INTRUSION DETECTION SYSTEMS FOR IOT: OPPORTUNITIES AND CHALLENGES OFFERED BY EDGE COMPUTING

Pietro Spadaccino[1,2] and Francesca Cuomo[1,2]
[1]DIET, Department of Information Engineering, Electronics and Telecommunications, Sapienza University of Rome, 00184 Rome, Italy, [2]CNIT, National Inter-University Consortium for Telecommunications, Parma, Italy,
(pietro.spadaccino, francesca.cuomo)@uniroma1.it

NOTE: Corresponding author: Pietro Spadaccino, pietro.spadaccino@uniroma1.it

Abstract – *Key components of current cybersecurity methods are the Intrusion Detection Systems (IDSs), where different techniques and architectures are applied to detect intrusions. IDSs can be based either on cross-checking monitored events with a database of known intrusion experiences, known as signature-based, or on learning the normal behavior of the system and reporting whether anomalous events occur, named anomaly-based. This work is dedicated to the application of IDS to the Internet of Things (IoT) networks, where also edge computing is used to support IDS implementation. Specific attention is given to IDSs which leverage device classification at the edge. New challenges that arise when deploying an IDS in an edge scenario are identified and remedies are proposed.*

Keywords – Device classification, edge computing, Internet of Things, intrusion detection systems

1. INTRODUCTION

An Intrusion Detection System (IDS) is a software or hardware component that identifies malicious actions on computer systems or networks, thus allowing security to be maintained. Host-based Intrusion Detection Systems (HIDS) target a single computer system, while Network-based Intrusion Detection Systems (NIDS) target a whole network. NIDS are devices or software components deployed in a network which analyze the traffic generated by hosts and devices [1]. NIDSs are the focus of this work and from now on the term IDS will indicate NIDS.

The concept of IDS applied to the Internet of Things (IoT) is not new and many solutions have been proposed [2][3]. Traditional IoT-oriented IDSs are placed at the device level or at the gateway level, as shown in Fig. 1 and in cases where operated by leveraging cloud computing. However, recent advances in Edge Computing (EC) have opened new possibilities of IoT that can be leveraged also from a security point of view. Indeed EC extends the cloud computing paradigm to the edge of the network. For example, edge computing devices, which are capable of intelligent computing, can reduce the network latency by enabling computation and storage capacity at the edge network and this is particularly significant when dealing with IoT. On the other hand, the presence of edge nodes opens new breaches which could be exploited by malicious parties for their attacks. Edge nodes could be a victim of unauthorized remote accesses or even of physical tampering, especially those nodes which are deployed in public areas. An attacker, gaining the control of an edge node, could alter arbitrarily all the traffic passing through it. It could selectively-forward some packets, or even injecting some new ones pretending to be a legitimate device. If IDSs were placed on the device level or gateway level, they would not have the possibility to detect such attacks, since the intrusion takes place in a different network section. On the other hand, by deploying IDSs at the network edge (red box in Fig. 1) new issues arise, which hinder the reliability of such IDSs. For these challenges to be solved, new IDSs specifically designed for the edge should be implemented.

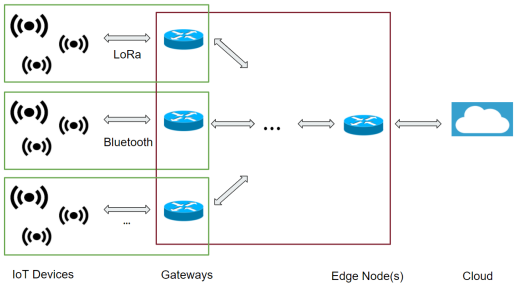


Fig. 1 – Network architecture of an edge-enabled IoT system. Traditional IoT IDSs are deployed at the device or gateway level (green boxes in figure). These systems protect the network against attacks generated by some malicious IoT or non-IoT devices in the specific network. However, the network edge offers new attack surfaces to be exploited by malicious parties. IDSs could be deployed at the network edge (red box in figure). In this case new challenges arise and have to be solved, as a consequence new IDSs specifically designed for the edge should be implemented.

In this framework, the goal of this work is threefold:

- to provide a taxonomy for IDSs and to discuss their applications in the IoT field;
- to present challenges and opportunities for the implementation of IDS at the edge;
- to discuss the open issues relevant to the adoption of edge-enabled architectures to IDS.

The rest of the paper is organized as follows. In Section 2 we classify signature-based and anomaly-based IDSs. Then, in Section 3, we discuss the application of IDS to IoT environments while Section 4 specifically addresses edge-enabled solutions. In Section 6 new challenges that arise when deploying an IDSs in an edge environment are identified. We illustrate how these challenge affects existing IDS and propose possible remedies. Conclusions are given in Section 7.

2. INTRUSION DETECTION SYSTEM TAXONOMY

The goal of an IDS is to prevent any unauthorized access to an information system. Any access could pose a threat to information confidentiality, integrity or availability. An IDS fulfills its duty by analyzing network traffic and/or resource usage and raising an alert when malicious activity is identified.

The IDSs can be categorized into two main families based on which strategy the system follows to detect intrusions, which can be either cross-checking monitored events with a database of known intrusion techniques or learning the normal behavior of the system and reporting whether some anomalous events are occurring. These strategies are named *signature-based* and *anomaly-based*, respectively.

2.1 Signature-based IDS

Signature-based IDSs are a class of systems that leverage a database of signatures of known attacks. Signatures of the current activities are extracted and matching methods and/or protocol conformance checks are then used to compare these signatures to the ones in the database. If a match is found an alarm is triggered. They can operate both in online mode, directly monitoring the hosts and raising alarms in real-time, and offline mode, where logs of the system activities are analyzed. This class of IDS is also known in the literature as misuse detection, specification-based or knowledge-based detection [4]. The main advantage of such an IDS is the high precision: if we have well-defined signatures of known attacks, an IDS can reliably raise alerts when they occur during operation. One downside of the signature-based approach is the extraction of traffic signatures, which may be a cumbersome and lengthy task to carry out, depending on which and how many traffic "features" are considered. Indeed signatures are often manually-crafted by experts having detailed knowledge of the exploits that the system is supposed to capture. Moreover, signature-based IDSs cannot cope with zero-day attacks, which are attacks whose signature is not in the IDS's database. The rising rate of zero-day attacks [5] makes less effective the overall performance of a signature-based IDS. For this reason anomaly-based IDSs were developed, a new class of IDSs which model the nominal behavior of a computer system and then reports any significant deviation from the baseline.

2.2 Anomaly-based IDS

Anomaly-based IDSs were developed to overcome the limitations of signature-based IDSs. Such IDSs usually have a training phase, during which they build a model of the nominal behavior of the system. When the IDS is deployed, it monitors computer hosts and compares their behavior with the nominal one. When a significant deviation between the hosts' behavior and the model is observed, the IDS may raise an alert. Potentially, this strategy gives an anomaly-based IDS the capability to capture zero-days attacks, since it does not perform any matching between the current hosts' behavior and attack signatures in a database. Another advantage of an anomaly-based IDS, is that it is difficult for an attacker to understand the normal behavior of a target host without doing transactions with it, since communicating with a target would likely make the IDS raise an alert [6], [7]. Moreover, anomaly-based IDSs could be exploited not just for security purposes, but also as a system analysis tool. If the IDS reports anomalies, it means that something is working differently from the baseline conditions, which can be an indication of not only an intrusion, but can also show the presence of a bug in a device's logic. A major limitation of an anomaly-based IDS, is the higher rate of false positives when compared to a signature-based IDS. Indeed, during operation, the targeted system can slightly or drastically change behavior without any intrusion taking place, and if an anomaly-based IDS is not aware of this possibility it can raise false alerts.

