A Machine Learning and UFW-based Security Architecture for Mitigating DoS Attacks on MQTT Networks

Matheus Castro[1], Luca Naja[1], Nelson Silva[1], Eneida Soares[1], Diego Amoedo[1], Andrey Bessa[1], Edma Mattos[1], Waldir Sabino[1], and Celso Carvalho[1]

[1]Federal University of Amazonas and Center for R&D in Elec. and Inf. Tech. (UFAM/CETELI), AM-Brazil

E-mails: {matheusfigueiredo, luca.naja, silva.nelson, eneida.soares, diegoamoedo, andrey.bessa, edmavalleria, waldirjr, ccarvalho_}@ufam.edu.br

Abstract— **The expansion of the Internet of Things (IoT) presents significant security challenges due to the limited resources of many IoT devices and their insufficient protection mechanisms. Lightweight protocols, such as Message Queueing Telemetry Transport (MQTT), are particularly susceptible to cyberattacks, underscoring the need for robust and adaptive mitigation strategies. Traditional security solutions, including conventional firewalls and signature-based detection systems, often struggle to counter sophisticated and evolving threats. To address these challenges, this work proposes a novel security architecture that integrates Machine Learning techniques with the Uncomplicated Firewall (UFW) to effectively detect and mitigate cyberattacks in IoT environments. The LightGBM (LGBM) classifier achieved a detection accuracy of 96%, with both precision and recall exceeding 95% in identifying malicious MQTT traffic, thereby validating its effectiveness. Furthermore, the proposed approach was tested under real attack scenarios on an IoT system, and the source code has been made publicly available to facilitate future research and improvements.**

Keywords— **Internet of Things, Message Queuing Telemetry Transport, Machine Learning, Uncomplicated Firewall, Cybersecurity.**

I. Introduction

The rapid expansion of the (IoT) has driven technological advancements across industrial, commercial, and consumer domains, fostering seamless connectivity among billions of devices. However, this widespread interconnectivity has also introduced significant security challenges, as many IoT devices - ranging from sensors and actuators to smart appliances - are inherently resource constrained and lack robust security mechanisms [1]. These vulnerabilities make IoT networks prime targets for cyber threats, particularly Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) attacks, which exploit the weaknesses of lightweight communication protocols such as MQTT [2]. In this context, the MQTT protocol is becoming the standard messaging protocol for many Machine-to-Machine (M2M) communications in IoT scenarios, due to its lightweight overhead, publish/subscribe (pub/sub) model, and bidirectional capabilities [3].

As IoT networks continue to expand, traditional firewall configurations and signature-based Intrusion Detection Systems (IDS) have struggled to adapt to evolving cyber threats, particularly as attackers develop more sophisticated evasion techniques [1]. These conventional security solutions often rely on predefined signatures or static rule sets, making them ineffective against novel attack patterns. Machine Learning-driven approaches have emerged as a promising alternative, enabling real-time anomaly detection and enhancing the capability of Network Intrusion Detection and Prevention Systems (NIDPS) to mitigate security breaches dynamically [4].

The primary objective of this work is to design and validate a comprehensive security architecture tailored for MQTT-based IoT environments. This architecture integrates Machine Learning-based anomaly detection with UFW-based mitigation mechanisms to effectively detect, block, and mitigate DoS attacks. By addressing the inherent vulnerabilities of resource-constrained IoT devices, our approach aims to provide a scalable, automated, and adaptive solution that significantly enhances the overall security posture against evolving cyber threats.

Contributions: (1) We propose a security architecture for cyberattack mitigation in IoT environments, combining Machine Learning methods and the UFW. (2) We demonstrate the effectiveness of the system in detecting malicious traffic and automatically mitigating suspicious activities. (3) We conduct real-world attacks on an IoT system and successfully achieve mitigation by deploying the proposed security architecture. The source code is publicly available for further research and development.

