Intelligent Systems with Applications 17 (2023) 200189

Contents lists available at ScienceDirect

Intelligent Systems with Applications

journal homepage: www.journals.elsevier.com/intelligent-systems-with-applications

Engineering the application of machine learning in an IDS based on IoT traffic flow

Nuno Prazeres [a], Rogério Luís de C. Costa [b,*], Leonel Santos [a,b], Carlos Rabadão [a,b]

[a] School of Technology and Management (ESTG), Polytechnic of Leiria, Leiria, 2411-901, Portugal
[b] Computer Science and Communication Research Centre (CIIC), Polytechnic of Leiria, Leiria, 2411-901, Portugal

ARTICLE INFO

Keywords:
Intrusion detection systems
Internet of things
Machine learning
Smart city
Cybersecurity

ABSTRACT

Internet of Things (IoT) devices are now widely used, enabling intelligent services that, in association with new communication technologies like the 5G and broadband internet, boost smart-city environments. Despite their limited resources, IoT devices collect and share large amounts of data and are connected to the internet, becoming an attractive target for malicious actors.
This work uses machine learning combined with an Intrusion Detection System (IDS) to detect possible attacks. Due to the limitations of IoT devices and low latency services, the IDS must have a specialized architecture. Furthermore, although machine learning-based solutions have high potential, there are still challenges related to training and generalization, which may impose constraints on the architecture.
Our proposal is an IDS with a distributed architecture that relies on Fog computing to run specialized modules and use deep neural networks to identify malicious traffic inside IoT data flows. We compare our IoT-Flow IDS with three other architectures. We assess model generalization using test data from different datasets and evaluate their performance in terms of Recall, Precision, and F1-Score. Results confirm the feasibility of flow-based anomaly detection and the importance of network traffic segmentation and specialized models in the AI-based IDS for IoT.

1. Introduction

The Internet of Things (IoT) paradigm is one of the drivers for a new generation of communication networks, combining a wide variety of hardware and software that provide customers easy-to-use experience and low-cost solutions. IoT devices are present in our cities, homes, industry, healthcare facilities, vehicles, and personal gadgets and can perform several critical tasks and make our lives more comfortable (Figueiredo et al., 2022, Tewari & Gupta, 2017). IoT networks are made up of a large number of devices and sensors that collect and share large volumes of data, including confidential and private ones (Neisse et al., 2014, Tewari & Gupta, 2020). The use of IoT devices has also boosted the creation of smart environments, such as smart cities. In these contexts, they provide a wide variety of services, increasing the well-being of the population and the conscious use of resources (Figueiredo et al., 2022).

Despite handling and generating a large volume of data, IoT devices are generally cheap and have low CPU capacity, low storage, and low memory resources. IoT devices may be vulnerable to attacks if there are no security measures, producing unexpected behaviors in the private networks, compromising services availability, data confidentiality, and the user's privacy (Butun et al., 2019, Hromada et al., 2021). Hence, the IoT ecosystem is a potential target for cybercriminals and requires novel solutions to deal with data protection and cybersecurity threats (Hromada et al., 2021, Tsimenidis et al., 2022).

Intrusion Detection Systems (IDS) are key security solutions for networks as they may detect non-authorized accesses and attacks against systems through the analyses of network communications and internal activities (Moustafa et al., 2019b). But the traditional knowledge-based IDS must be replaced by intelligent and data-driven solutions (Tsimenidis et al., 2022). Artificial Intelligence (AI) and machine learning methods have been used in the last years to analyze large volumes of data in diverse environments, correlating events, identifying patterns, and detecting anomalous behavior that otherwise would remain hidden (Berman et al., 2019). In this work, we deal with the use of machine

* Corresponding author.
E-mail addresses: 2192642@my.ipleiria.pt (N. Prazeres), rogerio.l.costa@ipleiria.pt (R.L.C. Costa), leonel.santos@ipleiria.pt (L. Santos), carlos.rabadao@ipleiria.pt (C. Rabadão).

https://doi.org/10.1016/j.iswa.2023.200189
Received 31 July 2022; Received in revised form 13 December 2022; Accepted 19 January 2023
Available online 24 January 2023

learning methods in an Intrusion Detection System (IDS) for the IoT environment of smart cities.

The development of an AI-driven IDS for IoT environments must consider several aspects, like using appropriate anomaly and intrusion detection mechanism, IDS placement and scalability, and the use of representative datasets to validate proposals (Ahmad & Alsmadi, 2021, Thakkar & Lohiya, 2021). Also, smart cities are complex environments (e.g., in terms of heterogeneity, number of nodes, and accessibility to providers' infrastructure) with very great volume of heterogeneous data and whose equipment are especially vulnerable to attacks as they are easily accessed in public spaces (Garcia-Font et al., 2018). These are some of the open issues we deal with in this paper.

Our IoT-Flow IDS operates on information and statics about traffic, using traffic flows. Flow records are information about a specific flow observed at an observation point (Claise et al., 2013, Velan, 2018), which include flow keys, such as characteristic properties of a flow (e.g., IP addresses and port numbers), and measured properties (e.g., packet and byte counters). Using flows to identify network traffic anomalies is a more scalable and interoperable approach than analyzing packets payload (Santos et al., 2021). Hence, our IDS must aggregate traffic data into flow information before looking for anomalous traffic. Only then, pre-processing steps somewhat usual in the data science pipeline (e.g., encoding and feature selection) take place. After that, a classifier (i.e., machine learning model) analyzes the transformed flows looking for anomalous traffic.

Another open challenge is IoT IDS placement as the performance of an IDS depends on the network's topology and available resources (Thakkar & Lohiya, 2021). IoT devices commonly have processing limitations. Also, in smart cities, several services have Quality-of-Service requirements related to availability and integrity. Therefore, considering environment limitations and application requirements, we use a distributed solution that uses traffic segmentation and relies on Fog computing. Fog computing is the extension of cloud computing towards the network edge to enable cloud-things service continuum (Diro & Chilamkurti, 2018). This paradigm reduces latency and energy consumption for the heterogeneous communication approaches in the smart cities applications (Singh et al., 2020). We place specialized components of our IoT IDS in Fog Nodes, which are network nodes with all the fog computing characteristics.

In machine learning (specially in Supervised Learning), a key challenge is the availability of labeled datasets representing the analyzed phenomenon. Also, the performance of deep models is highly dependent on data quality and training data size (Jiang et al., 2021). Although there are several works on using machine learning and other intrusion detection methods for IoT, most of the current evaluations do not use data on real IoT traffic. Instead, they use more general network traffic datasets like UNSW-NB15, NSL-KDD, and KDD99, which are not suitable for evaluating the performance of an IDS in IoT networks (Ashraf et al., 2021, Thakkar & Lohiya, 2021). In this work, we use two IoT traffic data datasets: the IoT-23 dataset (Garcia et al., 2020), which contains benign and malign traffic from real-world IoT equipment, and the MQTT-IoT-IDS2020 (Hindy et al., 2020b), that has data generated by a simulated MQTT network.

Also, many works claim the effectiveness of the features they use, but very few evaluate the robustness of selected features (Liu et al., 2021). We compare the performance of the optimal set of features with the one achieved using the features the datasets share. It is a relevant evaluation for the industry, as we show that near-optimal results may be achieved even though we use a considerably smaller set of features, thus leading to lower resource consumption. To avoid overfitting while training our models, we use a regularization technique (i.e., dropout) and early stopping. We show our proposal outperforms the results from the literature.

On the other hand, the heterogeneity of IoT devices, increases the difficult of applying machine-learning to anomaly detection (Mothukuri et al., 2022). We experimentally show the lack of generalization of our models by using the model trained with data from a dataset to identify anomalies in the other one. We also build a new set that combines data from IoT-23 and MQTT-IoT-IDS2020 and use such data to train new models. We show that the obtained model achieves worse performance than the specialized ones.

Therefore, the main contributions of this work include (i) the proposal of an architecture (including components, placement, and flow-based methods) for the machine learning-based IoT IDS for smart cities that considers the limitations of IoT devices and the complex environment of smart city services; (ii) an experimental evaluation using four scenarios; and (iii) the assessment of generalization and specialization of trained deep models.

In the following section, we review some background and related work. In Section 3, we describe the architecture of our IoT-Flow IDS. Then, Section 4 contains experimental results. Finally, Section 5 presents our conclusions and future works.

2. Background and related work

In the last few years, Internet of Things (IoT) devices turned into part of day-to-day technologies that enable a wide range of applications in smart environments. Currently, the use of IoT devices and cyber-enabled resources in machine-to-machine and human-to-machine interactions generate large amounts of data (Berman et al., 2019), including confidential, private, and high-value data. Also, in smart cities, IoT devices support a myriad of interconnected services, which increases the risk of cyberattacks (Figueiredo et al., 2022). Cybersecurity systems can be used together with machine learning to take advantage of its characteristics to develop increasingly robust attack detection methods and solutions.

