Passban IDS: An Intelligent Anomaly Based Intrusion Detection System for IoT Edge Devices

Mojtaba Eskandari, Zaffar Haider Janjua, Massimo Vecchio and Fabio Antonelli

OpenIoT Research Area, FBK CREATE-NET, Italy
Email: {eskandari, janjua, mvecchio, fantonelli}@fbk.eu

***Abstract*—Cyber-threat protection is today one of the most challenging research branches of Information Technology, while the exponentially-increasing number of tiny, connected devices able to push personal data to the Internet is doing nothing but exacerbating the battle between the involved parties. Thus, this protection becomes crucial with a typical Internet of Things (IoT) setup, as it usually involves several IoT based data sources interacting with the physical world within various application domains, such as agriculture, health care, home automation, critical industrial processes, etc. Unfortunately, contemporary IoT devices often offer very limited security features, laying themselves open to always new and more sophisticated attacks and also inhibiting the expected global adoption of IoT technologies. Not to mention those millions of IoT devices already deployed without any hardware security support. In this context, it is crucial to develop tools able to detect such cyber-threats. In this paper, we present *Passban*, an intelligent Intrusion Detection System (IDS) able to protect the IoT devices that are directly connected to it. The peculiarity of the proposed solution is that it can be deployed directly on very cheap IoT gateways (*e.g.*, single-board PCs currently costing few tens USD), hence taking full advantage of the Edge Computing paradigm to detect cyber-threats as close as possible to the corresponding data sources. We will demonstrate that Passban is able to detect various types of malicious traffic, including Port Scanning, HTTP and SSH Brute Force, and SYN Flood attacks with very low false positive rates and satisfactory accuracies.**

***Index Terms*—IDS, Intrusion Detection, Anomaly Detection, Cybersecurity, Internet of Things, Edge Computing; Open-source.**

I. Introduction

THE Internet of Things (IoT) is a constantly evolving umbrella of technologies aiming at connecting diverse devices and everyday objects (*e.g.*, wireless sensor nodes, smart-phones, IP cameras, but also TVs, fridges and other home appliances) to the Internet. The fact that all of these devices are connected to the Internet (*i.e.*, the "network of the networks") makes it possible to provide "smart" digital services in various application domains, such as monitoring the moisture of crop fields in smart agriculture, tracking the flow of products of a factory in smart industry, remotely monitoring patients in smart health, monitoring and controlling appliances in smart homes, and so on and so forth. In a sentence, IoT is already influencing modern society in various aspects, ranging from personal healthcare services to manufacturing products in industries [1].

However, embracing such a paradigm shift in our daily lives increases the risk of data privacy breaches and cyber-security attacks [2]. Loosely speaking, the fact that the IoT is able to directly connect the physical world with the digital one has drawn significant attention towards cyber-attacks specifically designed to target IoT devices, and security countermeasures as a way for securing the connection between these two originally-separated worlds. A typical configuration of an IoT deployment consists of cheap, resource-constrained devices connected to a local IoT gateway in a way that data collected by the connected devices can flow through the gateway to/from the Internet [3]. Some of the most famous cyber-attacks caused by infected IoT devices are of very serious nature, such as the one happened in a German steel mill. Briefly, in 2015 the German Federal Office for Information Security issued a report confirming that some hackers breached a steel plant in their country, compromising numerous systems, including components of the production network. As a result, the personnel of the milling sector was unable to shut down a blast furnace when required, resulting in massive damage of the system [4]. This is not the only recent case of cyber-attack targeting an IoT system and/or devices. Recall the recent cyber-attack happened in a Ukrainian power grid [5], the shutting down of a floating oil rig [6], the hacking of kitchen devices in the United Kingdom [7], and the smart fridges converted into spam-bots [8], just to mention a few examples. Certainly, the most infamous attacks in the IoT history was that performed by the *Mirai* malware in 2017, infecting more than $380,000$ IoT devices to create wide-scale malicious botnets. These are literally "networks of bots", *i.e.*, compromised computing devices controlled by a remote command and control server in order to carry out Distributed Denial of Service Attacks (DDoS). To understand the seriousness of the infection, according to a report published by the US CERT [9], this malware is still considered as the largest DDoS attack ever recorded.

In order to mitigate the risk of threats on IoT devices, efficient and well responding Intrusion Detection Systems can be of help. An *Intrusion Detection System (IDS)* continuously monitors incoming data streams generated by diverse sources, and analyzes it to detect possible signs of cyber-threats [10]. There are two major approaches for threat detection: signature based and anomaly based. A signature is a pattern crafted to identify one or more already known attacks. Hence, a signature based IDS (also referred to as *rule based IDS*) simply compares the signatures against observed events and reports a threat when they match each other. Several issues related to signature based IDSs exist and the most relevant ones are summarized below:

- the first (and probably the most severe) issue is that they can only detect already-known attacks with well-investigated characteristics, while they fail to detect zero-day (*i.e.*, unknown) ones. Unfortunately, attackers keep evolving their strategies, also varying attack behaviors to bypass traditional security mechanisms [11].
- the second issue is that an increase in the number of newly identified attacks increases the number of signatures, resulting in more comparisons between stored patterns and incoming events. This increases the burden on the detection system, directly affecting the system's response, being this a very serious issue for real-time threat detection systems. For this reason, often such IDSs drop inspection of events at certain rates depending on the available processing resources [12].
- the third limitation is that they require human experts in the loop in order to study, analyze, and craft such signatures (*i.e.*, rules) for the new threats. This is not a secondary problem, as some studies report that it could take up to one year to investigate the characteristics of a specific attack [13].

All the above-mentioned limitations can be addressed by anomaly detection based techniques. As a matter of fact, an anomaly based IDS observes a sequence of incoming events while the system is not under attack and builds a model of the system's normal behavior. Then, the trained model detects anomalies based on a similarity index between normal and abnormal observations. The major challenge with this approach is to model a unique normal behavior of a system which is, in reality, a mixture of several underlying varying behaviors generated by individual data sources. Indeed, different types of data sources may generate diverse events which can increase the false positive rate due to less similarity between incoming events and learned normal behavior [14].

The motivation of this work spurs from the following very basic assumption: it is very hard (if not impossible, as per the today's availability on the market) to efficiently deploy a signature based IDS on a typical edge device of an IoT system. We gave shape to this assumption after reading several research papers in this field where, in essence, authors raised concerns about the applicability of signature based IDSs in scenarios where attacks could be unknown (see, for instance, [15], [16], [17]). Indeed, a signature based IDS cannot detect such unknown attacks, as it cannot have these definitions in its vocabulary of attacks. Even worse, in an IoT environment, an IDS is remotely deployed on low-cost IoT gateways, if not on tinier end-devices. This makes it more difficult to regularly update the definitions of attacks, as required in a signature based approach. The main technical goal of this paper is to provide a *lightweight software implementation* of an IDS suitable for a typical IoT gateway. Throughout this paper, "lightweight software implementation" will mean a software framework that can be deployed and executed directly on a cheap, commercially available IoT gateway, without depleting all of its CPU and memory resources. To map this definition onto today's technology, we identify the *Raspberry Pi 3, model B*[1] as our reference deployment board, given the rich hardware equipment it offers (*e.g.*, 4X CPU@1.2GHz, 1GB RAM@900MHz, on-board wired and wireless network adapters) with respect to its affordable cost (*i.e.*, less than 40 USD). Hence, by leveraging on the edge computing paradigm and by adopting only less demanding one-class classification techniques to build low-demanding machine learning models for anomaly detection based IDS, we aim to reduce the hardware requirements necessary for the hosting device or, in other terms, to keep its cost at bay. As a side benefit, to provide an IDS directly deployable on an IoT gateway let us accommodate more complex system architectures: in a geographically extended network model, for instance, a gateway is further connected to *unreliable and untrusted Internet* to deliver gateway data to a cloud infrastructure. In this configuration, a gateway is naturally exposed to direct and indirect attacks, so it is essential to endow it with its own IDS capabilities.

In this paper, we have designed, implemented, and evaluated the so-called "*Passban*" IDS, that is an anomaly based IDS able to analyze data generated by multiple IoT sources. More in detail, we focus on the following objectives:

- ensure data protection near the IoT data sources. To this aim, we designed an IDS residing as close as possible to the IoT devices, that is on the edge of the network (if not directly on the edge device). From a software development point of view, one of the main objectives is to deploy a lightweight IDS, deployable and executable on typical resource-constrained IoT gateways. As anticipated, an edge based IDS not only reduces the processing burden on the overall network but also ensures better security and protection services, being an edge device more sensitive to privacy issues due to its closeness to a data source.
- design an IDS able to easily scale after deployment, being the latter able to manage always new (*e.g.*, unknown) definitions of threats, without requiring hardware upgrades. For this reason, from a mere Machine Learning perspective, we only focus on lightweight anomaly detection algorithms. Clearly, different and more sophisticated algorithms may be also accommodated within the proposed system. In other words, in this paper, we are not interested in proposing any novel machine learning algorithm able to perform better with respect to the current state of the art. Rather, our intention is to prove that also cheap IoT devices can be used as protection add-ons in real-world IoT scenarios.
- deliver an IDS yet characterized by a reduced false positive rate and satisfactory detection accuracy, as we consider the latter as crucial requirements for an IDS to be truly effective.

The main contributions of this work can be summarized as:

- we present Passban, a platform-independent anomaly based IDS operating directly on edge devices and able to learn from the normal behavior of incoming IoT traffic.

[1]https://www.raspberrypi.org/products/

- we implement Passban as a dedicated module of the *AGILE* framework [18] which is thoroughly evaluated in a real-world environment against four common cyber-attacks, namely Port Scanning, HTTP Login Brute Force, SSH Login Brute Force, and SYN Flood attacks. However, it is worth to mention that the proposed IDS is generally suitable for detecting cyber-attacks characterized by network flow patterns that are slightly different from legitimate counterparts.
- we evaluate Passban in two different scenarios. In the first scenario, we use Passban as an IDS directly running on the gateway receiving data from IoT devices and from the Internet. In the second scenario, we use Passban as a *"Security-in-a-box"* infrastructural element which is a dedicated device receiving traffic from the Internet and from a local gateway.
- we deploy a real IoT testbed in a home automation scenario for intrusion detection analysis and collect real datasets that can be reused by the research community. Indeed, an interesting feature of Passban is that it can be configured to generate new datasets for various testbed settings.
- we show that Passban can be deployed and executed on very resource-constrained IoT gateway devices while, regarding its detection capability, we show that it is able to detect almost all malicious traffic with very low false positive rates and relatively high accuracies.
- we pack all the components of Passban into a Docker container[2] which is publicly available for the research and development open communities[3].

