Cybersecurity

RESEARCH

Open Access

A novel botnet attack detection for IoT networks based on communication graphs

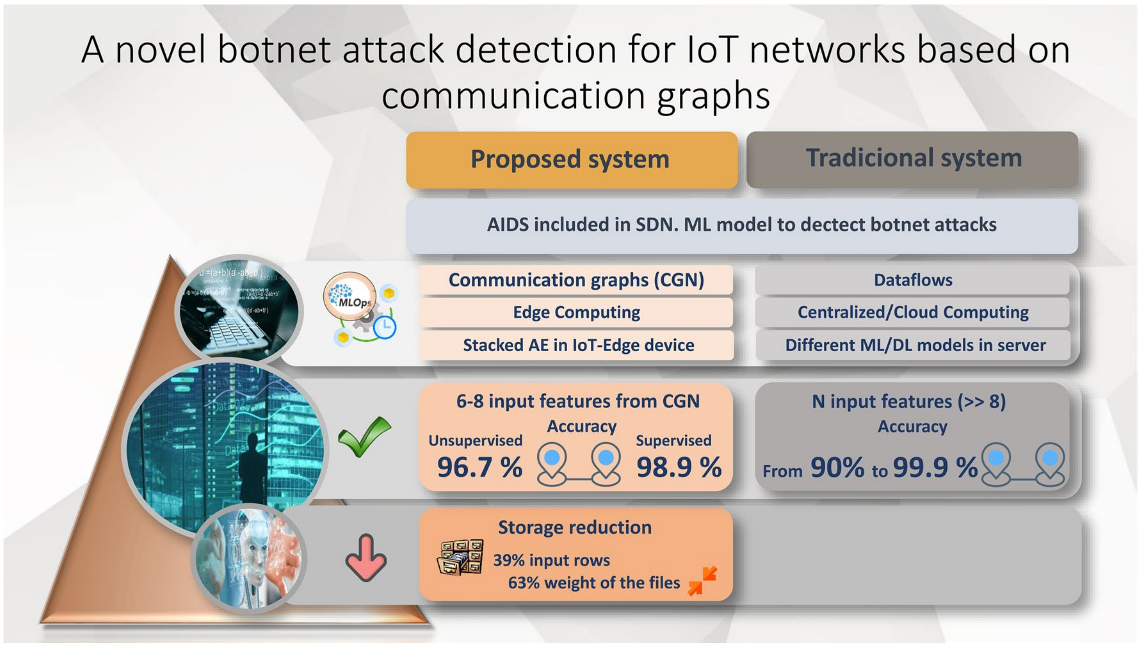
David Concejal Muñoz[1] and Antonio del-Corte Valiente[2*]

Abstract
Intrusion detection systems have been proposed for the detection of botnet attacks. Various types of centralized or distributed cloud-based machine learning and deep learning models have been suggested. However, the emergence of the Internet of Things (IoT) has brought about a huge increase in connected devices, necessitating a different approach. In this paper, we propose to perform detection on IoT-edge devices. The suggested architecture includes an anomaly intrusion detection system in the application layer of IoT-edge devices, arranged in software-defined networks. IoT-edge devices request information from the software-defined networks controller about their own behaviour in the network. This behaviour is represented by communication graphs and is novel for IoT networks. This representation better characterizes the behaviour of the device than the traditional analysis of network traffic, with a lower volume of information. Botnet attack scenarios are simulated with the IoT-23 dataset. Experimental results show that attacks are detected with high accuracy using a deep learning model with low device memory requirements and significant storage reduction for training.