2.1. Smart cities and IoT attacks

In the last decade, cities started to provide a wide range of information to their citizens based on IoT data, including car parking availability, transportation routes or schedules, traffic congestion. Also, they started to use IoT devices in critical services, like environmental disasters detection, energy and water supply management, and street surveillance, among others (Figueiredo et al., 2022). Smart cities became one of the major drivers for the IoT applications (Singh et al., 2020). All to prevent the waste of resources and provide a better quality of life to citizens in a controlled and secure environment. Smart cities usually share the IoT network architecture. Fig. 1 presents one of the most common representations of such architecture, composed of the perception, network, support layer, and application layers (Cui et al., 2018). The perception layer is the source of the data. It includes the sensors, the meters, and all the IoT heterogeneous hardware deployed across the city to provide services. The technologies that enable the communication between devices and the remaining layers are in the network layer, like Wi-Fi, 5G, Bluetooth, RFID, and Ethernet. The support layer provides services to the smart city context and its applications. In this layer, the network command center can be deployed and perform tasks like network monitoring and anomaly detection. The application layer is the top one and provides the end-user information and services regarding the collected data in the perception layer.

The most common concerns related to IoT security include the identification and authentication processes, the maintenance of data availability, integrity, confidentiality and privacy, and trust and access control (Hromada et al., 2021).

Each layer of the IoT architecture has its own vulnerabilities and is subject to different attacks (Butun et al., 2019). Some of the most frequent attacks include Denial of Service (DoS), botnets, brute force attacks, and ransomware.

DoS is an attempt by an attacker to prevent legitimate access to websites by overwhelming the amount of available bandwidth or resources

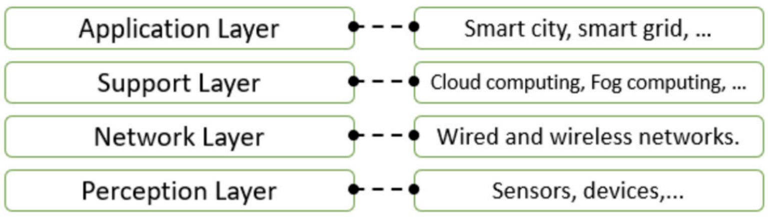
Application Layer
Smart city, smart grid, ...
Support Layer
Cloud computing, Fog computing, ...
Network Layer
Wired and wireless networks.
Perception Layer
Sensors, devices,...

Fig. 1. IoT based architecture for a smart city.

of the computer system (Moustafa et al., 2019a). This type of attack typically evolves to a Distributed Denial of Service (DDoS) when different types of equipment produce this type of attack against a specific target.

A botnet denotes the number of hijacked computer systems remotely operated by one or many malicious actors which coordinate their activities by Command and Control (C&C). These botnets are responsible for DoS or DDoS attacks where the owner of a hijacked device isn't aware of being part of an attack on a computer system, denoting only some slow processing of its device.

Brute force endeavors to illegally obtain pairs of user names and passwords by trying all predefined pairs to gain access to network services, with automated applications often used to guess password combinations. When this type of attack succeeds, the attacker gains access to all the data processed by the device. This device can be transformed into a bot or perform DoS attacks on other devices or computer systems. Ransomware is malware that harms computer and network systems by encrypting computer resources and blocking access till the victim pays a ransom (Moustafa et al., 2019a). It may result from a brute force attack and compromise service and data availability.

2.2. Intrusion detection systems

An Intrusion Detection System (IDS) is a solution to detect unauthorized accesses and attacks in network systems. It aims to detect any anomaly or attack in real-time and uses the network traffic as its data source. These systems work with different detection methods and must be strategically placed in the network (Thakkar & Lohiya, 2021).

The IDS may rely on multiple sensors that collect information about the devices installed inside the network and oversee the network communications. The IDS sensors placement may be host-based, network-based, or in a hybrid approach. The host-based approach is centered on a device, having a real-time perception of what is happening on that node and of the node's network communications. This approach depends highly on the device's processing capabilities. The network-based architecture is usually deployed in the network gateways. It has a broader view of the network as it converges multiple hosts, thus dealing with more data. The hybrid strategy tries to take advantage of the best of the host and network-based approaches.

In addition to the IDS placement, it is necessary to adjust the type of detection that is performed. There are several detection techniques related to IDSs, such as signature-based, anomaly-based, specification-based, and hybrid. The signature-based or the specification-based IDSs have preconceived data to look for on the network, and when a match occurs, it raises an alert. This strategy produces a small number of alerts. On the other hand, the IDS would not detect new threats or attacks not listed on its database. An anomaly-based IDS detects abnormal behaviors or anomalies by comparing network traffic with expected communications or usual behavior. This strategy leads to a higher number of alerts when compared with the signature and specification-based approaches, but it may detect new threats and attacks on time. A hybrid strategy may combine more than one technique, trying to detect new attacks or threats but raising a small number of alerts.

2.3. Machine learning and performance metrics

In the last decades, Machine Learning techniques have been used in several contexts where there is a need to process and analyze large volumes of data, spotting patterns, behaviors, and anomalies inside the datasets. There are two types of machine learning tasks: classification and prediction. In this work, our models classify network traffic flows. In classification tasks, the model tries to identify rules from the sample data and predicts the belonging of new elements (objects, individuals, and criteria) to a given class (Hussain et al., 2020).

In Supervised Learning, the model learns from labeled data, which means that training data includes both the input and the desired results (Chaabouni et al., 2019). A key challenge when using Supervised Learning is to get large labeled datasets representing the analyzed phenomenon.

Artificial neural networks (ANNs) are generic algorithms mimicking the biological functioning of a brain without being intended for a specific task (Chaabouni et al., 2019). A Multilayer Perceptron (MLP) is one of the simpler ANN. A perceptron is a device capable of computing all predicates which are linear in some given set ϕ of partial predicates (Minsky & Papert, 1969). In other words, it is a simple algorithm intended to perform binary classification. The MLP has an input layer that receives data, an output layer that outputs the decision or prediction about the input, and between those two, an arbitrary number of perceptrons or hidden layers that are the computational engine of the MLP. In Deep Learning (DL), ANN learns to represent the data as a nested hierarchy of concepts within the layers of the neural network (Chalapathy & Chawla, 2019). An advantage of DL over traditional machine learning is its superior performance in large datasets (Al-Garadi et al., 2020). Using adequate metrics is of major importance when assessing model performance. The most suitable indicator depends on the problem of interest. For instance, in this work, we use machine learning to classify network flows into normal or malicious. A binary classification problem has four possible outcomes. The correctly predicted negatives are the *true negatives* (TN), and the correctly predicted positives are *true positives* (TP). The incorrectly predicted negatives are the *false negatives* (FN). Finally, the incorrectly predicted positives are called *false positives* (FP).

One of the commonly used metrics is *precision*, which measures how accurate the classification model is. Precision is defined on the number of correctly classified elements, as represented in Equation (1).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

In practice, misclassifications may have different importance. For instance, classifying a malicious flow as a normal one may be more prejudicial than identifying a normal flow as a malicious flow. The TP rate (or recall) depends on the number of true positives and false negatives is defined by Equation (2).

$$Recall = TPR = \frac{TP}{TP + FN} \tag{2}$$

Models with high precision and recall values are highly dependable, as they do not misclassify benign flows and do not wrongly leave out malicious flows. On the other hand, models that achieve high precision values but low recalls miss out on many malicious flows. Therefore, these models should perform critical tasks. Lastly, models with high recall and small precision values would detect most of the malicious flows but also raise many false alarms, which can create entropy in the security system.

The *F1 Score* (or *F-score*) is useful to evaluate the performance of models on unbalanced datasets. It is the harmonic mean of the precision and recall, as defined in Equation (3).

$$F1\ Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

2.4. Related work

There are several cybersecurity-related challenges in the IoT environment. One of the possible solutions refers to protecting communications. In (Tewari & Gupta, 2017, 2020), authors present authentication methods which may be used to ensure security of IoT communications. Stergiou et al. (2020) present encryption algorithms for securing communications when using IoT with cloud computing in the field of telecommunications. Our work deals with using an AI-driven IDS to identify anomalous traffic in IoT networks, specially in the smart cities context.

Cui et al. (2018) describe the four-layered IoT-based architecture for smart cities and identify machine learning as a technique that would improve traditional intrusion detection systems in protecting the network. The work does not present any actual implementation details and does not even discuss the implementation of the IoT IDS with machine learning. Butun et al. (2019) describe the most common vulnerabilities of each layer and the corresponding attacks and countermeasures.

Elrawy et al. (2018) state that many IoT applications may run in real-time and that network delay and latency would affect their performance. The work refers to the need for robust IoT network security measures in environments like e-health systems, as attacks may become life-threatening in such environments. Authors state that the security solution must at the same time protect the IoT network and its resources without impacting the system performance or user privacy. The work describes the IDS operation as a 3-stage process. The first stage is a monitoring phase that relies on sensors that could be network or host-based. Then, there is the analysis phase, in which feature extraction or pattern identification methods are performed. The third stage is the detection one, in which the system detects anomalies or misuses. The authors also highlight the importance of choosing the adequate placement for the IDS in an IoT network cause this matter would affect the overall IDS efficiency. There is no practical implementation. Indeed, the selection of features remains an open issue in the use of deep learning for intrusions detection (Liu et al., 2021).

