



# One-Class Reconstruction Methods for Categorizing DoS Attacks on CoAP

Álvaro Michelena<sup>1</sup>, Antonio Díaz-Longueira<sup>1</sup>, Míriam Timiraos<sup>1,2</sup>,  
Esteban Jove<sup>1</sup>, Jose Azeiteiro-Mata<sup>3</sup>, Isaias García-Rodríguez<sup>4</sup>,  
María Teresa García-Ordás<sup>4</sup>, José Luis Calvo-Rolle<sup>1</sup>,  
and Héctor Alaiz-Moretón<sup>3</sup>

- <sup>1</sup> Department of Industrial Engineering, CTC, CITIC, University of A Coruña,  
Calle Mendizábal s/n, 15403 Ferrol, A Coruña, Spain  
{alvaro.michelena,a.diazl,miriam.timiraos.diaz,  
esteban.jove,jlcalvo,hector.quintian}@udc.es
- <sup>2</sup> Department of Water Technologies, National Technological Center,  
Fundación Instituto Tecnológico de Galicia, Cantón Grande 9,  
Planta 3, 15003 A Coruña, Spain  
mtimiraos@itg.es
- <sup>3</sup> Research Institute of Applied Sciences in Cybersecurity (RIASC), MIC,  
University of León, Campus de Vegazana, s/n, 24071 León, Spain  
{jose.aveiteiro,hector.moreton}@unileon.es
- <sup>4</sup> Department of Electrical and Systems Engineering, University of León,  
Campus de Vegazana, s/n, 24071 León, Spain  
{isaias.garcia,mgaro}@unileon.es

**Abstract.** Denial of Service (DoS) attack over Internet of Things (IoT) is among the most prevalent cyber threat, their complex behavior makes very expensive the use of Datagram Transport Layer Security (DTLS) for securing purposes. DoS attack exploits specific protocol features, causing disruptions and remaining undetected by legitimate components. This paper introduces a set of one-class reconstruction methods such as auto-encoder, K-Means and PCA (Principal Component Analysis) for developing a categorization model in order to prevent IoT DoS attacks over the CoAP (Constrained Application Protocol) environments.

**Keywords:** Anomaly detection · IoT · CoAP · autoencoders · PCA · K-Means

## 1 Introduction

The Constrained Application Protocol (CoAP) is a web-like transfer protocol specifically designed to facilitate communication at the application layer for energy-constrained IoT devices [25]. CoAP operates over the User Datagram Protocol (UDP) and adheres to the Representational State Transfer (REST) architectural framework. CoAP's architecture comprises two distinct layers: (1) the message layer and (2) the request/response layer. The message layer is

responsible for managing communication over the UDP protocol, while the request/response layer transmits the corresponding messages, using specific codes to mitigate and circumvent functional issues, such as message loss [17, 21].

One of the notable advantages of CoAP is its ability to seamlessly integrate with HTTP, thus enabling integration with existing web infrastructure while satisfying the specific demands of constrained environments. This integration is achieved by addressing specialized requirements such as support for multicast communication, minimization of overhead, and simplicity in constrained settings. The primary purpose of CoAP is to facilitate machine-to-machine (M2M) communication, particularly in domains like smart energy and building automation. As well as the request/response interaction model, the protocol also encompasses built-in discovery mechanisms for services and resources, aligning itself with the fundamental concepts of the Web [2, 20].

The security of CoAP primarily relies on the implementation of the Datagram Transport Layer Security (DTLS) protocol at the transport layer. DTLS ensures confidentiality, integrity, and non-repudiation of information and services [23, 30]. However, not all IoT devices and environments can make use of DTLS due to the computationally expensive cryptographic operations required by this technology or the necessary additional bytes for message encryption and integrity checks. These needs produce a higher energy consumption, reduced network throughput, and increased latency, which can negatively impact the overall performance of the IoT network [4, 22]. A number of research is being conducted in order to find lighter implementations for DTLS or new techniques that can jointly be used with these cryptographic approaches. One such techniques is the development of model-based intrusion detection systems that can help securing IoT environments while relieving devices from the burden of the task [9, 11]. These models can be based on simple known rules, however, these cannot be used to solve a categorization problem. As a result, the classifier implementation must go through a process of learning from a set of training items.