Keywords Autoencoders, Communication graphs, Cyberattacks, Internet of Things

*Correspondence:
Antonio del-Corte Valiente
antonio.delcorte@uah.es
Full list of author information is available at the end of the article

Graphical abstract


A novel botnet attack detection for IoT networks based on communication graphs
Proposed system
Tradicional system
AIDS included in SDN. ML model to dectect botnet attacks
MLOps
Communication graphs (CGN)
Edge Computing
Stacked AE in IoT-Edge device
Dataflows
Centralized/Cloud Computing
Different ML/DL models in server
6-8 input features from CGN
Accuracy
Unsupervised 96.7 %
Supervised 98.9 %
N input features (>> 8)
Accuracy
From 90% to 99.9 %
Storage reduction
39% input rows
63% weight of the files

Introduction

Cybersecurity is considered necessary from a strategic point of view. Reports from cybersecurity companies reveal an annual increase in the number of cyberattacks. Global cyberattacks increased by 38% in 2022 compared to 2021 (Check Point 2023). Multipurpose malware (a category for botnets and banking Trojans) accounts for 32% of all attacks.

The ever-increasing interest and continuous development of the Internet of Things (IoT) has reinforced the growth of data and devices. IoT is widely used in both industrial production and the social domain, offering significant advantages in terms of convenience, efficiency, and accessibility. However, its vulnerabilities have contributed to serious security and privacy threats (Zhou et al. 2019).

IoT devices collect data for a specific purpose. They are used for temperature control, home access control, biometric measurements, quality control, industrial process security, and in many other applications. These data are usually stored on servers hosted on local networks or in the cloud. The data may be confidential, and its alteration could cause damage.

IoT devices generally have weak security, making them vulnerable to exploitation as a backdoor for intrusion into the systems to which they connect. This can result in blocked reception of information, due to altered or missing data. Additionally, they can be recruited for botnets and actively participate in cyberattacks.

The current integration of IoT devices in the medical, industrial, and military fields requires more sophisticated security mechanisms to prevent information theft and mitigate material and physical damage.

A current line of defence in IoT networks are intrusion detection systems (IDS), based on machine learning techniques, due to the specific characteristics of these networks: global connectivity, limited energy and bandwidth, and heterogeneity (Thakkar and Lohiya 2019). IDS classifies network traffic, and identify abnormal patterns. Generally, they analyse the data flow in the network to detect a variety of attacks. However, flow-based approaches can often lead to computational overhead and do not fully capture the communication characteristics (Daya et al. 2019). Communications graphs of networks (CGN) overcome these limitations, and they are an alternative approach to determining the behaviour of connected devices without compromising data confidentiality because they are drawn from control information. For this reason, communication graph analysis has gained attention. In this line of research, software-defined networks (SDNs) provide major advantages by separating communication into a data plane and a control plane. This improves security and network management (Sarica and Angin 2020) but also simplifies the representation of behaviour by communication graphs.

We propose an architecture for an anomaly intrusion detection system (AIDS) integrated into a software-defined network. It uses stacked autoencoders to categorize the behaviour of IoT-edge devices on the network. This deep learning technique is suitable for unbalanced datasets, such as those gathered for botnet detection. Furthermore, research has shown the feasibility of running autoencoders on IoT-edge devices (Luo and Nagarajan 2018), despite their limited computational capabilities.

This deep learning model, hosted in the application layer of IoT-edge devices, individually evaluates its behaviour on the network. Each device periodically scans its communications graphs of the network, provided by the SDN controller. The system has a dedicated decision server that is responsible for taking actions based on the predictions. Moreover, the server regularly retrains the model, fitting it to the evolution of normal network traffic.

A stacked autoencoder-based model, with no supervision, has been implemented. It has low computational requirements and is supported by IoT-edge devices. The results are similar to those commonly reported in the literature but with reduced usage of computational resources. As the training and detection dataset have fewer features and items to represent the behaviour of the network, the generated CGN files are smaller compared to the original traffic dataset.

Our main contributions are summarized as follows:

- We propose a new defence against botnet attacks, which is trained from the CGN of the IoT device network flow. The attack detection process is performed by edge devices thereby distributing the computational cost. Training and detection require only control information which avoids a breach of confidentiality.
- We reveal that CGNs reduce the storage space and features needed to learn the benign behaviour of a network from deep learning models with low depth and parameters and thus obtaining a high accuracy in the detection of malicious attacks. This optimization permits detection to be performed on edge devices.
- We evaluated the model against six of the prevalent malware. Our experiments show that its performance is consistent with the current state of the art.