Zeadally and Tsikerdekis (2020) discuss the use of traditional network monitoring (like Intrusion Detection Systems) with the help of machine learning algorithms to give a viable alternative to existing IoT security solutions. They summarize the needs of host and network-based approaches to perform network traffic capture using machine learning to process the data. In this case, no option is given as optimal, highlighting only the strength and limitations of the machine learning algorithms regarding the IoT devices' characteristics.

Chaabouni et al. (2019) present a comprehensive survey pointing out the design challenges of IoT security and classification of IoT threats. As future research directions, the authors state that exploring the edge and fog computing paradigms would give the ability to push the intelligence and processing logic employment down near to data sources. Also, they identify that to train and deploy an IoT IDS based on machine learning, a real-world IoT-dedicated dataset is required.

Diro and Chilamkurti (2018) present a distributed attack detection scheme using a deep learning approach for IoT and place their IDS based on deep learning in the fog network. The fog nodes are responsible for training models and hosting attack detection systems at the edge of the distributed fog network since they are closer to the IoT data layer. A central node updates the parameters of each cooperative node and propagates the resulting update back to the worker nodes. No IoT-based dataset was used for model training. (Ling & Hao, 2022) present an algorithm for feature selection based on artificial immune with co-operative evolution of multiple operators. Authors evaluate their work using KDD99 and UNSW-NB15 datasets, which are not oriented for evaluating the performance of an IDS in IoT networks (Thakkar & Lohiya, 2021).

Ariyaluran Habeeb et al. (2019) highlight that adopting a real-time architecture for the smart city would assure effective and seamless communication among sensing devices within the smart city infrastructure. It also includes the quality of services support in the network, which is crucial for the real-time application for smart cities. Due to big data production in several environments, Habeeb et al. investigated real-time big data processing and machine learning with the possibility of detection anomalies. Li et al. (2022) discuss the use of machine learning to identify false alerts generated by IDS.

Austin (2021) used the IoT-23 dataset in his trials to answer which learning model that performs the best in terms of classification accuracy, recall, precision, F1 score and to discover which features have the best predictive power in the dataset. Austin (2021) described how the dataset was collected, the types of attacks used in the network, and how the learning approach helps to classify traffic in the dataset.

Mothukuri et al. (2022) use federated training rounds and gated recurrent units (GRUs) in anomaly detection for IoT. They focus on applications of industry domain and use a dataset built with Modbus to evaluate their proposals.

Ahmad and Alsmadi (2021) and Thakkar and Lohiya (2021) present recent surveys on machine learning-based security solutions and IDS for IoT. These works identify some open challenges, which include choosing the appropriate intrusion detection strategy and IDS placement strategies, the lack of scalable solutions, using validation datasets based on IoT network data, and providing IDS solutions that deal with the diverse types of existing IoT technologies.

These are some of the open challenges we deal with in this work. We describe a realistic smart city scenario, discussing IDS components and placement and network segmentation to deal with distinct application systems. Our AI-driven IDS operates over flow information extracted from captured network traffic. To assess the use of learning techniques in such a context, we experimentally evaluate deep models over two IoT traffic flow datasets, assessing the performance and generalization of models.

3. IoT-flow: machine learning over flow data IDS for IoT

Some of the current challenges related to deploying an AI-driven IDS for IoT environments are the anomaly and intrusion detection mechanism, IDS placement and scalability, and the use of representative datasets to validate proposals (Ahmad & Alsmadi, 2021, Thakkar & Lohiya, 2021). Also, smart cities are complex environments with very great volume of heterogeneous data (Garcia-Font et al., 2018). This section describes our IoT-Flow IDS that addresses the complexity of smart cities' environments through a distributed architecture which is scalable and efficiently deals with the diverse types of IoT technologies.

Our IDS operates on traffic flow information. A flow-based IDS analyzes traffic information and statistics rather than packets payload. Using packet flows in IDS for IoT can make these solutions more scalable and interoperable (Santos et al., 2021). However, this solution requires flow aggregation and export activities. Usually, the transformation of traffic data into flows is conducted by a process that consists of several steps (Sperotto et al., 2010). The first step is *Packet Observation* and consists in the process of capturing packets from the line and pre-processing them. Then, the *Flow Metering & Export* step is where packets are aggregated into flows and flow records are exported to a collector that receive, store and pre-process data from the flow exporters assuring the *Data Collection* step. Finally, *Data Analysis* is the final step and there are three main areas where analyses of flow data can be applied

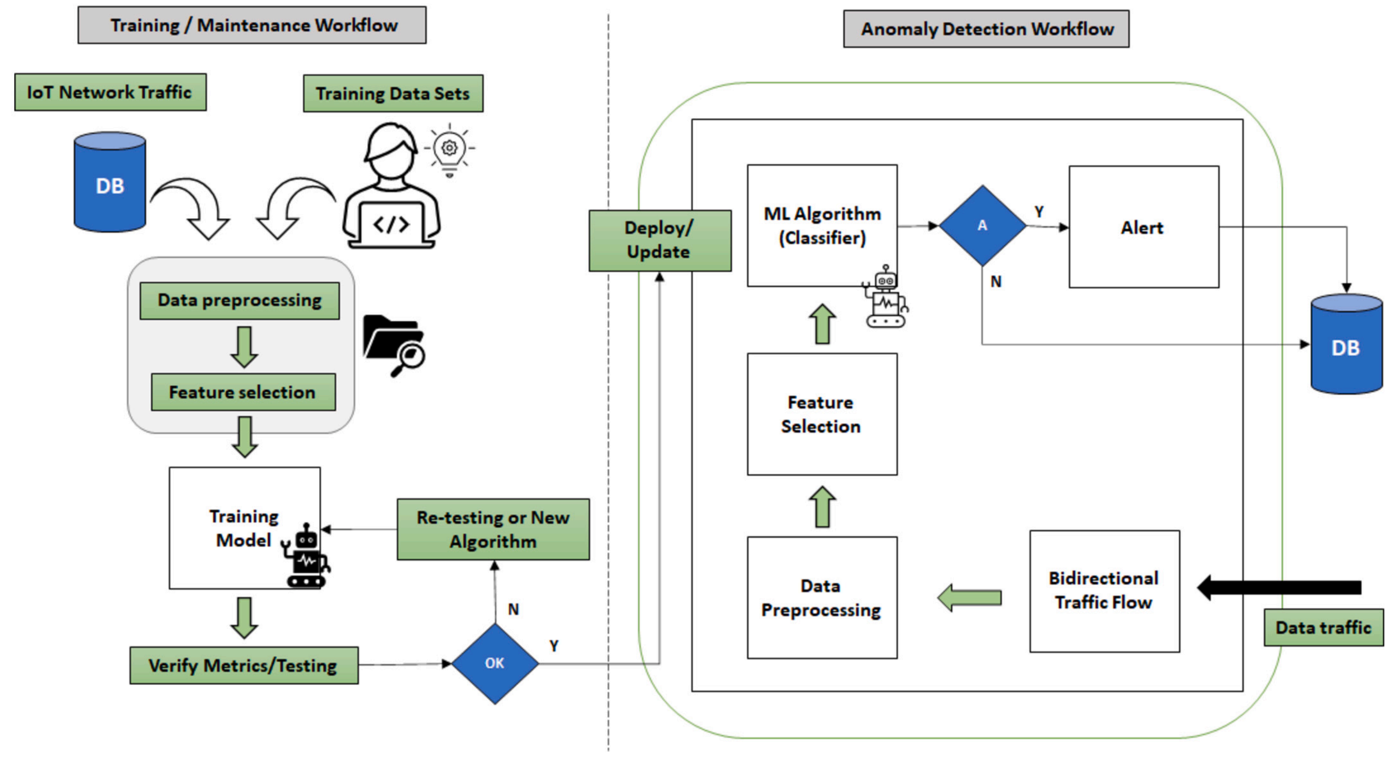

Fig. 2. Anomaly Detection Workflow Overview.

(Hofstede et al., 2014): (a) Flow analysis & reporting; (b) Threat detection; and (c) Performance monitoring.

3.1. *Workflow overview*

Fig. 2 presents the main steps in the flow of our solution in the model training and production environment. On its left side, Fig. 2 represents the steps related to model selection and training. In this context, models are trained and tuned using datasets representative of data flows with characteristics of the IoT environment of smart cities. The models with the highest combined value of precision and recall, in terms of performance metrics, are deployed in the production environment. On the right side of Fig. 2, there is the anomaly detection workflow with the various steps required in the production environment.

Sensors are responsible for capturing data from packets transmitted in the network. This is similar to the first step of the above-described transformation of traffic data into flows. The sensors are distributed in the perception layer as discussed in Section 3.2.