II. Related Work

A. Machine Learning-based Attack Detection

Nowadays, many researchers have proposed Machine Learning-based approaches to detect cyberattacks in IoT systems. These methods are based on models trained with features extracted from network packets, enabling the classification of packet types based on detected patterns. In [5], Figueiredo *et al.* compared the performance of various binary classifiers in identifying malicious traffic in MQTT-based smart home networks. Hussain *et al.* [6] developed a security framework for healthcare IoT using IoT-Flock to generate attack traffic and applied Machine Learning techniques for anomaly detection. Sultan *et al.* [7] evaluated five Machine Learning models for detecting man-in-the-middle (MiTM) attacks, considering multiple performance metrics. In [9], Xie implemented and compared four Machine Learning models to classify 12 different types of DDoS attacks in networks based on transport layer protocols. Although most studies focus on detecting cyberattacks, they often lack mechanisms for mitigating threats after detection. Similarly, Alaiz-Moreton *et al.* [8] created a dataset from MQTT networkd to propose an IDS leveraging ensemble methods and recurrent neural networks to classify attacks in IoT environments. The cited works achieved great results in detecting attacks; however, they only focused on offline detection. Furthermore, online detection was not addressed, which is a crucial aspect in real IoT scenarios. In our work, we propose an online detection approach for DoS attacks in a real-world setting using a Machine Learning model. This real-time capability is essential for practical applications, especially in systems that require immediate response to threats.

B. *Attack Mitigation Methods*

The methods for mitigating cyberattacks are usually based on advanced firewalls, DDoS prevention systems against volumetric attacks, and NIDPS. These defense strategies primarily rely on predefined rules, but modern systems also incorporate behavioral analysis and Artificial Intelligence-driven threat detection to enhance mitigation effectiveness. In the IoT context, attack mitigation approaches have been proposed in the literature. In [10], Jadhav and Sane proposed the implementation of the *Suricata*, an open-source IDS/IPS, on a Raspberry Pi to demonstrate the feasibility of running mitigation software on an embedded platform. Another study on IDS/IPS topic was conducted by Coscia *et al.* [4], in which the authors proposed an algorithm for the automatic generation of rules for *Suricata* using the *Decision Tree* Machine Learning model. Furthermore, some authors have proposed frameworks specifically designed to mitigate attacks in IoT environments. The works of [10] and [4] demonstrate good performance in mitigating attacks; however, they do not focus on MQTT-based networks and do not address detection methods, leaving a gap that may allow other types of attacks to occur. In [13], Jha *et al.* proposed a mitigation strategy that combines attack detection using Machine Learning models with real-time mitigation through firewalls. However, this work also does not focus on IoT systems. In [11], Illy *et al.* adopted Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) architectures to enable the effective implementation of an IDPS in home IoT networks. Zhou *et al.* [12] proposed an SDN-enabled framework capable of proactively adapting the attack surface of IoT networks, dynamically optimizing defense strategies, and rapidly deploying corresponding defense mechanisms. In our work, we focus on detection-based mitigation of attacks specifically in MQTT networks, ensuring higher accuracy in identifying threats whose behavior was previously learned by the Machine Learning model.

III. Scenario

This work aims to develop a security architecture to mitigate DoS attacks in IoT systems focusing on MQTT communications. Figure 1 illustrates the proposed scenario.

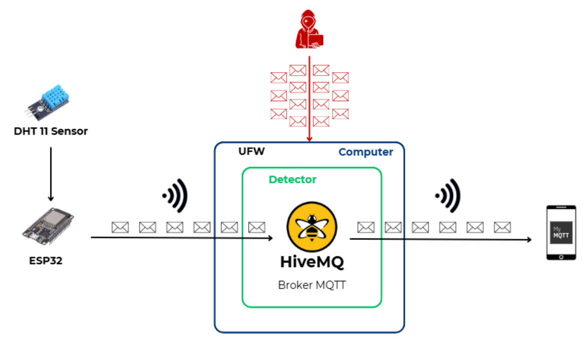


Fig. 1: Scenario proposed.