This work is devoted to develop a convenient model-based IDS for detecting DoS attacks on IoT scenarios, trying to find the best techniques to achieve this objective. DoS attacks based on amplification is one of the most usual and dangerous ones, as it can be performed even in secured encrypted scenarios where DTLS is used [6, 18]. The objective of this work is to get a good detector for this kind of specific attacks. This approach introduces an scalability issue, as the model training should be performed for each different IoT ecosystem where it is needed. Also, the approach is only useful for amplification-based DoS attacks and so different models should be trained for detecting other anomalies, if needed. Currently, a number of efforts are being carried out in order to achieve more generic and zero-day attack detectors [8, 13, 24], but these solutions are more prone to both false positives and false negatives. Also, these approaches may require more computational resources and processing time.

One-class makes reference to a specific situation in which the classifier must distinguish between a known class (target class) and an unknown class (non-target class) [10,28]. One-class classifiers can be implemented through three approaches: using density estimation functions to approximate the system behavior, delimiting the boundaries of the target set, or applying reconstruction methods. This method implements a model from the training data to minimize the reconstruction error. Then, objects from the non-target class would lead to high reconstruction error, thus facilitating the outlier detection [15].

The paper is divided into a number of sections. The case study, which details the particular IoT CoAP ecosystem and dataset being used, is described in the second section. The third part discusses the techniques that will be used, including information on the auto-encoder, K-Means and PCA (Principal Component Analysis). The experiments that were carried out and their outcomes are covered in section four. Section five addresses the results of set of experiments while the findings and recommendations for further research are presented in the concluding part.

## 2 Case Study

In the preceding section, the general features of CoAP were discussed, along with the cybersecurity challenges it faces. This section delves deeper into the workings of CoAP and elaborates on the implementation of a DoS attack.

CoAP works like client/server model, much like HTTP, and makes use of the REST API architecture for communication. This structure employs a REST API, where resources are identified by unique URLs and can be manipulated through HTTP methods such as GET, POST, PUT, and DELETE. Additionally, CoAP includes the “Observe” functionality [5], which allows a client to keep track of a server resource. The server delivers periodic updates of the resource to registered clients, enabling bidirectional communication among devices.

To generate a dataset, a testing environment is deployed to create genuine traffic within the CoAP framework, wherein DoS attack will be performed to assess the protocol’s debility, as delineated in RFC7252 [25]. This environment comprises a “Node.js” server furnished with the “node-coap” library to facilitate the CoAP protocol. A “DHT11” sensor is interfaced with a “NodeMCU” board, which is programmed using the “ESP-CoAP” library [19] to deliver temperature and humidity services. A JavaScript client presents the sensor data on the terminal, while a pair of “Copper4cr” clients facilitate the dispatching of requests and the reception of responses [16].

A DoS attack will be executed on the CoAP protocol within the devised environment, with the objective of generating a valuable dataset to aid in the identification of anomalies in the protocol and the mitigation of such threats.

When a request is received, CoAP servers produce a response packet. The size of this response packet can be considerably larger than the request packet, due to CoAP’s capability to transmit multiple blocks in various sizes, even remarkably small ones during an attack. This characteristic makes CoAP clients susceptible to Denial of Service (DoS) attacks [29].