The remaining parts of this paper are organized as follows: In section "State of the art", we present a review of related work and the state of the art on the subject of this article; in section "Objectives and methodology", we describe the research objectives and the methodology followed; in section "Contribution", we explain the proposed architecture and model for detecting botnet attacks on IoT-edge devices; in section "Evaluation", we show the results achieved in the experiments; in section "Discussion", we analyse and discuss the effectiveness of the models; and in section "Conclusions", we summarize our conclusions.

State of the art

A communication network is a collection of autonomously operating computers with the capability to exchange information with each other (Tanenbaum and Wetherall 2011). This definition can be extended to include IoT-edge devices.

Security paradigms

There are three main security objectives: network availability, information integrity, and confidentiality (Lu et al. 2010). Different security paradigms are used for the detection and mitigation of malicious attacks, such as IDS and SDN. The latest trends combine both paradigms.

An IDS is an automated detection process that monitors events on a system and looks for signs of intrusion (Hung-Jen Liao et al. 2013). It analyses traffic differently from traditional firewalls, seeking out characteristics that identify the traffic as abnormal. IDSs can be classified into two main groups (Khraisat et al. 2019): (i) a signature IDS searches for previously catalogued patterns in the data transmitted over the network; however, these systems are becoming less effective. (ii) An AIDS detects differences in the learned behaviour of the network using machine learning techniques. Attackers must be aware of normal behaviour to avoid detection.

An SDN is a paradigm that enables the design, implementation, and management of communications networks in which control and data flow are separated into two different planes (Benzekki et al. 2016). The control plane is centralized, making routing decisions and managing the logical topology of the network. The data plane is responsible for transmitting data, according to the control plane's policies.

It has a three-layer architecture (Sarica and Angin 2020) (Fig. 1): (i) Application: hosts applications, such as the IDS, communicating with the controller; (ii) Control: contains a controller managing a network overview and routing policies; (iii) Infrastructure: applies policies received from the controller to routing devices.

An SDN provides greater protection against some types of attacks (Shinan et al. 2021), however, the controller is a vulnerable point for failure due to the centralized network management (Ahmed et al. 2015).

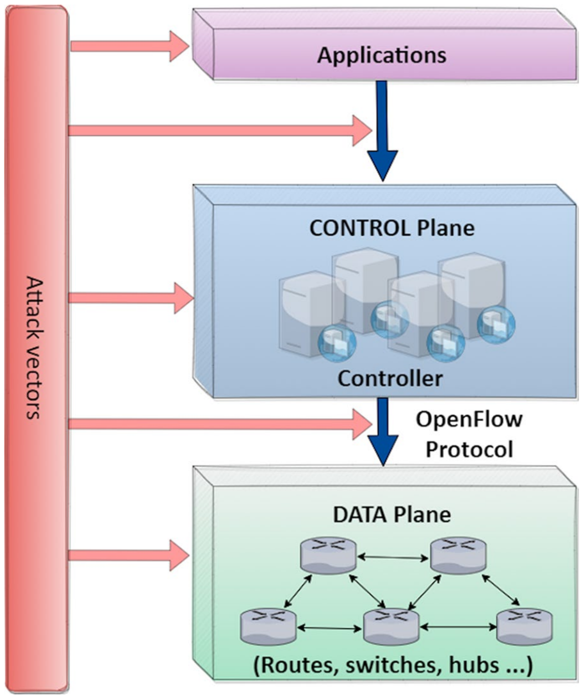


Fig. 1 Layers and attack vectors in the SDN architecture

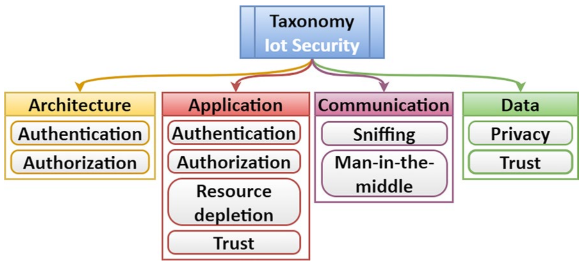


Fig. 2 Domains and features of IoT Network Security Taxonomy

IoT network security

IoT networks include components such as sensors, actuators, computation nodes, receivers, and communicators. There are three different layers in IoT networks (Horrow and Sardana 2012; Zhao and Ge 2013): (i) Application, providing multiple visible services; (ii) Perception, collecting and sending information; and (iii) Network Protocols.