2.2.1 Statistics-based IDS

During the learning phase, an IDS based on statistical techniques builds a probability distribution model of the computing system during its nominal behavior. The model is built by taking measurements of different parameters and events taking place in the computing system. When the IDS is deployed, it evaluates the probability of all the monitored events of the system, and raises alerts on low probability events. The simplest strategy to build the statistical model is the so-called "Univariate" strategy, and consists of considering each measurement independently from the others. An evolution of it is the "Multivariate" strategy, which consists of identifying correlations and relationships between two or more measurements. Hybrid approaches are also possible: Ye et al. [8] proposed a hybrid univariate and multivariate system, by building profiles of each measurement individually, and then discovering multivariate correlations to decrease the false positives rate. When dealing with a high number of measurements, using multivariate statistics techniques on the raw data may produce a high level of noise. To overcome this problem, systems in [9], [10], [11], [12] used Principal Component Analysis (PCA), a statistical technique which is used to reduce the dimensions of input vectors, before applying standard multivariate statistical techniques.

Another family of statistics-based techniques to detect anomalies takes advantage of time series data techniques [13], which have also been applied to network anomaly detection [14]. When calculating the probability of an event occurring also the time is considered, and an alert is raised if an event is unlikely to have happened in a specific time. Zhao et al. [15] exploited techniques to mine frequent patterns in network traffic, and applied time-decay factors to differentiate between newer and older patterns. This strategy helps such IDSs to update its system baseline, making the IDS able to cope with the highly dynamic behavior of users. When developing a statistics-based anomaly IDS, and in particular an IDS working on time series data, attention must be given to data seasonality. Seasonality is the presence of variations in the data, which occur periodically in a course of months, days or even hours. It can be caused by "human" factors like holidays and work-hours or can be also influenced by other factors, like weather, depending on the application. Reddy et al. [16] proposed an algorithm to detect outliers in seasonality-affected time series data using a double pass of Gaussian Mixture Models (GMMs). During the learning phase, they divide the time into seasonal time bins, GMMs are trained and outliers are removed from data. To improve performance, another set of GMMs are built on the cleaned data. Finally this second set of GMMs is used to carry out the final anomaly detection.

2.2.2 Machine learning-based IDS

Machine Learning (ML) has been extensively applied in the field of cybersecurity [17]. Many of the specialized branches of ML have been exploited to develop an anomaly-based IDS leveraging machine learning, including data mining [18], deep learning [19] [20], deep reinforcement learning [21] and lately adversarial learning [22]. ML-based IDSs leverage machine learning models to automatically learn a representation of the normal conditions of the computing system.

When designing an ML system, the first step is to identify the features of the data to be analyzed, and an IDS makes no exception [23]. Preliminary work is focused on evaluating the goodness of traffic features, by using publicly available datasets and applying baseline ML algorithms. Works from Khraisat [24], Bajaj [25] and Elhag [26] evaluate the importance of dataset features via Information Gain (IG), correlation attribute evaluation and by applying genetic-fuzzy rule mining methods. By exploiting this evaluation, they clean out features that bring low IG or carry the same information of another feature. They then apply algorithms such as C4.5 Decision Tree, Naïve Bayes, NB-Tree, Multi-Layer Perceptron, SVM, and k-means Clustering. Other techniques used for IDS feature selections include Principal Component Analysis (PCA) [27], [28] and Genetic Algorithms (GA) [29].

A machine learning model can be trained with or without ground-truth labels. The learning techniques take the name "Supervised Learning" and "Unsupervised Learning". An overview of the most used ML-based approaches and their requirements is in Fig 2. From the point of view of the input knowledge needed to build the ML model, supervised learning techniques are the ones that require the greatest input knowledge, since, in order to build an efficient model, a great amount of data may be needed. On the other hand, memory requirements tend to be higher with the unsupervised models. Indeed, instance-based models such as k-means or kNN require storing the entire dataset to make predictions, while ANNs or SVMs need only the trained model, which has a memory footprint of orders of magnitude lower than the one of a dataset. ANNs and SVMs require high complexity when building a model, and, in case of ANNs, they may require also specialized hardware e.g. GPUs. Instance-based classifiers on the other hand do not have any special requirement for training, since there is not a real training phase where a model is built. For this reason those classifiers have high computational requirements when making predictions, since it is likely that, in order to predict a sample, the algorithm has to go through the whole dataset. ANNs and SVMs instead can make predictions easily, requiring just some matrix multiplication operations.

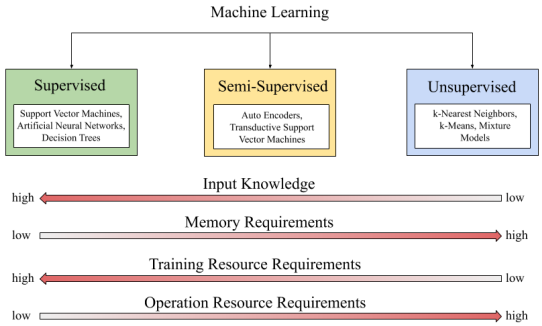


Fig. 2 – General taxonomy of machine learning techniques applied to IDS and their main requirements

2.3 Specification-based IDS

Specification-based IDSs fall into the category of the so-called expert systems. These systems leverage a knowledge source which represents the legitimate traffic signature. Every event that differs from this profile is treated as an anomaly. This knowledge is, most of the times, hand-crafted, and could contain rules about the nominal traffic patterns of the systems as well as Finite State Machines (FSMs) applied to Internet protocols such as IP, TCP, HTTP, etc. to ensure the compliance of the host to the aforementioned protocols. Ensuring protocol compliance via an FSM could be a hard task, since we have to model our state machine on top of the targeted protocol, which can be complex. If otherwise the communication protocol is implemented via writing code, one should always refer to existing implementations, e.g. open protocol stacks found in the Linux kernel. Another downside of specification-based IDSs is that they may fail to identify DoS attacks. In particular, these attacks may ex-

ploit messages and payloads that are foreseen by the protocol standard, therefore respecting the specification of the communication. For this reason, there could be no violation of the IDS specification and no alerts could be raised. These attacks may be more efficiently countered by anomaly-based IDSs. Moreover, specification-based IDSs don't scale well in a complex system formed by several distinct IoT protocols. In such a system, a different specification-based IDS needs to be installed for each IoT communication protocol we want to cover. This could increase the complexity of the setup and operation of the IDS.

3. IDS FOR IOT

As stated in Section 1, IDSs targeting the IoT can be categorized into IoT-specific and IoT-agnostic. An IoT-specific IDS targets devices using a particular communication technology, such as 6LoWPAN, BLE, LoRaWAN etc. This class of IDS should be deployed on the same network of the device. They usually carry out their predictions based on messages sent by the IoT devices leveraging control information of the specific technology, such as checking protocol compliance. On the other hand, IoT-agnostic IDSs do not depend on a particular IoT technology. They utilize information available regardless of which technology is currently used by the devices, such as TCP/IP traffic. This class of IDS is suitable to be used in an edge environment, since it can deal with traffic generated by heterogeneous devices leveraging different communication technologies.

An advantage of an IoT-specific IDS over an IoT-agnostic one is the ability to detect low-level attacks generated on the device level. On the other hand, a single IoT-agnostic IDS is able to deal with many IoT devices, without the need to deploy an IoT-specific IDS for every communication technology available.

An IoT-specific IDS commonly operates on the network section highlighted in green in Fig. 1, while an IoT-agnostic IDS on the one highlighted in red.

