Intelligent Anomaly Detection for Context-Oriented Data Brokerage Systems

Rawaa Al-Wani[a] and Mays Al-Naday[b]
School of Computer Science and Electronic Engineering, The University of Essex,Colchester, U.K.

Keywords: Internet of Things, Publish/Subscribe, FIWARE, Context-Awareness, Anomaly Detection, Machine Learning.

Abstract: Applications of the Internet of Things (IoT) face challenges related to interoperability and heterogeneity due to variations in data representation formats and the absence of connectivity standards across wireless networks. This has led to the emergence of context-oriented data brokering frameworks, with FIWARE being the most widely adopted. However, such frameworks are not able to differentiate malicious from benign data. Consequently, challenges related to data quality persist, and brokering overlays are susceptible to exploitation for the distribution of malicious data assets. We propose a novel Artificial Intelligence (AI) anomaly detection service that communicates with the FIWARE broker via the Fast Application Programming Interface (FastAPI). The system also uses the Publish/Subscribe (Pub/Sub) model of FIWARE to allow networking between brokers to validate data assets before disseminating them. This is to analyze the overhead that anomaly detection introduces as a cost of the solution. The results show that the solution can detect around 95% malicious data, with an approximate overhead of 12% increase in response time.

1 INTRODUCTION

The breakthrough that came with the Internet of Things (IoT) has changed almost all aspects of life, heralding a new era in which everyday objects are interconnected to the Internet. The IoT applications produce heterogeneous data at the device and network levels, and the spontaneous occurrence of numerous events, will pose a significant barrier for the development of diverse applications and services (Razzaque et al., 2015; Alberti et al., 2019). Consequently, to coherently model IoT objects and data from multiple sources with different formats, the Semantic Web of Things (SWT) based on the standards and technology of the World Wide Web Consortium (W3C) is used.

The W3C's Web of Things (WoT) architecture recommendations delineate the prerequisites for establishing a proxy that interlinks brokers with the IoT network and cloud computing systems. Cloud-based Publish/Subscribe (Pub/Sub) systems provide reliable solutions for the deployment of IoT data in the cloud and facilitate communication with applications or users subscribing to IoT entities (Amara et al., 2022). FIWARE is the most prominent cloud-based Pub/Sub platform. FIWARE facilitates data brokering through *Context Brokers* that implement a Pub/Sub model over entities using the Next Generation Service Interface (NGSI) protocol. FIWARE defines crucial components called Generic Enablers (GE). Orion GE acts as the context broker of FIWARE. Orion broker offers an Application Programming Interface (API) that implements the NGSI Context API (Bellini et al., 2023).

A *context* is defined as the information that characterizes the IoT data, and context-awareness involves using this information to comprehend the acquired facts (Barriga et al., 2022). However, FIWARE security capabilities focus on authentication and access control services using Keyrock GE and Wilma GE, without anomaly detection support for data assets (Munoz-Arcentales et al., 2021). As a result, protecting the information sent between broker systems is crucial. Machine learning (ML) has been used for anomaly detection in telecommunication networks, but it has not been applied yet in FIWARE-like brokerage systems to provide such detection capabilities.

This work proposes a novel Pub/Sub-based communication framework across FIWARE brokers, for anomaly detection in data assets. The framework enables the integration of ML-based anomaly detector

[a] https://orcid.org/0000-0001-6420-0296
[b] https://orcid.org/0000-0002-2439-5620

Al-Wani, R. and Al-Naday, M.
Intelligent Anomaly Detection for Context-Oriented Data Brokerage Systems.
DOI: 10.5220/0013478800003944
In *Proceedings of the 10th International Conference on Internet of Things, Big Data and Security (IoTBDS 2025)*, pages 442-449
ISBN: 978-989-758-750-4; ISSN: 2184-4976

as a "pluggable" service, allowing flexible incorporation of different ML models. ML service plugging includes: entity pre-processing to extract its data from the respective NGSI message and serve to the ML model; and, post-processing to package the prediction result as an NGSI feature update of the same entity, maintained by the verifying broker.