MQTT Network. In our scenario, we implemented a remote temperature and humidity monitoring system. A DHT11 sensor was used to collect measurements, and an ESP32 module handled communication with the broker, sending messages at a frequency of 60 seconds. The temperature and humidity data is received by an Android application called MyMQTT[1]. For this work, we used the HiveMQ broker. The communication network follows the MQTT 3.1.1 protocol. Authentication was not enabled and no username and password exchange is required to authenticate clients to the broker.

Denial of Service Attack. Denial of Service is a cyberattack that makes IoT systems or resources on the network unavailable to their intended legitimate users [14]. In this work, we implemented flooding attack in the network. In this case, an adversary sends many connection-establishment requests to the victim and each request causes the victim to allocate resources that maintain state for that connection [15]. Hence, the aim goal is to saturate the broker, by establishing several connections with the broker and sending, for each connection, the higher number of messages possible [16]. We adopted the tools the MQTTSA[2] and MQTT-Malaria[3] to execute this attack. Figure 1 shows the adversary in the network.

IV. Methodology

The methodology of this work consists in two steps. The first, is to develop a Machine Learning based-detector to indetify DoS attacks on the network and the second is to integrate the detector with the UFW to mitigate attacks.

A. *Machine Learning based-Detector*

To develop of the detector consists in four major stages. These modules include packets capturing, dataset creation, and Machine Learning model training.

1) Packets Capturing: The data capture from the network was performed using the TShark[4] tool. We collected normal and malicious traffic separately. As a result, the final files

[1]Available at `https://mymqtt.app/en`
[2]Available in `https://sites.google.com/fbk.eu/mqttsa`
[3]Available in `https://github.com/etactica/mqtt-malaria`
[4]Available at `https://www.wireshark.org/docs/man-pages/tshark.html`

consists in one containing traffic without DoS attacks and another with DoS attacks. Table I presents the total amount of collected data.

TABLE I: Number of packets captured.

Traffic	Number of packets
Without attack	24,559
With attack	105,981

2) Dataset Creation and Preprocessing: The data was collected and saved in CSV file format. To prevent class imbalance, we performed undersampling. First, we concatenated the legitimate and malicious traffic samples. Then, we randomly selected a number of samples from the majority class equal to that of the minority class. This strategy aims to reduce bias in the model during training.

For this study, we considered the following features for analysis:

- **tcp.flags**: A hexadecimal number that represents the TCP flags.
- **tcp.time_delta**: A floating-point number that indicates the time difference between the current packet and the previous one.
- **tcp.len**: An integer that represents the TCP payload length.

In particular, to ensure that the **tcp.flags** feature contained only numerical values, we applied a cleaning process to remove any non-numeric characters. This was achieved using a regular expression that retained only digits and periods. This step helps standardize the data and prevent inconsistencies that could affect the analysis and model training.

3) Machine Learning Model Training: To train our Machine Learning model, we used the LazyPredict[5] (LP) framework to perform a comparative evaluation of various classifiers. This evaluation was based on multiple metrics —- Accuracy, Receiver Operating Characteristics (ROC), Area Under The Curve (AUC) and F1 score —- which allowed us to identify the most promising algorithms for our problem.

The initial assessment considered several algorithms, and the models that achieved a precision greater than 90% are presented in the Table II:

TABLE II: Performance of models with accuracy above 95%.

Model	Accuracy	ROC AUC	F1 Score
LGBMClassifier	0.97	0.97	0.97
KNeighborsClassifier	0.97	0.97	0.97
RandomForestClassifier	0.96	0.96	0.96
BaggingClassifier	0.96	0.96	0.96
ExtraTreesClassifier	0.96	0.96	0.96
DecisionTreeClassifier	0.96	0.96	0.96
ExtraTreeClassifier	0.96	0.96	0.96
AdaBoostClassifier	0.96	0.96	0.96

Among the models evaluated for attack detection in IoT networks, several demonstrated excellent performance, achieving over 95% in Accuracy, ROC AUC, and F1-Score. Notably, both the LGBMClassifier and the KNeighborsClassifier reached 97% across all evaluated metrics. Other models, such as RandomForestClassifier, BaggingClassifier, and AdaBoostClassifier, also exhibited consistent results, with slightly lower but still robust performance for the proposed scenario.