The main characteristics of IDS targeting IoT are described in Fig. 3. Starting from left to right, we can distinguish the kind of device from where the IDS runs. This can be from fully implemented at the cloud server level to a very peripheral part of the network, i.e. the end devices. IDS may be designed targeting different access technologies (short, medium or long range) and specific vulnerabilities that may exist. The adopted techniques, as described in Section 2, as well as the computing architecture may distinguish the impact of the IDS. Finally, different performance objectives exist. Some IDSs may be oriented to use devices' energy as low as possible while others may target indicators like latency or quantity of data exchanged.

Several IoT-specific IDSs have been proposed for different communication technologies. These systems are usually expert systems which capture the traffic between hosts and check the compliance of each packet to technology-specific network protocols and/or search for known attack signatures. For Wi-Fi, authors in [30] have developed neural network-based detection approaches and they have tested it on the Aegean WiFi Intrusion Dataset (AIWD) [31]. The proposed techniques reached high accuracies on three different attack types, namely injection attacks, impersonation attacks and flooding attacks. For LoRaWAN, authors in [32] proposed a detection technique for a well-known vulnerability that ultimately disconnects an end device from a LoRaWAN network permanently. The proposed approach is based on the Hamming distance and on the Kullback Leibler Divergence (KLD). The authors have also set up a testbed and they have tested the system on it, reaching high detection rates with low false positives. Continuing with LoRaWAN, authors in [33] have developed an IDS which detects if a re-identification attack, which links the DevAddress and the DevEUI of an end device, is possible. The proposed IDS is based on pattern matching and it is able to scale well with an increasing number of end devices. For ZigBee, authors in [34] have proposed HANIDPS, which is a hybrid specification-based and machine learning-based IDS targeting the ZigBee protocol. The system leverages Q-learning to evolve the system with the interaction of the environment, and is able to protect the devices to new and unseen attacks. For Bluetooth, authors in [35] have developed a machine learning-based IDS which detects known denial-of-service attacks in BLE networks. In order to test their techniques and to overcome the limitation of the unavailability of a dataset for such attacks, the authors have also developed a data collection system based on ESP32.

Advanced systems, can also detect attacks on the physical network layer (PHY), e.g. jamming. Usually an attacker sending a PHY attack sends bits not following the communication protocol, preventing the data to be readable from an external IDS and making the attack extremely difficult to detect. For example, authors in [36] propose an attack on the BLE physical layer which selectively jams the signal on specific channels whenever a device tries to connect.

Instead of focusing on single protocols, other proposals have focused on creating IDSs capable of working on unified IoT environment. Authors in [37] have proposed a framework for an IDS based on an Artificial Immune System (AIS). The IDS is distributed among IoT devices, edge nodes and cloud nodes. On the IoT devices, lightweight detectors are deployed. On the edge, alerts are analyzed and processed using smart data concepts. Finally, the cloud clusters the data and trains the detectors. In this way the heavyweight detectors' model training is done on the cloud and only a lightweight application of them is carried out by IoT devices.

As with all expert systems, IoT-specific IDSs usually achieve high accuracy and low false positive rates, and often they are specialized into the detection of a specific, well-defined attack. They are, however, unable to detect zero-days or unusual usage of the network resources by the hosts. On the other hand, IoT-agnostic IDSs work independently on the communication technology between

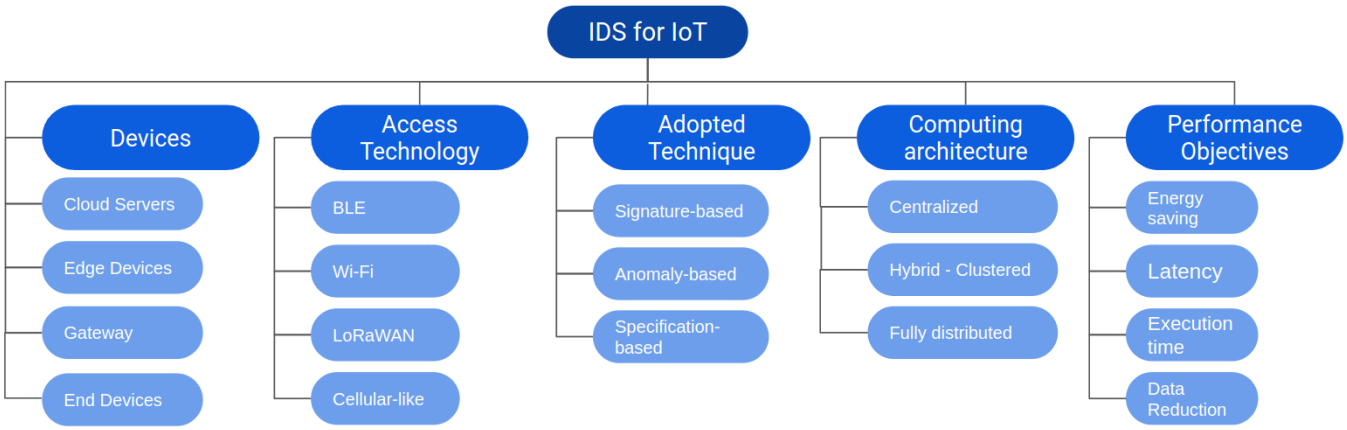


Fig. 3 – Classification of IDS targeting IoT devices

IoT devices. These IDSs could be deployed on IoT gateways, discarding PHY or MAC layer information, or also in another subnetwork, where they leverage TCP/IP traffic features.

Specific and new security issues arising in the IoT field are the ones related to the network management and operation such as routing, topology control, and network maintenance. As for the routing, new protocols for devices with constrained resources have been designed like the Routing Protocol for Low Power Lossy Network (RPL) [38]. The message exchange is based on the Destination Oriented Directed Acyclic Graph (DODAG), which is built by the devices following the protocol, enabling point-to-point, point-to-multipoint and multipoint-to-point communications. Many IDSs have been proposed to check the correct execution of the RPL protocol by the end devices. In RPL networks, there exists known attacks whose goal is to make the devices find a suboptimal route, which in the end can shorten the lifespan of the devices, or to damage the routing of the network itself, e.g. black holing. Mayzaud et al. [39] have proposed an IDS targeting specifically RPL version number attacks. Their solution is based on a distributed monitoring architecture, on which several detection algorithms are deployed. Moreover, the IDS is able to identify and locate the malicious node which is carrying the attack. Authors in [40] have proposed an IDS based on neural networks which targets version number, worst parent and hello flood attacks. The novelty of such an IDS, is that it considers not only routing-layer features but also link-layer features. Link-layer features are especially useful by decreasing the false positives and when considering version number attacks.

4. THE EDGE-ENABLED APPROACH

Edge computing was proposed to enhance the characteristics and the reliability of traditional IoT applications [41], [42] under several aspects. The IoT application can offload computational tasks, storage or management tasks to the edge nodes. Some of the expected quality enhancements include the minimization of the latency, real-time network management and better data management. In this context, also security applications, such as an IDS, could be "migrated" on the edge (see the red box in Fig. 1). An IDS could benefit from this transition, having more computational resources available, enabling it to use more complex algorithms, and also more storage capabilities, in order to store systems logs to be later analyzed or to carry out memory-intensive procedures. Also an edge node could offer lower latency than the cloud, which is crucial for real-time IoT applications. Moreover, an IDS deployed on the edge, should be IoT-agnostic, meaning that it does not depend on specific IoT communication technologies. If such an IDS is used, it can deal with many heterogeneous devices using different communication technologies in a unified manner, without having to deploy a single IoT-specific IDS for every subnetwork of devices.