The framework is implemented and evaluated experimentally using example dataset: the Canadian Institute of Cybersecurity (CIC) IoT dataset (Neto et al., 2023), which covers extensive attacks in IoT environments. Evaluation results show benefits of anomaly detection in data brokerage systems, compared to the overhead introduced by the detection framework.

We structure the rest of the paper as follows: Section 2 reviews the state-of-the-art related work. Section 3 describes the proposed Pub/Sub framework for anomaly detection. Section 4 evaluates the performance of the proposed solution, while Section 5 draws our conclusions.

2 RELATED WORK

Data interoperability and anomaly detection have been active research topics within the IoT domain, with a wide range of solutions being developed by the community (Martins et al., 2022; Zyrianoff et al., 2021; Baee et al., 2024). The work of (Anwar and Saravanan, 2022) applies apache spark for big data processing to classify network traffic and detect intrusions produced by IoT devices. To evaluate the effectiveness of intrusion detection, this study compares the performance of ML versus deep learning models. Both types of models are trained and evaluated in the distributed computing environment provided by Spark, ensuring scalability for handling the large volume of data in the BoT-IoT dataset. However, this work does not support IDS services in context-aware IoT networks.

The key features for the design and performance metrics of several open-source systems are explained in (Lazidis et al., 2022). These systems include RabbitMQ and Apache Kafka. A significant contribution of this work is the comprehensive evaluation of seven open-source systems. However, this work provides precise details on several Pub/Sub systems, but does not offer substantial guidance on the implementation. The work of (Ataei et al., 2023) introduces a comprehensive architectural framework based on the Pub/Sub technique, designed for real-time data processing in the broad field of Massive IoT (MIoT) utilizing the powerful features of Apache Kafka for data stream processing. However, this work does not support context awareness by itself. It needs to be in conjunction with other frameworks and technologies to create a context-aware system. The work of (Shukla et al., 2024) introduces a new approach for detecting distributed denial-of-service (DDoS) attacks on IoT data. It operates within a Kafka framework. Kafka is utilized to implement a portable, scalable, and distributed detection system. However, Apache Kafka does not evaluate or infer context; it merely facilitates the transfer and persistence of data, rather than adapting to changing contexts. Anomaly detection of IoT applications is increasingly using machine learning (ML). An Intrusion Detection System (IDS) based on ML classifier algorithm is used in the work of (Sirisha et al., 2021) to distinguish between normal and malicious traffic and lowers the risk of malicious activity. The ML algorithms used are trained on the UNSW-NB15 dataset. However, this work haven't addressed the interoperability challenge of IoT (not supporting context-aware platforms).

The work of (Martín et al., 2023) evaluates the compatibility of AI services with the FIWARE platform. The integration of cognitive AI services with IoT platforms is enabled by an abstraction layer that incorporates cognitive components, enhancing interoperability across diverse IoT domains. This work is particularly relevant to the research presented in this work. However, it did not provide an IDS services-based context-aware platform like FIWARE.

3 THE PROPOSED ANOMALY DETECTION SYSTEM

The proposed framework (implemented as a system) comprises: an off-line-trained machine learning model that is served as a pluggable anomaly detector service, verification brokers, and a Pub/Sub communication protocol to facilitate near-real-time anomaly detection.

3.1 Functional Components

The proposed framework illustrated in Figure 1 consists of: a collection of data verification brokers, each representing a distinct environment; (edge) service broker that enables the validation of data assets using an anomaly detector; and a modular ML-based anomaly detection microservice that predicts the nature of the data assets. The proposed brokers differ from the baseline broker by incorporating verification capabilities that regulate the management of data assets. These brokers communicate via a Pub/Sub paradigm facilitated by the NGSI-LD protocol. Fur-

thermore, the Fast Application Programming Interface (FastAPI) web framework is utilized to deploy the ML model as a pluggable microservice, facilitating flexible and modular integration of several ML models according to the scenario. Moreover, FastAPI is selected for its speed, high performance, and robustness, as well as its inherent capabilities for data validation, JSON serialization, and OpenAPI integration.