We then applied a stratified five-fold cross-validation to further assess the LGBMClassifier's generalization. This procedure yielded an average F1 Score of 0.9689 with a standard deviation of 0.0018, and individual fold scores of 0.9711, 0.9667, 0.9669, 0.9696, and 0.9699—confirming the model's robust and consistent performance across different data splits.

To partition the training set, the data was randomly shuffled while preserving the original class distribution and then divided into five balanced subsets. In each iteration, the model was trained on four of these subsets and validated on the remaining one.

Furthermore, to fine-tune the model and further optimize its performance, we used a grid search approach to select the optimal hyperparameters.

4) Development of the Online Capture Script: The development of the online capture script is the final step of the detector. This script is responsible for capturing and preprocessing values online and sending them to the trained Machine Learning model. The script was written in Python, and its structure is based on three steps, as described in Figure 2.

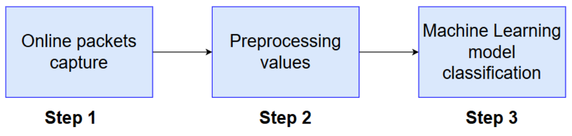


Fig. 2: Detector steps.

1) **Step 1**: Online packet capture was performed using the *PySharK* library. We used the local host interface and the *"tcp or mqtt"* capture filter in the script.
2) **Step 2**: To preprocess values from the packets, we created a function to process categorical values from the *tcp.flags* feature.
3) **Step 3**: The classification result is the output of the Machine Learning model.

B. Intregation of Detector and UFW

This step consists of mitigating attacks detected by the Machine Learning model. The main goal is to use the UFW to block network packets when the Machine Learning model output is 1 (attack detected). To achieve this, we implemented a strategy that dynamically updates the firewall rules upon attack detection. The approach consists of adding iptables rules to drop all MQTT traffic (port 1883) when an attack is identified. Furthermore, the system terminates the MQTT broker process to prevent further attack propagation. The Algorithm 1 illustrates the work of the system.

V. Experiments and Results

In this section, the practical application of methodology proposed is discussed. First, we show the results of Machine

[5]Available at `https://lazypredict.readthedocs.io/en/latest/`

Algorithm 1 Integration of Machine Learning Model with UFW.

```
1:  Initialization:
2:  Define the network interface for packet capture.
3:  Set the capture filter for TCP and MQTT packets.
4:  Load the pre-trained Machine Learning model from a file.
5:  Packet Capture and Processing:
6:  Start continuous packet capture on the defined network
    interface.
7:  while new packet is captured do
8:      if packet contains a TCP layer then
9:          Extract the following features:
10:             tcp.flags: Remove the "0x" prefix and format
    as an integer.
11:             tcp.time_delta: Replace "N/A" with zero and
    convert to a floating-point number.
12:             tcp.len: Replace "N/A" with zero and convert
    to an integer.
13:         Display the extracted features.
14:     end if
15:     Packet Classification:
16:     Use the Machine Learning model to classify the packet
    based on extracted features.
17:     Display the classification result.
18:     if packet is classified as an attack (label = 1) then
19:         Add iptables rules to drop MQTT packets on port
    1883.
20:         Display a message indicating attack mitigation.
21:         Terminate the MQTT broker process to prevent
    further propagation.
22:         Stop packet capture.
23:     end if
24: end while
25: Termination:
26: if user manually interrupts execution then
27:     Display termination message.
28: else if an unexpected error occurs then
29:     Display an error message and terminate the program.
30: end if
31: Display a message indicating that the network is out of
    service.
```

Learning model training and, finally, the implementation of the architecture on scenarios with attack.

A. Machine Learning Model Training Results

As mentioned in Section IV, we use LP to select an optimal model for packet classification. The chosen model was the LGBM Classifier. The best model configuration was determined using *GridSearchCV*, resulting in a model with a learning rate of 0.1, maximum depth of 7, and 200 estimators. This configuration achieved the highest performance in our experiments. The selected algorithm was implemented in Python using the *LightGBM* and *Scikit-learn* libraries.