Eskandari et al. [43] developed Passban IDS, a system which is able to apply a protection layer on IoT devices which are directly connected to it. The attacks targeted by the system are TCP/IP-oriented, not including IoT technology-dependent ones, such as Port Scanning, HTTP and SSH brute force and SYN flood. The system does not require intensive calculations and can be deployed also on cheap edge devices and/or IoT gateways, such as Raspberry Pis or equivalent. While the IDS aims to protect devices against a relatively low number of attacks, the system shows a very low false positive rate and high accuracy. A positive note about the system, is that is one of the few fully-implemented IDSs, from the detection algorithm to the alerting system leveraging a web user interface.

Terenzi et al. [33] have developed an IDS architecture targeting LoRaWAN devices. Such IDSs have the goal to alert the network operator in case an attacker is able to find the DevEUI of a device only by observing application traffic and the DevAddr carried inside the packets. The proposed technique assumes that devices transmit application packets with some periodicity, then exploits temporal linking of the application and join request messages to

find the exact matching between a DevAddr and DevEUI. Such IDSs can operate on the LoRaWAN gateways, which represent the network edge in the context of LoRaWAN. Moreover, the IDS does not require to decrypt packets and bases its predictions only on the timings of the arrival of packets.

Authors in [44] have investigated the identification of malicious edge devices. Indeed, edge devices are privileged for storing and processing data produced by potentially hundreds or thousands of IoT devices. For an attacker to gain control over such an edge node, would mean a potential control over the data sent by attached IoT devices. The authors proposed a framework which exploits a two-stage Markov Model, an anomaly-based IDS and a Virtual Honeypot Device (VHD). When an alert is raised by the IDS, it is forwarded to the two-stage Markov Model. The first stage categorizes the specific fog node and the second one predicts whether or not the VHD should be attached to the edge node for which the alert was raised. The VHD stores logs of all attached edge nodes, which can be later investigated by experts.

Authors in [45] introduced a system to improve the detection accuracy of an IDS by deploying fuzzy c-means and ANNs in the edge. They compared their approach with classic ANN techniques, and show high accuracy also on attacks with a low frequency.

Hafeez et al. [46] proposed a system to perform anomaly detection at the network edge gateways. The system represents the traffic with features that are agnostic with respect to the IoT communication technology, but only depends on TCP/IP features which can be observed by the edge. The advantage of this approach is that several systems, each one having heterogeneous IoT communication technologies, can be attached to the same IDS. As for the dataset, they have collected IoT data from a real-world testbed. They have also studied the distribution of the various considered features, and they have observed that the majority of them are well fitted by a heavy-tailed Gaussian. The final anomaly detection is performed through the use of fuzzy clustering. On their custom dataset, they have achieved high accuracy and low false positive rate.

Schneible et al. [47] et al. proposed a framework to perform a distributed anomaly detection on edge nodes. The system consists of deploying auto-encoder models on several edge nodes positioned in different network regions. The anomaly detection is carried out using the classic auto-encoder approach. The system also shows some degree of adaptivity: while deployed the edge nodes update their models based on new observations, identifying new trends in network traffic. An edge node then sends to a central authority its updated model, which aggregates them and sends the updates to the other edge agents. The authors observed that this approach reduces the overhead bandwidth, since the only generated network traffic carries the models of the auto-encoders instead of all observed data. In this context auto-encoders were leveraged to detect anomalies as well as an automatic system to extract features compressing observed data, to reduce traffic between edge nodes and the central authority.

While edge nodes have superior computing capabilities with respect to IoT devices, they could not provide resources to perform intensive tasks such as heavyweight ML model training. Previous work has foreseen this issue proposing systems which don't require intensive operations. Sudqi et al. [48] have proposed a host IDS running on energy-constrained devices. Sedjelmaci et al. [49] have proposed a more advanced system which makes a trade-off between energy consumption and detection accuracy. Their system is composed of a signature-based IDS, which is more energy efficient but may yield a high number of false positives, and an anomaly-based IDSs, which requires more power to operate but performs a more accurate analysis. During operation, only the signature-based IDS is active. When an alert is raised, it is forwarded to the anomaly-based IDS, which can confirm or discard it. Moreover, the system is formulated as a security game model, where the anomaly-based IDSs carry out its predictions based on the Nash Equilibrium. A drawback is that the cloud must be always up and running for the system to work correctly. Anomaly detection techniques could be used not just to detect network intrusions, but could also be used as a means of detecting bugs in devices' firmware or deviations from the normal state of a system. In the context of Industrial IoT (IIoT), work has been proposed to detect such anomalies.

Utomo et al. [50] develop a system performing anomaly detection on power grids sensor readings. Anomaly alerts could be used not only as an indication of an illegal intrusion, but also as a means to ensure grid safety preventing failures and blackouts. To perform the anomaly detection, due to the high non-linearity of the readings, an ANN based on Long-Short Term Memory (LSTM) cells is used. LSTM neural networks belong to the family of Recurrent Neural Networks (RNNs), a class of ANN architecture which excels in processing data in sequence, such as a sequence sensor readings or a sequence of words in the field of Natural Language Processing (NLP).

Niedermaier et al. [51] found that a single IDS running on the network perimeter could not be able to monitor, capture and analyze all the events. They proposed a distributed IDS based on multiple IIoT agents' edge devices and a central unit which unifies the logs produced by them. At its core, the IDS performs anomaly detection using one-class classification techniques: the authors assume that they know the normal behavior of the system, which can be learned by the agents. The IDS is suitable to be run on low-power microcontrollers, since it does not require any intensive calculation. The authors have also developed a proof-of-concept implementation of the system, which is not usually done in similar work.

Hafeez et al. [52] proposed a lightweight technique, named IOT GUARD, to distinguish between malicious and benign IoT traffic, using a semi-supervised approach. Their approach is almost completely unsupervised, but it requires a small portion of labels to be verified by hand,

which makes the algorithm technically semi-supervised. It is based on Fuzzy C-Mean (FCM) clustering. To improve performances, the authors performed aggregation of same-host and same-service features of devices. This aggregation strategy is not based on packet timestamps, but over the n latest device connections. This brings an advantage since time-based aggregation aggregates features over a definite time, e.g. number of connections made in last t seconds between two devices A and B. The time-based aggregation strategy is not suited for detecting attacks where an attacker introduces a time delay between successive connection attempts. In contrast, connection-based aggregation techniques aggregate features over last n connections i.e. out of last n connections made by A how many terminated at device B. This technique accommodates the time delay added to successive connections. The evaluation was carried out using a private dataset which was not made available to the public. The achieved accuracy is good, however practical comparison with other existing solutions or any baseline algorithm was not made.

5. DEVICE CLASSIFICATION AT THE EDGE

Recently, efforts have been made to identify and classify IoT devices based on their network traffic fingerprint. Using network packets, classifiers could be built to categorize devices based on their device class (e.g. motion sensors, security cameras, smart bulb and plugs, etc.) or to learn device signatures. The construction of such signatures are a prerequisite for building an IDS, since they serve, for example, as ground truth when comparing known signatures to the ones extracted during operation. In this way, if unauthorized devices connect to the network or existing devices abruptly change their behavior signature, an IDS can promptly raise alerts. Detecting intrusions based only on network traffic is a requirement for IDSs designed to be deployed on the edge.