The remaining of this paper is organized as follows: Sec. II reviews the most recent related literature. Sec. III delves into the design and the implementation details of the proposed IDS, while Sec. IV presents the main advantages and limitations of our approach. Then, Sec. V describes the deployed testbed and the datasets construction, while Sec. VI deals with an extensive analysis of the obtained results. Finally, in Sec. VII we draw our conclusions.

II. Related Works

The recent related literature is rich in research works dealing with intelligent IDSs operating within traditional networks. Less has been done specifically for IDSs within IoT environments. Historically, the most challenging technological obstacle when dealing with IDSs deployed on IoT gateways is represented by the limited computational power of such devices, which makes it difficult to run a full-fledged IDS. Therefore, several approaches, even recently, had suggested addressing this issue by running the IDS out of the IoT device [19]. An analysis of the computational power of two traditional open-source IDS tools, namely SNORT [20] and BRO [21], required by a Raspberry Pi 2 board is provided in [22]. There, the authors used three cyber-attacks, namely SYN Flood, Address Resolution Protocol (ARP) Spoofing, and Port Scanning, with background traffic to compare the performance and detection rates of the aforementioned open-source IDSs. Interestingly, they reported that both tools were able to detect all the initiated network attacks while, regarding the measured overhead, SNORT performed slightly better than BRO. However, in our opinion, the most important finding of this experimental campaign was that even a tiny single-board PC such as Raspberry Pi 2 could be eventually able to host an IDS installation, at least in small-scale network scenarios.

Kasinathan et al. [23] introduced an IDS architecture for the protection of a Wireless Sensor Networks (WSN) and IoT wireless nodes based on the Suricata IDS [24]. In the proposed architecture, the IDS nodes are connected to a base station IDS through a wired connection, as the latter makes it resistant against wireless DoS attacks while also guaranteeing an instant message delivery service. However, since Suricata was not designed for non-IP networks, the novel contribution of this work can be relegated in the implementation of the required decoders that Suricata needed to understand IPv6 protocols.

Oh et al. introduced an optimized pattern matching approach for constrained devices [25] and evaluated it with intrusion pattern sets (*i.e.*, signatures) from SNORT for conventional network attacks and ClamAV [26], that is an open-source anti-virus for conventional operating systems.

However, most of the traditional intrusion detection systems (also including Suricata, SNORT, and BRO) are rule based (*i.e.*, they rely on a list of signatures, or *rules*, to detect attacks). Large rule lists are normally available for traditional networks having a large number of nodes. However, in small IoT networks, these lists incur heavy overheads, with bad consequences in terms of performance of the whole detection system. Furthermore, as introduced in Sec. I, due to their intrinsic nature, signature based IDSs are not able to detect new attacks. On the other hand, Passban (*i.e.*, the IDS proposed in this paper) is a lightweight IDS that uses an anomaly detection method which is able to detect new attacks with relatively high accuracy and a low false alarm rate. There exists similar anomaly detection IDSs, however, significant research is still required to develop an efficient IDSs for IoT systems. The following paragraph briefly reviews the already developed IDSs for IoT based scenarios.

Habibi et al. introduced a white list IDS approach called Heimdall to give protection from DDoS attacks like Mirai [27]. Briefly, Heimdall queries each URL (or DNS reply) to VirusTotal [28] in order to mark it either as malicious or benign. Then, the URL is added either to a whitelist or a blacklist and, from that moment on, only network traffic from whitelisted destinations is allowed to come to/go from devices behind the gateway. The major drawback with this approach is that its security relies on VirusTotal, which means that the portion of zero-day attacks not analyzed by VirusTotal yet remains undetectable by the gateway. Furthermore, a whitelist based approach restricts the functionality of the system by keeping the traffic to a new host in a queue to be analyzed by a remote service (*i.e.*, VirusTotal).

Wallgren et al. investigated the protection capabilities of the RPL against selective forwarding attacks [29].In a typical WSN, each node participates in communication by sending and receiving data packets. In this type of attack, the attacker

[2]https://www.docker.com/
[3]https://hub.docker.com/r/mojiz/passban-ids

node announces to its neighbors that it knows the shortest path to the desired destination, aiming at diverting the traffic to the certain path in order to discard the packets and harm the network communication.

Amaral et al. introduced the design of an IDS for WSN based on traffic signature detection for abnormal behaviors [30]. The IDS includes three modules that monitors the incoming traffic, namely *(i)* a packet monitoring module, which buffers and inspects the packets; *(ii)* a detection module, which applies all the specified rules that would trigger an alert; and *(iii)* an action module, which notifies anomalies to network administrator who would then take the necessary action.

Jun et al. developed a Complex Event Processing (CEP) engine for real-time anomalous pattern detection in the traffic generated by different IoT devices in a network [31]. They designed a rule based IDS based on an Event Processing Model. The rules have to be defined by an expert and be stored in a repository used by the IDS. According to the obtained result, their approach is quite CPU intensive but consumed less memory.

Riecker et al. presented an energy consumption analysis for IDSs specifically designed for WSN [32]. In this work, the authors monitored the power consumption of nodes in a WSN and then used a linear regression model to detect changes in power consumption matching denial of service attack patterns. The approach is able to distinguish several types of attacks made on WSNs and provides almost constant sensing information that makes the energy consumption nearly constant. However, energy consumption is not always steady in IoT devices due to the occurrence of diverse devices in a complex IoT network. In such an IoT network, often, nodes are heterogeneous and use various network protocols, settings, and dynamics. Therefore, this approach is not suitable to efficiently protect an IoT network.

SVELTE is a novel IDS for IoT, running on IPv6 networks [33]. It comprises two main modules, namely a server and a client. On the one hand, a server module builds the IPv6 network topology, maps it with IDS parameters and protects the sensor system from attacks coming from the Internet. On the other hand, The client module provides an IPv6 network topology mapper and computes lost packets. The authors evaluated SVELTE using a software simulator running on the Contiki Operating System [34] and it was able to detect rank attacks, inconsistencies in the network topology, sinkhole attacks, and selective forwarding attacks. The major limitation with this approach is that it requires the modification of sensors' software, which is difficult to carry out for networks having large numbers of sensing nodes, being this the typical case for many IoT application domains. Moreover, its focus is sharp on WSN related attacks, which are not always applicable to IoT deployments.

Linda et al. proposed an anomaly based IDS for critical infrastructures (*e.g.*, power plants, water supply facilities). The proposed IDS first analyzes the network traffic to build a reference model during a learning phase. The learned model is then used to make comparisons with real-time network traffic [35]. The authors used Artificial Neural Network (ANN) techniques to construct a model using both attack and normal traffic flows. Similarly, Hodo et al. used ANN to detect DoS/DDoS attacks in IoTs using a simulated network [36]. However, these schemes produced a significant number of false alarms and are intrinsically not able to recognize new/unknown attacks, as their classification models were built using known examples.

Lee et al. introduced a lightweight method that monitors node energy consumption for detecting intrusions [37]. By focusing only on one single node parameter, the authors attempted to minimize the computational resources needed for intrusion detection. In the distributed placement, nodes may also be responsible for monitoring their neighbors. Nodes that monitor their neighbors are referred to as watchdogs. This technique is useful for homogeneous networks like a WSN, while in an IoT network nodes are usually heterogeneous and show a varying power consumption behavior. Therefore, power consumption duty cycle pattern cannot be a distinguishable parameter for intrusion detection.

Krimmling et al. proposed a modular IDS framework for securing smart city IoT networks using the Constrained Application Protocol (CoAP) [38]. They simulated a number of routing attacks (replay, drop, and insertion attacks) and situations such as bit flips, byte exchanges, and modifications of entire data fields that can be related to a man-in-the-middle attack for evaluation. Their approach is lightweight, however, it is limited to mitigating only the routing attacks. Moreover, due to the poor performance of their solution, the authors concluded that a hybrid approach for intrusion detection involving both signature based and anomaly based IDSs should be the path for improving detection performances.

Cervantes et al. proposed an approach named INTI (Intrusion detection of Sinkhole attacks on 6LoWPAN for IoT) that combines concepts of trust and reputation with watchdogs [39]. They composed a hierarchical structure by classifying nodes into two groups of leaders and associated (or member) nodes. The role of each node can change over time due to network reconfiguration or an attack event. Each node monitors a superior node by estimating its inbound and outbound traffic. When a node detects an attack, it broadcasts a message to alert other nodes and to isolate the attacker. However, the authors did not discuss the impact of the solution in low capacity nodes.

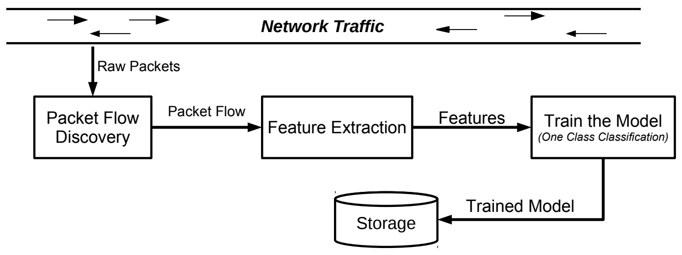
Midi et al. introduced Kalis, which observes a network and configures an IDS based on network parameters and configurations in order to activate detection methods that are possible/suitable for that particular network [40]. Kalis uses both signature based and anomaly based detection methods. Furthermore, it obtains knowledge from modules installed in the network and attempts to prevent DoS attacks based on the current network topology, traffic analysis, and mobility information. The major limitations of Kalis are that it focuses on routing attacks only and it requires installation of particular detection modules for detecting each type of attack. This raises the network complexity and could lead to poor detection performance.

Machine Learning tools have been widely used in intrusion detection systems to detect threats. In [41], Elrawy et al. presented a comprehensive survey of IDSs along with techniques for IoT systems. In another similar work, Chaabouni et al. discussed IoT security threats and challenges in detail [42]. In [43], Li et al. discussed the feasibility and sustainability of statistical methods to be used in IoT based IDS.

