comparison with constrained edge hardware and demonstrate that Isolation Forest-based inference operates efficiently under sustained workload conditions.

5.5 Phase 2: Realistic Telemetry Evaluation

5.5.1 Dataset Description and Experimental Consistency

To evaluate model behavior under more realistic conditions, a second evaluation pass was conducted using the *realistic_v2* telemetry dataset. This dataset keeps the same event structure and schema used in prior experiments (timestamp, device_id, metric, value), while introducing more subtle anomaly pattern and more overlap between normal and abnormal telemetry values.

To ensure comparability with the controlled evaluation pass, the experimental configuration was kept the same.

- Model family: isoforest_v1
- Evaluation size: 9,999 events
- Feature set: identical to the prior pass
- Scoring logic: anomaly = (score < threshold), where the threshold was derived from the 1% baseline quantile during model training (contamination = 0.01)

The final merged evaluation dataset contained 9,999 records due to a single-row formatting issue encountered during CSV merge. This did not materially affect class distribution or reported evaluation metrics.

Because the model, features, and scoring logic were not changed, any performance differences reflect the increased realism of the telemetry, not configuration changes.

5.5.2 Overall Detection Performance

Compared to the controlled injection pass, the realistic_v2 dataset introduced higher variance and overlapping telemetry distributions. This reduced separability between normal and anomalous observations.

The realistic dataset produced:

- ROC AUC: 0.783
- Average Precision (PR AUC): 0.334

At the operational threshold:

- True Positives (TP): 121
- False Positives (FP): 348
- True Negatives (TN): 9,412
- False Negatives (FN): 118

Derived metrics:

- Precision: 0.258
- Recall: 0.506

- F1 Score: 0.342
- False Positive Rate: 0.0357

Figure 5.5 shows the overall ROC curve for the realistic evaluation dataset.

Figure 5.5

Overall ROC Curve (Realistic Evaluation)

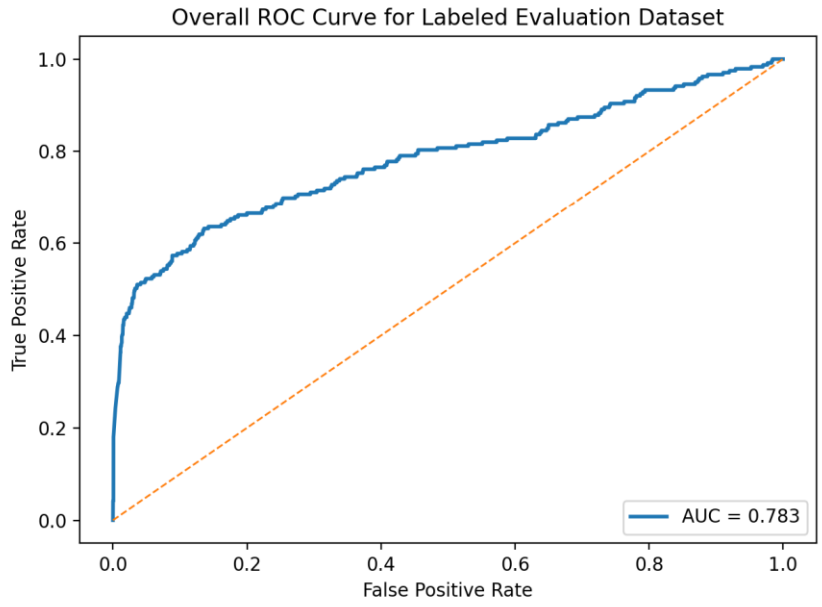
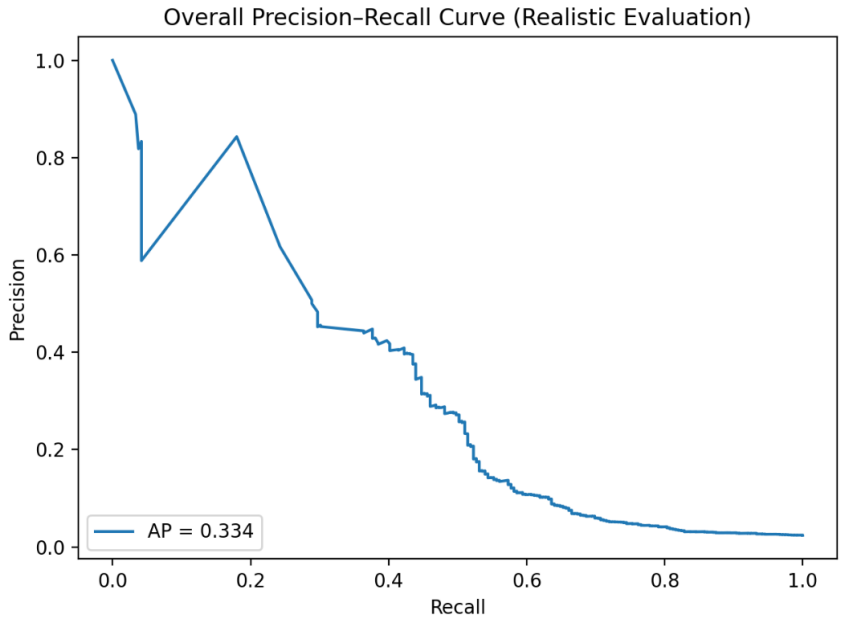


Figure 5.6 shows the corresponding Precision–Recall curve.

Performance drops compared to the synthetic baseline (AUC = 1.000), which is expected given the increased complexity of the realistic dataset. The lower AUC reflects greater overlap between normal and anomalous telemetry values, not instability or drift in the model itself.

Figure 5.6

Overall Precision–Recall Curve (Realistic Evaluation)



5.5.3 Detection Performance by Telemetry Metric

Per-metric ROC AUC values:

- battery_pct: 0.77
- heart_rate_bpm: 0.75
- infusion_rate_ml_hr: 0.80
- temp_f: 0.77

Per-metric Average Precision (AP):

- battery_pct: 0.38
- heart_rate_bpm: 0.16
- infusion_rate_ml_hr: 0.53

- temp_f: 0.17

Infusion rate telemetry was easier for the model to distinguish from normal behavior. In contrast, heart rate and temperature values vary more naturally, which makes subtle anomalies harder to separate from routine fluctuations.

Figure 5.7 presents ROC curves segmented by telemetry metric.