In this context, the signature or fingerprint of a device is a representation of the traffic generated by the device itself. Several combinations of features could be leveraged to construct the signature. To understand which features are more suitable than others, Desai et al. [53] developed a feature-ranking system for IoT device classification. The utility of each feature is based on statistical methods. In order to extract features from traffic flows, they considered time windows of 15 minutes, and a sub-portion of them, which they named "activity period", corresponding to the time passing from the reception of the first packet to the reception of the last packet device-wise. Based on the class of the device, this activity period can assume different lengths. In their testing, they trained classifiers using both all of the features and only the top-k ranked ones. They found that classifiers trained using only top-k features with $k = 5$, show only a relative $\approx 6\%$ drop in accuracy, meaning a great reduction in computational tasks can be achieved without impacting the accuracy.

Thangavelu et al. [54] proposed a distributed device fingerprinting technique, named DEFT, which recognizes IoT device fingerprint. In the system, the IoT gateways extract features from devices' traffic sessions. These features are then sent to central edge nodes, which gather them and train ML models and classifiers. These classifiers are then sent back to the gateways, which perform the final identification of the device. The system does not need to know in advance traffic signatures of the connected devices, since it can autonomously recognize new devices based on the extracted fingerprint. In particular, when a new device is connected to the network (or an existing device changes its usual traffic e.g. due to a firmware update) the classifier on the gateways marks its traffic as low-probability. In this case the gateway sends the captured features to the edge node. When another device belonging to the same unknown class (i.e. with the same traffic signature) connects to another gateway, this one also sends features to the edge node. Now the edge node is able to clusterize and to identify the new device category. If there is not a second device connecting to another gateway this strategy does not work. The whole system can be controlled as a Software Defined Network (SDN) function. The classification is carried out not packet-wise, which can be expensive in terms of resources, but flow-wise, selecting a fixed time window frame. This technique could be used for extracting fingerprints of IoT devices and ultimately as a building block for an anomaly-based IDS. Such an IDS could be deployed on the edge of the network, since the final application of the trained models is done at the gateway level.

Bai et al. [55] propose a device classification technique to identify new and unseen devices. This is a novelty when compared to the majority of other works, which in order to recognize a device they must have had some sort of training on that exact device. This technique is particularly useful to IDSs which have the objective of tracking and alerting when new or unauthorized devices connect to the network. The classification is done using information streams generated by devices and then using an LSTM-CNN model leveraging time dependencies of network traffic. First, for each captured packet features such as timestamp, length, and various addresses are saved. Then features are extracted dividing the traffic into time windows of length T seconds. The authors do not specify the value of T that they used in their experiments and it seems to be fixed, non-adaptive. They extracted features differentiating between incoming or outgoing packets and user (TCP, UDP, MQTT, HTTP) or control packets (ICMP, ARP, DNS). Various statistics of the packets are extracted. Finally, the processed data is given to an LSTM network learning an encoding of the data. The LSTM is then attached to a CNN network which learns the final classification. The network is trained with standard backpropagation algorithms. The achieved results are quite good in accuracy, even if there is room for improvement.

6. OPEN ISSUES FOR EDGE-ENABLED ARCHITECTURES

The edge network creates new attack surfaces to be exploited by malicious parties. In Fig. 1 is illustrated the architecture of an edge-enabled IoT application. Traditional IoT-oriented IDSs are placed on the gateway level or device level and their focus is to protect against malicious IoT devices. However, attacks may target specifically the edge network, making an edge node become malicious. These could be caused by a remote attack of if the node gets physically tampered with. Due to the pervasivity offered by the edge, nodes could be deployed in public areas, which facilitates an attacker to tamper the device. Any attacker in control of an edge node, could alter all the traffic passing through it. They can generate packet streams in the edge pretending to be a legitimate IoT gateway or device, or they can selectively-forward packets of interest and discard the others. The management of malicious edge nodes was considered in the literature, but most of the time it is not an automatic process.
Already existing IDSs could be used and deployed in an edge scenario, however some new challenges arise and hinder the reliability of the intrusion detection system. They include:

- *Traffic Encryption.* The IDS can be deployed on IoT gateways or more external edge nodes. If deployed on edge nodes, the traffic it observes is encrypted, assuming that the IoT devices and the cloud communicate through secure protocols. The same could happen if the IDS is deployed on IoT gateways and the IoT devices have a TCP/IP stack, i.e. they can directly communicate with the cloud and the gateway performs routing operations only. Packet encryption means that an IDS is not able to know the contained information and it can only perform operations based on non-encrypted fields, such as TCP/IP headers, timestamp etc.
- *High Resource Variability.* IDSs can leverage several techniques to carry out the detection, which can be highly variable in terms of required computational resources. However, also edge nodes show high computational resource variability, which could range from a commodity PC with specialized hardware to a Raspberry Pi. The problem that may arise is that the requested resources for the IDS to work are too high for the edge node which is running the system, which could add communication latency and could block the whole system execution. On the other hand, an edge node that offers a lot of resources costs more, and if the resources are not exploited by the IDS the extra cost is wasted. Edge IDSs should be adaptive to the available resources, using a variety of algorithms requiring different capabilities and selecting them based on the current execution platform.
- *Distributed IDS architecture on Edge/IoT.* Due to the resource variability between the IoT and the edge, the execution of IDS for the network edge should be somewhat distributed. A single IDS could be composed of many subsystems which cooperate for the correct working of the system or to improve the detection performance. The cooperation of different subsystems, however, brings distributed systems challenges into the intrusion detection system, increasing its complexity.
- *Aggregated traffic.* If the protocol stack of IoT devices and the protocol stack of the edge differ, it could make an observer on the edge, including an IDS, unaware of the source end device of a packet. This is caused by IoT gateways which receive packets from end devices using their specific IoT communication technology and craft new packets using the protocols of the network edge, such as TCP/IP. This issue and its aftermath will be illustrated in more detail in Section 6.1.

In order to develop communication schemes which are resilient to malicious edge nodes theory of distributed systems could be leveraged, treating edge nodes as potentially byzantine nodes [56] and treating each packet that goes through the edge as a byzantine consensus problem. However, theorems [56] state that, in a non-authenticated and partially synchronous communication scheme, it must hold $N > 3f$, where N is the number of parties and f is the maximum number of tolerated byzantine nodes, in order for a byzantine consensus to be successful. This, however, would require a transmission of the same packet from multiple edge nodes. Moreover, if the packet was originated from an IoT device, it would require the same device to send the same packet to multiple edge nodes, which is a waste of energy and network resources.
A possible solution to the aforementioned problem is using IPsec [57]. IPsec is a network protocol guaranteeing network layer security, offering authenticity, integrity (AH and ESP modes) and encryption (ESP mode only) to the packet header and data. In this way malicious edge nodes are no longer able to craft packets pretending to be a legitimate user of the network. However, some issues still persist:

- *IPsec does not protect against traffic rerouting or selective-forwarding.* Attackers could decide which packets to forward and which ones to discard (selective-forwarding). They could also route the packets with additional delay, which can impact the real time characteristic of the IoT application.
- *IPsec fails to guarantee security specification in a physical tampering scenario.* If a device gets tampered, attackers have the possibility to access the private keys of an edge node, compromising the whole IPsec secure communication scheme for the device.
- *IPsec increases fractional overhead.* In a usual IoT application, packets sent from IoT devices are a few

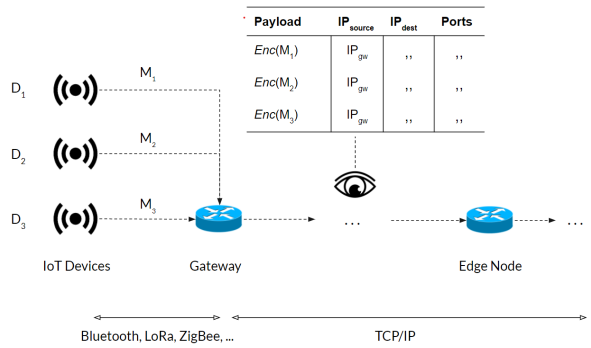
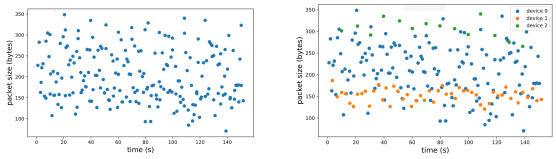


Fig. 4 – An example of why the edge node may be able to observe only the cumulative traffic, thus being unable to identify the end device which generated the observed packet. Some IoT devices send their data to the cloud. They first communicate to their gateway using their specific IoT communication technology. The gateway then crafts packets which will be sent to the edge and forwarded to the cloud, assuming using TCP/IP as protocols. The packets crafted by the gateway will have the same source IP, possibly the same destination IP (the same application server) and could use the same TCP ports. This causes any observer after the gateway, including an edge node, to be unaware of the devices behind the gateway. The edge node is only able to observe the cumulative traffic, without being able to identify the source device of an observed packet.