The security requirements for IoT networks are different from those for traditional networks. Alaba et al. (2017) proposes a specific taxonomy related to security for IoT networks, comprising four domains (Fig. 2): architecture, application, communication, and data.

The presence of attacks on IoT-edge devices has increased in recent years, with attention focusing on aspects such as their low security and vulnerabilities (Mendes et al. 2019). These devices are often recruited by botnets to participate in distributed denial-of-service attacks.

Threats in communications networks

Communication networks are exposed to multiple threats, which can be divided into two main categories (Pawar and Anuradha 2015): passive and active.

Passive attacks analyse the data flow over the network, threatening the confidentiality of information. The main types of passive attacks are traffic analysis, in which sensitive information is inferred (Hafeez et al. 2019), and eavesdropping or sniffing, in which information is gathered from messages passing through the network.

Active attacks threaten the integrity of information and the accessibility of network services. The principal types are: (i) Spoofing: Impersonating an authenticated device to send information on its behalf (Jindal et al. 2014), (ii) Modification: Changes in the routing of messages, causing delays in the delivery of messages, (iii) Wormhole: Packets transmitted over a network are recorded and sent to a new location (Hu et al. 2003), (iv) Fabrication: Generation of false routing messages, (v) Denial-of-Service: Reducing or interrupting access to services offered by a network, (vi) Sinkhole: a compromised device on the network attracting traffic to remove it (Kibirige and Sanga 2015), (vii) Sybil: a malicious device presenting multiple identities on peer-to-peer (P2P) networks (Douceur 2002), (viii) Black hole: interruptions or delays in packet delivery performed by network routing services, (ix) Rushing: Sending message forwarding requests before authenticated devices, (x) Replay: A legitimate message in one context is being injected into a different context (Malladi et al. 2002), and (xi) Byzantine: a set of authenticated devices arbitrarily blocking the services offered (Geetha and Sreenath 2016).

Botnets

A botnet is a set of compromised devices that are controlled by a botmaster (Choi et al. 2007). They are heterogeneous devices connected via a communications network. A botnet is not inherently malicious and it allows for the coordinated execution of commands on multiple devices. These features have been exploited by cybercriminals for fraudulent purposes. The architecture has evolved from P2P configurations to hybrid configurations and from internet relay chat protocols to hypertext transfer protocols, P2P protocols, or a hybrid model combining both.

The components of a botnet are (Silva et al. 2013) (i) vulnerable host: devices that have been infected with a malicious piece of code; (ii) bots: potentially malicious

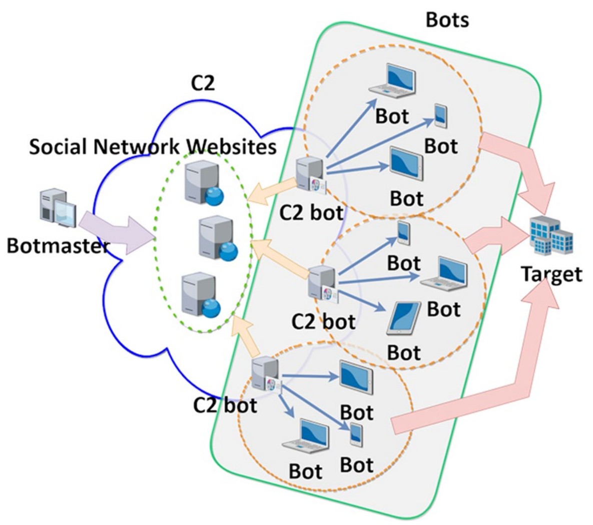


Fig. 3 Hybrid model of botnet architecture. Communication between C2 and botmaster through social networks

programs that acquire control of the host and execute commands received from a third party; (iii) botmaster: who controls the botnet and is responsible for sending commands to the bots; and (iv) command-and-control (C2) infrastructure: which allows the botmasters to communicate with the bots. Limarunothai and Munlin (2015) decomposes the C2 architecture into servers and protocols.