Figure 5.7

ROC Curves by Telemetry Metric

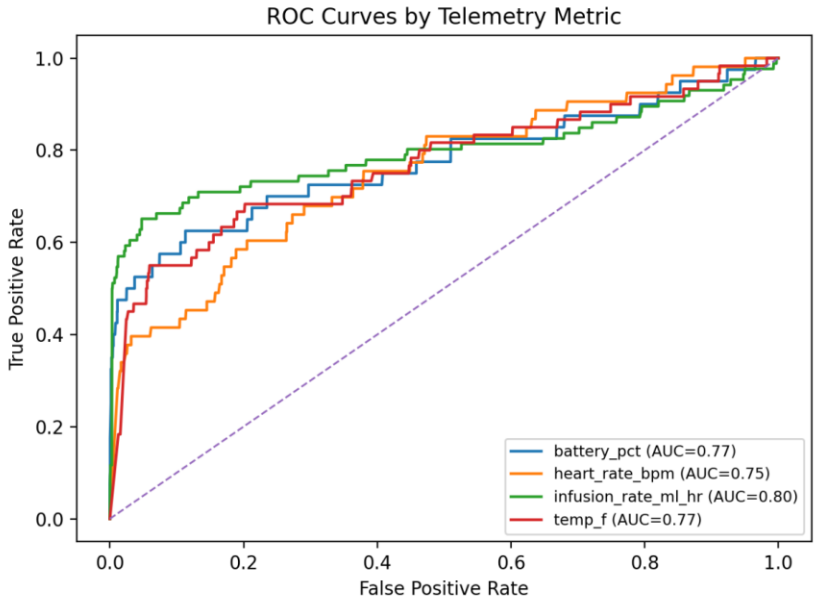
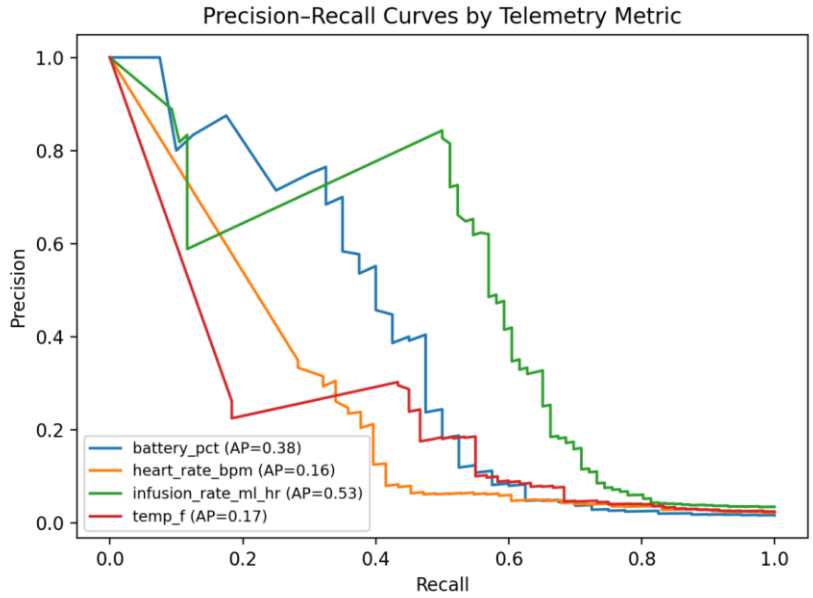


Figure 5.8 presents the corresponding Precision–Recall curves.

Figure 5.8

Precision-Recall Curves by Telemetry Metric (Realistic Evaluation)



5.5.4 Threshold Operating Point Analysis

ROC AUC and Average Precision show how well the model ranks anomalies overall, but in practice a specific score cutoff must be chosen to decide which events generate alerts. To understand the impact of that choice, Performance was evaluated across the full range of possible score cutoffs by sweeping the threshold over all observed anomaly score values. This produces the ROC and Precision–Recall curves. From that sweep, two representative deployment thresholds were selected for analysis. This makes it possible to see how tightening or relaxing the cutoff changes precision, recall, and alert volume.

1.Maximum F1 Overall

- Threshold: -0.04585
- Precision: 0.409

- Recall: 0.423
- F1 Score: 0.416
- False Positive Rate: 0.0150
- TP: 101 | FP: 146 | TN: 9,614 | FN: 138

This threshold identifies a reasonable number of true anomalies while keeping the number of false alerts manageable.

2.Constrained False Positive Rate FPR ≤ 1%

To reflect stricter alert-volume requirements, a second threshold was selected that limits the false positive rate to at most 1%.

- Threshold: -0.05856
- Precision: 0.507
- Recall: 0.289
- F1 Score: 0.368
- False Positive Rate: 0.0069
- TP: 69 | FP: 67 | TN: 9,693 | FN: 170

This operating point substantially reduces false positives, but at the cost of missed anomalies.

Figure 5.9 shows the ROC curve with annotated operating points.

Figure 5.9

ROC Curve with Operating Points (Realistic Evaluation)

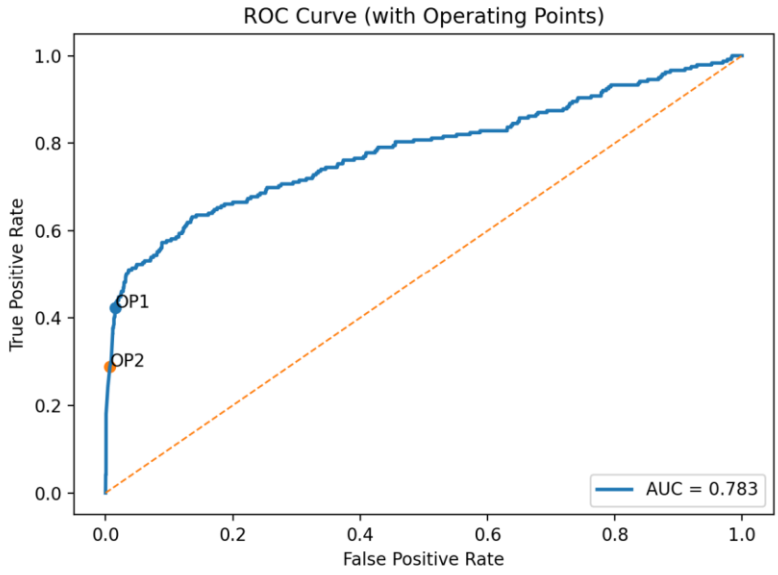
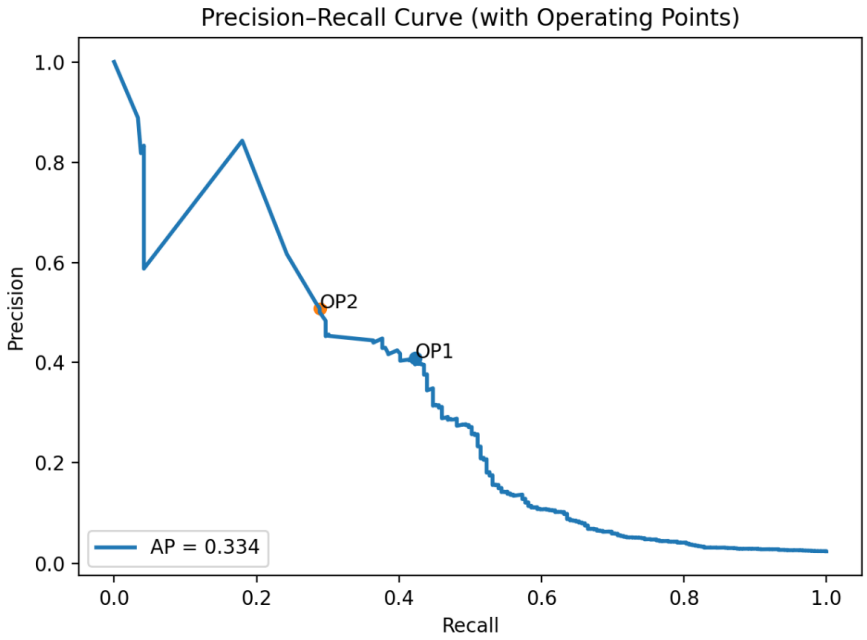