(a) *Unlabeled packets* **(b)** *Labeled packets*

Fig. 5 – Consider three IoT devices. Each device produces packets with its own mean length, its own mean time between them and own variances. In the plots, each dot represents a network packet. As we said in Section 6, the edge is not able to tell which IoT devices are connected and therefore it cannot assign a packet to its most likely IoT producer device. So what the edge observes is an "unlabeled" flow of packets, in Fig. 5a, not knowing the source/destination device of a packet. Applying anomaly detection strategies on the cumulative traffic yields poor performance, since too much variance is experienced by the algorithms. In Fig. 5b is depicted the same traffic but with packets labeled with their producer device. Applying anomaly detection on the labeled flow, should help algorithms to improve detection accuracy.

bytes long, meaning a low ratio of payload data over header data. The use of an additional control header increases even more the payload data, making the communication even more inefficient in terms of fractional overhead.

6.1 Aggregated traffic

Another problem is that the edge may not have the possibility to differentiate the traffic flows coming from the IoT devices, in other words it could only observe the aggregated traffic generated by all the devices combined as if it was generated by a single device. This issue verifies whether IoT devices and the edge have different protocol stacks and the gateway has to translate the protocols used by the IoT to the ones used by the edge/cloud. The observation of the aggregated traffic will cause both signature-based and anomaly-based IDS to carry out unreliable predictions.

Let us consider the scenario depicted in Fig. 4. We have IoT devices connected to the gateways with some IoT specific communication technology (BLE, LoRa, etc.) and the gateways connected to the edge and the cloud via TCP/IP. When the IoT devices send data to the cloud, they send a packet to their gateway using their IoT communication technology. The gateway then crafts a new TCP/IP packet and forwards it to the edge and to the cloud. This newly created packet by the gateway will have as source IP address the one of the gateway, regardless of which IoT end device produced it. Moreover, these packets could have the same IP destination address (same application server) and could use the same TCP ports for every IoT end device. This causes any observer beyond the gateways, including the edge nodes, to be unable to tell the source device of an observed packet. Being unable to separate the TCP flows, the edge node would regard the observed traffic as it was generated by a single device, since it has no means of knowing which devices are connected beyond the gateways.
The aggregated traffic poses problems for existing IDSs, both signature-based and anomaly-based:

- Signature-based IDSs cannot isolate packets coming from or going to the same device. This causes the inability to extract patterns from the observed traffic stream, thus making an IDS unable to recognize an attack signature. Methods could be developed to adapt existing signature-based IDSs to solve this issue, for example by mining patterns from the cumulative traffic. However, since the observed traffic is the sum of the various traffic streams generated by each device, there could be cases where a signature could be mistakenly marked as malicious. For instance, let's consider a pattern which is malicious only if generated by a single device (e.g. a particular exchange of messages between it and the server). If two or more devices generate non-malicious messages, it could be that when mining attack patterns, the sum of these flows generates a signature match. This increases the ratio of false positives.
- Anomaly-based IDSs will have to deal with the high variance of the aggregated traffic, since it is presumable for the cumulative traffic to have a higher variance than the traffic flows generated by each single IoT device. To carry out anomaly detection, an IDS has to learn the state of a system in a normal condition i.e. without an anomaly taking place. Then an anomaly is reported when the observed state deviates substantially from the expectation. If the normal state is learned via the cumulative traffic, too much variance could be experienced by the anomaly detection algorithm. The higher variance poses the risk that malicious anomalies are marked as non-malicious oscillations of the expectation, since these oscillations are acceptable given the variance of the normal system state. This increases the ratio of false negatives.

IDS approach	Effects of cumulative traffic observation	Result
Signature-based	Unable to reliably extract signatures from cumulative traffic	Increase of false positives.
Anomaly-based	Anomaly detection algorithm experiences too much variance	Increase of false negatives.

Table 1 – Summary of the expected issues that causes the observation of the cumulative traffic on the edge by an IDS. In the case of signature-based IDSs, the system is not able to extract precisely patterns and signatures from the traffic, ultimately increasing the ratio of false positives. In the case of anomaly-based systems, their algorithms would experience too much variance during the learning phase. This will cause an inexact anomaly report with a high ratio of false negatives.

Table 1 summarizes the effects of the cumulative traffic on existing anomaly-based and signature-based IDSs.
An example of anomaly detection on the aggregated traffic is illustrated in Fig. 5. An anomaly detection algorithm deployed in the edge, should learn the normal system behavior from the cumulative network traffic instead of device-wise traffic. However the cumulative traffic presents more variance than the traffic split in a device-wise manner, which could drastically impact the performance of the anomaly detection strategy. One first step to improve anomaly detection in the edge, could be to split the cumulative traffic into flows, one for each IoT device. Once this split is done, existing algorithms could be used to learn the normal behavior of the system, not from the cumulative traffic but from the flows of each device. However, this task could not be carried out by an edge node alone, since it doesn't have the knowledge of which IoT devices are connected beyond the gateways.

7. CONCLUSIONS

This work presents intrusion detection systems for IoT, both under the architectural perspective and under the methodologies that are used to let them capture anomalies and cyber attacks. As for the architectural perspective, while traditional IoT IDSs are deployed at the device level or gateway level the strong interest in having edge computing solutions offers new attack sides to be exploited by malicious parties. We discussed these new issues and present solutions that have been introduced to face them. New IDSs, specifically designed for the edge, are addressed. We also focus on the adoption of device classification techniques at the edge by discussing methodologies that could be leveraged by new IDSs targeted to IoT.

REFERENCES

[1] Anukool Lakhina, Mark Crovella, and Christophe Diot. "Diagnosing network-wide traffic anomalies". In: *ACM SIGCOMM Computer Communication Review* 34.4 (Oct. 2004), p. 219. DOI: 10.1145/1030194.1015492.

[2] A. L. Buczak and E. Guven. "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: *IEEE Communications Surveys Tutorials* 18.2 (2016), pp. 1153–1176. DOI: 10.1109/COMST.2015.2494502.

[3] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani. "A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security". In: *IEEE Communications Surveys Tutorials* 22.3 (2020), pp. 1646–1685.

[4] Ansam Khraisat, Iqbal Gondal, and Peter Vamplew. "An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2018, pp. 149–155. DOI: 10.1007/978-3-030-04503-6_14.

[5] Symantec. *Internet security threat report 2017*.

[6] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2.1 (July 2019). DOI: 10.1186/s42400-019-0038-7.

[7] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review". In: *Journal of Network and Computer Applications* 36.1 (Jan. 2013), pp. 16–24. DOI: 10.1016/j.jnca.2012.09.004.