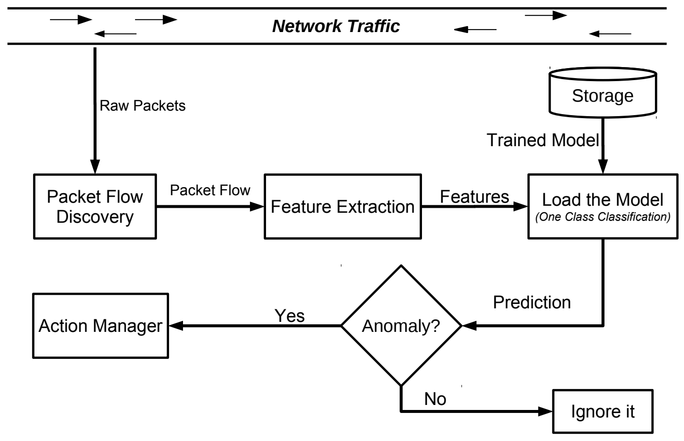
However, due to the limited hardware resources of the IoT devices that we address in this work, we have to carefully select the data analysis tools and algorithms to deploy. Similarly to [44], [45], where the authors employed one-class classification algorithms to detect specific botnet threats, we leverage on the same family of machine learning algorithms to detect different threats. Indeed, one-class classification algorithms constitute a relatively new concept within the framework of IoT based IDSs, though they result to be particularly effective whenever the stringent hardware constraints of a typical IoT device prevent the adoption of more sophisticated classification methods.

Some one-class classification techniques, such as isolation forest, have been widely used in various domains to detect anomalies. However, there are not many related works in the domain of IDSs for IoT scenarios. For example, in [46] Ding and Fei proposed a model using an isolation forest algorithm to detect anomalies in streaming data. Their work is conceptually close to ours, as IoT data is streaming data by definition. However, the nature of intrusions could vary in each case, which makes it more difficult and challenging to detect anomalies (threats) as compared to the simple case of stream data analysis. In another similar work, Vartouni et al. proposed a model to detect web attacks [47]. There, the authors used auto-encoders for feature extraction and applied an isolation forest algorithm to detect the anomalies. However, this work specifically targeted web applications, without considering specific IoT data, applications, and devices.

For the sake of a more visual recap, Table I summarizes the most significant research works reviewed in this section.



(a) Training phase



(b) Prediction phase

Fig. 1. Main operational phases of the Passban IDS.

III. *The Passban IDS*

In this section, we explain how the *Passban* IDS works. First of all, Fig. 1 shows the block diagrams of the proposed IDS during its main operational phases, namely the training and prediction phases. These diagrams also show the networks flow direction and the interfaces between the involved components of the IDS. More in detail, Passban is composed of the following main components:

a) Packet Flow Discovery block: it constantly observes network traffic by capturing raw packets and extracting network flows.

b) Feature Extraction block: it calculates network flow statistics and builds a feature set used to train the machine learning model. The feature set is usually built with the support of a network security expert to ensure an optimized performance of IDS. The feature set used in this work is aligned with the state of the art in this domain. Therefore, the interested reader is referred to the dedicated research literature (*e.g.*, [48], [49], [50], [51]). Passban IDS utilizes a Network Measurement and Accounting Meter tool called *"NetMate"* [52]. This tool simply listens to the network traffic and converts raw packets into network flows and then computes metrics for the flows. To capture the network traffic, NetMate employs the LibPCAP [53] library.

c) Train/Load model block: the training phase of the IDS is done on normal network flow, *i.e.*, while the system is running under an attack free mode. At the end of a training phase, the learned model is stored/deployed in the internal memory of the gateway so that it can be used to detect attacks for the new incoming network traffic. In order to keep the trained model updated, a user should initiate a new training session whenever the network configuration changes (*e.g.*, after the provision of new IoT devices). To keep the re-training procedure simple, in Passban a new training session can be launched by just pushing a software button in the web based management interface, while no prior technical knowledge on network administration or machine learning is required to the user.

d) Action Manager procedure: this module is responsible to take necessary actions according to the decisions generated by the learned model for attack detection. A few examples of such actions are logging attack incidents, blocking the incoming/outgoing traffic to a target device, notifying a particular user about the attack, etc.

e) Web Management Interface: it enables user's interaction with the IDS, including cleaning and exploring logs, controlling the activation status of IDS, managing the training procedure, etc.

TABLE I
A SUMMARY OF THE INVESTIGATED APPROACHES TO DESIGN INTRUSION DETECTION SYSTEMS WITHIN IOT CONTEXTS.

Approach	**Detection Method**	**Security Threat(s)**	**Evaluation**	**Dataset**	**ML Algorithm**
Kasinathan et al. [23]	Signature based	DOS (flooding attack)	Empirical	generated by Scapy	N/A*
Oh et al. [25]	Signature based	Several conventional attacks	Empirical	N/A	N/A
Habibi et al. [27]	White List	DDOS (Selective forwarding attacks)	Empirical	A designed testbed	N/A
Wallgren et al. [29]	A heartbeat protocol	Routing attacks, DOS	Empirical	N/A	N/A
Amaral et al. [30]	Specification based	N/A	Simulation	N/A	N/A
Jun et al. [31]	Specification based	N/A	N/A	N/A	N/A
Riecker et al. [32]	Anomaly based	Routing attacks, DOS	Simulation	N/A	Linear Regression
Raza et al. [33]	Network Topology Analysis	Routing attacks (Sinkhole and selective-forwarding attacks)	Simulation	N/A	N/A
Linda et al. [35]	Anomaly based	N/A	Simulation	Generated by the authors	Neural Networks
Hodo et al. [36]	Anomaly based	DOS, DDOS	Simulation	N/A	Neural Networks
Lee et al. [37]	Anomaly based	Routing attacks, DOS	Simulation	N/A	N/A
Krimmling et al. [38]	Hybrid (Anomaly+Signature)	Routing attacks	Simulation	N/A	N/A
Cervantes et al. [39]	Hybrid (Anomaly+Signature)	Sinkhole attack	Simulation	N/A	N/A
Midi et al. [40]	Hybrid (Anomaly+Specification)	Routing attacks, DOS	Simulation	N/A	N/A

* The parameter is not available in the referenced paper.

A. Machine Learning in Passban IDS

Passban is an intelligent intrusion detection system which relies on a machine learning technique to learn the normal behavior of the system. After the training phase, it uses the learned model to detect anomalous events occurring in the incoming network traffic. In general, there are two categories of machine learning techniques to detect interesting patterns present in data: supervised and unsupervised machine learning. In the specific case of IDSs, supervised techniques can be used to distinguish between two known types of network traffic (*i.e.*, malicious and benign). Based on the labeled data, a supervised machine learning model defines boundaries to uniquely classify incoming data into such distinct classes. To produce classification models able to distinguish normal traffic from malicious one with satisfactory accuracies, it is essential to use well labeled and balanced training data. However, in the case of anomaly based IDSs, it is difficult to build such balanced training sets containing well-labeled examples from both the normal and abnormal classes. Indeed, the availability of such data is limited, due to the scarce number of examples and the high frequency of emergence of new attacks. For this reason, supervised learning approaches are more useful is signature based IDSs.

In this paper, we address the challenging situation in which network attacks are new, hence unknown. Briefly, exploiting the fact that the characteristics of an attack are often significantly different from normal traffic, we focus on learning the normal behavior of the network traffic flow with the aim of detecting those patterns that are slightly different from it, to tag them as *anomalies*. In other words, we only assume to know the behavior of normal traffic, whereas the behavior of any attack is unknown or, at least, few examples are known and used to train a machine learning model. In this specific situation, a very useful learning strategy is available, namely *one-class classification*. The latter is a special case of unsupervised machine learning since it does not need labeled data to build its classification model (*i.e.*, all training instances conceptually belong to one class). Once the model is trained, it is used to detect patterns which have not been learned as *normal* by the model during its training phase. Since the model inference is not computationally intensive, this strategy is particularly useful for the real-time network traffic monitoring, in which anomalies are required to be promptly detected.

Several one-class classification algorithms are available, which are mainly based on two types of approaches, namely profiling (*i.e.*, defining a boundary/profile of normal instances) and isolation (*i.e.*, explicitly isolating anomaly instances from normal instances). In general, profiling methods are known to suffer from drawbacks which can negatively affect the detection of anomalies in IoT data, as explained next. First, a profiling method learns key features of normal behavior and it is optimized to detect what looks normal as compared to other instances (possible anomalies), without optimizing anomaly detection. Consequently, it may result in an increased false positive rate, which must be avoided by any effective IDS. Second, profiling methods are normally computationally intensive and in some cases, the computational cost increases exponentially with an increase in data instances or in the number of features. On the other hand, isolation methods are found to be computationally more economical as compared to profiling methods, as they converge rather quickly. Moreover, isolation methods work better even when some attributes are irrelevant to detect anomalies [54]. In the case of IoT contexts, this latter feature can be particularly useful as data coming from various IoT devices have different attributes and some attributes might not be useful or relevant to detect a particular type of anomaly. In this paper, we explore both approaches to better understand and empirically prove which one performs better in detecting anomalies directly on an Edge device. To this aim, we consider the Isolation Forest as an isolation based algorithm, and the Local Outlier Factor as a profiling based method. A brief description of these methods is given in the following:

*a) Isolation forest (*i*Forest):* it is an ensemble method which exploits the fact that data instances containing anomalies are few and characterized by attribute values which are quite different from the normal data instances. Therefore, it is possible to isolate such data instances from the normal data ones. The algorithm generates a forest of data induced random trees, in which each tree is built by recursively partitioning the instances until all the instances are isolated. The anomalous instances are detected by taking the length of branches of all trees in the forest. As a consequence of random partitioning, the instances having anomalies are represented by shorter paths in the tree, whereas the normal instances are represented with relatively longer paths. Thus, having the shortest branch length, the anomalies are automatically isolated from the normal tree. For a more detailed review of this algorithm, the interested reader is referred to as [54].

b) Local Outlier Factor (LOF): it is a density based method in which the distance between given data points is measured to estimate their density in a locality [55]. The density estimation is based on a comparison between distances measured of a point with its k-nearest neighbors. Considering, these measured distances, the data points belonging to denser regions having similar density are considered normal, against data points occurring in the lower density regions which are considered outliers. The LOF algorithm makes a profile of normal data points and anomalies lie outside of the developed profile. Since LOF is based on measuring distances between given data points, the computation power and complexity increases with the increase in the data points in the sample space. Thus, considering an edge device, LOF is not a resource-friendly solution to be used to detect anomalies.