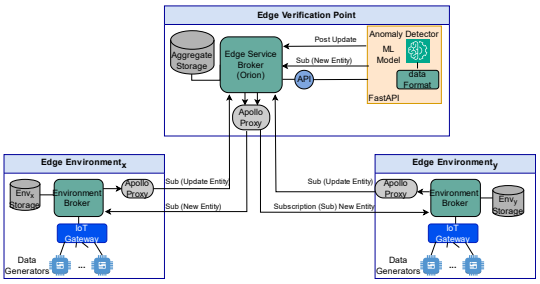

Figure 1: Agents-based System.

3.1.1 Data-Verifying Brokers

Each environment is represented by at least one broker, making it easier to verify data assets prior to publication. Data created in the environment is transferred to the broker via the appropriate gateway, where it is represented as a context entity and momentarily added as a new entity to the environment database (MongoDB). Before confirming the entity admission to the brokerage system, the broker publishes the the new entity to a verification service broker and subscribes for the response channel with the verification broker. The response channel is identified by a commonly agreed subscription identifier between the environment and the service brokers. The response itself is an entity update that confirms whether a data entity is benign or malicious, and what type of malicious attack it is likely to be caused by. The entity is identified by its `Entity Id`. The service broker directly informs the anomaly detector about the new entity, as indicated by the subscription illustrated in Figure 5 for the CIC entity. Upon notification from the anomaly detector that the data is benign, the service broker verifies the entity's admission and processes the data in accordance with the management policy specific to benign data within the environment. Otherwise, if the entity is malicious, the broker may act on it with an alternative management policy for malicious data. For example, to delete the entity from the environment database and raise an alarm to relevant systems; or redirect the data to a honeypot. It should be noted that the interaction between the environmental verification broker and the verification counterpart uses two subscriptions (asynchronous Pub/Sub paradigm), as detailed in Section 3.2. We implement the verifying broker as a FIWARE Orion supported by Apollo proxy, which handles data extraction and maintains context subscriptions by turning broker notifications into context entities.

3.1.2 Verification Service Broker

This broker interacts with an anomaly detector to analyze data entities for legitimacy assessment. The broker initially provides its services to the environment by subscribing to the new entities obtained from the environmental brokers. Specifically, the service broker subscribes once to each of the environment counterparts for a particular type of entity, represented by a common `type` attribute. This implies treating `type` as a 'context group', and allows for verifying any number of entities of a particular type for the lifetime of the subscription. This work assumes that each environment broker represents a distinct type of entity. Consequently, the service broker establishes a number of subscriptions that does not exceed the total number of environment brokers within the system. The anomaly detector and the service broker subscribe to new entities. When the service broker receives a publication of a new entity from an environment broker, it extracts and passes the entity data to the anomaly detector. The latter analyzes the data and responds to the service broker with a prediction of the entity's nature. The prediction result is sent as an HTTP POST message. Since the anomaly detector is accessed directly via an API, the service broker is not required to explicitly subscribe to a response channel. The subscription is presumed to be implied. Upon reception of the prediction result from the anomaly detector, the service broker updates the entity in its own database and publishes the result to the respective environment broker over the response channel. To this end, the service broker is assumed to operate over less constrained infrastructure than environment brokers, and can be managed by the same or different stakeholders as the environment brokers. The broker is too integrated as an extended Orion, able to communicate with the anomaly detector.

3.1.3 Pluggable Anomaly Detector (ML-Based)

The anomaly detector is represented as a self-contained service that is pluggable to the service broker using the NGSI-LD interface on one direction and RESTful API on the other. Firstly, the ML model is trained offline through a separate pipeline as shown in Figure 2. The pipeline includes applying data preprocessing, feature reduction component that decreases

the data volume to the minimum required to provide scalability-/efficiency-by-design. For this, we apply the K-best feature selection technique. The (K-best function) selects the features according to their relevance to the output variable using one of these functions (chi-squared, ANOVA F-test, and mutual information). Chi-square test has been chosen to select the features with the highest scores for the final feature subset. Secondly, the produced (pluggable) ML model is deployed as a microservice enabled by FastAPI, which subscribes to new entities from the service broker and provides, in return, an API endpoint to process incoming publications and post prediction notifications.