Figure 5.10 shows the Precision–Recall curve with the same operating thresholds.

These results show that while overall ranking performance (AUC = 0.783) remains the same, threshold selection allows the system to be tuned either to minimize alert fatigue or maximize anomaly coverage.

Figure 5.10

Precision–Recall Curve with Operating Points (Realistic Evaluation)



5.6 Operational Evaluation of SOC-Scored Alerts

Strong ranking performance alone does not mean the system is easy to use in a SOC environment. This section examines how raw anomaly scores translate to SOC-facing alerts.

5.6.1 SOC Score Normalization and Monotonicity

Raw Isolation Forest scores were transformed into a normalized SOC score ranging from 0 to 100. Higher SOC scores correspond to stronger anomaly confidence. This transformation keeps the relative ordering of anomalies the same but expresses the scores on a 0–100 scale that is easier for analysts to interpret.

As anomaly scores increase, SOC scores increase as well. Recomputing severity labels independently produced identical results to those generated during runtime scoring, indicating that the severity thresholds were applied correctly.

Figure 5.11 shows console output indicating an independent check of severity labels derived from SOC score thresholds. The mismatch rate of 0.0 confirms that runtime-assigned severity values align exactly with the defined SOC score ranges.

Figure 5.11

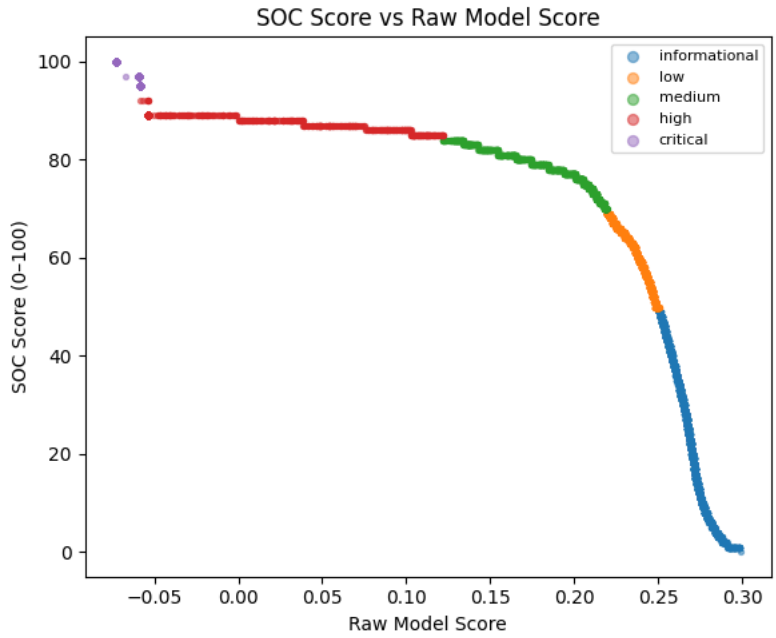
Validation of SOC Score to Severity Mapping

```
(.venv) payton@payton-VirtualBox:~/thesis-lab$ python3 - << 'EOF'
import pandas as pd

df = pd.read_csv("notes/edge_measurements/soc_validation_scored.csv")
df = df.dropna(subset=["soc_score","severity"])

# define expected bins
def expected_sev(s):
    if s >= 95: return "critical"
    if s >= 85: return "high"
    if s >= 70: return "medium"
    if s >= 50: return "low"
    return "informational"

df["expected"] = df["soc_score"].apply(expected_sev)
mismatch = (df["expected"].str.lower() != df["severity"].str.lower()).mean()

print("Rows w/ soc_score+severity:", len(df))
print("Mismatch rate:", mismatch)

print("\nSOC score ranges by severity:")
print(df.groupby(df["severity"].str.lower())["soc_score"].agg(["count","min","median","max"]))
EOF
Rows w/ soc_score+severity: 10000
Mismatch rate: 0.0
```

Figure 5.12 shows the relationship between raw anomaly scores and normalized SOC scores, colored by severity. The mapping is monotonic, and severity thresholds align with increasing anomaly strength.

Figure 5.12

SOC Score vs. Raw Model Score

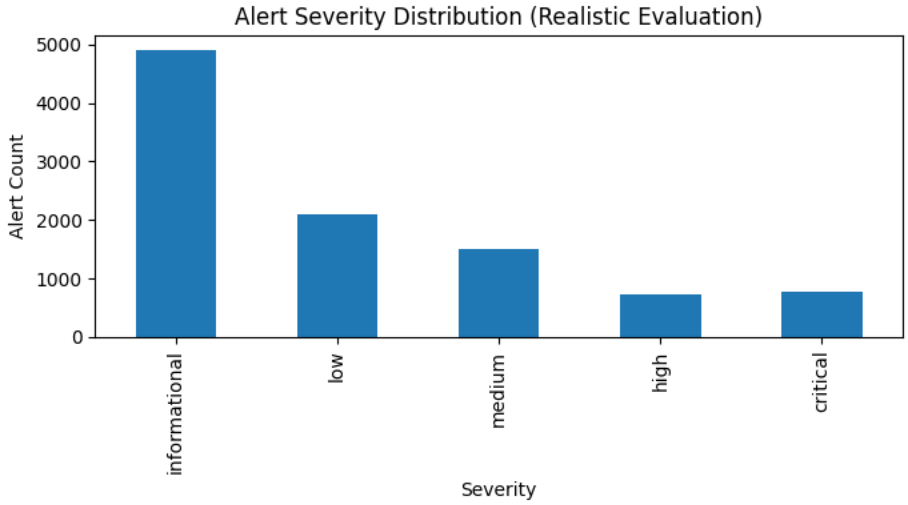


5.6.2 Severity Distribution and Interpretability

Figure 5.13 shows the distribution of alert severities generated during the realistic evaluation pass.