The next step in the transformation of traffic data into flows is the aggregation of packets into flows, which is represented in the *Bidirectional Traffic Flow* step in Fig. 2. Traffic flows within the communication networks have a large series of statistics and characteristics, which may be standardized. IPFIX (Claise, 2008, Claise et al., 2008, Claise & Trammell, 2013) stands for IP Flow Information Export and is an IETF protocol that was born due to the need of having a common universal standard for exporting IP flow information from network devices and probes that facilitates services such measurement, accounting, and billing. This protocol is used to transport information elements (IE) that allow network administrators to have a broader view of the traffic that flows in and out of the network. IPFIX can be deployed across network elements like routers where it performs passive flow measurements (Zseby et al., 2006). The IEs are grouped into 12 groups according to their semantics and their applicability (Claise et al., 2008, Claise & Trammell, 2013):

1. Identifiers
2. Metering and Exporting Process Configuration
3. Metering and Exporting Process Statistics
4. IP Header Fields
5. Transport Header Fields
6. Sub-IP Header Fields
7. Derived Packet Properties
8. Min/Max Flow Properties
9. Flow Timestamps
10. Per-Flow Counters
11. Miscellaneous Flow Properties
12. Padding

Examples of this information include source and destination IP addresses and ports, the number and total size of transmitted packets, and the protocol used. With IPFIX IEs a network admin may answer questions like *who is originating a network flow? Which devices are intervening in this communication? What application is generating this flow? Where did this flow take place? How do a flow is working and what are its characteristics?*

Access to IPFIX IEs can be made by using a flowmeter installed on the network sensors. This software generates the flows containing information about every connection of the observed traffic. Yet Another Flowmeter (YAF) (Inacio & Trammell, 2010) is an open-source software tool that can be used for this task since it captures network packages or reads files from network captures, transforming and exporting them into IPFIX flows.

After transforming traffic data into traffic flows, data pre-processing begins, with tasks such as encoding and filling in missing values. Categorical features (strings) are encoded with numerical values and replaced with zero if received as null or blanks.

The next step comprises selecting the features relevant to anomaly identification, which the machine learning model (classifier) performs. The feature selection is accomplished by identifying the features that have a major effect on the target attribute. In our implementation, we use the ExtraTreesClassifier Pedregosa et al. (2011) algorithm, as described in Section 4.

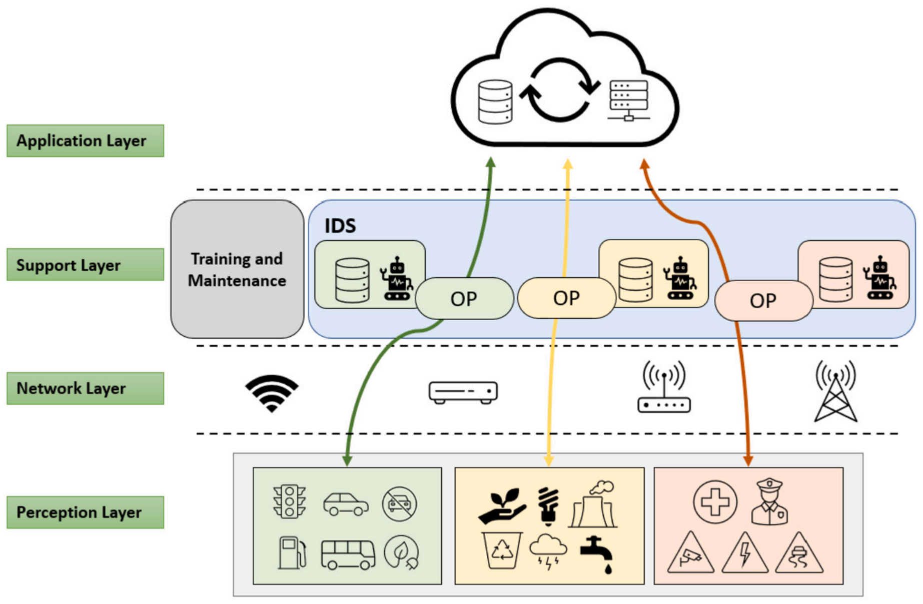

Fig. 3. IoT-Flow IDS architecture proposal.

The result will be the classification of flows into normal or anomalous, in which case the system issues an alert. But in both cases, selected flows are stored in a database for future analysis, in case of an attack or to check for relevant changes in the patterns of network communications, and will be used as input for training new models. The flow data stored at the database may also be used to support other activities, like forensic analyses, for instance, as it may provide evidence on the source and timing of anomalous behavior in network communications.

3.2. IoT-flow IDS distributed architecture

To reach the full potential of an IoT IDS based on machine learning, one should look for a computational solution that guarantees the required resources for data processing. Such a solution should deal with the processing limitations of IoT devices and be aware of the requirements of the services they support. Our solution relies on the use of Fog computing. Due to its edge location, it may provide network context information to fog applications, such as local network conditions, traffic statistics, and client status information (Sucharitha et al., 2019). Due to its location awareness, fog computing allows the creation of network nodes (i.e., Fog Nodes) that offer local network services and resources. Each fog node has all the characteristics associated with fog computing.

Also, to place a cybersecurity solution in the smart city IoT environment, one must consider the layers that represent the architecture of the smart city and the existence of distinct requirements due to the variety of services. Fig. 3 presents the distributed architecture of our solution.

Starting in the perception layer, where we place the sensors, we segment the IoT network traffic flow, which may be based on requirements and criticality of services or by the application messaging protocols (e.g., MQTT or CoAP). Satisfying the different needs of each service would be easier with network segmentation. But the segmentation would also allow us to create observation points (OP) of each segment or service, making it simpler to perform network analysis and understand network behaviors. The network layer has the necessary technology to transport data to the support layer. Whether through Wi-Fi access points, Ethernet, or 4G/5G radio links. Devices' gateways would be set to the support layer node that provides services before the data reaches the application layer. In the support layer, we assume that we will have an infrastructure based on fog computing, capable of providing all the necessary processing and storage capacity resources. It would be the layer through which all the traffic generated by the perception layer or originating from the application layer would pass.

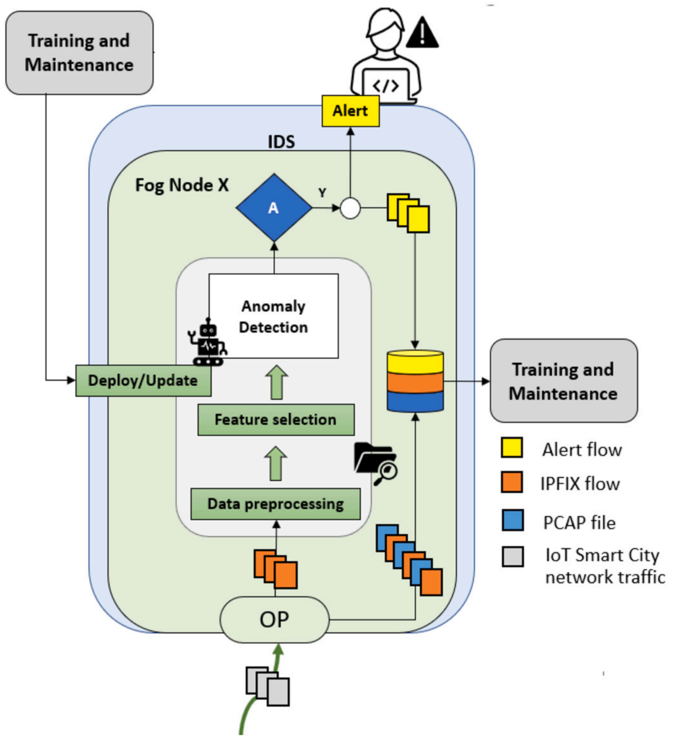

Fig. 4. Fog Node with Learning-based IDS.

3.3. Specialized components

Fog nodes, like the one in Fig. 4, are the network elements responsible for observing, storing, transforming, classifying, and forwarding the data generated by the IoT network. The network traffic will be transformed into standard (e.g., IPFIX) flows that go through the preprocessing module before being delivered to the model that classifies the flow. Regardless of the classification result, it will be saved together with the flows and packets that originated it. Likewise, the flow forwarding to or from the application layer is guaranteed since it is just a detection system.

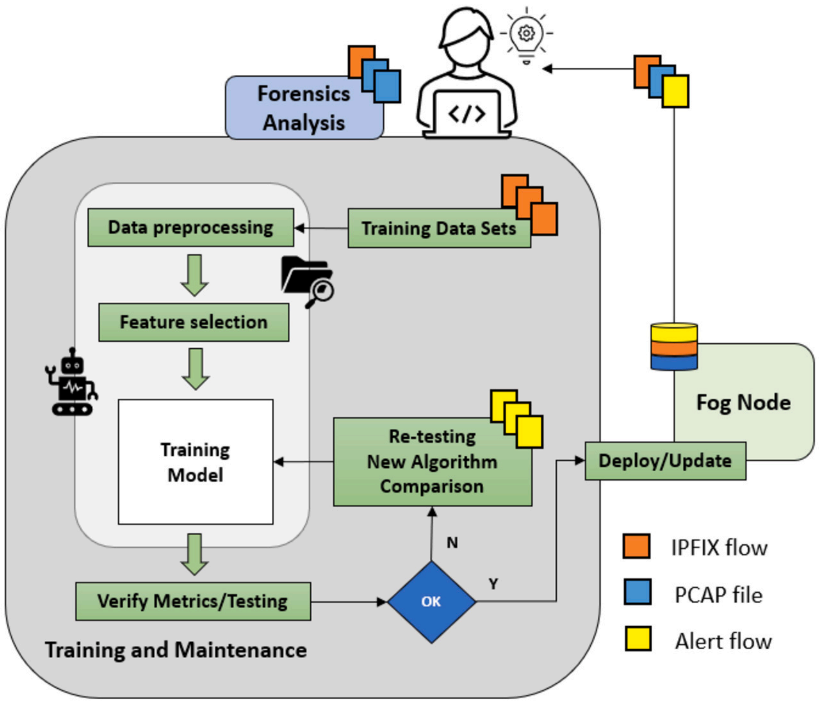

Fig. 5. Support layer Training/Maintenance Module.