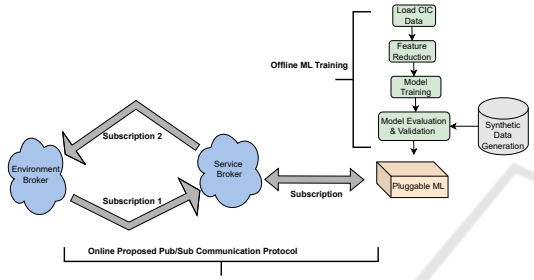

Figure 2: Proposed Pub/Sub Communication Protocol and Offline ML Training.

Secondly, the anomaly detector deploys the (pluggable) ML model as a microservice using FastAPI and subscribes to the new entities from the service broker. Thirdly, in FASTAPI the received entity published by the service broker is processed first to serialize the data, remove message header(s) as well as feature names from the entity leaving only the values of the features to be used in the loaded pluggable ML. The latter analyzes the data and returns the prediction of whether the entity is benign or malicious and what type of malicious. The API endpoint pushes the prediction result to the service broker as an HTTP POST to update the existing entity using its ID. We used CIC IoT 2023 dataset for training and testing the ML model offline. CIC IoT dataset involves seven groups of attacks, namely DDoS, DoS, Recon, Web-based, brute force, spoofing, and Mirai.

3.2 Publish/Subscribe Message Exchange

This work introduces an innovative online Pub/Sub communication protocol to enhance interaction between the environment and the service brokers, as well as between the service broker and the anomaly detector. Using three subscriptions that circulate the IoT entity through the proposed system beginning from the environment broker and ending by returning the prediction result back to the same broker as shown in Figure 2. First, the environment broker creates a context entity of the data received from IoT devices, and store the pending entity in the environment database. The service broker subscribes to the new entities under a specific type, `subscription 1 (new entity)`. The subscription specifies: a `name`, an identifier (`id`) and an entity `type`. This constitutes a form of service channel between the two brokers. The subscription further specifies the entity attributes to be included in the notification along with the destination endpoint (for sending the publication). An example of `subscription 1` is shown in Figure 4 based on CIC dataset. Similarly, the anomaly detector subscribes to the service broker (see Figure 5), using the `name`, `id` and `endpoint` of the service channel between the service broker and the anomaly detector. The cascade subscription enables asynchronous forwarding of the new entity from the environment broker to the anomaly detector.

When an environment broker receives `subscription 1`, it responds back with notification of any pending entity - i.e. for which there is no prediction result - to the service broker. Following that publication, the environment broker subscribes to the prediction results, expected as an updates of the existing (pending) entities. Meanwhile, the service broker stores new entities in its aggregate database and publishes them to FastAPI anomaly detector. The latter processes the received entity to classify it and post the predicted result back to the service broker, as a new feature of this entity. Subscription to the prediction result - by the service broker - is implicit, as FastAPI sends the result back as an HTTP POST message. When the service broker receives the prediction result, it will notify the environment broker due to `subscription 2 (entity update)` - illustrated in Figure 6. This subscription does not specify a specific entity; instead, each notification response is anticipated to include an entity ID that corresponds to an existing entity. The subscriptions and alerts for each broker are managed by the corresponding Apollo proxy linked to the broker. The workflow of the Pub/Sub model is shown in Figure 3.

4 EVALUATION

This section evaluates the performance of the proposed intelligent anomaly detection solution experimentally, using our FIWARE-based implementation. The overhead of achieving anomaly detection is quantified as a solution cost, relative to its benefit in mit-

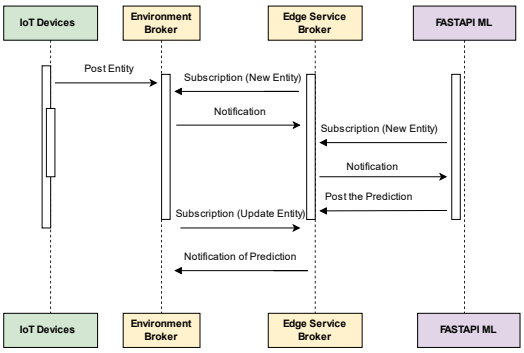

Figure 3: Pub/Sub model Workflow.