Botnets can have different architectures; as described in Zeidanloo and Manaf (2009), there are three models: (i) centralized, where all communications go through a C2 server; (ii) distributed, where hosts can simultaneously function as C2 and bot; and (iii) hybrid, which uses social networks as a means to broadcast commands (Fig. 3).

The life cycle of a botnet consists of five steps (Limarunothai and Munlin 2015): (i) initial infection: devices are infected to install the code, transforming them into bots; (ii) secondary injection: the bot code is downloaded and installed; (iii) Domain Name Server (DNS) lookup: the internet protocol (IP) address of the C2 server is located; (iv) rallying: The bot establishes a connection with the C2 server; and (v) malicious commands: maintenance and updating.

Knowledge of the life cycle of a botnet is important for the definition of detection methods (Silva et al. 2013). The three principal mechanisms are: (i) DNS-based: consisting of detecting behavioural patterns of queries made by bots to DNS services; (ii) HoneyNet networks: purposely configured with vulnerabilities with the aim of attracting attacks, although they do not allow botnet detection by themselves (Karim et al. 2014), and (iii) IDS.

Intrusion detection systems

Multiple studies have been conducted since 2015 (Shinan et al. 2021), proposing an AIDS solution on an SDN architecture located in the application or control layer. Khraisat et al. (2019), Shinan et al. (2021) enumerate the most widely used machine learning methods: (i) supervised: decision trees, Naïves-Bayes, artificial neural networks, support vector machines, and K-nearest neighbour; and (ii) unsupervised: K-means (clustering) and genetic algorithms. These have been used less often (Murray et al. 2014).

Machine learning models have traditionally been applied on a set of characteristics of the data flow. Promising lines of research propose that network behaviour is similar to the patterns of social networks (Shinan et al. 2021; Chowdhury et al. 2017; Daya et al. 2019). Therefore, novel machine learning models are trained with CGN. The nodes represent the devices, and the arcs describe the data flows, with one arc existing for each port-IP address tuple (Fig. 4).

CGN avoids comparing data flows (Venkatesh et al. 2015) and is a more efficient method. Chowdhury et al. (2017) suggests eight features: (i) in degree (IDM): the number of input flows to a device, which is high on the C2 server; (ii) out degree (ODM): the number of outgoing flows from a device, which is high on the C2 server and bots; (iii) in weight degree (IWM): the total number of incoming packets received by a device from its neighbours, where it is assumed that all bots on a network will receive the same number of packets from the C2 server; (iv) out weight degree (OWM): the total number of outgoing packets sent by a device from its neighbours, where it is assumed that bots will send the same packets

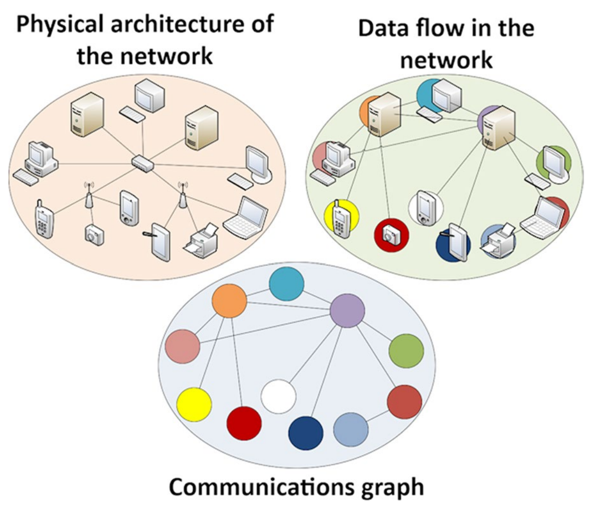


Fig. 4 Relationship between the communication graph and the physical architecture of a network