B. Feature Extraction

We extract several features from data to apply machine learning techniques on it. The full list of the features we use in our experimentation is given in Table II; there, each feature is also briefly introduced. These features were selected after a thorough analysis of the recent domain-specific literature, even if they did not specifically tackle IoT scenarios (*e.g.*, [48], [49], [50], [51]). To alleviate the fact that these studies were not IoT-specific, we decided to build our domain-specific datasets rather than leveraging on the generic, publicly available ones (see Sec. V-B).

C. Training and prediction phases

During the training phase (see Fig. 1(a)), a machine learning algorithm is trained to learn the concept of normality hidden in normal data instances. At the end of this phase, the trained model is stored in the local memory of the edge device (*e.g.*, an IoT gateway). Then, during the prediction phase (see Fig. 1(b)), the learned model is loaded back from the local memory of the edge device which can start predicting eventual anomalies in the network traffic. The output of the prediction block is a binary value (*i.e.*, *'Yes'* in case of an anomaly is detected in the data flow, *'No'* otherwise). Finally, all the detected anomalies are sent to an *Action Manager*, which is responsible to take necessary action(s) for the detected anomalies.

When Passban is executed for the first time, it trains a one-class classification algorithm modeling the normal behavior of the monitored network. For the sake of completeness, the training procedure is described in Algorithm 1, by means of pseudocode. Briefly, it takes the network interface Ni to be monitored by the IDS as the input argument, internally collects the network flow and trains a machine learning model object ML, returning such object as the output argument. Notice that at the beginning, the algorithm looks for an existing trained model (*line 2*); if such model exists in the local memory, then it is loaded (*line 3*), otherwise a new training session is started (*lines 6-12*) and the resulting training model is then stored in the local memory (*line 13*). Regarding the training session, it is executed for a predefined duration set by the user (possibly according to an expert's recommendation). Clearly, during this phase, we assume that Passban only receives normal traffic instances (*i.e.*, without anomalies). Moreover, as already introduced, to keep the model up-to-date (especially after a network topology change), new training sessions can be manually initiated by the user. In these cases, a new one-class classification algorithm is learned from scratch and, upon completing the training session, the new model is applied as a filter on the incoming network traffic to detect anomalies.

```
Input: $Ni$: Network interface;
Output: $ML$: Trained Machine Learning Object;
1  $ML$ = new MachineLearning();
2  if ( Exist trainedModel) then
3      $ML$.loadModel( trainedModel);
4  end
5  else
6      $ds$ = new Dataset();
7      $nf$ = new NetFlow();
8      for ( $nf$ in $Ni$.getNetFlow()) do
9          $nf$ = $Ni$.getNetFlow();
10         $ds$.add($nf$);
11     end
12     trainedModel = $ML$.train($ds$);
13     store( trainedModel);
14 end
15 return $ML$;
```
Algorithm 1: Passban initialization procedure.

D. The Action Manager Module

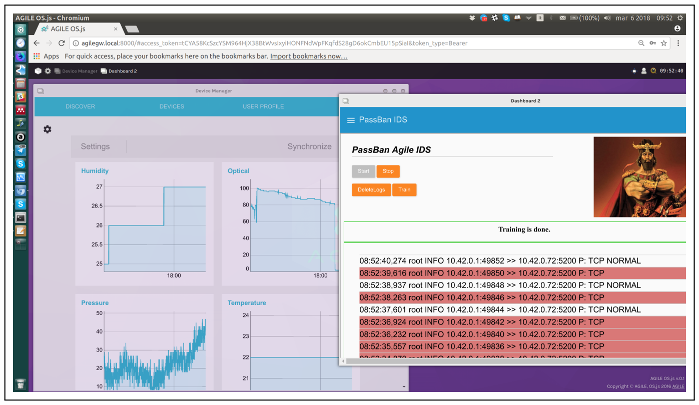
Whenever Passban tags an incoming network flow as an anomaly, the action manager module (AM) is notified in order to take proper action(s). The default action is to log the details of detected attacks. The AM module provides interfaces for API callback in order to communicate with other applications or devices in order to act based on the detected anomalies. As simple examples, it can block the traffic, send notifications to the network administrator, switch off critical devices based on a predefined threat alert policy, etc. However, any further development of such module falls outside the focus of this work.

E. The Web Manager Interface

The web manager interface listens on a predefined network port and provides an interface to the Passban IDS. The web

TABLE II
FEATURE SET USED IN THE EXPERIMENTATION.

Type	Feature ID	Description
Traffic volume	max_fpktl	The size of the largest packet sent in the forward direction (in bytes)
	max_bpktl	size of the largest packet sent in the backward direction (in bytes)
	mean_fpktl	mean size of packets sent in the forward direction (in bytes)
	mean_bpktl	mean size of packets sent in the backward direction (in bytes)
	min_fpktl	size of the smallest packet sent in the forward direction (in bytes)
	min_bpktl	size of the smallest packet sent in the backward direction (in bytes)
	sflow_fbytes	average number of bytes in a sub flow in the forward direction
	sflow_bbytes	average number of bytes in a sub flow in the backward direction
	total_bvolume	total bytes in the backward direction
	total_fvolume	total bytes in the forward direction
	std_bpktl	standard deviation from the mean of the packets sent (in bytes)
	total_bhlen	total bytes used for headers
Packets statistics	sflow_fpackets	average number of packets in a sub flow in the forward direction
	sflow_bpackets	average number of packets in a sub flow in the backward direction
	total_bpackets	total packets in the backward direction
	total_fpackets	total packets in the forward direction
Time statistics	mean_active	The mean amount of time that the flow was active before going idle (in microseconds)
	mean_fiat	mean amount of time between two packets sent in the forward direction (in microseconds)
	max_active	maximum amount of time that the flow was active before going idle (in microseconds)
	max_fiat	maximum amount of time between two packets sent in the forward direction (in microseconds)
	min_active	minimum amount of time that the flow was active before going idle (in microseconds)
	min_biat	minimum amount of time between two packets sent in the backward direction (in microseconds)
	min_fiat	minimum amount of time between two packets sent in the forward direction (in microseconds)
	duration	duration of the flow (in microseconds)



```
08:52:40,274 root INFO 10.42.0.1:49852 >> 10.42.0.72:5200 P: TCP NORMAL
08:52:39,616 root INFO 10.42.0.1:49850 >> 10.42.0.72:5200 P: TCP
08:52:38,937 root INFO 10.42.0.1:49848 >> 10.42.0.72:5200 P: TCP NORMAL
08:52:38,263 root INFO 10.42.0.1:49846 >> 10.42.0.72:5200 P: TCP
08:52:37,601 root INFO 10.42.0.1:49844 >> 10.42.0.72:5200 P: TCP NORMAL
08:52:36,924 root INFO 10.42.0.1:49842 >> 10.42.0.72:5200 P: TCP
08:52:36,232 root INFO 10.42.0.1:49840 >> 10.42.0.72:5200 P: TCP
08:52:35,557 root INFO 10.42.0.1:49836 >> 10.42.0.72:5200 P: TCP
```

Fig. 2. The Web interface of the proposed IDS running on an popular open source IoT gateway. The red rows indicate the suspicious network flows detected by Passban.

interface shows the status of the IDS and displays alerts generated for the suspicious network flows. Moreover, it enables a user to start/stop the IDS, change the mode of the IDS from detection to training mode, and is able to clean the detection logs. Technically speaking, the web management interface has a PHP back-end interacting with the Passban core modules and log manager through Linux pipes [56] and an HTML/Javascript front-end. A screenshot of the web Manager interface is shown in Fig. 2.e focus of this work.

IV. Passban as an IDS for IoT gateways

The emergence of open-source hardware has facilitated a growth on the product development for the IoT [57]. One of the most promising -yet fully open at both software and hardware levels- IoT frameworks is the *AGILE* (Adaptive Gateways for dIverse muLtiple Environments) project [18]. Briefly, it provides a modular hardware and software gateway for the IoT with support for protocol interoperability, device and data management, IoT app execution, and external cloud communication. In order to promote the widest adoption of the proposed IDS, we designed, implemented and deployed it in a Docker container environment [58] which is fully integrated into the AGILE gateway. Since it is provided in the form of a container, it can be seamlessly integrated into any IoT gateway supporting Docker containers, hence becoming truly platform-independent.

Fig. 3 depicts the overall system architecture, confirming that Passban is physically deployed on the IoT gateway, hence very close to the IoT devices that are, at their turn, connected to the latter. As we discuss in the following, this locality represents one of the main advantages of Passban.

A. The locality advantage

Passban focuses on collecting incoming and outgoing network traffic of IoT devices and constructing a comprehensive model of network traffic of the system when it is working regularly (*i.e.*, in an attack-free situation). In an IoT network hierarchy (see Fig. 3), as we move from the IoT end devices layer towards the Cloud layer, the data streams generated from individual devices are merged together at each higher level. Thus, at IoT device level the data stream is composed only of data generated by the device itself while at gateway level data is already the merge of data coming from several IoT devices. At the next higher level (*e.g.*, at router level) data may be the merge of traffic coming from several gateway devices, and so on and so forth. This aggregation of multiple streams at each level may exhibit more generic characteristics,

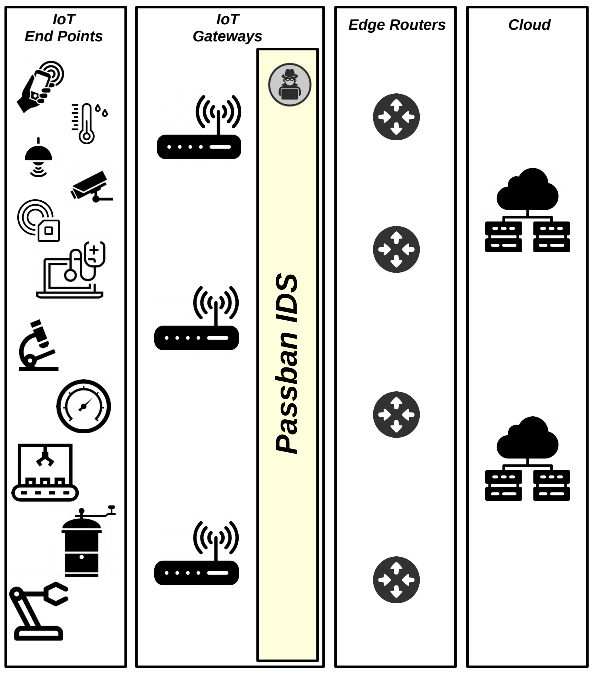


Fig. 3. Block diagram of the system architecture (note where the Passban IDS is logically located with respect to the involved hardware entities).

rather than device-specific characteristics, which may reduce the performance of an IDS when detecting threats. In principle, in order to achieve the highest accuracy for an IDS, the best option for an IDS installation would be locally (*i.e.*, at IoT device level itself). However, common IoT devices usually have very limited processing capability and, in some cases, they are battery-powered. Therefore, a sub-optimal place for an IDS to provide protection is at the IoT gateway, which is the case of Passban.