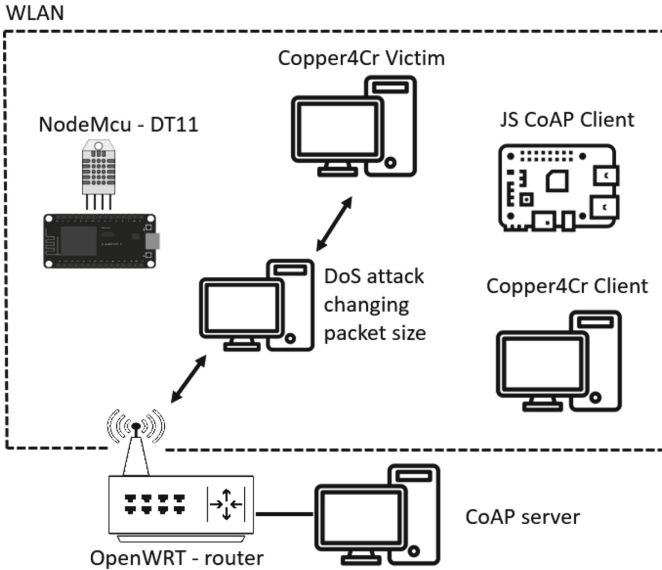


Fig. 1. CoAP environment with DoS attack

An attacker can launch an amplification attack by provoking a denial of service and falsifying the victim’s IP address as the source address in a request packet. This action prompts the server to dispatch a larger packet directed at the victim. To perform this attack within the environment, the attacker poses as a Copper4Cr client, spoofing the client’s IP address, leading the server to reply to their requests instead of the legitimate client’s. To impede the client’s service, the attacker adjusts the response packet to employ exceptionally small block sizes. As a consequence, the server is compelled to deliver an increased volume of response packets to the client. Figure 1 shows how the DoS attack is carried out in the CoAP environment.

All traffic is meticulously captured with the intent of procuring a pcap file, which is subsequently employed to analyze the frames of the generated traffic and extract universally shared fields among them. These fields encompass system times, relative capture times, and all fields pertinent to the CoAP protocol. The frames are labeled in accordance with their timestamp at the time of capture, signifying whether they correspond to a DoS attack or typical traffic.

The dataset collected for this study contains three types of fields: frame level fields, CoAP protocol fields, and a particular “type” field used to identify frames under DoS attack. The frame level fields allow for easy pattern recognition in the generated data. The CoAP protocol fields provide information specific to the frames using this protocol and can be found in the CoAP section of the Wireshark Display Filter Reference. The “type” field is used to indicate the

type of attack and frames under DoS attack are labeled with the “DoS” tag. The dataset is stored in a CSV file and contains a total of 30,319 frames, with 21,269 frames representing normal traffic and 9,050 frames representing traffic under attack.

### 3 One-Class Reconstruction Methods

This section describes the different reconstruction methods applied to the training set to develop anomaly detection. It is important to emphasize that only information about normal operations is registered.

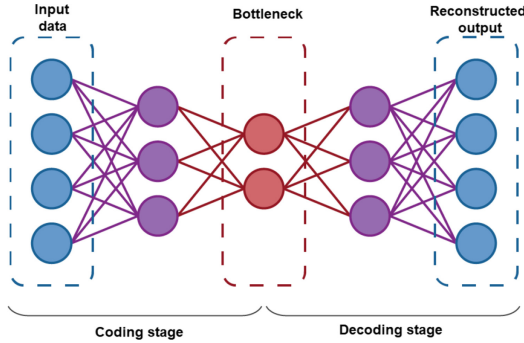
#### 3.1 Autoencoders

*Autoencoder* is a type of supervised neural network that is based on the dimensional reduction or compression of information, which is later decompressed to recreate the original input data so that the final representation is as close as possible to the original one. Figure 2 shows the architecture of an autoencoder network that presents two stages:

- Coding stage: it is made up of the input layer, in which the data is entered; one or more hidden layers of dimensional reduction and a last bottleneck layer, in which there is a compressed representation of the original data.
- Decoding stage: from the bottleneck layer, the information is decompressed by passing it through one or more hidden layers to be displayed in the output layer with the same dimension as at the network’s input.

The hidden bottleneck layer contains a number of hidden  $h_{auto}$  neurons [28,31]. Once the network is trained, it is assumed that test instances that do not belong to the target set will present a great reconstruction error. It is calculated through Eq. 1, where  $f_{auto}(p; w)$  represents the reconstructed output of the network.

$$e(p) = \| f_{auto}(p; w) - p \|^2 \tag{1}$$



**Fig. 2.** Autoencoder topology

### 3.2 K-Means

The K-Means is an unsupervised algorithm commonly used for machine learning problems [27, 28]. This clustering algorithm is based on the distances between objects to determine their memberships. It assumes that the data is grouped into *clusters* that must be selected by the user, and that they can be characterized by a series of  $\theta_k$  prototype objects located after minimizing the function in Eq. 2. These prototypes create a partition of the entire feature space.

$$\varepsilon_{K\text{-means}} = \sum_i \left( \min_k \|x_i - \theta_k\|^2 \right) \quad (2)$$