Name: New_Entity_Ebroker_to_Sbroker
Id: Urn: NGSI-LD: Subscription: Ebroker_to_Sbroker
Type: Subscription
Entities:
Type: NetFlow_CIC
Watched Attributes: [Header_Length, S_Rate, ...] #15 features
Notification:
Attributes: [Header_Length, S_Rate, ...] #15 Features
Endpoint:
URI: http://EdgeServiceBroker.notification-proxy.docker:8080/notification
Accept: Application/Json

Figure 4: Subscription 1 from Edge Service Broker to Environment Agent.

Name: New_Entity_Sbroker_to_ML
Id: Urn: NGSI-LD: Subscription: Sbroker_to_ML
Type: Subscription
Entities:
Type: NetFlow_CIC
Watched Attributes: [Header_Length, S_Rate, ...] #15 features
Notification:
Attributes: [Header_Length, S_Rate, ...] #15 Features
Endpoint:
URI: http://Backend.docker:8000/notification
Accept: Application/Json

Figure 5: Subscription from Fast API to Edge Service Broker.

igating the spread of malicious data. We illustrate our argument by comparing the system performance with and without the proposed solution. We refer to the FIWARE system without our solution as *baseline*, whereas a system that integrates our solution is identified as *proposed*. Experiments are conducted in a containerized virtual environment utilizing the generated load from the custom-built entity generator and/or

Name: Entity_Verification_Result_Sbroker_to_Ebroker
Id: Urn: NGSI-LD: Subscription: Sbroker_to_Ebroker
Type: Subscription
Entities:
Type: NetFlow_CIC
Watched Attributes: Prediction_Result
Notification:
Attributes: Prediction_Result
Endpoint:
URI: http://EnvironmentAgent.notification-proxy.docker:8081/notification
Accept: Application/Json

Figure 6: Subscription 2 from Environment Agent to Edge Service Broker.

the Locust load tester[1]. The entity generator enables adaptive creation of entities, according to the response rate of the service broker; while Locust was used to scale the load introduced in the system. Moreover, the entity generator resembles the behavior of realistic data generators (IoT devices). The rate of entity generation and the total number of entities have been configured differently to assess each of the KPIs, and it has been described in their respective sections below. A set of Key Performance Indicators (KPIs) have been used: ***Response Time***; ***Response Throughput***; and ***ML model performance***. The physical edge node runs on Linux Ubuntu 24.04, using intel(R), Xeon(R) core CPU of 1.60GHz - 2.11 GHz, and 8GB RAM.

4.1 Response Time

Response Time is the elapsed time between sending a notification of a new entity from an environment broker to its service counterpart and receiving a prediction result back.

4.1.1 Per Entity

The empirical cumulative distribution function (ECDF) is used to present the response time per entity in the Baseline versus Proposed system. The response time of 30 entities was collected independently to be analyzed as shown in Figure 7. Generally, the distribution pattern in both systems is analogous with $\approx 12.5\%$ proposed system overhead. Additionally, ($\approx 90\%$) of baseline responses are received with ≈ 70 msec compared to $\approx 95-97$ msec for the proposed system. The maximum baseline response time was recorded at ≈ 97 msec, as opposite to the maximum proposed system response time at ≈ 120 msec. The proposed system overhead is

[1]https://locust.io/

mainly driven by the processing delay of the anomaly detector and the communication time between the service broker and the detector.

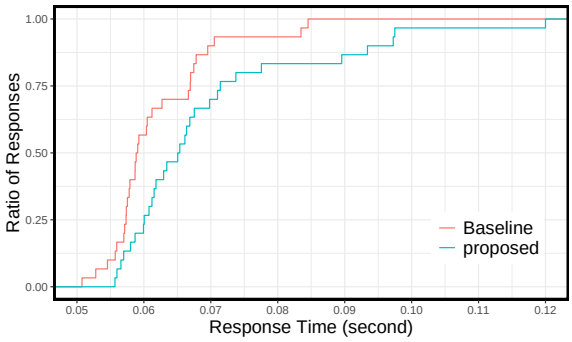

Figure 7: ECDF comparison of Response Time.