Figure 5.13

Alert Severity Distribution



Most alerts fall within informational and low categories, with progressively fewer medium, high, and critical alerts. This reflects expected behavior and suggests that severity inflation does not occur under the evaluated conditions.

Figure 5.14 shows SOC score distribution by severity category, demonstrating clear separation between severity bins.

Figure 5.14

SOC Score Ranges by Severity

```
SOC score ranges by severity:
               count  min  median  max
severity
critical         763   95    97.0  100
high             737   85    88.0   92
informational   4901    0    25.0   49
low             2100   50    59.0   69
medium          1499   70    77.0   84
```

5.6.3 Temporal and Device-Level Stability

Alert volume remained stable over time without sustained spikes.

Figure 5.15 shows alert volume by severity across the evaluation window.

Figure 5.15

Alert Volume by Severity Over Time

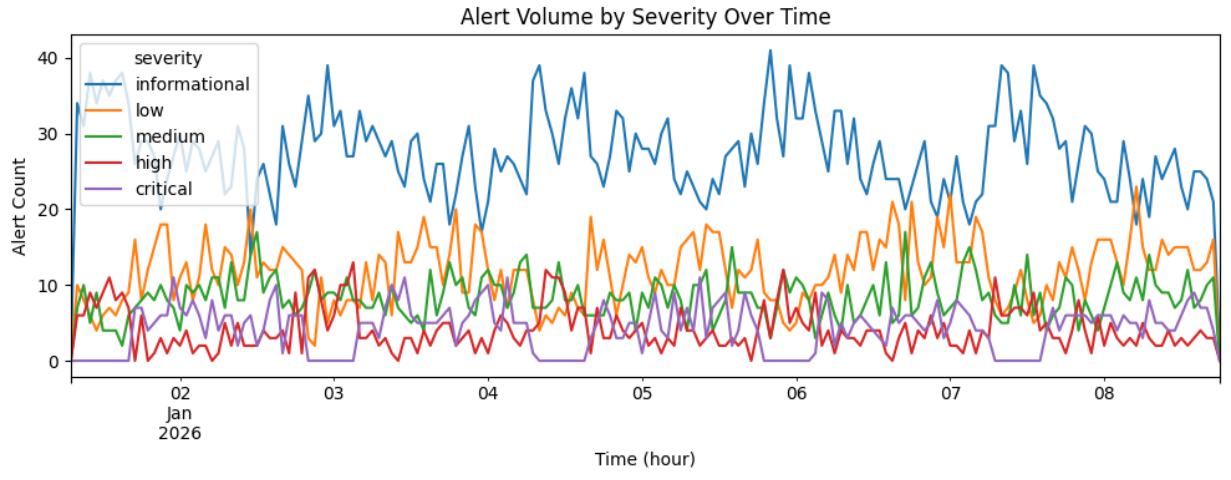


Figure 5.16 shows alert counts by device.

Figure 5.16

Distribution of Anomaly Alerts by Device During Evaluation

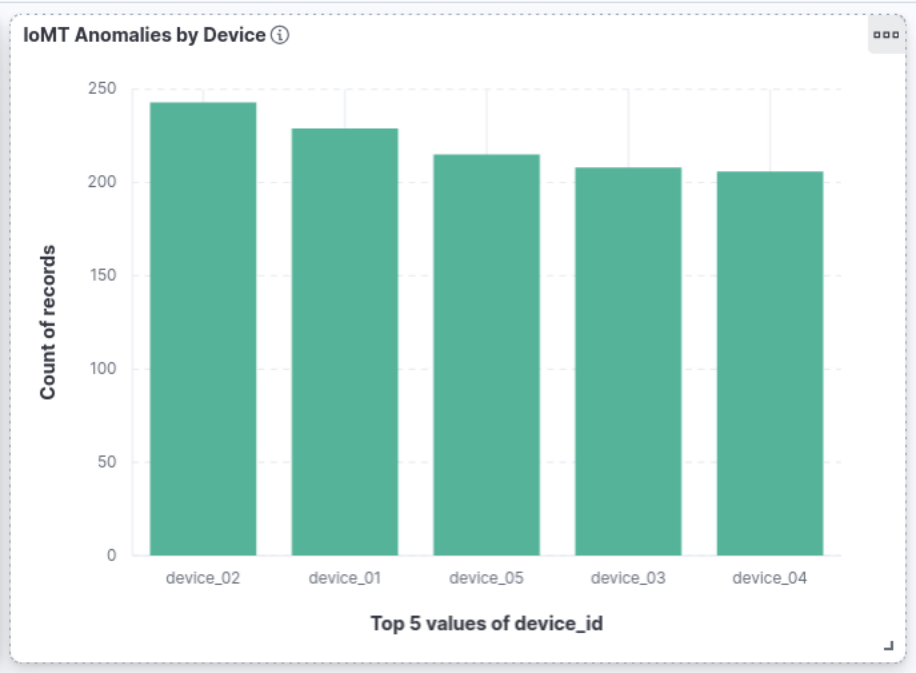
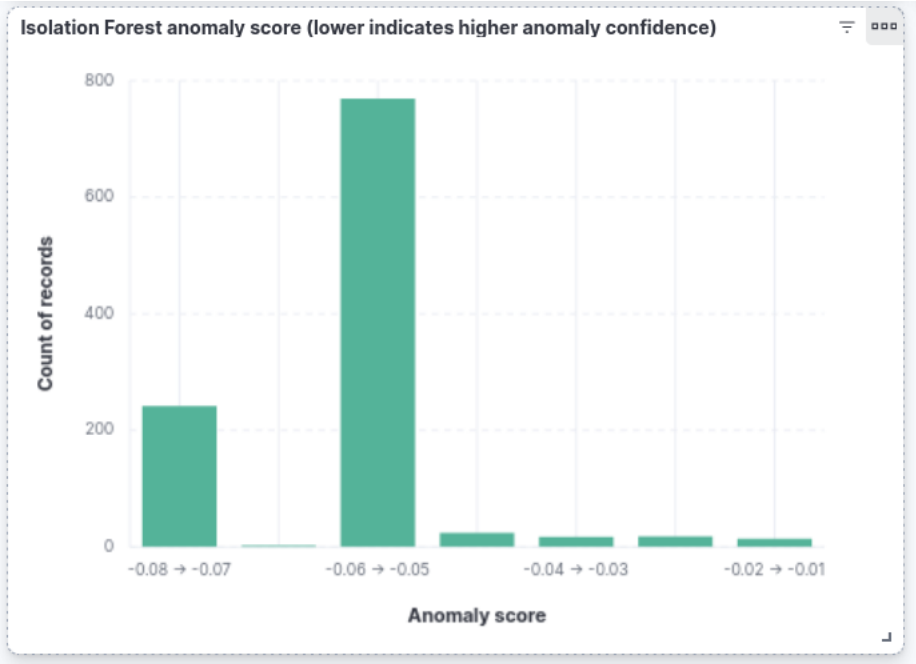


Figure 5.17 shows the distribution of raw anomaly scores.