Table III presents the classification report of the LGBM Classifier, where class 0 represents normal traffic and class 1 indicates attack traffic. To validate these results, we consider common Machine Learning evaluation metrics described in [17]: True Positive, False Positive, False Negative, True Negative, Accuracy, Precision, Recall, and F1-Score.

TABLE III: Classification Report Metrics

Class	Precision	Recall	F1-score
0	0.9717	0.9642	0.9679
1	0.9645	0.9720	0.9683
Accuracy		0.9681	

Furthermore, Table IV shows the model's performance in terms of hits and misses in percentage format.

TABLE IV: Confusion matrix (%) of the LGBM classifier.

Actual \ Predicted	**Normal**	**Attack**
Normal	96.42	3.58
Attack	2.80	97.20

B. Attack Scenario

To execute attacks on the systems, we use the MQTTSA and MQTT-Malaria tools with the followings sets: In the first time of attack, we employ 100 amount of connections for the flooding-based DoS with 10 megabytes the payload size and 100 messages to test of messages queued by the browser. In the second time, we execute 8 processes in sequence, each with 10000 messages of 100 megabytes.

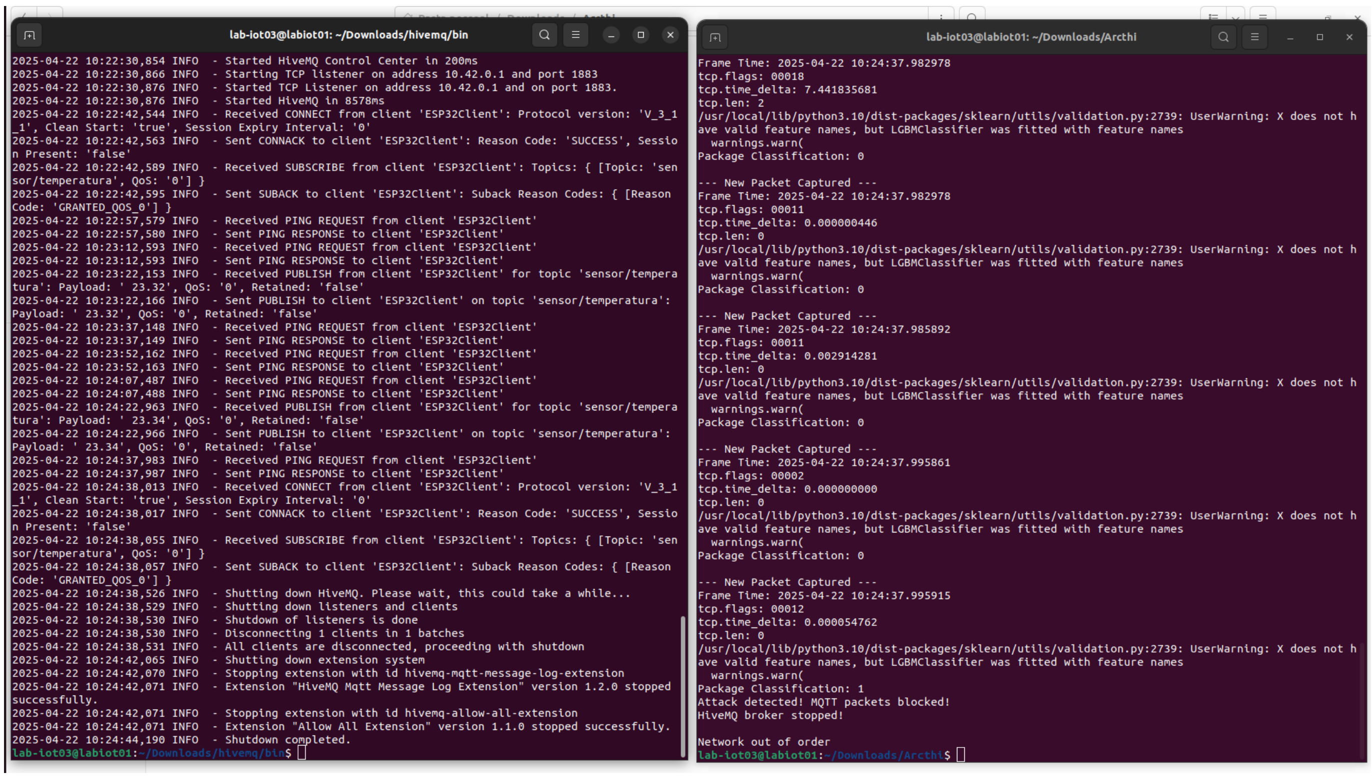
C. Implementation of Security Architecture