to the devices that they are going to attack; (v) clustering coefficient (CCM): evaluates the closeness between the neighbours of a device, where a high value is assumed for P2P-type botnets; (vi) node betweenness (BCM): the number of times a device is in the set composed of the shortest paths between each pair of devices, where a high value is assumed in P2P-type botnets; (vii) node closeness (LCM): the mean of the shortest distance of all devices that can reach another device, which is relevant in P2P botnets (Sengupta et al. 2021), and (viii) eigenvector centrality (EVM): the weight of the device in the graph.

Autoencoders for anomaly detection

Autoencoders are a type of machine learning model introduced by Rumelhart et al. (1986). They are unsupervised neural networks that are trained to extract the main features of the input so that it can be reconstructed. The inputs are encoded and then decoded, resulting in a loss of information. The difference between the input and output is minimized. Autoencoders are a generalization of principal component analysis. Instead of finding linear relationships, they learn the non linear ones (Bank et al. 2020), thus achieving dimensional reduction.

Stacked autoencoders are layer-trained autoencoders. Each layer of the encoder is the input of another more internal autoencoder, until it reaches the deepest level (bottleneck). Similarly, the decoder layers are considered to be the output of another autoencoder.

Mirsky et al. (2018), Luo and Nagarajan (2018), Zhou and Paffenroth (2017) presented models for detecting attacks in the cybersecurity domain. Niyaz et al. (2017) describes the use of autoencoders for the detection of distributed denial-of-service attacks in SDN networks, learning the normal behaviour of the network with a stacked autoencoder. The model classifies up to eight different types of attacks with a very low rate of false-positives. Their experiments achieve 99.82% accuracy in identifying network attacks.

Autoencoders can be run for anomaly detection on IoT-edge devices (Luo and Nagarajan 2018). The edge computing paradigm is applicable when a copy of an autoencoder is placed on the edge devices. Moreover, the machine learning model can be periodically retrained from data traffic. This architecture allows the model to evolve with the latest network behaviour.

Summary of conclusions

The most recent research tends to integrate AIDS with SDN networks. Previous papers have proposed this architecture for IoT networks, dedicating servers to detection. Other articles propose AIDS in wireless sensor networks where the detection is carried out by autoencoders in the IoT-edge devices. All the proposed models examine the data flow traffic. To the best of our knowledge, AIDS systems for IoT networks that analyse CGN have not yet been suggested.

Objectives and methodology

The primary objective is the development of an AIDS prototype integrated into an SDN, for the detection of botnet attacks on IoT networks. We represent the network behaviour using CGN. We expect to reduce the amount of evaluation and training data by at least 30%, and achieve an accuracy greater than 90%.

This objective has been divided into more specific goals for IoT networks: (i) determining the components to integrate an AIDS in an SDN architecture, (ii) identifying at least six characteristics of CGN for botnet attack detection, (iii) evaluating the accuracy and precision of the stacked autoencoder in detecting behavioural anomalies, (iv) reducing the required resources, (v) verifying that the solution is a valid alternative, and (vi) exploring the use of CGN to increase security.

The life cycle chosen for the prototype is machine learning model operationalization management (MLOps), with a focus on the tasks associated with machine learning engineering. However, deployment and monitoring work have not been covered.

According to the MLOps methodology, the tasks are distributed across three preparation pipelines: data, model, and software.

Training data are acquired, explored, validated, and prepared for the duration of the data pipeline. The source of the data required for both model training and classification of network traffic is the control flows managed in the SDN controller. First, the benign behaviour of the network is captured by recording the number of packets exchanged for each ip and port during a time interval. These captures contain the information needed to generate the CGNs with the benign behaviour of the network and is used as a data source for the preparation of training data. In later steps, the controller maintains a CGN that it will share with the devices for classification of device behaviour on the network. For this experiment, the input data source is the Aposemat IoT-23 dataset (Garcia et al. 2020), composed of labelled data flows from a real IoT network. CGNs are calculated from this dataset and versioned in Data Version Control (DVC).