Figure 5.17

Distribution of Isolation Forest Anomaly Scores During Evaluation



Alert counts were distributed across devices rather than concentrated on a single source. Most anomaly scores cluster near the baseline-derived decision threshold (score < -1.15×10^{-17}), with a smaller number of events showing substantially stronger deviations.

5.6.4 Operational Implications

The SOC scoring and severity assignment components behave consistently and transparently. By separating raw anomaly scores from analyst-facing severity labels, the system makes it easier to integrate into SOC workflows. The results demonstrate that anomaly detection output can be translated into policy-aligned alert severity without retraining or manual adjustment.

5.7 Edge Deployment Performance (Raspberry Pi)

To evaluate feasibility on constrained edge hardware, the trained Isolation Forest model was deployed to a Raspberry Pi–class device and executed in inference-only mode using the same 9,999-event realistic_v2 evaluation dataset described in Section 5.5. This ensures that latency differences reflect hardware constraints instead of dataset variation.

Inference completed successfully for all events without memory exhaustion or process termination.

5.7.1 Runtime and Resource Utilization (VM vs Edge)

In both environments, CPU capacity was the primary limiting factor.

Key observations:

- ~4× increase in mean latency on Raspberry Pi
- Comparable memory footprint (~160 MB)
- Increased page faults on SD-backed storage
- No swap activity

Despite higher latency, most events were processed within tens of milliseconds, supporting feasibility for edge deployment. Most IoMT devices report telemetry at intervals measured in seconds, so per-event inference latencies around 20–30 ms is acceptable for edge scoring.

Detailed /usr/bin/time -v output confirmed CPU saturation with stable memory usage and no swap activity. Full runtime statistics are summarized in Table 5.1.

Table 5.1 compares runtime performance metrics between the virtual machine baseline (Section 5.4) and the Raspberry Pi.

Table 5.1

Runtime Performance Comparison (VM vs Raspberry Pi)

Metric	**Virtual Machine (VM)**	**Raspberry Pi (Edge)**	**Notes**
Events processed	9,999	9,999	Identical workload
Mean latency (ms)	5.67 ms	22.94 ms	~4× slowdown on Pi
Median latency (ms)	4.71 ms	22.19 ms	Stable central tendency
Std dev latency (ms)	23.58 ms	66.86 ms	Higher variance on Pi
Min latency (ms)	2.94 ms	21.86 ms	Baseline processing cost
75th percentile (ms)	5.50 ms	22.27 ms	Tight IQR on both
Max latency (ms)	2340.93 ms	6705.23 ms	Rare tail outliers
CPU utilization	~101%	~98%	CPU-bound workload
Max RSS memory	~165 MB	~160 MB	Comparable footprint
Major page faults	0	347	Expected on SD-backed Pi
Swaps	0	0	No memory exhaustion
Model family	isoforest_v1	isoforest_v1	Identical model