To validate our approach, we implemented the architecture within the attack scenario detailed in Section V-B. The integration of the online detector with UFW, described in Section IV, was tested on the same machine hosting the broker. Figure 3 illustrates the implemented architecture, with the broker logs shown on the left and the packet classification script on the right.

The complete project, including all code and demonstration videos, is available in a public GitHub repository: https://github.com/matheus-fig/Architecture_Security_MQTT.

VI. Conclusion

The research presented explores the use of Machine Learning techniques for detecting and mitigating cyberattacks in MQTT-based IoT environments. The proposed approach integrates an online detection system with the UFW, enabling real-time attack detection and automated mitigation. The results demonstrated that the system effectively identifies and mitigates attacks, making it a viable solution for implementation in critical IoT systems. However, blocking all network packets when a threat is detected is not the most efficient solution for such systems. Future work will focus on studying methods to define precise firewall rules that target only malicious packets, rather than disabling the entire network. These studies will contribute to making the proposed approach more practical for real-world deployment in IoT environments with stringent security and performance requirements.


lab-iot03@labiot01: ~/Downloads/hivemq/bin
lab-iot03@labiot01: ~/Downloads/Arcthi

```
2025-04-22 10:22:30,854 INFO  - Started HiveMQ Control Center in 200ms
2025-04-22 10:22:30,866 INFO  - Starting TCP listener on address 10.42.0.1 and port 1883
2025-04-22 10:22:30,876 INFO  - Started TCP Listener on address 10.42.0.1 and on port 1883.
2025-04-22 10:22:30,876 INFO  - Started HiveMQ in 8578ms
2025-04-22 10:22:42,544 INFO  - Received CONNECT from client 'ESP32Client': Protocol version: 'V_3_1
_1', Clean Start: 'true', Session Expiry Interval: '0'
2025-04-22 10:22:42,563 INFO  - Sent CONNACK to client 'ESP32Client': Reason Code: 'SUCCESS', Sessio
n Present: 'false'
2025-04-22 10:22:42,589 INFO  - Received SUBSCRIBE from client 'ESP32Client': Topics: { [Topic: 'sen
sor/temperatura', QoS: '0'] }
2025-04-22 10:22:42,595 INFO  - Sent SUBACK to client 'ESP32Client': Suback Reason Codes: { [Reason
Code: 'GRANTED_QOS_0'] }
2025-04-22 10:22:57,579 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:22:57,580 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:23:12,593 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:23:12,593 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:23:22,153 INFO  - Received PUBLISH from client 'ESP32Client' for topic 'sensor/tempera
tura': Payload: ' 23.32', QoS: '0', Retained: 'false'
2025-04-22 10:23:22,166 INFO  - Sent PUBLISH to client 'ESP32Client' on topic 'sensor/temperatura':
Payload: ' 23.32', QoS: '0', Retained: 'false'
2025-04-22 10:23:37,148 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:23:37,149 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:23:52,162 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:23:52,163 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:24:07,487 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:24:07,488 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:24:22,963 INFO  - Received PUBLISH from client 'ESP32Client' for topic 'sensor/tempera
tura': Payload: ' 23.34', QoS: '0', Retained: 'false'
2025-04-22 10:24:22,966 INFO  - Sent PUBLISH to client 'ESP32Client' on topic 'sensor/temperatura':