When using supervised learning, one must train the learning module to distinguish the usual behavior of the network from anomalous behavior. Model training may use datasets built and labeled in controlled laboratory environments or through pre-production networks that use protocols or equipment similar to those in the smart city. Hence, our architecture comprises the Training and Maintenance module described in Fig. 5, which has a close working process with the IDS fog node.

In this module, the machine learning models with the best performance are identified and later implemented in the production IDS module. In the production environment, a database stores the captured network traffic (PCAP files), their flow representation (e.g., in IPFIX), and the classification of the flows. The Training and Maintenance module contains a replica of the production database and uses it in model training. Hence, the training module feeds the production IDS (with newly trained models), and production IDS nodes feedback the training module with real-world (and labeled) data. Such real-world labelled data may be obtained from historical data, and the traffic previously identified as malicious may be validated as result from network auditing and forensic analysis.

4. Experimental evaluation

To experimentally evaluate our proposals, we use two datasets with IoT network flows, namely IoT-23 (Garcia et al., 2020) and MQTT-IoT-IDS2020 (Hindy et al., 2020b). We used Python (Van Rossum & Drake, 2009), Jupyter Notebooks (Kluyver et al., 2016), and Tensorflow to train and test several deep models. We evaluated three architectures, i.e., our IoT-Flow IDS with segmented traffic flow information, a merged flow architecture, and a federated learning approach.

4.1. Evaluated scenarios

We evaluated four scenarios on the use of flow data for anomaly detection: IoT-Flow IDS, Cross-flow, Merged flow and Flow-based federated learning.

IoT-Flow IDS - the first scenario simulates our proposals with network traffic segmentation. We used the IoT-23 and MQTT-IoT-IDS2020 datasets to evaluate our proposals. Each dataset represents an application system. We applied separate IDS training for each dataset and assessed the IDS performance using test data from the corresponding system. Fig. 6 represents the evaluation scenario.

Cross-flow - in this scenario, we applied separate IDS training for each dataset and assessed the IDS performance using test data from other dataset, thus representing the use of the models to detect attacks in systems other than the ones on which they were trained. Fig. 7 represents the Cross-flow evaluation scenario.

Merged flow - this scenario considers the merge of flows from distinct systems and its use for training of a single IDS, as represented in Fig. 8. Such system would be used to identify anomalous traffic over the merged traffic data.

Flow-based federated learning - we also considered a federated learning approach, on which each client's model is trained considering distinct datasets. Then, a new server model is built by the combination (average) of the parameters of trained client models, as represented in Fig. 9. Test data of both datasets is used to evaluate the server model.

4.2. Datasets description

Both used datasets contain IoT network traffic data. But they have a distinct set of features.

IoT-23 - The IoT-23 dataset results from the Malware Capture Facility Project from the Czech Technical University ATG Group and contains normal and malicious traffic. Real hardware (not simulated), including a smart door lock (Somfy), a smart LED lamp (Philips), and a home intelligent personal assistant (from Amazon), is used to produce benign traffic. Network attacks are mainly based on known botnets like Mirai and in trojan software that helps the malicious actors take over the equipment remotely. A passive open-source network traffic analyzer called Zeek captured the flows, whose main data structure is a connection that follows typical flow identification mechanisms that correlate with IPFIX. IoT-23 contains 23 features of binary flows, including origin and destination addresses and ports, protocol and service type, and counts of bytes from source and destination. Table 1 presents the dataset features and the corresponding IPFIX elements.

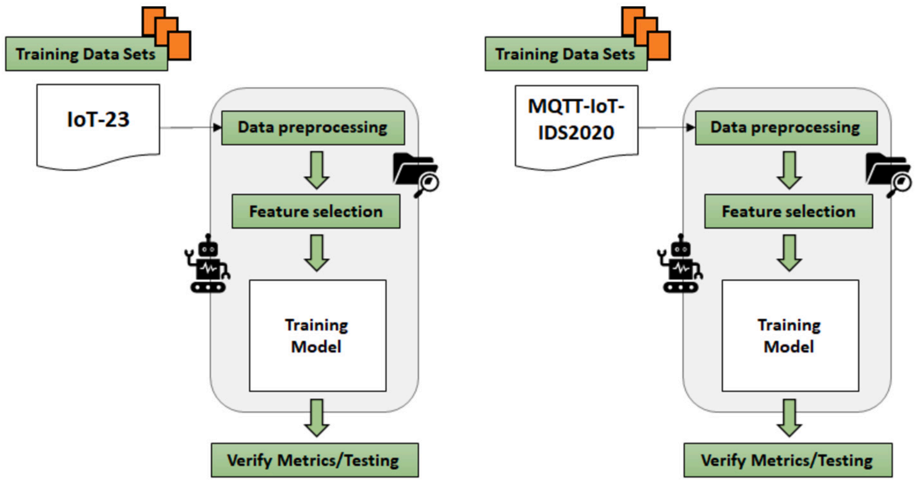

Fig. 6. IoT-Flow evaluation architecture.

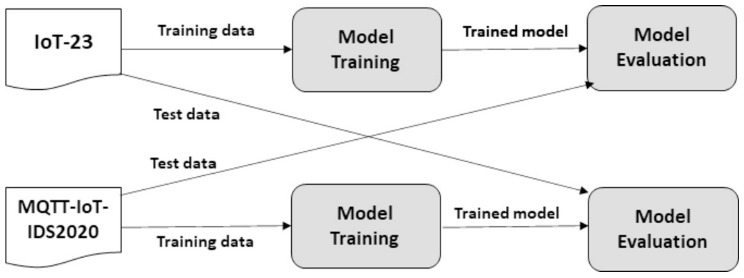

Fig. 7. Cross-flow evaluation architecture.

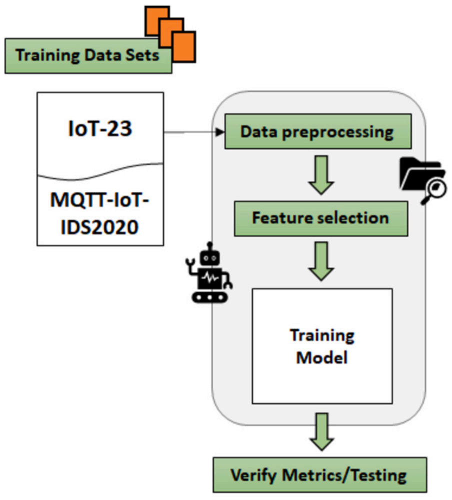

Fig. 8. Evaluation architecture for merged flow data.

Table 1
IoT23 features - Zeek flow features vs IPFIX.

IoT23 / Zeek field	IPFIX	
conn.log file	**ID**	**Name**
ts	22	flowStartSysUpTime
uid	148	flowId
id.orig_h	8	sourceIPv4Address
id.orig_p	7	sourceTransportPort
id.resp_h	12	destinationIPv4Address
id.resp_p	11	destinationTransportPort
proto	4	protocolIdentifier
service	5	ipClassOfService
duration	161	flowDurationMilliseconds
orig_bytes	231	initiatorOctets
resp_bytes	232	responderOctets
conn_state	136, 218, 219, 220, 221, 222, 223	flowEndReason, tcpSynTotalCount, tcpFinTotalCount, tcpRstTotalCount, tcpPshTotalCount, tcpAckTotalCount, tcpUrgTotalCount
local_orig	149	observationDomainId
local_resp	149	observationDomainId
missed_bytes	165	ignoredOctetTotalCount
history	6	tcpControlBits
orig_pkts	298	initiatorPackets
orig_ip_bytes	1	octetDeltaCount
resp_pkts	299	responderPackets
resp_ip_bytes	1	octetDeltaCount
tunnel_parents	148	flowId
label (*not a Zeek field*)	n.a.	n.a.
detailed-label (*not a Zeek field*)	n.a.	n.a.

Originally, traffic flows were split into log files accordingly to the malware type. To build a sample dataset representing the various types of network attacks produced in the lab, we took samples from several log files. Our final dataset has 1,244,220 flows, with the total number of malicious flows close to the number of normal flows. The not benign flows include horizontal port scans (to gather information to perform further attacks), Okiru malware, DDoS (Distributed Denial of Service) attack, and command and control attacks (C&C).

MQTT-IoT-IDS2020 - The MQTT-IoT-IDS2020 dataset contains data generated by a simulated MQTT network. Dataset contains raw pcap files, and unidirecional and bidirecional flow features. Data include normal operation traffic, aggressive and UDP scan traffic, Sparta SSH brute-force attack, and MQTT brute-force attack. The dataset contains distinct sets of features for pcaps, unidirecional flows and bidirecional flows. For bidirecional flows, it contains 32 features, including addresses and ports, protocol, packet length statics and flags. We merged the files containing bidirecional flow data, creating a dataset with almost 180 thousand flows.