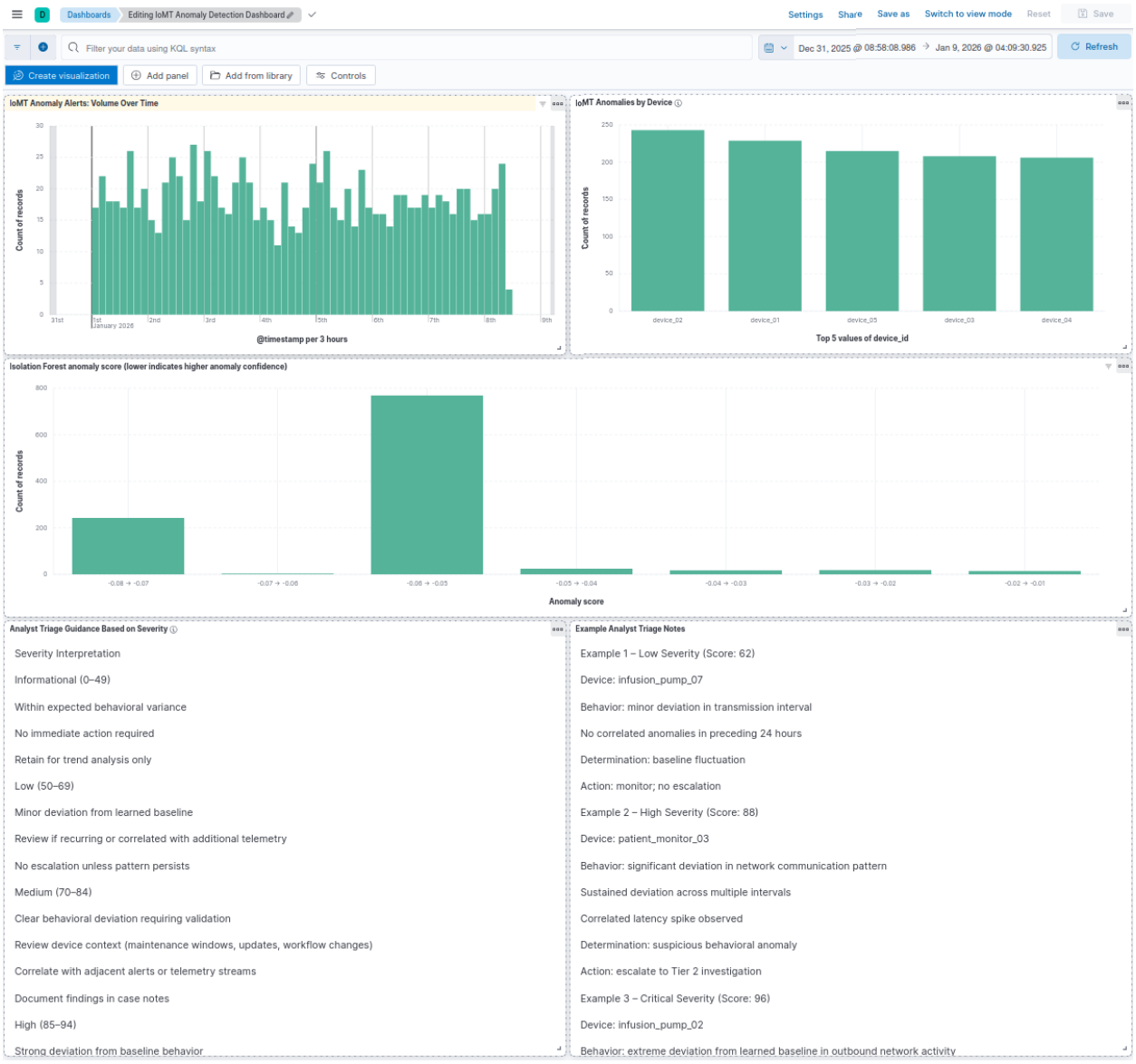
5.8 Integrated SIEM Dashboard View

Figure 5.18 shows the integrated Kibana dashboard for SOC analysts. Anomaly detection outputs for this dashboard include anomaly volume over time, score distribution, anomalies by device, and embedded analyst triage guidance. Instead of presenting raw model output alone, structured

severity definitions and example triage annotations within the same interface demonstrate how unsupervised anomaly scores can be integrated into a practical SOC workflow.

Figure 5.18

Integrated SIEM Dashboard Displaying Anomaly Volume, Score Distribution, Device Distribution, and Analyst Triage Guidance



Chapter 6: Discussion

This chapter interprets the results from Chapter 5, outlines the limitations of the approach, and discusses what the findings mean for IoMT environments and SOC workflows.

6.1 Interpretation of Results

The results presented in Chapter 5 highlight two key findings. The results from Chapter 5 highlight two main points. First, the Isolation Forest model produced anomaly scores with a clear ranking structure under controlled conditions. Second, those scores could be translated into usable alerts that fit into a SIEM-driven workflow. The results vary depending on which evaluation pass is considered, which is an important distinction.

6.1.1 Score Separability Under Controlled Conditions

In the controlled injection-based evaluation, the model achieved near-perfect ROC-AUC and high average precision. While these numbers look impressive, they must be interpreted carefully.

The injected anomalies were deliberately designed to be distinguishable from baseline behavior. In other words, the abnormal telemetry was intentionally separated from normal device behavior. Under those conditions, it is not surprising that the model ranked anomalous events above normal ones with high consistency.

What this pass demonstrates is that the scoring mechanism behaves as expected. When abnormal behavior is clearly present, the scoring pipeline recognizes it and assigns stronger anomaly values accordingly. It confirms that when a clear deviation is present, the model can consistently rank it above baseline behavior.

Importantly, these results should not be interpreted as claims of real-world detection accuracy. It does not prove resilience against adversarial behavior, nor does it guarantee performance in a live hospital network. The controlled evaluation was intended to validate ranking mechanics, not to simulate production-level readiness.

6.1.2 Performance Under More Realistic Conditions

During the realistic evaluation pass, the telemetry distributions began to overlap (meaning normal and abnormal values were not cleanly separated). As expected, performance decreased to more moderate levels (e.g., ROC-AUC dropped to approximately 0.78 and precision and recall both declined). This reduction aligns with prior literature showing that unsupervised models tend to struggle when anomaly distributions overlap with normal behavior.

In real IoMT environments, physiological measurements and device readings naturally fluctuate. A slightly elevated heart rate, a small temperature change, or a temporary battery deviation does not always indicate malicious activity. When abnormal and normal behavior begin to resemble each other, perfect separation becomes impossible without context from other data sources in the network.

The key is that the model still retained a meaningful ranking structure. The threshold sweep showed that the system can be tuned based on how much alert noise a security team is able to handle**.** For example, the system could be configured to maintain a false positive rate below 1% while still detecting a portion of anomalous events. This supports one of the central claims of the thesis, that the system is meant to help sort and prioritize events, not to label every event perfectly.

As a longtime SOC analyst, I am aware that a large percentage of alerts investigated in real environments ultimately turn out to be benign. False positives are not unusual in security operations; they're part of daily SOC work. In that context, the performance observed in the realistic evaluation pass is not weak or discouraging. Instead, it reflects the real conditions under which anomaly detection systems operate. Security teams rarely work with perfectly separable data. They work with noisy detections that require judgment and frequent tuning. The realistic pass ultimately reinforces the idea that anomaly detection should be evaluated as a tool that helps analysts triage activity, not a system that's expected to attain perfect precision.

6.1.3 Prioritization Rather Than Diagnosis

An important design decision within this system is that anomaly detection is used for prioritization, not diagnosis. The model does not determine root cause. It does not assess clinical safety or attribute activity to any particular threat actor or group. Instead, it simply ranks events by relative abnormality to baseline behavior.

The addition of normalized SOC scores and severity labels reinforces this intent. Analysts are not expected to interpret raw model scores. Instead, they receive structured alerts formatted in a way that supports their existing SOC tooling. The anomaly detection component is designed to flag unusual activity early, not to make the final call on whether something is malicious. This distinction is important because the system is meant to uncover unusual behavior early so that human investigation and correlation to other events can follow quickly.

6.2 Limitations

While the results demonstrate feasibility and structured anomaly ranking, several limitations must be considered. This section addresses what the system does not prove.

6.2.1 Synthetic Data and Injection-Based Evaluation

The telemetry in this study was synthetically generated. While this allows for controlled experimentation and reproducibility, synthetic data can never fully capture the degree to which telemetry varies or becomes unpredictable in a real hospital environment.

Injected anomalies were designed to simulate deviations in IoMT device behavior, but they do not perfectly represent adversarial techniques, firmware issues or software faults, or real-world network misconfigurations. For these reasons, the performance metrics reported in Chapter 5 should be interpreted as validation of the pipeline's mechanics, not proof of real attack detection rates.

6.2.2 Absence of Clinical or Adversarial Ground Truth

No live clinical telemetry or real adversarial traffic was used in model evaluation. As a result, this thesis does not claim detection of specific malware, ransomware, or ICS manipulation types. It also does not measure patient impact or have the ability to benchmark against real hospital data.

What this work actually evaluates is how consistently the model ranks injected anomalies above normal behavior within a controlled experiment setup. The work does not claim operationally usable incident response performance and should not be used as such.

6.2.3 Model and Feature Constraints

The model uses a very simple feature representation based on the telemetry metric and its numeric value. Each event is evaluated independently using only that information. In other words, the system looks at one data point at a time rather than tracking long-term patterns across hours or days. It does not attempt to detect multi-step attack sequences or coordinated, persistent

behavior across multiple devices. The model also does not automatically adapt if normal device behavior slowly changes over time. If devices start behaving differently over time, the model would need to be retrained to reflect that new normal. These constraints were meant to keep the system lightweight and focused, but they do limit how broadly the results can be generalized.