B. Main limitations

Although Passban is a promising research-oriented tool, from a development perspective it still lacks full maturity. Indeed, the following are the main limitations that should be addressed before evolving it into a technologically ready software:

1) Passban assumes that the IoT network is not under attack while the IDS is in training mode. However, in this domain, this is a typical assumption, shared also with the recent state of the art literature, see for instance [15], [51];
2) Passban may signal an anomaly even if the incoming traffic contains a pattern which is not an attack, but somehow it diverges from the routine traffic (*e.g.*, a false positive);
3) a new training session might be necessary when the underlying network changes (*e.g.*, network topology changes). Such a process usually imposes a significant overhead on the gateway that during this phase may not able to quickly serve the requests;
4) when the system is under an SYN Flood attack, the gateway's computational resources are exhausted and therefore, Passban's detection rate is lower than its detection rate for other types of attacks. This aspect will be better discussed in Sec. V.

V. A REAL TESTBED FOR PERFORMANCE ASSESSMENT

To accurately assess the performance efficiency in terms of threat detection accuracy and computational power requested by the proposed IDS, a customized testing environment (*i.e.*, an *IoT testbed*) is needed. Indeed, an accurate measurement campaign means, first of all, the availability of suitable datasets containing traces of various types of threats commonly present in real IoT deployments. To this aim, we set up a real IoT testbed. This testbed is then configured in two different scenarios and, on these two scenarios, real network traffic is collected to produce a number of datasets that are then used to extensively evaluate the performance of the proposed IDS. In this section, first, we explain the testbed setting and the two specific deployments. Then, we summarize the main characteristics of the built datasets, giving also information on the attacks we performed on our testbed. Finally, we dive into the software implementations details.

A. Testbed Setup

In order to set up a typical testbed for home automation scenario, we used a *Raspberry Pi 3 model B* running the AGILE gateway software. Such gateway directly interacts with a range of IoT devices and with a cloud back-end. We consider the following IoT devices for our experimental setup:

- *Texas instrument BLE SensorTag*[4] endowed with the following sensors
  - *TMP007*: Temperature sensor;
  - *BMP280*: Altimeter/Air pressure sensor;
  - *OPT3001*: Ambient light sensor;
  - *DHC1000*: Humidity sensor;
  - *MPU-9250*: 9-axis motion sensor.
- *FosCam FI8910W* as WiFi IP Camera[5]

Fig. 4 depicts the architecture of our testbed. Briefly, the Texas Instrument sensors communicate with the AGILE gateway through Bluetooth technology. We used an LED on a sensor board as an actuator and configured the AGILE gateway to turn it on when the temperature reaches a specific threshold limit value. Then, in order to include a variety of IoT devices generating various types of network traffic, we configured the AGILE gateway in such a way that it is able to automatically scan for new devices, connect to multiple IP cameras through the local WiFi network, and to automatically transmit the collected video stream to a cloud endpoint. As Fig. 4 shows, Passban IDS monitors incoming and outgoing traffic from/to the Internet on the AGILE gateway.

We used TCPDump [53] as a Network Dumping Tool to collect raw network traffic coming to/from the AGILE gateway. Since we need to evaluate the anomaly detection method of Passban IDS when it is under attack, we had to include an attacker component in our testbed. Such an

[4]http://www.ti.com/sensortag
[5]http://foscam.ca/products/foscam-fi8910w-wireless-ip-camera.html

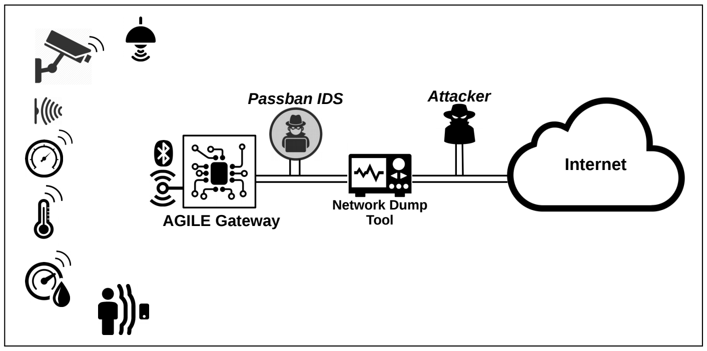


Fig. 4. The testbed architecture.

attacker component is responsible for generating various types of attacks while considering possible vulnerabilities in an IoT setup. For the attacker component, we considered the Metasploit framework (*i.e.*, a purposely designed tool to develop and execute exploit code against remote target machines [59]) running on a Kali Linux distribution (*i.e.*, a Debian derived Linux distribution specifically designed for digital forensics and penetration testing [60]).

In order to perform a particular attack on an IoT gateway, first, we need to discover what sort of services externally accessible is in execution on the gateway. As a matter of fact, if the gateway is not running any web server, it does not make sense to perform a web attack on it. An effective way to gather such information is to perform a *port scanning* procedure. We used *Nmap* tool to perform port scanning [61]. Nmap (Network Mapper) is a security scanner tool designed to discover hosts and services on a computer network, thus building a map of the network. In order to accomplish this goal, Nmap sends special packets to the target host and then analyzes corresponding responses. It provides a number of features for probing computer networks, including host discovery and service and operating system detection. Therefore, it reveals publicly accessible services running on the gateway. It is important to notice that port scanning itself is considered as a threat, as it is used to initiate an attack. Therefore, even if the AGILE gateway was not an open-source framework, by making use of this attack, we were able to discover that it is running at least one web server and that the *SSH* port was open. Therefore, we configured Metasploit for relevant attacks and collected the network traffic while AGILE gateway was under attack.

B. Dataset Construction

There exist IoT datasets publicly available in the literature, such as the ones used in [44], [45], [62]. However, these datasets could not perfectly match our requirements. First of all, they are related to very specific attacks (*e.g.*, only Botnet, only Telnet, etc.). Conversely, our goal is to develop an IDS able to detect a wider range of attacks targeting IoT gateways. Moreover, we need a scalable dataset which enables us to add new attacks in various settings. Therefore, we opt for building our own dataset collecting network traffic from our in-house testbed. The latter can be purposely configured to collect network flows and then to extract some features commonly accepted within the related literature. In doing so, we avoid considering features which are static with respect to the environment (*e.g.*, source/destination IP addresses and source/destination port numbers), only including those related to flow of packets, timings involved in the traffic flow, and data flow statistics over a particular time interval. Since Passban is an anomaly based IDS, we can exploit the fact that it can be trained while it is observing the network traffic of the target system in normal state (*e.g.*, not under attack). During this phase, the IDS is able to build a model representing the system's normal behavior. In other words, by using one-class classification algorithms to detect anomalies, the proposed IDS only needs normal traffic instances during its training phase. From a methodological point of view, this also means that for Passban the distribution of malicious network flows against the benign flows does not affect the detection accuracy, and this simplify the validation phase. Indeed, with this approach, we can build our own test sets (*i.e.*, attack datasets) simply by collecting network traffic while conducting real attacks on the target network and then labeling as attack flows those containing network packets with packet information of the specific attack (*e.g.*, IP addresses of the known infection). The counter-effect is that we cannot have full control on the distribution of normal flows against malicious one for a given attack performed.

With this approach, to evaluate the performance of Passban, we collect individual datasets by configuring our testbed in two different scenarios:

- in Scenario 1, we setup the testbed as shown in Fig. 4;
- in Scenario 2, we setup Passban in a *"Security-in-a-box"* configuration and collected network packets passing through Passban to construct the dataset.

a) Scenario 1: to collect the data for this scenario, we configured the AGILE gateway to communicate with an IP camera and other sensors. The gateway collects data from these sensors and transmits it to a cloud infrastructure, while *Tcpdump* captures inbound and outbound traffic from the cloud, dumping it in PCAP files. In order to extract features from the raw network traffic of each PCAP file, we calculate network statistics using NetMate tool. To acquire training data, we let the system run without any attack (*i.e.*, attack free mode) for 12 hours, collecting around 17.3 millions of raw network packets. Afterward, we launched four different types of attacks on the AGILE gateway, collecting the inbound/outbound raw traffic. A brief description of these attacks is given in the following:

- **Port Scanning**: as mentioned earlier, port scanning enables reconnaissance on the target system to discover possible vulnerable points.
- **HTTP Brute Force**: AGILE, like almost every other IoT gateway, provides a web interface to interact with various sensors, applications, and services. Thus, it runs a local web server on the gateway itself. Usually, this web interface is protected via a pair of *username/password* credentials. Therefore, it creates an attack vector for an intruder in order to brute force a dictionary of credentials

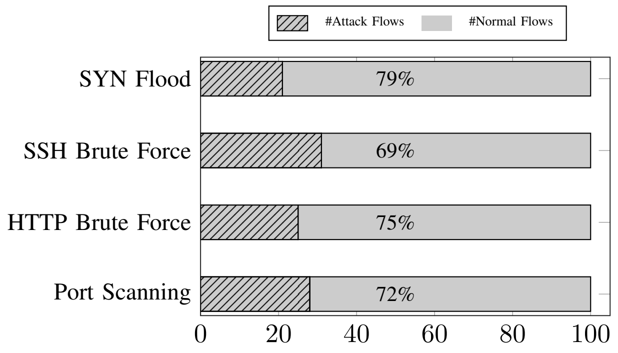


Fig. 5. Scenario 1: Distributions of attack flows (dark gray) and normal flows (light gray) against the total flows of the collected traffic (percentage values are referred to the ratio of normal flows against malicious ones).

and get unauthorized access to the gateway.