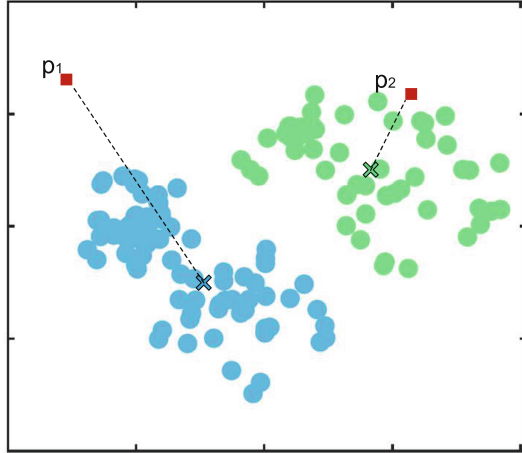
The use of K-Means for one-class purposes lies in the calculation of the reconstruction error. Once the different centroids are determined, a test instance  $p$  reconstruction error is the minimum Euclidean distance from the object to its closest prototype, as shown in Eq. 3.

$$e(p) = \min_k \|p - \theta_k\|^2 \quad (3)$$

An example of K-Means application in a two-dimensional dataset is shown in Fig. 3. The target set is separated into two clusters. The test point  $p_1$  is considered anomalous since the distance to its nearest centroid (blue cross) is greater than all maximum distances of blue points to that centroid. On the contrary,  $p_2$  belongs to the target set because it is closer to the green cross than many green points.

### 3.3 Principal Component Analysis

*Principal Component Analysis* (PCA) is a statistical method commonly applied to analyze multivariate data. Its use extends to many branches as a tool for dimensional reduction, clustering, and classification problems to anomaly detection. PCA focuses on finding the relationship between data by obtaining the



**Fig. 3.** Clasificador K-Means vs K-Centers (Color figure online)

orthonormal subspace that reflects the greatest variation between the different variables [1, 26, 28]. Then, the eigenvectors of the covariance matrix of the training data are calculated, whose eigenvalues are maximum, to later build a base  $\mathcal{W}$  of the generated subspace with them, onto which the data will be projected. From a test object  $p$ , the reconstruction error will be calculated to check if it belongs to the target data set. To do this, its projection in the subspace will be calculated first (see Eq. 4). The reconstruction error is calculated as the difference between the original and projected points, as shown in Eq. 5.

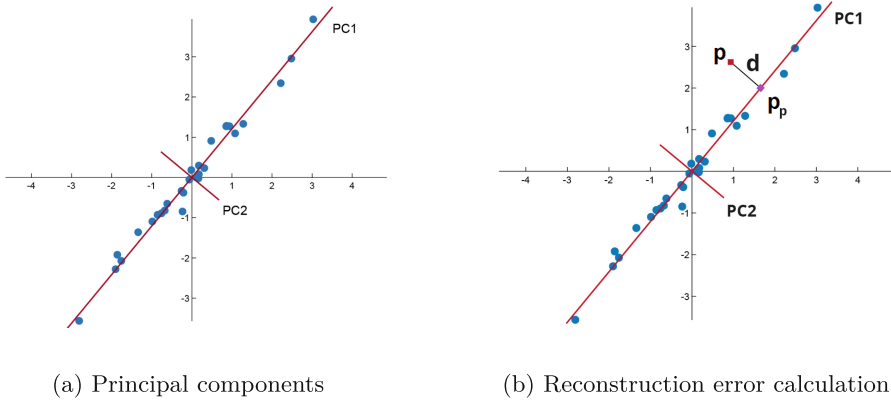
$$p_p = \mathcal{W}(\mathcal{W}^T \mathcal{W})^{-1} \mathcal{W}^T z \quad (4)$$

$$e(p) = \| p - p_p \|^2 \quad (5)$$

By default, the  $k$  eigenvectors with the largest eigenvalues are commonly used, although there it is possible to use eigenvectors with the smallest eigenvalues. Figure 4(a) shows an example of how two principal components are represented in two dimensions. In this case, PC1 corresponds to a greater eigenvalue. To check whether a test point  $p$  (red square) belongs to the target class, the square of the distance  $d$  from that point to its projection on the principal component  $p_p$  (purple square), as shown in Fig. 4(b). The point belongs to the target set if this value is below the limit established during the training stage. Otherwise, it will be considered as non-target [12, 14].