[8] N. Ye, S. M. Emran, Q. Chen, and S. Vilbert. "Multivariate statistical analysis of audit trails for host-based intrusion detection". In: *IEEE Transactions on Computers* 51.7 (2002), pp. 810–820.

[9] Roberto Magán-Carrión, José Camacho, Gabriel Maciá-Fernández, and Angel Ruíz-Zafra. "Multivariate Statistical Network Monitoring–Sensor: An effective tool for real-time monitoring and anomaly detection in complex networks and systems". In: *International Journal of Distributed Sensor Networks* 16.5 (May 2020), p. 155014772092130. DOI: 10.1177/1550147720921309.

[10] V. Chatzigiannakis, S. Papavassiliou, and G. Androulidakis. "Improving network anomaly detection effectiveness via an integrated multi-metric-multi-link (M3L) PCA-based approach". In: *Security and Communication Networks* 2.3 (May 2009), pp. 289–304. DOI: 10.1002/sec.69.

[11] D. Brauckhoff, K. Salamatian, and M. May. "Applying PCA for Traffic Anomaly Detection: Problems and Solutions". In: *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*. IEEE, Apr. 2009. DOI: 10.1109/infcom.2009.5062248.

[12] Haakon Ringberg, Augustin Soule, Jennifer Rexford, and Christophe Diot. "Sensitivity of PCA for traffic anomaly detection". In: *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems - SIGMETRICS '07*. ACM Press, 2007. DOI: 10.1145/1254882.1254895.

[13] Ane Blázquez-García, Angel Conde, Usue Mori, and Jose A. Lozano. "A review on outlier/anomaly detection in time series data". In: *ArXiv e-prints* (2020). eprint: 2002.04236.

[14] Kieran Flanagan, Enda Fallon, Paul Connolly, and Abir Awad. "Network anomaly detection in time series using distance based outlier detection with cluster density analysis". In: *2017 Internet Technologies and Applications (ITA)*. IEEE, Sept. 2017. DOI: 10.1109/itecha.2017.8101921.

[15] Ying Zhao, Junjun Chen, Di Wu, Jian Teng, Nabin Sharma, Atul Sajjanhar, and Michael Blumenstein. "Network Anomaly Detection by Using a Time-Decay Closed Frequent Pattern". In: *Information* 10 (Aug. 2019), p. 262. DOI: 10.3390/info10080262.

[16] Aarthi Reddy, Meredith Ordway-West, Melissa Lee, Matt Dugan, Joshua Whitney, Ronen Kahana, Brad Ford, Johan Muedsam, Austin Henslee, and Max Rao. "Using Gaussian Mixture Models to Detect Outliers in Seasonal Univariate Network Traffic". In: *2017 IEEE Security and Privacy Workshops (SPW)*. IEEE, May 2017. DOI: 10.1109/spw.2017.9.

[17] Idan Amit, John Matherly, William Hewlett, Zhi Xu, Yinnon Meshi, and Yigal Weinberger. "Machine Learning in Cyber-Security -Problems, Challenges and Data Sets". In: *ArXiv e-prints* (Feb. 2019).

[18] Sumeet Dua and Xian Du. *Data Mining and Machine Learning in Cybersecurity*. Auerbach Publications, Apr. 2016. DOI: 10.1201/b10867.

[19] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. "Machine Learning and Deep Learning Methods for Cybersecurity". In: *IEEE Access* 6 (2018), pp. 35365–35381. DOI: 10.1109/access.2018.2836950.

[20] Arwa Aldweesh, Abdelouahid Derhab, and Ahmed Z. Emam. "Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues". In: *Knowledge-Based Systems* 189 (Feb. 2020), p. 105124. DOI: 10.1016/j.knosys.2019.105124.

[21] Thanh Thi Nguyen and Vijay Janapa Reddi. "Deep Reinforcement Learning for Cyber Security". In: *ArXiv e-prints* (July 2019).

[22] Shilin Qiu, Qihe Liu, Shijie Zhou, and Chunjiang Wu. "Review of Artificial Intelligence Adversarial Attack and Defense Technologies". In: *Applied Sciences* 9.5 (Mar. 2019), p. 909. DOI: 10.3390/app9050909.

[23] Ammar Alazab, Michael Hobbs, Jemal Abawajy, and Ansam Khraisat. "Developing an Intelligent Intrusion Detection and Prevention System against Web Application Malware". In: *Communications in Computer and Information Science*. Springer Berlin Heidelberg, 2013, pp. 177–184. DOI: 10.1007/978-3-642-40597-6_15.

[24] Ansam Khraisat, Iqbal Gondal, and Peter Vamplew. "An Anomaly Intrusion Detection System Using C5 Decision Tree Classifier". In: *Trends and Applications in Knowledge Discovery and Data Mining*. Ed. by Mohadeseh Ganji, Lida Rashidi, Benjamin C. M. Fung, and Can Wang. Cham: Springer International Publishing, 2018, pp. 149–155. ISBN: 978-3-030-04503-6.

[25] Karan Bajaj and Amit Arora. "Dimension Reduction in Intrusion Detection Features Using Discriminative Machine Learning Approach". In: *IJCSI International Journal of Computer Science*. Vol. 10. 2013.

[26] Salma Elhag, Alberto Fernández, Abdullah Bawakid, Saleh Alshomrani, and Francisco Herrera. "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems". In: *Expert Systems with Applications* 42.1 (Jan. 2015), pp. 193–202. DOI: 10.1016/j.eswa.2014.08.002.

[27] F. Salo, A. B. Nassif, and Aleksander Essex. "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection". In: *Comput. Networks* 148 (2019), pp. 164–175.

[28] Eduardo De la Hoz, Emiro De La Hoz, Andrés Ortiz, Julio Ortega, and Beatriz Prieto. "PCA filtering and probabilistic SOM for network intrusion detection". In: *Neurocomputing* 164 (Sept. 2015), pp. 71–81. DOI: 10.1016/j.neucom.2014.09.083.

[29] Jorge R. Vergara and Pablo A. Estévez. "A review of feature selection methods based on mutual information". In: *Neural Computing and Applications* 24.1 (Mar. 2013), pp. 175–186. ISSN: 1433-3058. DOI: 10.1007/s00521-013-1368-0.

[30] Shaoqian Wang, Bo Li, Mao Yang, and Zhongjiang Yan. *Intrusion Detection for WiFi Network: A Deep Learning Approach*. 2019. DOI: 10.1007/978-3-030-06158-6_10. URL: http://dx.doi.org/10.1007/978-3-030-06158-6_10.

[31] Constantinos Kolias, Georgios Kambourakis, Angelos Stavrou, and Stefanos Gritzalis. "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset". In: *IEEE Communications Surveys & Tutorials* 18.1 (2016), pp. 184–208.

[32] S. M. Danish, A. Nasir, H. K. Qureshi, A. B. Ashfaq, S. Mumtaz, and J. Rodriguez. "Network Intrusion Detection System for Jamming Attack in LoRaWAN Join Procedure". In: *2018 IEEE International Conference on Communications (ICC)*. 2018, pp. 1–6.

[33] F. Terenzi, P. Spadaccino, and F. Cuomo. "Edge computing enabling the Internet of Things". In: *23rd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*. 2022.

[34] Paria Jokar. "Intrusion Detection and Prevention for ZigBee-Based Home Area Networks in Smart Grids". In: *IEEE Transactions on Smart Grid* PP (Aug. 2016), pp. 1–1. DOI: 10.1109/TSG.2016.2600585.