4.1.2 For Multiple Entities

We evaluate the overall response time for multiple entities. We measure this parameter by calculating the total time required to publish multiple entities and receive their responses. To evaluate this KPI, we perform 10 experiments for both the baseline and proposed systems. Each experiment involves generating several entities, ranging from 1 to 10. The custom entity generator has been used here to control the number of entities. The results are presented in Figure 8 as scatterplots. The response time of the two systems shows an upward trend along with the increase in the number of successful responses (entities). It is worth mentioning that the elapsed time in Figure 8 is in the order of 500-750 msec as compared to 70-100 msec in Figure 7; this difference is because the latter presented the response time of getting one entity while Figure 8 presented the total elapsed time for getting responses of multiple entities, ranging from one entity to ten. The total elapsed time of the baseline system is between 50 msec for 1-entity experiments and 450 msec for 10-entity experiments, with a variation of $\approx 70-100$ msec. Whereas, the total proposed system elapsed time is recorded at ≈ 750 msec when the number of entities is 10, with ≈ 200 msec variation. The maximum difference in the average response time was ≈ 300 msec when the number of entities was 10. This shows that under light load conditions, both systems have similar response times, with overhead from the proposed becoming observable as load increases. This is due to the additive effect of the processing delay taken by the anomaly detector and the communication latency between the service broker and the anomaly detector. Moreover, the response time increases at a slower rate in the baseline system, with a slope percentage of $\approx 4.4\%$. The growth rate is faster in the proposed system, with a slope percentage of $\approx 7.7\%$. This shows that under light load conditions, both systems perform relatively similar, with overhead of the proposed becoming observable as the load increases. This is due to the additive effect of the processing delay taken by the anomaly detector and the communication latency between the service broker and the anomaly detector.

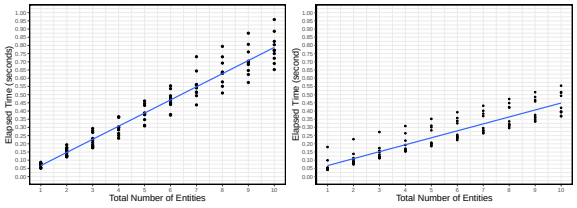

Figure 8: (a) Response Time of Proposed System (b) Response Time of Baseline System.

4.2 Response Time Percentile

Here, the overall response time is assessed when the entities are published concurrently. We used the locust load tester to scale the number of users within a designated time period and obtain the response time percentile for completed entities. Locust configuration involved specifying the number of users, the ramp-up users, and the run time. We extract the evaluation report as a Comma Separated Values (CSV) file and use it to produce the results. Each active user uses the `client.post` method to submit one entity to the environment broker. The entity could be benign or malicious, classified as an attack in the experimental dataset.

The proposed and baseline response time percentiles are depicted in Figure 9. We tested three scales of active users (50, 500, 2000). In the baseline system, 50% of the entities received responses within ≈ 10 msec when the number of users is 50. Whereas, in the proposed system, it reaches ≈ 90 msec for the same ratio of completed entities. The proposed system response time was ≈ 250 msec or below, compared to ≈ 130 msec in the baseline, when the ratio of completed entities reached 99%. Overall, for all three scales of active users and at the $90\% - 99\%$ of completed entities, the response time almost doubles in the proposed system compared to the baseline. This

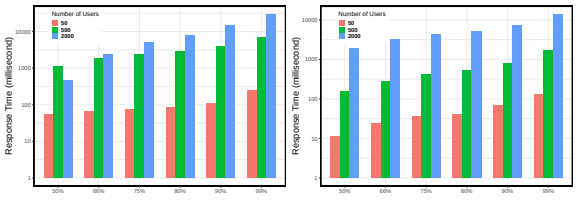

Figure 9: (a) Proposed System Response Time Percentile (b) Baseline System Response Time Percentile.

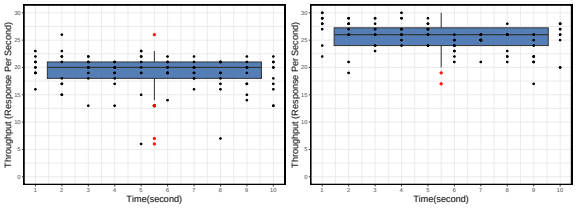

Figure 10: (a) Proposed system RPS over 10 seconds for 10 experiments (b) Baseline system RPS over 10 seconds for 10 experiments.

is similar to the results shown earlier in Figure 8. The anomaly detector, with its single deployment instance in the testbed, drives this processing and queuing delay.