Payload: ' 23.34', QoS: '0', Retained: 'false'
2025-04-22 10:24:37,983 INFO  - Received PING REQUEST from client 'ESP32Client'
2025-04-22 10:24:37,987 INFO  - Sent PING RESPONSE to client 'ESP32Client'
2025-04-22 10:24:38,013 INFO  - Received CONNECT from client 'ESP32Client': Protocol version: 'V_3_1
_1', Clean Start: 'true', Session Expiry Interval: '0'
2025-04-22 10:24:38,017 INFO  - Sent CONNACK to client 'ESP32Client': Reason Code: 'SUCCESS', Sessio
n Present: 'false'
2025-04-22 10:24:38,055 INFO  - Received SUBSCRIBE from client 'ESP32Client': Topics: { [Topic: 'sen
sor/temperatura', QoS: '0'] }
2025-04-22 10:24:38,057 INFO  - Sent SUBACK to client 'ESP32Client': Suback Reason Codes: { [Reason
Code: 'GRANTED_QOS_0'] }
2025-04-22 10:24:38,526 INFO  - Shutting down HiveMQ. Please wait, this could take a while...
2025-04-22 10:24:38,529 INFO  - Shutting down listeners and clients
2025-04-22 10:24:38,530 INFO  - Shutdown of listeners is done
2025-04-22 10:24:38,530 INFO  - Disconnecting 1 clients in 1 batches
2025-04-22 10:24:38,531 INFO  - All clients are disconnected, proceeding with shutdown
2025-04-22 10:24:42,065 INFO  - Shutting down extension system
2025-04-22 10:24:42,070 INFO  - Stopping extension with id hivemq-mqtt-message-log-extension
2025-04-22 10:24:42,071 INFO  - Extension "HiveMQ Mqtt Message Log Extension" version 1.2.0 stopped
successfully.
2025-04-22 10:24:42,071 INFO  - Stopping extension with id hivemq-allow-all-extension
2025-04-22 10:24:42,071 INFO  - Extension "Allow All Extension" version 1.1.0 stopped successfully.
2025-04-22 10:24:44,190 INFO  - Shutdown completed.
lab-iot03@labiot01:~/Downloads/hivemq/bin$
```

```
Frame Time: 2025-04-22 10:24:37.982978
tcp.flags: 00018
tcp.time_delta: 7.441835681
tcp.len: 2
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not h
ave valid feature names, but LGBMClassifier was fitted with feature names
  warnings.warn(
Package Classification: 0

--- New Packet Captured ---
Frame Time: 2025-04-22 10:24:37.982978
tcp.flags: 00011
tcp.time_delta: 0.000000446
tcp.len: 0
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not h
ave valid feature names, but LGBMClassifier was fitted with feature names
  warnings.warn(
Package Classification: 0

--- New Packet Captured ---
Frame Time: 2025-04-22 10:24:37.985892
tcp.flags: 00011
tcp.time_delta: 0.002914281
tcp.len: 0
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not h
ave valid feature names, but LGBMClassifier was fitted with feature names
  warnings.warn(
Package Classification: 0

--- New Packet Captured ---
Frame Time: 2025-04-22 10:24:37.995861
tcp.flags: 00002
tcp.time_delta: 0.000000000
tcp.len: 0
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not h
ave valid feature names, but LGBMClassifier was fitted with feature names
  warnings.warn(
Package Classification: 0

--- New Packet Captured ---
Frame Time: 2025-04-22 10:24:37.995915
tcp.flags: 00012
tcp.time_delta: 0.000054762
tcp.len: 0
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not h
ave valid feature names, but LGBMClassifier was fitted with feature names
  warnings.warn(
Package Classification: 1
Attack detected! MQTT packets blocked!
HiveMQ broker stopped!

Network out of order
lab-iot03@labiot01:~/Downloads/Arcthi$
```

Fig. 3: Simultaneous execution of the online detector with UFW.

ACKNOWLEDGEMENTS

The authors would like to thank the Center of Electronic and Information Technology Research and Development (CETELI/UFAM) for the financial and institutional support to the authors of the article.