- **SSH Brute Force**: the SSH protocol is usually used by a system administrator to communicate with the gateway, for maintenance purposes. It opens another attack vector for an intruder to get unauthorized access to the gateway via SSH.
- **SYN Flood**: it is a form of denial of service (DOS) attack in which an attacker bombards a target system with a large number of SYN (*i.e.*, synchronize) requests. It is an attempt to consume enough server resources in order to make the system unresponsive to legitimate traffic. Since this attack requires less computational resources to be launched (*e.g.*, a Raspberry Pi or a laptop can be sufficient), it could be especially harmful to IoT gateways. In order to perform SYN flood attacks, we use *Hping3*, a TCP/IP packet assembler/analyzer [63] which is also used for security auditing and testing of firewalls and networks.

Fig. 5 depicts the distribution of attack flows (malicious network flows) in the collected traffic for this scenario. The detailed number of attack flows versus benign flows is given in Table III. In this dataset, the benign traffic comprises of 3761 network flows which are used to model the normal behavior of the system. There are 57 port scanning malicious network flows which collect the required information in order to perform an attack. HTTP and SSH brute force attacks are composed of 36 and 389 network flows respectively. To have a pragmatic approach, the datasets are built by mixing a number of benign network flows and malicious ones. Finally, the SYN flood attack is composed of 31 attack flows and 67 normal network flows.

TABLE III
SCENARIO 1: NUMBER OF ATTACK FLOWS AND NORMAL FLOWS.

Attack Name	#Normal Flows	#Attack Flows
Normal Traffic	3761	0
Port Scanning	148	57
HTTP Brute Force	106	36
SSH Brute Force	870	389
SYN Flood	117	31

b) Scenario 2: in the second scenario, we configured Passban IDS as a hardware security element usable to protect an off-the-shelf home automation IoT system. For this experiment, we employed the FIBARO Home Center Lite (HCL) [6]. We modified the testbed setup for this experiment as illustrated in Fig. 6. It can be noticed that Passban provides the Internet connectivity to FIBARO HCL and observes its network traffic flowing from/to the cloud. We configured FIBARO HCL to use the following IoT end devices:

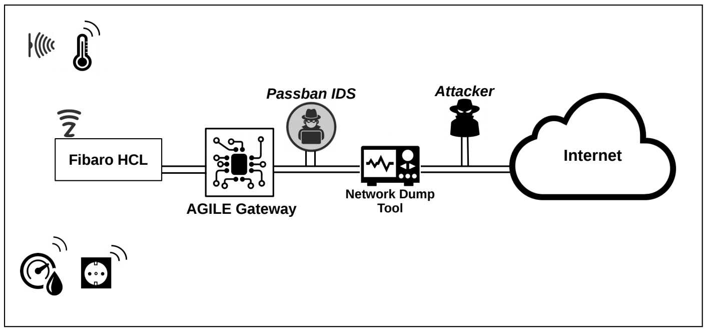


Fig. 6. The architecture of Scenario 2 (*e.g.*, *"Security-in-a-box"*), where an AGILE gateway running the Passban IDS protects a commercial home automation IoT system.

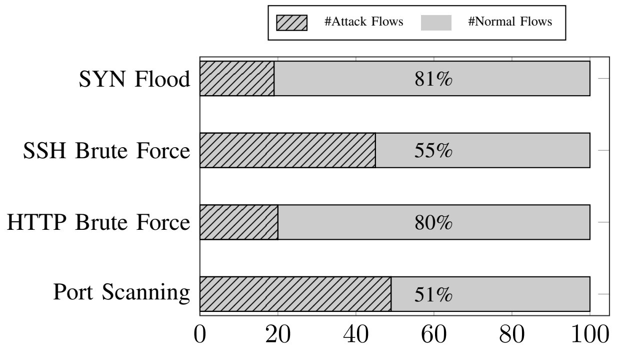


Fig. 7. Scenario 2: Distributions of attack flows (dark gray) and normal flows (light gray) against the total flows of the collected traffic (percentage values are referred to the ratio of normal flows against malicious ones).

- Smart wall plug & power meter
- Motion sensor
- Flood sensor
- Door/window sensor
- Smoke/CO sensor

The FIBARO gateway interacts with the IoT end devices, collects data and transmits it to a private cloud infrastructure. Thus, Passban IDS stays in between the gateway and the cloud to observe IoT data, learn the normal behavior of the network traffic and to report suspicious network flows.

We perform the same attacks on the FIBARO system as we did in the previous scenario. Fig. 5 depicts the distribution of attack flows (malicious network flows) in the collected traffic for the "Security-in-a-box" scenario. Numerical details are also provided in Table IV. In this dataset, the benign traffic comprises of 6845 network flows, which are collected in about 23 hours of letting the system run, collect sensory data and transmit it to the cloud. There are 58 port scanning malicious network flows while HTTP and SSH brute force attacks are

[6]https://www.fibaro.com/en/

composed of 45 and 184 network flows respectively. Finally, the SYN flood attack is composed of 16 attack flows and 67 normal network flows. As already remarked, since the experiment is done in a real-world scenario, the collected datasets are composed of a mixed number of benign network flows and malicious ones.

TABLE IV
SCENARIO 2: NUMBER OF ATTACK FLOWS AND NORMAL FLOWS.

Attack Name	#Normal Flows	#Attack Flows
Normal Traffic	6845	0
Port Scanning	61	58
HTTP Brute Force	176	45
SSH Brute Force	227	184
SYN Flood	67	16

C. Software Implementation

We implemented various components of the Passban IDS using *Python*. Specifically, we used the Scikit-learn[7] implementations of the anomaly detection algorithms (*i.e.*, *i*Forest, and LOF). Each algorithm required a preliminary tuning phase of the characterizing parameters. The majority of these parameters were tuned using a mixture of grid and manual searches [64], also guided by a certain familiarity matured with the technological framework and adopted techniques [65], [66]. For the sake of brevity, the details of this preliminary phase are omitted, while in the following we only mention the values of some key parameters, as they were set during our evaluation campaign. Specifically, for *i*Forest, we set the number of base estimators in the ensemble to 50, while for the LOF algorithm, we opt for the Euclidean distance to calculate distances between points in the feature space.

VI. RESULTS AND DISCUSSIONS

In this Section, we summarize and analyze the results obtained by Passban on the two scenarios, in terms of machine learning related metrics (see Sec. VI-A) and hardware resources utilization (see Sec. VI-B).

A. Machine Learning based performance evaluation and analysis

In order to assess the performance of Passban from a pure machine learning perspective, we start from the classical terms of a confusion matrix [67], namely true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), recalling that, in our experiments, a TP occurs when Passban correctly detects an attack (*i.e.*, a *hit*), a TN occurs when Passban correctly detects a normal flow (*i.e.*, a *correct rejection*), an FP occurs when Passban incorrectly detects an attack (*i.e.*, a *false alarm*), and an FN occurs when Passban fails to detect an attack (*i.e.*, a *miss*).

However, it is also important to recall that Passban is an anomaly detection system. Regardless of the domain, by definition, an anomaly is very rarely manifested, as compared to normal instances. In all these circumstances, usually, the terms of the confusion matrix are further aggregated to compute Precision (P), Recall (R) and their harmonic mean (*i.e.*, the F-measure, also known as $F1$ score). The equations used to calculate these accuracy metrics are given in Table V). Finally, it is important to recall that FP is still an expressive performance indicator when assessing an IDS. Indeed, a large number of false alarms not only reduces the system performance but also decreases the reliability of the whole system. Therefore, one of the main objectives of Passban remains that of minimizing FP. In terms of the aggregated metrics of Table V, this is equivalent to say that a good anomaly detection system has to be characterized by high values of P (which is a metric telling how many anomalies were correctly detected by the IDS).

[7]https://scikit-learn.org/stable/

TABLE V
MACHINE LEARNING BASED EVALUATION METRICS USED TO ASSESS THE PERFORMANCE OF PASSBAN.

Performance Measure	Formula
Precision	$P = \frac{TP}{TP+FP}$
Recall	$R = \frac{TP}{TP+FN}$
F-Measure	$F1 = 2 \times \frac{(P \cdot R)}{P+R}$

Table VI and Table VII summarize the main performance indicators computed for Scenario 1 and Scenario 2, respectively, grouped by type of attack. There, to improve the readability, for each group and each aggregated metric, the best values are marked in bold.

In Scenario 1, LOF and *i*Forest are able to detect all the tested attacks with satisfactory accuracies. More in detail, *i*Forest reaches always the best values in terms of both precision and recall, hence also in terms of $F1$: 0.99, 0.96, 0.96 and 0.90 for Port Scanning, HTTP Brute Force, SSH Brute Force, and SYN Flood, respectively. In Scenario 2, that is when the proposed IDS is deployed on a dedicated device, receiving aggregated traffic from the IoT gateway, as expected, the configuration is not optimal. This is confirmed by the decreased performances of the LOF algorithm on three of the four tested attacks (*i.e.*, this algorithm is only able to accurately detect SSH Brute Force attacks). *i*Forest seems more robust to the change of scenario, always preserving his relative ranking against the other algorithm, and also its accuracy, except when detecting SYN Flood attacks (in this case, the $F1$ score decreases to 0.79).

B. Resource utilization based performance evaluation and analysis

Machine Learning related metrics are only one side of the medal. As said, we are equally interested in assessing the possibility of deploying Passban on a common edge device. For this reason, in this Section, we summarize the results in terms of hardware resources needed by a Raspberry Pi 3 model B to run the proposed IDS. For the sake of brevity, in the following we focus on the Scenario 1, when using a trained *i*Forest model deployed on the target board and, for the latter, we will measure the values of CPU load, memory usage, and network throughput. More in detail, to measure the CPU load

TABLE VI
PERFORMANCE EVALUATION RESULTS FOR SCENARIO 1.

Attack	*Technique*	*#Normal*	*#Attack*	*FP*	*TP*	*FN*	*TN*	*Precision*	*Recall*	*F1*
Port Scanning	iForest	148	57	1	57	0	147	**0.98**	**1**	**0.99**
	LOF	148	57	10	52	5	138	0.84	0.91	0.87
HTTP Brute Force	iForest	106	36	2	35	1	104	**0.95**	**0.97**	**0.96**
	LOF	106	36	7	35	1	99	0.83	**0.97**	0.89
SSH Brute Force	iForest	870	389	9	370	19	861	**0.98**	**0.95**	**0.96**
	LOF	870	389	7	302	87	863	**0.98**	0.78	0.87
SYN Flood	iForest	117	31	2	27	4	115	**0.93**	**0.87**	**0.9**
	LOF	117	31	5	27	4	112	0.84	0.87	0.85

TABLE VII
PERFORMANCE EVALUATION RESULTS FOR SCENARIO 2.