[35] Andrea Lacava, Emanuele Giacomini, Francesco D'Alterio, and Francesca Cuomo. "Intrusion Detection System for Bluetooth Mesh Networks: Data Gathering and Experimental Evaluations". In: *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. 2021, pp. 661–666.

[36] S. Bräuer, A. Zubow, S. Zehl, M. Roshandel, and S. Mashhadi-Sohi. "On practical selective jamming of Bluetooth Low Energy advertising". In: *2016 IEEE Conference on Standards for Communications and Networking (CSCN)*. 2016, pp. 1–6.

[37] Farhoud Hosseinpour, Payam Amoli, Juha Plosila, Timo Hämäläinen, and Hannu Tenhunen. "An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach". In: *International Journal of Digital Content Technology and its Applications* 10 (Dec. 2016).

[38] Sarika Choudhary and Nishtha Kesswani. "A Survey". In: *International Journal of Information Security and Privacy* 13.1 (Jan. 2019), pp. 86–105. DOI: 10.4018/ijisp.2019010107.

[39] Anthea Mayzaud, Remi Badonnel, and Isabelle Chrisment. *A Distributed Monitoring Strategy for Detecting Version Number Attacks in RPL-Based Networks*. June 2017. DOI: 10.1109/tnsm.2017.2705290. URL: http://dx.doi.org/10.1109/TNSM.2017.2705290.

[40] Erdem Canbalaban and Sevil Sen. "A Cross-Layer Intrusion Detection System for RPL-Based Internet of Things". In: *Ad-Hoc, Mobile, and Wireless Networks*. Oct. 2020.

[41] O. Salman, I. Elhajj, A. Kayssi, and A. Chehab. "Edge computing enabling the Internet of Things". In: *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*. 2015, pp. 603–608. DOI: 10.1109/WF-IoT.2015.7389122.

[42] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran. "The Role of Edge Computing in Internet of Things". In: *IEEE Communications Magazine* 56.11 (2018), pp. 110–115. DOI: 10.1109/MCOM.2018.1700906.

[43] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli. "Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices". In: *IEEE Internet of Things Journal* 7.8 (2020), pp. 6882–6897.

[44] Rajinder Sandhu, Amandeep Sohal, and Sandeep Sood. "Identification of malicious edge devices in fog computing environments". In: *Information Security Journal: A Global Perspective* 26 (July 2017), pp. 1–16. DOI: 10.1080/19393555.2017.1334843.

[45] N. Pandeeswari and Ganesh Kumar. "Anomaly Detection System in Cloud Environment Using Fuzzy Clustering Based ANN". In: *Mobile Networks and Applications* 21 (Aug. 2015). DOI: 10.1007/s11036-015-0644-x.

[46] Ibbad Hafeez, Markku Antikainen, Aaron Yi Ding, and Sasu Tarkoma. "IoT-KEEPER: Detecting Malicious IoT Network Activity Using Online Traffic Analysis at the Edge". In: *IEEE Transactions on Network and Service Management* 17.1 (Mar. 2020), pp. 45–59. DOI: 10.1109/tnsm.2020.2966951.

[47] Joseph Schneible and Alex Lu. "Anomaly detection on the edge". In: *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)* (2017), pp. 678–682.

[48] Belal Sudqi Khater, Ainuddin Wahid Bin Abdul Wahab, Mohd Yamani Idna Bin Idris, Mohammed Abdulla Hussain, and Ashraf Ahmed Ibrahim. "A Lightweight Perceptron-Based Intrusion Detection System for Fog Computing". In: *Applied Sciences* 9.1 (Jan. 2019), p. 178.

[49] H. Sedjelmaci, S. M. Senouci, and M. Al-Bahri. "A lightweight anomaly detection technique for low-resource IoT devices: A game-theoretic methodology". In: *2016 IEEE International Conference on Communications (ICC)*. 2016, pp. 1–6.

[50] D. Utomo and P. Hsiung. "Anomaly Detection at the IoT Edge using Deep Learning". In: *2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW)*. 2019, pp. 1–2.

[51] Matthias Niedermaier, Martin Striegel, Felix Sauer, Dominik Merli, and Georg Sigl. "Efficient Intrusion Detection on Low-Performance Industrial IoT Edge Node Devices". In: *ArXiv e-prints* (2019). arXiv: 1908.03964 [cs.CR].

[52] Ibbad Hafeez, Aaron Yi Ding, Markku Antikainen, and Sasu Tarkoma. "Real-Time IoT Device Activity Detection in Edge Networks". In: *Network and System Security*. Springer International Publishing, 2018, pp. 221–236. DOI: 10.1007/978-3-030-02744-5_17.

[53] Bharat Desai, Dinil Mon Divakaran, Ido Nevat, G. Peters, and Mohan Gurusamy. "A feature-ranking framework for IoT device classification". In: Jan. 2019. DOI: 10.1109/COMSNETS.2019.8711210.

[54] V. Thangavelu, D. M. Divakaran, R. Sairam, S. S. Bhunia, and M. Gurusamy. "DEFT: A Distributed IoT Fingerprinting Technique". In: *IEEE Internet of Things Journal* 6.1 (2019), pp. 940–952.

[55] Lei Bai, Lina Yao, Salil S. Kanhere, Xianzhi Wang, and Zheng Yang. "Automatic Device Classification from Network Traffic Streams of Internet of Things". In: *ArXiv e-prints* (2018). arXiv: 1812.09882 [cs.NI].

[56] C. Dwork, N. Lynch, and L. Stockmeyer. "Consensus in the presence of partial synchrony". In: *Journal of the ACM (JACM)* 35.2 (Apr. 1988), pp. 288–323. DOI: 10.1145/42282.42283.

[57] S. Frankel, K. Kent, R. Lewkowski, A. D. Orebaugh, R. W.Ritchey, and S. R. Sharma. "Guide to IPsec VPNs". In: *National Institute of Standards and Technology* (2005). DOI: 10.6028/NIST.SP.800-77r1.

AUTHORS

Pietro Spadaccino received his B.S. in computer engineering and his M.S. in computer engineering from the University of Rome Sapienza, Italy in 2018 and 2020, respectively. He is currently pursuing a Ph.D. degree in information and communication technologies engineering at Sapienza. His research interests focus on Internet of Things, LoW power area networks and network security.

Francesca Cuomo received a Ph.D. in information and communications engineering in 1998 from Sapienza University of Rome. From 2005 to October 2020 she was an associate professor and from November 2020 she joined "Sapienza" as a full professor teaching courses in telecommunication networks. Prof. Cuomo has advised numerous master's students in computer engineering, and has been the advisor of 13 PhD students in networking. Her current research interests focus on: vehicular networks and sensor networks, low power wide area networks and IoT, 5G networks, multimedia networking, energy saving in the Internet and in the wireless system. Francesca Cuomo has authored over 158 peer-reviewed papers published in prominent international journals and conferences. Her Google Scholar h-index is 31, >3947 citations. Relevant scientific international recognitions: 2 Best Paper Awards. She has been on the editorial board of Computer Networks (Elsevier) and now is a member of the editorial board of Ad-Hoc Networks (Elsevier), IEEE Transactions on Mobile Computing, Sensors (MDPI), Frontiers in Communications and Networks Journal. She has been the TPC co-chair of several editions of the ACM PE-WASUN workshop, TPC Co-Chair of ICCCN 2016, TPC Symposium Chair of IEEE WiMob 2017, General Co-Chair of the First Workshop on Sustainable Networking through Machine Learning and Internet of Things (SMILING), in conjunction with IEEE INFOCOM 2019; Workshop Co-Chair of AmI 2019: European Conference on Ambient Intelligence 2019. She is an IEEE senior member.