4.3 Throughput

Throughput is the number of responses within each second is recorded as the RPS. Figure 10 shows the throughput received by an environment broker, measured during 10 experiments. Each experiment runs for a total duration of 10 seconds. Each second involves sequential generation and posting of a new entity after successfully receiving a response of the previous entity. Figure 10-(a) shows that the average RPS achieved by the proposed system is 20 RPS and the maximum is 26. This is $\approx 20\%$ lower than the RPS achieved by the baseline, shown in Figure 10-(b). The latter exhibits an average RPS of 26 with a maximum of 30. In general, the throughput of the proposed system is lower than the throughput of the baseline counterpart as a result of the added overhead of the anomaly detector component, along with communication overhead on the forwarding channel from the service broker to the detector.

4.4 Offline ML Performance Evaluation

Two ML training and testing pipelines have been assessed: one without a dimensionality reduction function, hence including the full feature set, and one with the reduction function to minimize the processing requirement of the model. The goal is to quantify the performance loss associated with the reduction, rather than the training cost in CPU resources and training time. We have trained and tested each pipeline offline using four distinct classification algorithms: K Nearest Neighbors (KNN), Decision Trees (DT), Gradient Boosting (GB), and Random Forest (RF). An example dataset, CIC IoT 2023, has been used to train and validate each model. We measured four ML KPIs: *Accuracy*, which shows the percentage of correct predictions; *Precision* and *Recall*, which show the percentage of fewer false alarms; and *F1-score*, which shows

Table 1: Comparison of the full-feature and feature-reduction pipelines for the CIC dataset.

ML Algorithm	Number of features	Accuracy	Precision	Recall	F1-Score	Train Time (s)	CPU Usage %
KNN	15 Features	0.9705	0.6611	0.9705	0.9691	4.7541	14.5
DT		0.9925	0.8557	0.8351	0.8436	6.5025	11.9
GB		0.9786	0.6899	0.9786	0.9798	1157.1868	16.8
RF		0.9926	0.9925	0.9926	0.9921	253.0454	11.7
KNN	41 Features	0.9705	0.6611	0.9705	0.9691	4.6831	14.0
DT		0.9925	0.8538	0.8378	0.8445	7.5184	13.4
GB		0.9786	0.6899	0.9786	0.9798	1329.2544	17.1
RF		0.9927	0.9926	0.9927	0.9922	368.8559	15.8

how accurate the models really are. Table 1 presents the performance and cost results of the two pipelines over the CIC validation dataset. The first pipeline includes all 41 features, while the second includes only the 15 most important ones. Cost is measured by the time it takes to train a model and the percentage of CPU used in training. First, the results show that the average accuracy of all models is $\approx 98\%$. Both pipelines exhibit this, with negligible differences between them. On the other hand, the F1 score exhibits higher variation across models, with RF achieving the highest score of $\approx 99\%$ and DT achieving the lowest counterpart of $\approx 84\%$. Across pipelines, there is a negligible reduction in F1-score, except for DT, where the score is less by $\approx 0.1\%$. Cost-wise, the RF feature-reduction pipeline requires $\approx 30\%$ less training time of ≈ 253 seconds than its full-feature counterpart, taking ≈ 369 seconds. Similarly, the CPU percentage required for the RF feature-reduction pipeline is $\approx 26\%$ less than that of the full-feature counterpart. The three other models have yielded comparable results. Overall, the results show that similar performance can be achieved with considerably fewer resources and reduced training time, promoting better sustainable ML edge models.