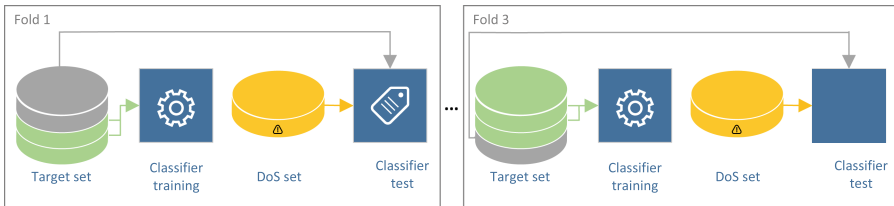
## 4 Experiments

The dataset is divided into two general groups, normal operation and Denial of Service situations. Considering that the training stage is based on data from nor-



**Fig. 4.** Example of PCA for anomaly detection

mal operations, a k-fold iteration is followed to validate the classifiers. Figure 5 shows an example with three folds, where the green sets are used to learn the target class patterns, and the gray and yellow sets represent the normal and DoS operation, respectively. These last two sets determine the performance of the classifier for each technique and configuration during the classifier test.



**Fig. 5.** Example of k-fold with  $k = 3$

The well-known Area Under the Receiving Operating Characteristics Curve (AUC) measurement is used to assess each classifier configuration’s performance. The true positive and false positive rates are combined to provide the AUC, which provides an interesting idea of the classifier performance. From a statistical perspective, this number represents the possibility that a random positive event will be classified as positive [7]. Furthermore, AUC presents the advantage of not being sensitive to class distribution, particularly in one-class tasks, when compared to other measures like sensitivity, precision, or recall [3]. The times needed to train the classifier and to label a new test sample are also registered as a measure of the classifier goodness.

Furthermore, to seek the best classifier performance, the algorithms were trained with raw data, with a 0 to 1 normalization and with a Z-Score normaliza-



tion. Besides these three different configurations, several hyperparameters were swept for each technique.

*Autoencoders:* The number of neurons in the hidden layer was tested from 1 to 11, that is the number minimum dimensional reduction of the coding stage. Furthermore, the threshold is tested considering different anomalies percentage in the target set: 0%, 5%, 10%, 15%.

*K-Means:* The number of clusters in which the data is grouped was tested from 1 to 15. Furthermore, the threshold is tested considering different anomalies percentage in the target set: 0%, 5%, 10%, 15%.

*PCA:* The components tested from 1 to 11 are the number of features minus one. Furthermore, the threshold is tested considering different anomalies percentage in the target set: 0%, 5%, 10%, 15%.

## 5 Results

Table 1 summarizes the results achieved by Autoencoders classifiers. It is important to remark that the training times are greater than the ones obtained with K-Means and PCA. This fact is especially significant when the dataset is not preprocessed. However, the raw data leads to the best Autoencoder results, with 8 neurons in the hidden layer and considering outliers 15% of the training data.

**Table 1.** Results achieved by Autoencoders classifiers

Normalization	Hidden size	Outlier (%)	AUC (%)	train (s)	label ( $\mu s$ )
Raw	8	15	80.4	246.5	5.3
0 to 1	9	10	68.5	29.9	4.7
Zscore	5	0	76.0	17.5	3.6

The greatest AUC values correspond to a K-Means classifier with 13 clusters, with an AUC of 82.72% (Table 2). Significant differences are also noticeable in terms of training times between classifiers. They are derived from the number of clusters; a greater number of clusters results in a slower training process.

Finally, PCA results are shown in Table 3, with Zscore normalization, seven principal components, and 10 (%) of outlier fraction as the best configuration, although it does not overcome K-Means results.

**Table 2.** Results achieved by K-Means classifiers

Normalization	Clusters	Outlier (%)	AUC (%)	train (ms)	label ( $\mu s$ )
Raw	13	10	82.72	72.15	1.90
0 to 1	11	15	59.97	51.67	1.48
Zscore	5	0	50.01	20.65	1.04

**Table 3.** Results achieved by PCA classifiers

Normalization	Component	Outlier (%)	AUC (%)	train (ms)	label ( $\mu s$ )
Raw	8	15	67.09	77.57	1.76
0 to 1	8	10	66.42	65.71	1.17
Zscore	7	10	81.03	47.25	1.03

## 6 Conclusions and Future Works

The significance of security in IoT devices has grown in recent times, making it increasingly important to employ systems for knowing networks behavior in order to help identify and categorize emerging attack methods, particularly in industrial processes and communication protocols such as CoAP. Early detection of these attacks is essential to ensure the resilience of these processes.