6.2.4 Deployment and Resource Constraints

The Raspberry Pi evaluation demonstrates that inference can run within constrained hardware limits. However, latency increased compared to the virtualized baseline, and page fault activity was higher. This is expected behavior when running statistical models on limited CPU and SD-backed storage. Inference completed without memory exhaustion or swapping, which supports feasibility. However, the edge evaluation was still conducted under relatively controlled conditions. Real-world gateways may experience thermal throttling, power fluctuations, competing processes, or network instability. The results therefore demonstrate viability, but not guaranteed production performance.

6.3 Implications for SOC and IoMT Monitoring

Despite these limitations, the results suggest practical implications for IoMT security operations.

First, the research shows that lightweight unsupervised anomaly detection can operate within realistic edge hardware constraints. This matters because many IoMT environments cannot tolerate heavy, resource-intensive monitoring agents.

Second, translating raw anomaly scores into normalized SOC scores and severity levels addresses a real operational gap in the literature. Analysts do not work directly with raw model outputs; they interact with alerts. Building score normalization and severity labels directly into

the scoring pipeline makes anomaly detection usable without requiring analysts to understand the underlying model.

Third, the realistic evaluation pass shows that anomaly detection in IoMT environments should be viewed as a ranking system. In security operations, some amount of false positives is unavoidable. The ability to tune thresholds based on an acceptable false positive rate allows organizations to align detection sensitivity with their staffing capacity and desired alert fatigue tolerance.

Finally, formatting anomaly detections as SIEM events enables correlation. When anomaly alerts are indexed alongside other data sources like authentication logs and network telemetry, their operational value increases because they can be correlated within a broader security context. The system acts as an early signal that feeds into the broader security context, rather than trying to function as a complete detection solution on its own.

6.4 Future Work

6.4.1 Edge Deployment and Resource-Aware Evaluation

Future work should include running the system for longer periods under more realistic load conditions. This would give a clearer picture of how the system performs outside of a clean test environment. Exploring lighter model variants or compression techniques could also reduce memory use on highly constrained devices.

6.4.2 Temporal and Drift-Aware Enhancements

Incorporating time-driven features like short rolling windows or simple trend tracking, could help the system recognize gradual changes or multi-device anomalies. In a real

deployment, periodically retraining the model would likely be necessary so that it reflects current device behavior.

6.4.3 Deeper SIEM Integration

While this thesis demonstrates basic SIEM ingestion and severity mapping, future work could explore more robust analyst workflows, including:

- correlating anomalies across multiple devices instead of evaluating them in isolation
- attaching basic asset context so analysts immediately know what type of device generated an alert
- mapping alerts to common ICS/IoMT attack patterns for better investigative context
- allowing analysts to adjust detection thresholds over time based on alert volume and experience

These improvements would further align anomaly detection with real SOC workflows.

6.5 Summary

This thesis demonstrates that a lightweight, unsupervised anomaly detection model can be integrated into an IoMT-style telemetry pipeline and transformed into SOC-aligned alerts suitable for centralized monitoring.

Performance under realistic overlap conditions declined, as expected. However, the system retained meaningful ranking structure and configurable threshold trade-offs.

The primary contribution is not improved detection accuracy alone. Instead, it shows that anomaly scores can be turned into structured alerts that help security teams spot unusual behavior early, even in environments with limited resources.

The results support the feasibility of edge-based anomaly detection as a practical early-warning mechanism, while being transparent about the limits of using synthetic data and a static model configuration.

Chapter 7: Conclusion

This thesis began by asking whether a lightweight anomaly detection model could run at the edge of an IoMT environment and still produce alerts that are useful in a SOC. Over the course of this research, the emphasis shifted from model performance to whether this system would be practical for security teams.

The Isolation Forest model was able to assign meaningful anomaly scores without relying on labeled attack data. Under controlled conditions, the performance was very strong, which was expected given that the injected anomalies were clearly separated from baseline telemetry. Under more realistic conditions, where normal and abnormal behavior overlapped, performance declined, which was also expected.

What ultimately matters is that the model still distinguished more abnormal activity from normal behavior. Even when the separation was imperfect, the model still produced scores that could be tuned using thresholds, which allowed alert volume to be adjusted in a way that mirrors real SOC operations. In practice, security teams do not see perfect classification in their tooling. They need a way to sort through the noise of false positives and prioritize alerts based on risk. This system demonstrated that this kind of prioritization can work in practice, even under constrained conditions.

Equally important was how the anomaly scores were represented. Raw model outputs are not helpful to analysts. Converting them into normalized scores and severity labels made the output usable inside an existing SIEM workflow. This step was central to making anomaly detection operational instead of just theoretical.

The runtime results reinforced that this approach is feasible. Running inference on edge-class hardware increased latency compared to the virtualized baseline, but the system remained stable. There were no crashes, no memory failures, and no instability. A model that cannot run reliably under real hardware constraints has limited, if any, practical value, regardless of performance metrics.

This work reinforces the fact that anomaly detection is most valuable when it fits the environment it is deployed in. In constrained IoMT settings, adding complexity usually translates directly into higher resource use. A simple, lightweight model that runs reliably and produces interpretable outputs for SOC teams may be more valuable than a more complex system that cannot operate within real hardware limits.

This project was not about achieving perfect metrics. The system was designed to run within edge constraints and to generate alerts in a format that security teams already understand. Getting the technical design to align with real SOC workflows ended up mattering more than any raw performance number.

In summary, this thesis demonstrates that lightweight, unsupervised anomaly detection can be deployed in a resource-constrained environment and integrated into centralized monitoring workflows in a way that is reproducible and operationally practical.

References

Al-Abadi, Abdullah & Mohamed, Mbarka & Fakhfakh, A.. (2023). Robust and Reliable Security Approach for IoMT: Detection of DoS and Delay Attacks through a High- Accuracy Machine Learning Model. International Journal on Recent and Innovation Trends in Computing and Communication. 11. 10.17762/ijritcc.v11i6.7558.