5 CONCLUSION

Context-oriented data brokerage platforms, like FIWARE, offer standard contextual representations of data assets. This platform makes it easy to share and use IoT data. However, so far FIWARE systems lack the ability to verify the legitimacy of data before acting on them. This involves determining whether a data asset is benign or malicious, as well as the specific type of malicious activity. This limitation poses a critical risk of exploiting FIWARE to spread malicious data and significantly impact data consumers, AI applications being the most prominent ones. This work addressed the limitation with a novel, edge-native, solution for intelligent anomaly detection. The proposed solution integrates a ML-based microservice anomaly detector, in a pluggable manner using FastAPI. The solution also had a group of data-verifying brokers that leverage the FIWARE Pub/Sub model and the NGSI-LD to make it possi-

ble for data and verification messages to be sent and received in a flexible, asynchronous way. The prototype implementation of the solution has been evaluated experimentally to analyze the overhead of the solution as a cost indicator, compared to the benefit of reducing the spread of malicious data. Evaluation results have shown the solution to require $\approx 12\%$ longer response time per data entity and reduce the response throughput by $\approx 20\%$. At the same time, the results show the ability to accurately detect over 95% of malicious data, allowing FIWARE to handle them accordingly.

REFERENCES

Alberti, A. M., Santos, M. A., Souza, R., Da Silva, H. D. L., Carneiro, J. R., Figueiredo, V. A. C., and Rodrigues, J. J. (2019). Platforms for smart environments and future internet design: A survey. *IEEE Access*, 7:165748–165778.

Amara, F. Z., Hemam, M., Djezzar, M., and Maimor, M. (2022). Semantic web and internet of things: Challenges, applications and perspectives. *Journal of ICT Standardization*, 10(2):261–291.

Anwar, F. and Saravanan, S. (2022). Comparison of artificial intelligence algorithms for iot botnet detection on apache spark platform. *Procedia Computer Science*, 215:499–508.

Ataei, M., Eghmazi, A., Shakerian, A., Landry Jr, R., and Chevrette, G. (2023). Publish/subscribe method for real-time data processing in massive iot leveraging blockchain for secured storage. *Sensors*, 23(24):9692.

Baee, M. A. R., Simpson, L., and Armstrong, W. (2024). Anomaly detection in the key-management interoperability protocol using metadata. *IEEE Open Journal of the Computer Society*.

Barriga, J. A., Clemente, P. J., Hernández, J., and Pérez-Toledano, M. A. (2022). Simulateiot-fiware: Domain specific language to design, code generation and execute iot simulation environments on fiware. *IEEE Access*, 10:7800–7822.

Bellini, P., Palesi, L. A. I., Giovannoni, A., and Nesi, P. (2023). Managing complexity of data models and performance in broker-based internet/web of things architectures. *Internet of Things*, 23:100834.

Lazidis, A., Tsakos, K., and Petrakis, E. G. (2022). Publish–subscribe approaches for the iot and the cloud: Functional and performance evaluation of open-source systems. *Internet of Things*, 19:100538.

Martín, D. G., Florez, S. L., González-Briones, A., and Corchado, J. M. (2023). Cosibas platform—cognitive services for iot-based scenarios: Application in p2p networks for energy exchange. *Sensors*, 23(2):982.

Martins, I., Resende, J. S., Sousa, P. R., Silva, S., Antunes, L., and Gama, J. (2022). Host-based ids: A review and open issues of an anomaly detection system in iot. *Future Generation Computer Systems*, 133:95–113.

Munoz-Arcentales, A., López-Pernas, S., Conde, J., Alonso, Á., Salvachúa, J., and Hierro, J. J. (2021). Enabling context-aware data analytics in smart environments: An open source reference implementation. *Sensors*, 21(21):7095.

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., and Ghorbani, A. A. (2023). Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13):5941.

Razzaque, M. A., Milojevic-Jevric, M., Palade, A., and Clarke, S. (2015). Middleware for internet of things: a survey. *IEEE Internet of things journal*, 3(1):70–95.

Shukla, P., Krishna, C. R., and Patil, N. V. (2024). Kafkashield: Kafka streams-based distributed detection scheme for iot traffic-based ddos attacks. *Security and Privacy*, 7(6):e416.

Sirisha, A., Chaitanya, K., Krishna, K., and Kanumalli, S. S. (2021). Intrusion detection models using supervised and unsupervised algorithms-a comparative estimation. *International Journal of Safety and Security Engineering*, 11(1):51–58.

Zyrianoff, I., Heideker, A., Sciullo, L., Kamienski, C., and Di Felice, M. (2021). Interoperability in open iot platforms: Wot-fiware comparison and integration. In *2021 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 169–174. IEEE.