In the current study, was employed a DoS CoAP dataset for developing a model based on the training data with the aim of minimizing the reconstruction error. As a result, instances belonging to the non-target class will yield high reconstruction errors, making it easier to detect outliers. Being, the best way the implementation of a K-Means classifier with 13 clusters.

Future research will explore the application of the set of methods studied in this paper to other CoAP datasets, which consist of Man-in-the-Middle and Cross-Protocol family attacks.

**Acknowledgments.** Álvaro Michelena’s research was supported by the Spanish Ministry of Universities (<https://www.universidades.gob.es/>), under the “Formación de Profesorado Universitario” grant with reference FPU21/00932.

Míriam Timiraos’s research was supported by the Xunta de Galicia (Regional Government of Galicia) through grants to industrial Ph.D. (<http://gain.xunta.gal>), under the Doutoramento Industrial 2022 grant with reference: 04<sub>I</sub>N606D<sub>2</sub>022<sub>2</sub>692965. CITIC, as a Research Center of the University System of Galicia, is funded by Consellería de Educación, Universidade e Formación Profesional of the Xunta de Galicia through the European Regional Development Fund (ERDF) and the Secretaría Xeral de Universidades (Ref. ED431G 2019/01).

## References

1. Basurto, N., Arroyo, A., Cambra, C., Herrero, A.: A hybrid machine learning system to impute and classify a component-based robot. *Log. J. IGPL* **31**(2), 338–351 (2022). <https://doi.org/10.1093/jigpal/jzac023>
2. Bormann, C., Castellani, A.P., Shelby, Z.: CoAP: an application protocol for billions of tiny internet nodes. *IEEE Internet Comput.* **16**(2), 62–67 (2012)
3. Bradley, A.P.: The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recogn.* **30**(7), 1145–1159 (1997)
4. López, V., et al.: Intelligent model for power cells state of charge forecasting in EV. *Processes* **10**(7), 1406 (2022)
5. Correia, N., Sacramento, D., Schutz, G.: Dynamic aggregation and scheduling in CoAP/observe-based wireless sensor networks. *IEEE Internet Things J.* **3**, 923–936 (2016)
6. Crespo Turrado, C., Sánchez Lasheras, F., Calvo-Rollé, J.L., Piñón-Pazos, A.J., de Cos Juez, F.J.: A new missing data imputation algorithm applied to electrical data loggers. *Sensors* **15**(12), 31069–31082 (2015)
7. Fawcett, T.: An introduction to roc analysis. *Pattern Recogn. Lett.* **27**(8), 861–874 (2006)
8. Fernandes, B., Silva, F., Alaiz-Moreton, H., Novais, P., Neves, J., Analide, C.: Long short-term memory networks for traffic flow forecasting: exploring input variables, time frames and multi-step approaches. *Informatica* **31**(4), 723–749 (2020)
9. Fernandez-Serantes, L., Casteleiro-Roca, J., Calvo-Rolle, J.: Hybrid intelligent system for a half-bridge converter control and soft switching ensurement. *Rev. Iberoamericana Autom. Inform. Industr.* (2022)
10. Gonzalez-Cava, J.M., et al.: Machine learning techniques for computer-based decision systems in the operating theatre: application to analgesia delivery. *Log. J. IGPL* **29**(2), 236–250 (2020). <https://doi.org/10.1093/jigpal/jzaa049>
11. Granjal, J., Silva, J.M., Lourenço, N.: Intrusion detection and prevention in CoAP wireless sensor networks using anomaly detection. *Sensors* **18**(8) (2018). <https://www.mdpi.com/1424-8220/18/8/2445>
12. Jove, E., et al.: Comparative study of one-class based anomaly detection techniques for a bicomponent mixing machine monitoring. *Cybern. Syst.* **51**(7), 649–667 (2020)
13. Jove, E., Casteleiro-Roca, J.L., Quintián, H., Zayas-Gato, F., Vercelli, G., Calvo-Rolle, J.L.: A one-class classifier based on a hybrid topology to detect faults in power cells. *Log. J. IGPL* **30**(4), 679–694 (2021). <https://doi.org/10.1093/jigpal/jzab011>
14. Jove, E., et al.: Hybrid intelligent model to predict the remifentanil infusion rate in patients under general anesthesia. *Log. J. IGPL* **29**(2), 193–206 (2021)
15. Khan, S.S., Madden, M.G.: One-class classification: taxonomy of study and review of techniques. *Knowl. Eng. Rev.* **29**(3), 345–374 (2014)
16. Kovatsch, M.: Github - mkovatsc/Copper4Cr: Copper (Cu) CoAP user-agent for chrome (javascript implementation) (2022). <https://github.com/mkovatsc/Copper4Cr>
17. Mali, A., Nimkar, A.: Security schemes for constrained application protocol in IoT: a precise survey. In: Thampi, S.M., Martínez Pérez, G., Westphall, C.B., Hu, J., Fan, C.I., Gómez Mármol, F. (eds.) *SSCC 2017. CCIS*, vol. 746, pp. 134–145. Springer, Singapore (2017). [https://doi.org/10.1007/978-981-10-6898-0\\_11](https://doi.org/10.1007/978-981-10-6898-0_11)