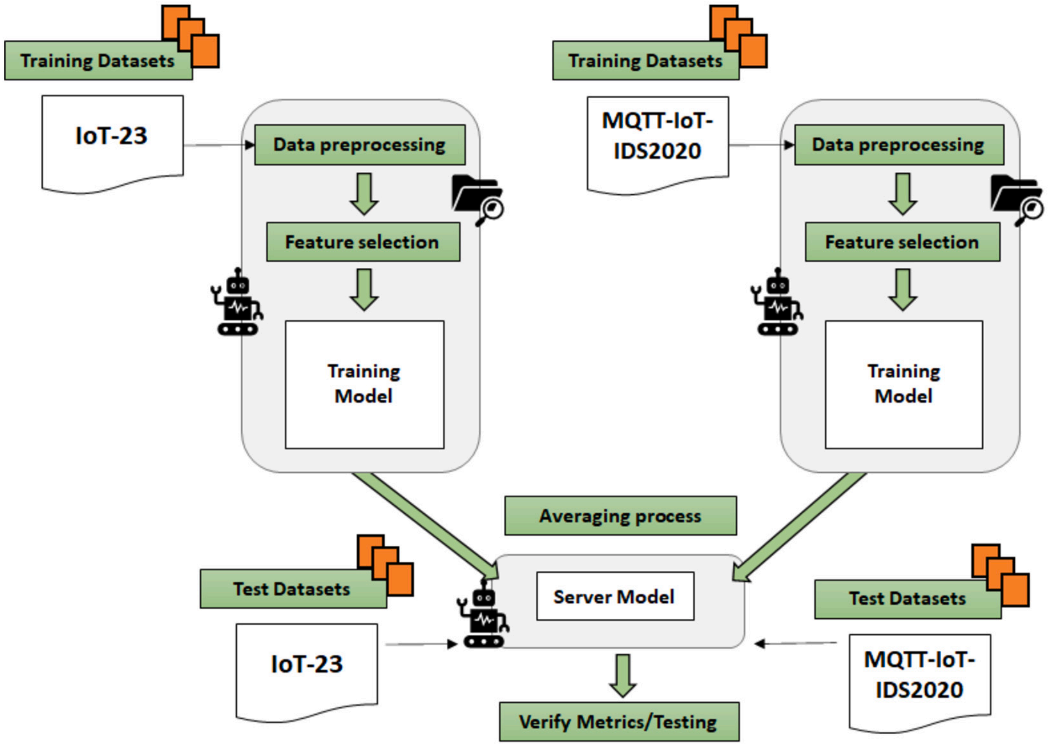

Fig. 9. Federated learning based on specialized flow data.

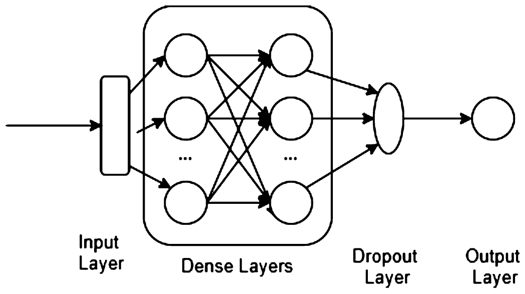

Fig. 10. Model representation.

4.3. Model configuration, and training, validation and testing strategies

We created a deep model composed of fully connected hidden layers, as represented in Fig. 10. Hyperparameters like the number of neurons in each hidden layer, the activation functions, and the output layer were manually set through experimentation. In the evaluation of the proposed IoT-Flow IDS architecture to process the IoT-23 dataset, we achieved the best results using three hidden layers with 64, 32, and 16 neurons, respectively. For all the other considered scenarios and data, we got the best results using two hidden layers with 50 and 25 neurons, respectively. Each hidden layer uses ReLu as the activation function. The output layer uses Softmax. We used Keras to build our model, the Adam optimizer, and a Dropout layer with 20% between the last hidden layer and the output one. Other hyperparameters remained in their default values.

We split each dataset into training (80%) and testing (20%) data. Validation data corresponds to 20% of the training data. We used Early Stopping with a patience of 10.

4.4. Feature selection and pre-processing

Reducing the number of features can lead to better and faster training. Our IoT-Flow IDS architecture does not require all the datasets to have the same features. On the other hand, the other three considered architectures require the flows to have compatible features.

IoT-23 - The first phase of our feature selection strategy was to remove the columns that always have the same value (i.e., *local_orig* and *local_resp*) and the columns whose values are all distinct (i.e., *ts* and *UID*). We also removed the tunnel-parents column as it is related to the removed *UID* column. The *detailed-label* represents the type of attack which we do not consider in these experiments. Thus, the *detailed-label* column was also removed.

Then, we preprocessed the remaining features and selected features using automatic methods. We transformed the features from categorical (strings) to numerical values and normalized numerical values. We used the LabelEncoder and the SimpleImputer (Pedregosa et al., 2011) to transform strings into values, and the StandardScaler (Pedregosa et al., 2011) algorithm for normalization. Then, we used the ExtraTreesClassifier (Pedregosa et al., 2011) algorithm to verify which features have a bigger contribution to flow classification inside our sample. The algorithm identified four columns as having a contribution over 0.05, namely the *proto* (Protocol), *Id.resp_p* (Destination port), and the *Id.orig_p* (Source port) and *history*. The latter is related to the TCP control bits. Since TCP is one of the drivers for malicious flows in this network, the *history* feature is a relevant feature for classification in our sample data.

Hence, we evaluated our IoT-Flow IDS architecture considering the four features identified by the ExtraTreesClassifier.

MQTT-IoT-IDS2020 - We merged the file with bidirectional flow data for normal traffic with four files with attack traffic flows data. Then, we removed the *ip_src* (IP address for traffic source), *ip_dst* (IP address for traffic destination) *proto* columns (the latter for comparison with (Hindy et al., 2020b)). We also removed the columns that store empty or single values.

In each numeric column, we replaced missing values with the median of the column's values and applied normalization using a Z-Score. The resulting feature set was used to evaluate the IoT-Flow IDS architecture.

Table 2
Source and destination port number transformation to categories.

Source/Destination Port Number	Category
< 5,000	1
> = 5,000 & < 10,000	2
> = 10,000 & < 15,000	3
> = 15,000 & < 20,000	4
> = 20,000 & < 25,000	5
> = 25,000 & < 30,000	6
> = 30,000 & < 35,000	7
> = 35,000 & < 40,000	8
> = 40,000 & < 45,000	9
> = 45,000 & < 50,000	10
> = 50,000 & < 55,000	11
> = 55,000 & < 60,000	12
> = 60,000	13

Additional operations to build the compatible datasets - After making the above described operations over the IoT23 and the MQTT-IoT-IDS2020 datasets, we had to make additional preprocessing operations to make them compatible, and thus enable their use to evaluate the Cross-flow, Merged flow and Flow-based federated learning scenarios.

Therefore, we selected the set of features they share, namely origin port number, destination port number, protocol, count of bytes generated by the source, and count of bytes generated by destination. We also had an attribute that identifies the traffic as normal or suspicious/malign.

Hence, besides the specific proprocessing operations for each dataset we described above, we applied the following transformations:

- Protocol - In IoT-23, we replaced the protocol described by the corresponding number. Then, for both datasets, we applied One-hot Encoding on the protocol value.
- Origin and destination port number - For port numbers (origin and destination), we created 13 classes. If a port number is below 5,000 then it is class 1. Port number between 5,000 and 9,999 is of class 2, and so on. All port numbers over 60,000 are from class 13. Table 2 presents the specified category for each port number. After transforming the origin and destination ports of each dataset into classes, we applied One-hot Encoding.
- Count of bytes generated by origin and destination - Missing values in the counts of generated bytes were transformed into zero. Then, we applied normalization using a Z-Score.

We also verified that both sets have the same attributes after the executed transformations (missing attributes were created and filled with a standard value). Finally, each of the pre-processed datasets had 31 attributes, i.e., three for protocol, two for transferred bytes, one for benign/malicious traffic identification, and the remaining containing information on the origin and destination port classes.

4.5. Experimental results

Each dataset was split into training, validation and testing data as described in the previous section. First, we evaluated the performance of models using training and testing data from the same dataset (i.e., IoT23 or MQTT-IoT-IDS2020). Then, we assessed models with training and testing data from distinct origins. Finally, we evaluated using a dataset composed of data merged from IoT23 and MQTT-IoT-IDS2020.

4.5.1. IoT-flow IDS evaluation

Table 3 presents the confusion matrix for the proposed IoT-Flow IDS.

IoT23 - The evaluated network achieved high true positive and negative values over IoT23, with only 0.2% of false positives and 0.1% of false negatives, as represented in the confusion matrix of Table 3a.

Table 3
Confusion matrices - Training and testing data selected from the same dataset.

Actual value \ Prediction outcome	0	1
0	50.0%	0.2%
1	0.1%	49.6%

(a) IoT23

Actual value \ Prediction outcome	0	1
0	72.5%	0%
1	0%	27.5%

(b) MQTT-IoT-IDS2020

Table 4
IoT23 Dataset - Performance evaluation.