Attack	*Technique*	*#Normal*	*#Attack*	*FP*	*TP*	*FN*	*TN*	*Precision*	*Recall*	*F1*
Port Scanning	iForest	61	58	2	56	2	59	**0.97**	**0.97**	**0.97**
	LOF	61	58	1	25	33	60	0.96	0.43	0.59
HTTP Brute Force	iForest	176	45	1	43	2	175	**0.98**	**0.96**	**0.97**
	LOF	176	45	4	21	24	172	0.84	0.47	0.6
SSH Brute Force	iForest	227	184	3	183	1	224	**0.98**	**0.99**	**0.98**
	LOF	227	184	18	176	8	209	0.91	0.96	0.93
SYN Flood	iForest	67	16	1	11	5	66	**0.92**	**0.69**	**0.79**
	LOF	67	16	2	9	7	65	0.82	0.56	0.67

and memory usage, we installed `sysstat` tool pack [68] on the target device. Then, a purposely written shell script invokes the `sar` tool from `sysstat` to monitor the CPU load and memory usage every second, storing this information in two separate files. Regarding the network performance evaluation, we use `iperf` [69], a tool able to generate traffic to measure bandwidth capacity and network throughput. More in detail, it first creates a server and a client instance and then transmits data between them. In our case, the target Raspberry Pi is set as the server, while a dedicated PC is configured as the client.

Fig. 8 shows the average network throughput, the CPU load, and the memory usage of the system with and without Passban on our experiment. Regarding memory consumption in Passban, there are clearly multiple factors involved. First of all, Passban is running a tiny web-server instance that is used to serve the UI controlling and observing the IDS activities. Second, the packet flow discovery module requires to hold a number of network packets in the memory and process them to extract network flow for next level modules. Third, the classification module needs to keep the trained model in the memory to compare the incoming flows against it. Fourth, Passban uses Linux pipes as a shared memory to communicate amongst its various modules. Our experiment reveals that the gateway's memory usage is 24.68% when Passban is executing and 19.41% when it is not executing. In other words, Passban requires 54 MB of the main memory of the Raspberry Pi to execute the *i*Forest model on Scenario 1. On the same experiment, we measure an average CPU load of 47.17% when Passban is in one of these states: *i)* active and waiting for packets; *ii)* analyzing the network traffic; 3) with the web user interface open and a user is interacting with it. Finally, regarding the network capacity, when the network traffic rate is too high, then Passban uses all the available CPU to try to analyze as many network flows as possible. Our experiment reveals that the Raspberry Pi can handle maximum 93.9 Megabits per second (Mbit/s) data on its Ethernet port. Then, when we activate Passban, this bandwidth is reduced to 77.2 (Mbit/s).

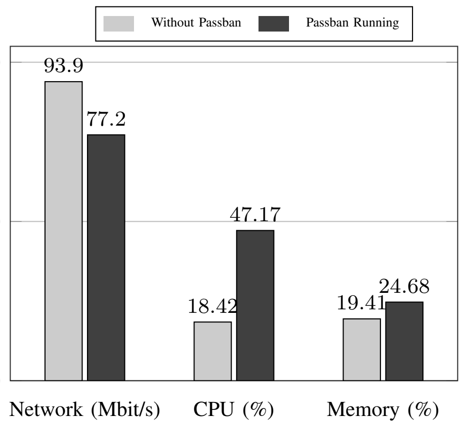


Fig. 8. Resource utilization of the target device (*i.e.*, a Raspberry Pi 3 Model B) when Passban IDS is in execution.

However, real-world network traffic has varying data rates. In order to evaluate the resource utilization of Passban in more realistic conditions, we perform another experiment which involves an incremental data rate for network traffic. More in details, we use `tcpreplay` [70] to replay previously collected network traces at the given speed, using the target device's Ethernet interface. The collected network traffic is composed of $246,137$ packets which yield $17,182$ network flows. We configure Passban to log all network flows including the suspicious and normal ones.

Fig. 9(c) shows the fraction of network flows successfully processed by Passban at varying data rates. Briefly, the capture rate is initially almost 100%, then it starts decreasing as the data rate increases. Comparing this graph with the CPU load one (Fig. 9(a)) reveals that there is a breaking point where the flow capture rate starts decreasing: this happens, indeed, when the CPU approaches full utilization. As a matter of fact, as the data rate increases, the number of network flows received grows and consequently the number of interruptions generated, until overcoming the maximum that the CPU can handle with its resources and will not be able to serve them all, hence new network flows will be discarded. However, as depicted in Fig. 9(b), the memory usage for varying rates of incoming data remained almost the same. We can conclude that the data rate does affect the memory required to handle the incoming traffic.

All in all, we conclude saying that, in a real IoT environment, when reasonable data rates are expected, then Passban can be successfully hosted on a Raspberry Pi 3 model B and used as an IDS.

VII. Conclusion

In this paper, we presented Passban, an intelligent anomaly based intrusion detection system (IDS) purposely designed to be directly hosted and executed by a typical edge device (*e.g.*, a cheap IoT gateway). First of all, we built an IoT testbed able to resemble a typical smart home automation environment. Then, with the goal of assessing the performance of the proposed IDS in terms of both threat detection accuracy and computational resources needed by the executing device, we configured the IoT testbed as two different scenarios, namely Scenario 1 and Scenario 2. Briefly, in Scenario 1, the IDS is directly deployed and executed on the IoT gateway: in this case, Passban is able to protect the latter and all the IoT devices directly connected to it. In Scenario 2, Passban is provided as a separate add-on device independently connected to the network it has to protect: in this case, it can monitor incoming and outgoing network traffic of the IoT devices connected to the gateway, inspecting them for suspicious patterns occurring in the network traffic. One of the most attractive features of the proposed IDS is that it trains itself autonomously using legitimate traffic flows of the IoT target network using lean one-class classification machine learning algorithms. In this respect, an anomaly is meant (and reported by Passban) as a network flow significantly deviating from the learned normal behavior. We implemented Passban using two one-class classification techniques (namely *i*Forest and LOF) and evaluated it against four common attacks (namely, port scanning, HTTP brute force, SSH brute force, and SYN flood attack). In terms of threat detection accuracy, our experimental evaluation revealed that using *i*Forest, Passban IDS can achieve F1 scores greater than 0.9 on some attacks (0.99 in the best case and 0.79 in the worst case). In terms of resource utilization, we proved that Passban can be executed even on cheap IoT gateway boards like a Raspberry Pi 3 model B, especially when the data rate on its network interface is not expected to exceed 40-50 Mbits/s.

Acknowledgment

This work was supported by the AGILE project (grant agreement No 688088) within the Horizon 2020 programme of the European Union.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[2] M. Ammar, G. Russello, and B. Crispo, "Internet of things: A survey on the security of iot frameworks," *Journal of Information Security and Applications*, vol. 38, pp. 8–27, 2018.

[3] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, 2017.

[4] R. Becker, "Cyber attack on german steel mill leads to massive real world damage," *PBS Magazine*, 2015, last accessed: January 28, 2020. [Online]. Available: http://www.pbs.org/wgbh/nova/next/tech/cyber-attack-german-steel-mill-leads-massive-real-world-damage/

[5] L. Robert M., A. Michael J., and C. Tim, "Analysis of the cyber attack on the ukrainian power grid," *Electricity Information Sharing and Analysis Center (E-ISAC) Online at https://ics.sans.org/media/E-ISACSANSUkraineDUC5.pdf*, 2016.

[6] J. Wagstaff. (2014) All at sea: global shipping fleet exposed to hacking threat. Last accessed: January 28, 2020. [Online]. Available: http://reut.rs/1rnmjdI

[7] J. Lima, "Iot security breach forces kitchen devices to reject junk food," *Computer Business Review Magazine*, 2015, last accessed: January 28, 2020. [Online]. Available: https://www.cbronline.com/news/iot-security-breach-forces-kitchen-devices-to-reject-junk-food-4544884/

[8] M. Starr, "Fridge caught sending spam emails in botnet attack," *CNET Magazine*, 2015, last accessed: January 28, 2020. [Online]. Available: https://cnet.co/2oPzNJC

[9] "Heightened DDoS threat posed by mirai and other botnets," The Cybersecurity and Infrastructure Security Agency (CISA), Tech. Rep. Alert (TA16-288A), 2016, last accessed: January 28, 2020. [Online]. Available: https://www.us-cert.gov/ncas/alerts/TA16-288A

[10] K. Scarfone and P. Mell, "Guide to intrusion detection and prevention systems (idps)," National Institute of Standards and Technology (NIST), Tech. Rep. SP 800-94, 2007, last accessed: January 28, 2020. [Online]. Available: https://www.nist.gov/publications/guide-intrusion-detection-and-prevention-systems-idps

[11] "2016 Internet Security Threat Report," Symantec Co., Tech. Rep. 21, 2016. [Online]. Available: https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf

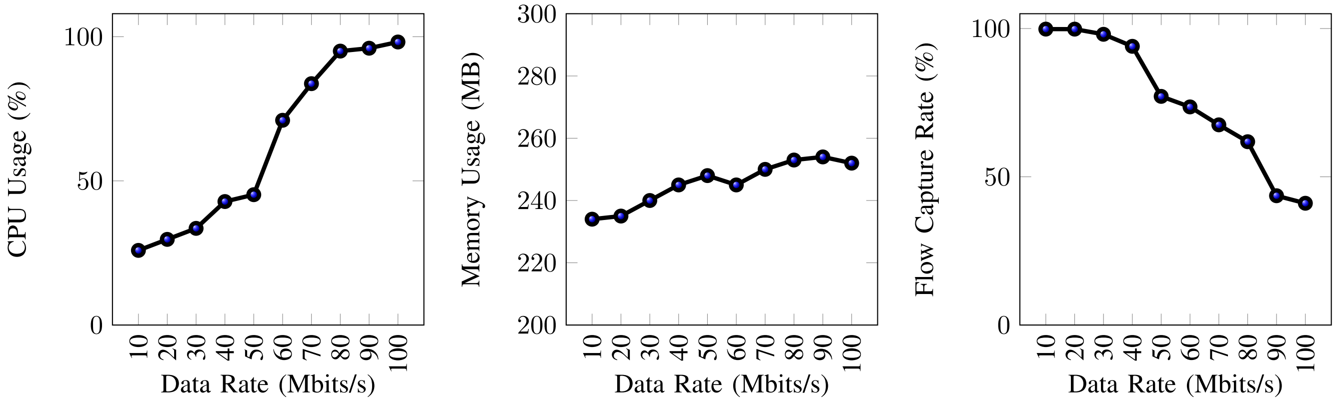
[12] E. Albin and N. C. Rowe, "A realistic experimental comparison of the suricata and snort intrusion-detection systems," in *Proc. of the 26th International Conference on Advanced Information Networking and Applications Workshops*, 2012, pp. 122–127.