18. Mattsson, J.P., Selander, G., Amsüss, C.: Amplification attacks using the constrained application protocol (CoAP). Internet-Draft draft-irtf-t2trg-amplification-attacks-02, Internet Engineering Task Force (2023). <https://datatracker.ietf.org/doc/draft-irtf-t2trg-amplification-attacks/02/>. Work in Progress
19. lovelesh patel: Commits · automote/esp-coap · github (2021). <https://github.com/automote/ESP-CoAP/commits?author=lovelesh>
20. Porras, S., Jove, E., Baruque, B., Calvo-Rolle, J.L.: A comparative analysis of intelligent techniques to predict energy generated by a small wind turbine from atmospheric variables. *Log. J. IGPL* **31**, 648–663 (2022). <https://doi.org/10.1093/jigpal/jzac031>
21. Quintian Pardo, H., Calvo Rolle, J.L., Fontenla Romero, O.: Application of a low cost commercial robot in tasks of tracking of objects. *Dyna* **79**(175), 24–33 (2012)
22. Radoglou Grammatikis, P.I., Sarigiannidis, P.G., Moscholios, I.D.: Securing the internet of things: challenges, threats and solutions. *Internet Things* **5**, 41–70 (2019). <https://www.sciencedirect.com/science/article/pii/S2542660518301161>
23. Rahman, R.A., Shah, B.: Security analysis of IoT protocols: a focus in CoAP. In: 2016 3rd MEC International Conference on Big Data and Smart City (ICBDSC), pp. 1–7 (2016)
24. Rodríguez, E., et al.: Transfer-learning-based intrusion detection framework in IoT networks. *Sensors* **22**(15), 5621 (2022). <https://dx.doi.org/10.3390/s22155621>
25. Shelby, Z., Hartke, K., Bormann, C.: The constrained application protocol (CoAP). RFC 7252 (2014). <https://www.rfc-editor.org/info/rfc7252>
26. Simić, S., et al.: A three-stage hybrid clustering system for diagnosing children with primary headache disorder. *Log. J. IGPL* **31**(2), 300–313 (2022). <https://doi.org/10.1093/jigpal/jzac020>
27. Simić, S., Simić, S.D., Banković, Z., Ivkov-Simić, M., Villar, J.R., Simić, D.: Deep convolutional neural networks on automatic classification for skin tumour images. *Logic J. IGPL* **30**(4), 649–663 (2021). <https://doi.org/10.1093/jigpal/jzab009>
28. Tax, D.M.J.: One-class classification: concept-learning in the absence of counter-examples [Ph. D. thesis]. Delft University of Technology (2001)
29. Thomas, D.R., Clayton, R., Beresford, A.R.: 1000 days of UDP amplification DDoS attacks. eCrime researchers Summit, eCrime, pp. 79–84 (2017)
30. Zayas-Gato, F., et al.: Intelligent model for active power prediction of a small wind turbine. *Log. J. IGPL* **31**, 785–803 (2022). <https://doi.org/10.1093/jigpal/jzac040>
31. Zayas-Gato, F., et al.: A novel method for anomaly detection using beta Hebbian learning and principal component analysis. *Log. J. IGPL* **31**(2), 390–399 (2022). <https://doi.org/10.1093/jigpal/jzac026>