Alexander, O., Belisle, M., & Steele, J. (2020). *MITRE ATT&CK ® for Industrial Control Systems: Design and Philosophy McLean, VA*. https://attack.mitre.org/docs/ATTACK_for_ICS_Philosophy_March_2020.pdf

Alwaisi, Z., Kumar, T., Harjula, E., & Soderi, S. (2024). Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention. *Internet of Things*, *28*, 101398. https://doi.org/10.1016/j.iot.2024.101398

Balhareth, G., & Ilyas, M. (2024). Optimized Intrusion Detection for IoMT Networks with Tree-Based Machine Learning and Filter-Based Feature Selection. *Sensors*, *24*(17), 5712. https://doi.org/10.3390/s24175712

Chandekar, P., Mehta, M., & Chandan, S. (2024). *Enhanced Anomaly Detection in IoMT Networks using Ensemble AI Models on the CICIoMT2024 Dataset*. ArXiv.org. https://arxiv.org/abs/2502.11854

Das, R., & Luo, T. (2023). LightESD: Fully-Automated and Lightweight Anomaly Detection Framework for Edge Computing. *ArXiv (Cornell University)*. https://doi.org/10.1109/edge60047.2023.00032

Ea, P., Vo, Q., Salem, O., & Mehaoua, A. (2024). Unsupervised Anomaly Detection in IoMT Based on Clustering and Online Learning. *2024 IEEE International Conference on E-Health Networking, Application & Services (HealthCom)*, 1–6. https://doi.org/10.1109/healthcom60970.2024.10880810

Food and Drug Administration. (2025, June 27). *Cybersecurity in medical devices: Quality system considerations and content of premarket submissions — Guidance for industry and FDA staff*. https://www.fda.gov/media/119933/download

Hadjer Goumidi, & Pierre, S. (2025). Real-Time Anomaly Detection in IoMT Networks Using Stacking model and a Healthcare-Specific Dataset. *IEEE Access*, *13*, 1–1. https://doi.org/10.1109/access.2025.3563158

Hussein, S. F., Dallalbashi, Z. Ez., & Mohammed, A. B. (2023). Anomaly detection in internet of medical things with artificial intillegence. *Eastern-European Journal of Enterprise Technologies*, *1*(4 (121)), 56–62. https://doi.org/10.15587/1729-4061.2023.274575

Idowu, J. (2025). Deploying Isolation Forest at the Edge: A Synthetic Data-Driven Approach for Real-Time IoT Anomaly Detection. *ResearchGate*, *8*(7). https://doi.org/10.13140/RG.2.2.23518.14408

Koutras, D., Stergiopoulos, G., Dasaklis, T., Kotzanikolaou, P., Glynos, D., & Douligeris, C. (2020). Security in IoMT Communications: A Survey. *Sensors*, *20*(17), 4828. https://doi.org/10.3390/s20174828

Liu, Y., Peng, X., Zhang, X., Chen, J., Xie, Y., & Yang, W. (2021). *Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach*. IEEE Internet of Things Journal, 8(8), 6348–6358. https://doi.org/10.1109/JIOT.2020.3011726

Li, Z., & Zhang, X. (2024). An Enhanced Autoencoder-based Anomaly Detection Model for time series Data from Wearable Medical Devices. *IEEE Journal of Biomedical and Health Informatics*, *29*(10), 1–13. https://doi.org/10.1109/jbhi.2024.3434420

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*. https://doi.org/10.1109/icdm.2008.17

Loiy Alsbatin, Basem Mohamad Alrifai, Firas Zawaideh, & Alawneh, T. A. (2025). Advancing IoMT Security: Machine Learning-Based Detection and Classification of Multi-Protocol

Cyberattacks. *Journal of Wireless Mobile Networks Ubiquitous Computing and Dependable Applications*, *16*(2), 228–247. https://doi.org/10.58346/jowua.2025.i2.015

Messinis, S., Temenos, N., Protonotarios, N. E., Rallis, I., Kalogeras, D., & Doulamis, N. (2024). Enhancing Internet of Medical Things security with artificial intelligence: A comprehensive review. *Computers in Biology and Medicine*, *170*, 108036. https://doi.org/10.1016/j.compbiomed.2024.108036

Ngo, M. V., Chaouchi, H., Luo, T., & Tony. (2020). *Adaptive Anomaly Detection for IoT Data in Hierarchical Edge Computing*. ArXiv.org. https://arxiv.org/abs/2001.03314

Ni, C., Wu, J., & Wang, H. (2025). Energy-Aware Edge Computing Optimization for Real-Time Anomaly Detection in IoT Networks. *Applied and Computational Engineering*, *139*(1), 42–53. https://doi.org/10.54254/2755-2721/2025.22280

NIST. (2024). The NIST cybersecurity framework (CSF) 2.0. *The NIST Cybersecurity Framework (CSF) 2.0*, *2.0*(29). https://doi.org/10.6028/nist.cswp.29

Nozza, V. (2023). *The Role of Real-Time Threat Analytics, Mitigating Data Integrity Risks; Within the IoMT Healthcare Environment* (Order No. 30313304). Available from ProQuest Dissertations & Theses Global. (2792157325). https://www.proquest.com/dissertations-theses/role-real-time-threat-analytics-mitigating-data/docview/2792157325/se-2

Postmarket Management of Cybersecurity in Medical Devices Guidance for Industry and Food and Drug Administration Staff. (2016). https://www.fda.gov/files/medical%20devices/published/Postmarket-Management-of-Cybersecurity-in-Medical-Devices---Guidance-for-Industry-and-Food-and-Drug-Administration-Staff.pdf

Sadhu, P. K., Yanambaka, V. P., Abdelgawad, A., & Yelamarthi, K. (2022). Prospect of Internet of Medical Things: A Review on Security Requirements and Solutions. *Sensors*, *22*(15), 5517. https://doi.org/10.3390/s22155517

Sindhuja R, Kapse Arvind S, & Kapse Avinash S. (2023). A Survey of Internet of Medical Things (IoMT) Applications, Architectures and Challenges in Smart Healthcare Systems. *ITM Web of Conferences*, *56*(56), 05013–05013. https://doi.org/10.1051/itmconf/20235605013

Stouffer, K. (2023). Guide to Operational Technology (OT) Security. *Guide to Operational Technology (OT) Security*. https://doi.org/10.6028/nist.sp.800-82r3

Swati Lipsa, Dash, R. K., & Nikola Ivković. (2025). An interpretable dimensional reduction technique with an explainable model for detecting attacks in Internet of Medical Things devices. *Scientific Reports*, *15*(1). https://doi.org/10.1038/s41598-025-93404-8

Tabassum, M., Mahmood, S., Bukhari, A., Alshemaimri, B., Daud, A., & Khalique, F. (2024). Anomaly-based threat detection in smart health using machine learning. *BMC Medical Informatics and Decision Making*, *24*(1), 347. https://doi.org/10.1186/s12911-024-02760-4

Zachos, G., Mantas, G., Kyriakos Porfyrakis, C.S, M., & Rodriguez, J. (2025). Anomaly-Based Intrusion Detection for IoMT Networks: Design, Implementation, Dataset Generation and ML Algorithms Evaluation. *IEEE Access*, *13*, 1–1. https://doi.org/10.1109/access.2025.3547572