Metric	IoT-Flow IDS	Austin (2021)	
		Random Forest	Naive Bayes
Precision	0.996	0.999	0.997
Recall	0.997	0.949	0.898
F1-Score	0.996	-	-

Table 5
MQTT-IoT-IDS2020 - Performance evaluation.

Metric	IoT-Flow IDS	Hindy et al. (2020a)	
		Linear Regression	Decision Trees
Precision	0.999	0.983	0.996
Recall	0.999	0.982	0.999
F1-Score	0.999	0.981	0.996

Table 4 presents the Precision, Recall, and F1-Score metrics obtained when using the data from the IoT23 dataset and compare the results with other work from the literature.

The proposed architecture achieved a precision compatible with the ones of other works, and clearly outperforms than in terms of recall, which is a key performance metric in the considered context, where having false negatives may be considered worse than having false positives.

MQTT-IoT-IDS2020 - Table 3b presents the confusion matrix obtained when evaluating the model using the MQTT-IoT-IDS2020 dataset. The network correctly classified almost all flows. Table 5 presents the computed metrics using the IoT-Flows IDS and the results obtained by (Hindy et al., 2020a). The table confirms our results outperforms the ones from the literature.

4.5.2. Closs-flow scenario

Table 6a presents the confusion matrix obtained when we evaluated the performance of a model trained with the IoT23 dataset when identifying anomalous traffic on the MQTT-IoT-IDS2020. The model misclassified most of the flows (more the 66% were false positives) and only achieved about 6% of true negatives.

Table 6b represents the confusion matrix obtained when using a model trained with data from MQTT-IoT-IDS2020 to identify anomalous traffic on the IoT23 dataset. The results were slightly better than the ones represented in Table 6a. Still, the model misclassified almost 63% of the flows.

Table 7 presents the values of Precision, Recall, and F1-Score. It is possible to confirm that the models didn't achieve acceptable generalization levels, as the metrics computed when using training and testing data from distinct datasets were below 50%.

Table 7 also presents the performance achieved when training models using only the features shared by both datasets to identify anomalies in the same dataset they were trained on.

Table 6
Confusion matrices - Training and testing data selected from distinct datasets.

Actual value	Prediction outcome 0	Prediction outcome 1
0	6.1%	66.4%
1	10.4%	17.1%

(a) Model trained with IoT23 - MQTT-IoT-IDS2020-based test data

Actual value	Prediction outcome 0	Prediction outcome 1
0	14.7%	35.4%
1	28.5%	21.4%

(b) Model trained with MQTT-IoT-IDS2020 - IoT23-based test data

Table 7
Performance evaluation - Cross-flow scenario.

Training data origin	Metric	Testing data origin	
		IoT-23	MQTT-IoT-IDS2020
IoT-23 (shared set of features)	Precision	0.979	0.326
	Recall	0.979	0.232
	F1-Score	0.979	0.185
MQTT-IoT-IDS2020 (shared set of features)	Precision	0.355	0.849
	Recall	0.358	0.836
	F1-Score	0.355	0.815

Table 8
Performance evaluation - Merged dataset.

Training data origin	Metric	Testing data origin	
		IoT-23	MQTT-IoT-IDS2020
Merged dataset	Precision	0.887	0.528
	Recall	0.860	0.300
	F1-Score	0.860	0.219

4.5.3. Merged flow scenario

We also built a dataset composed of the training data from IoT-23 and MQTT-IoT-IDS2020. We used 20% for such data for validation while training a new model. Then, we evaluated such a model on the testing sets we built for IoT-23 and MQTT-IoT-IDS2020. Achieved performance values are summarized in Table 8. Results show that the use of specialized models (i.e., the IoT-Flow IDS) achieves better performance than a single model for distinct types of network traffic.

4.5.4. Flow-based federated learning scenarios

We also evaluated the use of a federated learning approach. We trained a model with IoT23 data and another with MQTT-IoT-IDS2020. Then, the parameters of trained models were combined (averaged) and loaded into a new (server) model. We evaluated the performance of such server model using test data from both datasets.

Table 9a represents the confusion matrix obtained when using a model trained with data from MQTT-IoT-IDS2020 to identify anomalous traffic on the IoT23 dataset. The results were significantly better than the ones represented in Table 9b.

Table 10 presents the values of Precision, Recall, and F1-Score. It is possible to confirm that the server model didn't achieve acceptable generalization level, as the metrics computed when using testing data from MQTT-IoT-IDS2020 were far below 50%.

4.6. Discussion

Our IoT-Flow IDS proposal outperformed the results from the literature considered in Section 4.5.1. It achieved the best results from all the considered scenarios as summarized in Tables 11 and 12. Indeed, results of the Cross-flow and Federated learning scenarios confirm that evaluated models achieved reasonable performance when identifying attacks on the same sets they were trained on, but also that such models have poor generalization performance, as they were not capable of identifying anomalous traffic on data from another dataset.

Also, the performance became significantly worse when training the model with the merged dataset than when using models specialized to each flow dataset, which indicates that maintaining distinct IDS systems for individual application systems in smart cities is a promising approach.

By comparing the results of Table 7 achieved when using the *shared set of features* (in the Cross-folder scenario) to test data from the same origin of training data with the performance achieved using our IoT-Flow IDS, it is possible to conclude that the use of the deep model over the small set of features is a reasonable choice for the IoT23 dataset, as the performance was only slightly worse to one of the models that use a larger set of features. On the other hand, the results obtained when using data from the *shared set of features* of MQTT-IoT-IDS2020 were significantly worse than the ones of obtained with the full set of features, which confirms the need for the execution of the feature selection strategy for each dataset and, then, the need for the proposed segmented architecture.

Table 9
Confusion matrices - Flow-based federated learning.

Actual value	Prediction outcome 0	Prediction outcome 1
0	46.9%	3.2%
1	0.1%	49.7%

(a) IoT23 test data

Actual value	Prediction outcome 0	Prediction outcome 1
0	6.1%	66.4%
1	9.6%	17.9%

(b) MQTT-IoT-IDS2020 test data

Table 10
Performance evaluation - Federated learning.

Metric	Testing data origin	
	IoT-23	MQTT-IoT-IDS2020
Precision	0.968	0.342
Recall	0.967	0.241
F1-Score	0.967	0.189

5. Conclusions and future work

The large-scale use of IoT devices enabled the implementation of several intelligent services and boosted the creation of smart cities. In this context, devices collect and share a large amount of data. It is an attractive environment for potential attackers. Thus, it must have cyber protection measures, such as a specialized flow-based IDS. In this work, we describe a machine learning-based distributed IDS for smart cities' IoT environments that considers the usual limitations of IoT devices and the distinct classes of services and requirements in such context.

Our proposal captures IoT traffic data, aggregates communication packages into flows, applies pre-processing techniques and feature selection on the generated flows, and uses machine learning models to classify them. We describe a general architecture named IoT-Flow IDS, the specialized elements required for executing the workflow steps and their positioning considering segmented networks and fog nodes. We also discuss training and assessing new models over real-world data for continuous improvement.

Table 11
Performance metrics comparison - Models trained using IoT23 data.

Metric	IoT-Flow IDS - our proposal - (IoT23 test data)	Cross-Flow (MQTT-IoT-IDS2020 test data)	Merged Data Set (IoT23 test data)	Federated learning (IoT23 test data)
Precision	0.996	0.326	0.887	0.968
Recall	0.997	0.232	0.860	0.967
F1-Score	0.996	0.185	0.860	0.967

Table 12
Performance metrics comparison - Models trained using MQTT-IoT-IDS2020 data.

Metric	IoT-Flow IDS - our proposal - (MQTT-IoT-IDS2020 test data)	Cross-Flow (IoT23 test data)	Merged Data Set (MQTT-IoT-IDS2020 test data)	Federated learning (MQTT-IoT-IDS2020 test data)
Precision	0.999	0.355	0.528	0.342
Recall	0.999	0.358	0.300	0.241
F1-Score	0.999	0.355	0.219	0.189

We experimentally evaluated deep models to identify anomalous traffic in IoT traffic flows of two datasets. We use deep models in four distinct scenarios. The results from Cross-flow and Federated learning scenarios show that the models are not generalizable, even though we use techniques to avoid overfitting. We built a new set that combines data from two datasets. The results obtained by the combined dataset are considerably worse than the ones we got by specialized models.

In future work, we intend to expand the analysis of deep models with new datasets and evaluate the resilience of our solution to adversarial machine learning.

CRediT authorship contribution statement

Nuno Prazeres: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Rogério Luís de C. Costa:** Conceptualization, Investigation, Project administration, Methodology, Software, Supervision, Visualization, Writing – original draft, Writing – review & editing. **Leonel Santos:** Conceptualization, Project administration, Supervision, Writing – original draft, Writing – review & editing. **Carlos Rabadão:** Conceptualization, Project administration, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work is funded by national funds through FCT – Fundação para a Ciência e a Tecnologia, I.P., under the Scientific Employment Stimulus - Institutional Call - CEECINST/00051/2018 and in the context of the project UIDB/04524/2020.