REFERENCES

[1] I. Stellios, P. Kotzanikolaou, M. Psarakis, C. Alcaraz, and J. Lopez, “A Survey of IoT-Enabled Cyberattacks: Assessing Attack Paths to Critical Infrastructures and Services,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3453–3495, 2018.

[2] I. Vaccari, M. Aiello, and E. Cambiaso, “SlowITe, a Novel Denial of Service Attack Affecting MQTT,” *Sensors*, vol. 20, no. 10, p. 2932, May 2020.

[3] M. Amoretti, R. Pecori, Y. Protskaya, L. Veltri and F. Zanichelli, “A Scalable and Secure Publish/Subscribe-Based Framework for Industrial IoT,” *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 3815-3825, 2021.

[4] A. Coscia, V. Dentamaro, S. Galantucci, A. Maci, and G. Pirlo, “Automatic Decision Tree-Based NIDPS Ruleset Generation for DoS/DDoS Attacks,” *Journal of Information Security and Applications*, vol. 82, p. 103736, 2024.

[5] M. Figueiredo, D. P. Sodré, R. M. de Medeiros, V. F. Lucena Jr and I. Bessa, “Detection of Cyberattacks in IoT Networks Using Artificial Intelligence: A Comparative Study," *Proceedings of the 2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2024.

[6] F. Hussain, S. G. Abbas, G. A. Shah, I. M. Pires, U. U. Fayyaz, F. Shahzad, N. M. Garcia and E. Zdravevski, “A Framework for Malicious Traffic Detection in IoT Healthcare Environment," *Sensors*, vol. 21, nº 9, p. 3025, 2021.

[7] A. B. M. Sultan, S. Mehmood and H. Zahid “Man in the Middle Attack Detection for MQTT based IoT devices using different Machine Learning Algorithms," *Proceedings of the 2022 2nd International Conference on Artificial Intelligence (ICAI)*, 2022.

[8] H. Alaiz-Moreton, J. Aveleira-Mata, J. Ondicol-Garcia, A. L. Muñoz-Castañeda, I. García and C. Benavides, “Multiclass Classification Procedure for Detecting Attacks on MQTT-IoT Protocol,” *Complexity*, vol. 2019, p. 1-11, 2019.

[9] Y. Xie, “Machine Learning-Based DDoS Detection for IoT Networks,” *Applied and Computational Engineering*, vol. 29, pp. 99–107, 2023.

[10] A. S. Jadhav and S. S. Sane, “Enhancing IoT Security Affordably with Raspberry Pi and Open-Source IDS/IPS," *Proceedings of the 2022 IEEE 9th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2022.

[11] P. Illy, G. Kaddoum, K. Kaur and S. Garg, “ML-Based IDPS Enhancement With Complementary Features for Home IoT Networks,” *IEEE Transactions on Network and Service Management*, vol. 10, nº 2, pp. 772-783, 2022.

[12] Y. Zhou, G. Cheng and S. Yu, "An SDN-Enabled Proactive Defense Framework for DDoS Mitigation in IoT Networks," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5366-5380, 2021.

[13] P. Jha, G. Singh, A. Kumar, D. Agrawal, Y. S. Patel, and J. Forough, "NetProbe: Deep Learning-Driven DDoS Detection with a Two-Tiered Mitigation Strategy," *Proceedings of the 26th International Conference on Distributed Computing and Networking (ICDCN ’25)*, 2025.

[14] E. Gyamfi e A. Jurcut, “Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets," *Sensors*, vol. 22, nº 10, p. 3744, 2022.

[15] A. D. Wood e J. A. Stankovic, “Denial of Service in Sensor Networks," *Computer*, vol. 35, nº 10, pp. 54–62, 2002.

[16] I. Vaccari, G. Chiola, M. Aiello, M. Mongelli e E. Cambiaso, “MQTTset, a New Dataset for Machine Learning Techniques on MQTT," *Sensors*, vol. 20, nº 22, p. 6578, 2020.

[17] M. Hasan, Md. M. Islam, Md I. I. Zarif, M.M.A. Hashem,, “Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches,” *Internet of Things*, vol. 7, 100059, 2019.