[13] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proc. of the ACM Conference on Computer and Communications Security*, 2012, pp. 833–844.

[14] A. L.-H. Muna, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, 2018.

[15] H. H. Al-Maksousy, M. C. Weigle, and C. Wang, "Nids: Neural network based intrusion detection system," in *2018 IEEE International Symposium on Technologies for Homeland Security (HST)*. IEEE, 2018, pp. 1–6.

[16] R. Tahir, "A study on malware and malware detection techniques," *International Journal of Education and Management Engineering*, vol. 8, no. 2, p. 20, 2018.



(a) CPU Usage (b) Memory Usage (c) Network flow capture rate

Fig. 9. Relationship between monitored variables for Passban's resource usage performance and various network data rates. The variables include: CPU usage (a), Memory usage (b) and flow capture rate (c).

[17] H. Larijani, J. Ahmad, N. Mtetwa *et al.*, "A heuristic intrusion detection system for internet-of-things (iot)," in *Intelligent Computing-Proceedings of the Computing Conference*. Springer, 2019, pp. 86–98.

[18] The Agile IoT project website. Last accessed: January 28, 2020. [Online]. Available: http://agile-iot.eu

[19] L. Santos, C. Rabadao, and R. Gonçalves, "Intrusion detection systems in internet of things: A literature review," in *Proc. of the 2018 13th Iberian Conference on Information Systems and Technologies*. IEEE, 2018, pp. 1–7.

[20] M. Roesch, "Snort - Lightweight Intrusion Detection for Networks," in *Proc. of the 13th USENIX Conference on System Administration*, 1999, pp. 229–238.

[21] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Comput. Networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[22] A. K. Kyaw, Y. Chen, and J. Joseph, "Pi-ids: Evaluation of open-source intrusion detection systems on raspberry pi 2," in *Proc. of the 2nd International Conference on Information Security and Cyber Forensics*, 2016, pp. 165–170.

[23] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. a. Spirito, "DEMO: An IDS framework for Internet of things empowered by 6LoWPAN," in *Proc. of the ACM SIGSAC Conference on Computer & Communications Security*, 2013, pp. 1337–1340.

[24] The Suricata homepage. Last accessed: January 28, 2020. [Online]. Available: https://suricata-ids.org/

[25] D. Oh, D. Kim, and W. W. Ro, "A malicious pattern detection engine for embedded security systems in the internet of things," *Sensors*, vol. 14, no. 12, pp. 24 188–24 211, 2014.

[26] T. Kojm, "Clamav open source antivirus," 2019. [Online]. Available: https://www.clamav.net/

[27] J. Habibi, D. Midi, A. Mudgerikar, and E. Bertino, "Heimdall: Mitigating the internet of insecure things," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 968–978, 2017.

[28] The VirusTotal homepage. Last accessed: January 28, 2020. [Online]. Available: https://www.virustotal.com

[29] L. Wallgren, S. Raza, and T. Voigt, "Routing attacks and countermeasures in the rpl-based internet of things," *Int. J. Distrib. Sens. Netw.*, vol. 9, no. 8, p. 794326, 2013.

[30] J. P. Amaral, L. M. Oliveira, J. J. Rodrigues, G. Han, and L. Shu, "Policy and network-based intrusion detection system for ipv6-enabled wireless sensor networks," in *Proc. of the IEEE International Conference on Communications*. IEEE, 2014, pp. 1796–1801.

[31] C. Jun and C. Chi, "Design of complex event-processing ids in internet of things," in *Proc. of the 6th International Conference on Measuring Technology and Mechatronics Automation*. IEEE, 2014, pp. 226–229.

[32] M. Riecker, S. Biedermann, R. El Bansarkhani, and M. Hollick, "Lightweight energy consumption-based intrusion detection system for wireless sensor networks," *Int. J. Inf. Secur.*, vol. 14, no. 2, pp. 155–167, 2015.

[33] S. Raza, L. Wallgren, and T. Voigt, "Svelte: Real-time intrusion detection in the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, 2013.

[34] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *Proc. of the Workshop on Embedded Networked Sensors*, 2004.

[35] O. Linda, T. Vollmer, and M. Manic, "Neural network based intrusion detection system for critical infrastructures," in *Proc. of the International Joint Conference on Neural Networks*, 2009, pp. 1827–1834.

[36] E. Hodo, X. Bellekens, A. Hamilton, P.-L. Dubouilh, E. Iorkyase, C. Tachtatzis, and R. Atkinson, "Threat analysis of iot networks using artificial neural network intrusion detection system," in *Proc. of the International Symposium on Networks, Computers and Communications*, 2016, pp. 1–6.

[37] T.-H. Lee, C.-H. Wen, L.-H. Chang, H.-S. Chiang, and M.-C. Hsieh, "A lightweight intrusion detection scheme based on energy consumption analysis in 6lowpan," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing*. Springer, 2014, pp. 1205–1213.

[38] J. Krimmling and S. Peter, "Integration and evaluation of intrusion detection for coap in smart city applications," in *Proc. of the IEEE Conference on Communications and Network Security*. IEEE, 2014, pp. 73–78.

[39] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6lowpan for internet of things." in *Proc. of the IFIP/IEEE International Symposium on Integrated Network Management*, 2015, pp. 606–611.

[40] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalisa system for knowledge-driven adaptable intrusion detection for the internet of things," in *Proc. of the IEEE 37th International Conference on Distributed Computing Systems*. IEEE, 2017, pp. 656–666.

[41] M. F. Elrawy, A. I. Awad, and H. F. A. Hamed, "Intrusion detection systems for iot-based smart environments: a survey," *Journal of Cloud Computing*, vol. 7, no. 1, p. 21, Dec 2018.

[42] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for iot security based on learning techniques," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[43] F. Li, A. Shinde, Y. Shi, J. Ye, X. Li, and W. Z. Song, "System statistics learning-based iot security: Feasibility and suitability," *IEEE Internet Things J.*, pp. 1–1, 2019.

[44] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiotnetwork-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul 2018.

[45] V. H. Bezerra, V. G. Turrisi da Costa, S. Barbon Junior, R. Sanches Miani, and B. Bogaz Zarpelão, "One-class classification to detect botnets in iot devices," in *Proc. of the XVIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*. Porto Alegre, RS, Brasil: SBC, 2018, pp. 43–56. [Online]. Available: https://sol.sbc.org.br/index.php/sbseg/article/view/4242

[46] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12 – 17, 2013, 3rd IFAC Conference on Intelligent Control and Automation Science ICONS 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474667016314999

[47] A. M. Vartouni, S. S. Kashi, and M. Teshnehlab, "An anomaly detection method to detect web attacks using stacked auto-encoder," in *Proc. of the 6th Iranian Joint Congress on Fuzzy and Intelligent Systems*, Feb 2018, pp. 131–134.

[48] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Machine Learning*, vol. 101, no. 1-3, pp. 59–84, 2015.

[49] J. Jabez and B. Muthukumar, "Intrusion detection system (ids): anomaly detection using outlier detection approach," *Procedia Comput. Sci.*, vol. 48, pp. 338–346, 2015.

[50] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[51] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[52] The NetMate source-code repository. Last accessed: January 28, 2020. [Online]. Available: https://github.com/danielarndt/netmate-flowcalc

[53] The TCPDump & Libpcap homepage. Last accessed: January 28, 2020. [Online]. Available: http://www.tcpdump.org/

[54] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, no. 1, pp. 1–39, 2012.

[55] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, May 2000.

[56] Pipes: A brief introduction. Last accessed: January 28, 2020. [Online]. Available: http://www.linfo.org/pipe.html

[57] J. M. Pearce, "Emerging business models for open source hardware," *Journal of Open Hardware*, vol. 1, no. 1, 2017.

[58] The Docker homepage. Last accessed: January 28, 2020. [Online]. Available: https://www.docker.com

[59] The Metasploit homepage. Last accessed: January 28, 2020. [Online]. Available: https://www.metasploit.com/

[60] The Kali Linux homepage. Last accessed: January 28, 2020. [Online]. Available: https://www.kali.org/

[61] The Network Mapper (Nmap) homepage. Last accessed: January 28, 2020. [Online]. Available: https://nmap.org/

[62] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: A novel honeypot for revealing current iot threats," *Journal of Information Processing*, vol. 24, no. 3, pp. 522–533, 2016.

[63] The hping homepage. Last accessed: January 28, 2020. [Online]. Available: http://www.hping.org/

[64] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. Feb, pp. 281–305, 2012.

[65] M. Antonini, M. Vecchio, F. Antonelli, P. Ducange, and C. Perera, "Smart audio sensors in the internet of things edge for anomaly detection," *IEEE Access*, pp. 67 594–67 610, 2018.

[66] Z. H. Janjua, M. Vecchio, M. Antonini, and F. Antonelli, "IRESE: an intelligent rare-event detection system using unsupervised learning on the IoT edge," *Engineering Applications of Artificial Intelligence*, vol. 84, pp. 41–50, 2019.

[67] D. Powers, "Evaluation: From precision, recall and fmeasure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, pp. 37–63, 01 2007.

[68] SAR - Collect, report, or save system activity information. Last accessed: January 28, 2020. [Online]. Available: https://linux.die.net/man/1/sar/

[69] iperf - perform network throughput tests . Last accessed: January 28, 2020. [Online]. Available: https://linux.die.net/man/1/iperf

[70] tcpreplay - Replay network traffic stored in pcap files. Last accessed: January 28, 2020. [Online]. Available: https://linux.die.net/man/1/tcpreplay