References

Ahmad, R., & Alsmadi, I. (2021). Machine learning approaches to iot security: A systematic literature review. *Internet of Things*, *14*, Article 100365.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for Internet of things (IoT) security. *IEEE Communications Surveys and Tutorials*, *22*, 1646–1685. https://doi.org/10.1109/COMST.2020.2988293. arXiv:1807.11023.

Ariyaluran Habeeb, R. A., Nasaruddin, F., Gani, A., Targio Hashem, I. A., Ahmed, E., & Imran, M. (2019). Real-time big data processing for anomaly detection: A survey. *International Journal of Information Management*, *45*, 289–307. https://doi.org/10.1016/j.ijinfomgt.2018.08.006.

Ashraf, J., Keshk, M., Moustafa, N., Abdel-Basset, M., Khurshid, H., Bakhshi, A. D., & Mostafa, R. R. (2021). Iotbot-ids: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustainable Cities and Society*, *72*, Article 103041.

Austin, M. (2021). IoT malicious traffic classification using machine learning IoT malicious traffic classification using machine learning. Master's thesis, Statler College of Engineering and Mineral Resources - West Virginia University.

Berman, D. S., Buczak, A. L., Chavis, J. S., & Corbett, C. L. (2019). A survey of deep learning methods for cyber security. *Information (Switzerland)*, *10*. https://doi.org/10.3390/info10040122.

Butun, I., Österberg, P., & Song, H. (2019). Security of the Internet of things: Vulnerabilities, attacks, and countermeasures. *IEEE Communications Surveys and Tutorials*, *22*, 616–644.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys and Tutorials*, *21*, 2671–2701. https://doi.org/10.1109/COMST.2019.2896380.

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. Preprint (pp. 1–50). arXiv:1901.03407.

Claise, B. (2008). Specification of the IP flow information export (IPFIX) protocol for the exchange of IP traffic flow information. RFC 5101. https://rfc-editor.org/rfc/rfc5101.txt. https://doi.org/10.17487/RFC5101.

Claise, B., Quittek, J., Meyer, J., Bryant, S., & Aitken, P. (2008). Information model for IP flow information export. RFC 5102. https://rfc-editor.org/rfc/rfc5102.txt. https://doi.org/10.17487/RFC5102.

Claise, B., & Trammell, B. (2013). Information model for IP flow information export (IPFIX). RFC 7012. https://rfc-editor.org/rfc/rfc7012.txt. https://doi.org/10.17487/RFC7012.

Claise, B., Trammell, B., & Aitken, P. (2013). Specification of the IP flow information export (IPFIX) protocol for the exchange of flow information. Technical Report.

Cui, L., Xie, G., Qu, Y., Gao, L., & Yang, Y. (2018). Security and privacy in smart cities: Challenges and opportunities. *IEEE Access*, *6*, 46134–46145. https://doi.org/10.1109/ACCESS.2018.2853985.

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of things. *Future Generations Computer Systems*, *82*, 761–768. https://doi.org/10.1016/j.future.2017.08.043.

Elrawy, M. F., Awad, A. I., & Hamed, H. F. (2018). Intrusion detection systems for IoT-based smart environments: A survey. *Journal of Cloud Computing*, *7*, 1–20. https://doi.org/10.1186/s13677-018-0123-6.

Figueiredo, B. J., Costa, R. L. d. C., Santos, L., & Rabadão, C. (2022). Cybersecurity and privacy in smart cities for citizen welfare. In *Smart cities, citizen welfare, and the implementation of sustainable development goals* (pp. 197–221). IGI Global.

Garcia, S., Parmisano, A., & Erquiaga, M. J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. https://doi.org/10.5281/zenodo.4743746.

Garcia-Font, V., Garrigues, C., & Rifà-Pous, H. (2018). Difficulties and challenges of anomaly detection in smart cities: A laboratory analysis. *Sensors*, *18*, 3198.

Hindy, H., Bayne, E., Bures, M., Atkinson, R., Tachtatzis, C., & Bellekens, X. (2020a). Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset). In *International networking conference* (pp. 73–84). Springer.

Hindy, H., Tachtatzis, C., Atkinson, R., Bayne, E., & Bellekens, X. (2020b). Mqtt-iot-ids2020: Mqtt Internet of things intrusion detection dataset. https://dx.doi.org/10.21227/bhxy-ep04.

Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., & Pras, A. (2014). Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. *IEEE Communications Surveys and Tutorials*, *16*, 2037–2064.

Hromada, D., Costa, R. L. d. C., Santos, L., & Rabadão, C. (2021). Security aspects of the Internet of things. In *IoT protocols and applications for improving industry, environment and society* (pp. 207–233). IGI Global.

Hussain, F., Hussain, R., Hassan, S. A., & Hossain, E. (2020). Machine learning in IoT security: Current solutions and future challenges. *IEEE Communications Surveys and Tutorials*, *22*, 1686–1721. https://doi.org/10.1109/COMST.2020.2986444.

Inacio, C. M., & Trammell, B. (2010). {YAF}: Yet another flowmeter. In *24th large installation system administration conference (LISA 10)*.

Jiang, B., Li, J., Yue, G., & Song, H. (2021). Differential privacy for industrial Internet of things: Opportunities, applications, and challenges. *IEEE Internet of Things Journal*, *8*, 10430–10451.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides, & B. Schmidt (Eds.), *Positioning and power in academic publishing: Players, agents and agendas* (pp. 87–90). IOS Press.

Li, S., Qin, D., Wu, X., Li, J., Li, B., & Han, W. (2022). False alert detection based on deep learning and machine learning. *International Journal on Semantic Web and Information Systems*, *18*, 1–21.

Ling, Z., & Hao, Z. J. (2022). An intrusion detection system based on normalized mutual information antibodies feature selection and adaptive quantum artificial immune system. *International Journal on Semantic Web and Information Systems*, *18*, 1–25.

Liu, Y., Wang, J., Li, J., Niu, S., & Song, H. (2021). Machine learning for the detection and identification of Internet of things devices: A survey. *IEEE Internet of Things Journal*, *9*, 298–320.

Minsky, M. L., & Papert, S. (1969). *Perceptrons, expanded edition an introduction to computational geometry*. MIT Press.

Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2022). Federated-learning-based anomaly detection for iot security attacks. *IEEE Internet of Things Journal*, *9*, 2545–2554. https://doi.org/10.1109/JIOT.2021.3077803.

Moustafa, N., Hu, J., & Slay, J. (2019a). A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, *128*, 33–55. https://doi.org/10.1016/j.jnca.2018.12.006.

Moustafa, N., Hu, J., & Slay, J. (2019b). A holistic review of network anomaly detection systems: A comprehensive survey. *Journal of Network and Computer Applications*, *128*, 33–55.

Neisse, R., Steri, G., Baldini, G., Tragos, E., Fovino, I. N., & Botterman, M. (2014). Dynamic context-aware scalable and trust-based iot security, privacy framework. In *Internet of things applications-from research and innovation to market deployment*. IERC Cluster Book.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Santos, L., Gonçalves, R., Rabadão, C., & Martins, J. (2021). A flow-based intrusion detection framework for Internet of things networks. *Cluster Computing*, 1–21. https://doi.org/10.1007/s10586-021-03238-y.

Singh, P., Nayyar, A., Kaur, A., & Ghosh, U. (2020). Blockchain and fog based architecture for Internet of everything in smart cities. *Future Internet*, *12*, 1–12. https://doi.org/10.3390/FI12040061.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. (2010). An overview of ip flow-based intrusion detection. *IEEE Communications Surveys and Tutorials*, *12*, 343–356.

Stergiou, C. L., Plageras, A. P., Psannis, K. E., & Gupta, B. B. (2020). Secure machine learning scenario from big data in cloud computing via Internet of things network. In *Handbook of computer networks and cyber security* (pp. 525–554). Springer.

Sucharitha, V., Prakash, P., & Iyer, G. N. (2019). Agrifog-a fog computing based IoT for smart agriculture. *International Journal of Recent Technology and Engineering*, *7*, 210–217.

Tewari, A., & Gupta, B. B. (2017). A lightweight mutual authentication protocol based on elliptic curve cryptography for iot devices. *International Journal of Advanced Intelligence Paradigms*, *9*, 111–121.

Tewari, A., & Gupta, B. B. (2020). Secure timestamp-based mutual authentication protocol for iot devices using rfid tags. *International Journal on Semantic Web and Information Systems*, *16*, 20–34.

Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of ids for iot: Recent updates, security issues, and challenges. *Archives of Computational Methods in Engineering*, *28*, 3211–3243.

Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in iot intrusion detection. *Journal of Network and Systems Management*, *30*, 1–40.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. Scotts Valley, CA: CreateSpace.

Velan, P. (2018). Improving network flow definition: Formalization and applicability. In *NOMS 2018-2018 IEEE/IFIP network operations and management symposium* (pp. 1–5). IEEE.

Zeadally, S., & Tsikerdekis, M. (2020). Securing Internet of things (IoT) with machine learning. *International Journal of Communication Systems*, *33*, Article e4169. https://doi.org/10.1002/dac.4169.

Zseby, T., Boschi, E., Hirsch, T., & Mark, L. (2006). Ipfix/psamp: What future standards can offer to network security. *Proceedings FloCon, 2006*.