The machine learning model is prepared, evaluated, and packaged for deployment throughout the model pipeline. Even when autoencoders are not supervised, the original labelling is utilized as ground truth, and

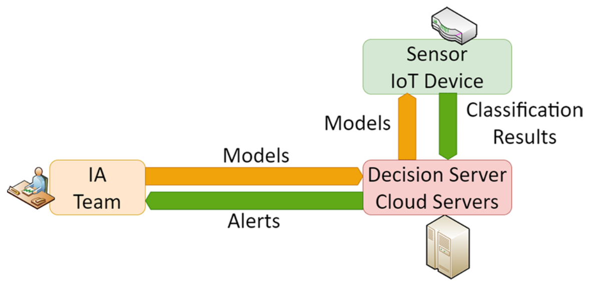


Fig. 5 Workflow of model preparation by the AI teams and their broadcast to the system

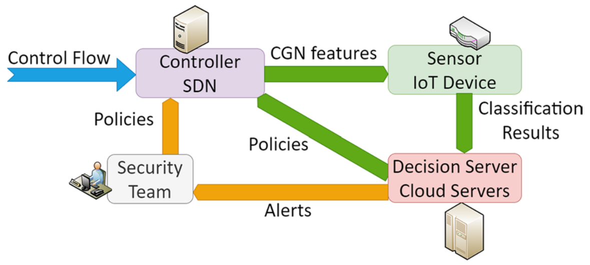


Fig. 6 Data flow between the different components of the system for attack detection

the training process is cross-validated with supervised metrics over fivefold. Thus, the total metrics are the mean accuracy and F1-score. We have sought the best model configuration. Therefore, a Cartesian hyperparameter search is used to generate the combinations to be evaluated. All tested models have been registered on the mlflow platform, thus guaranteeing traceability and reproducibility.

Figure 5 shows the model preparation workflow. AI teams prepare the classification models, and they are registered in the decision server. The new model is distributed to the sensors that will be classified with the updated model. The decision server evaluates whether there is a model decay based on the classification results, and upon a possible deviation, alerts are generated to the AI teams to adjust the current model.

The software pipeline is dedicated to the development of prototype programs. There are components for input data transformation, model training, and AIDS integration.

Figure 6 shows the data flow of the attack detection process. The SDN controller generates the CGN from the control flow. It sends the features to each device of its traffic to evaluate its own behaviour. The results obtained are sent to the decision server that can generate policies to mitigate a detected attack and generate alerts to security administrators to evaluate and act on a detected threat.

Finally, Table 1 shows the canvas proposed by MLops adapted to this prototype.

Contribution

In this section, we detail the IDS prototype that we have developed for research purposes. We have not fully implemented all components of the IDS, but a real runtime environment is emulated.

Figure 7 shows the proposed architecture and its features. It describes the integration into an SDN network and also represents the relationship among the different actors in the MLops life cycle. It is continuous and cyclical; thus, data analysis and modelling tasks alternate with their deployment in an operational environment.

Model development

The machine learning model development tasks begin with data preparation. An SDN has monitoring tools that trace the communication data flows. CGNs are generated from the collected information to extract model training data. We chose the IoT-23 dataset as the source of the data flows. It is publicly and freely available for cybersecurity research; it contains different types of attacks and normal traffic of a real IoT network.

Table 1 MLOps life cycle. Detail of the different blocks and specific objectives of the proposed prototype

Blocks	Objectives	Project scope
Value proposal	Define problem and importance	AIDS with lower computational cost
Data sources	Identify main sources	Network data flow
Prediction task	Model type to use	Stacked autoencoder
Features	How to represent input	CGN
Offline evaluation	Define methods and metrics	Accuracy and F1 score and MSE
Decisions	How to use predictions	Generate alerts
Making predictions	When and how	Batch periodically
Collecting data	Cost of new data	No labeling required
Build models	Frequency and cost	Periodic re-training
Evaluation and monitoring	How to supervise	Metrics with human oversight

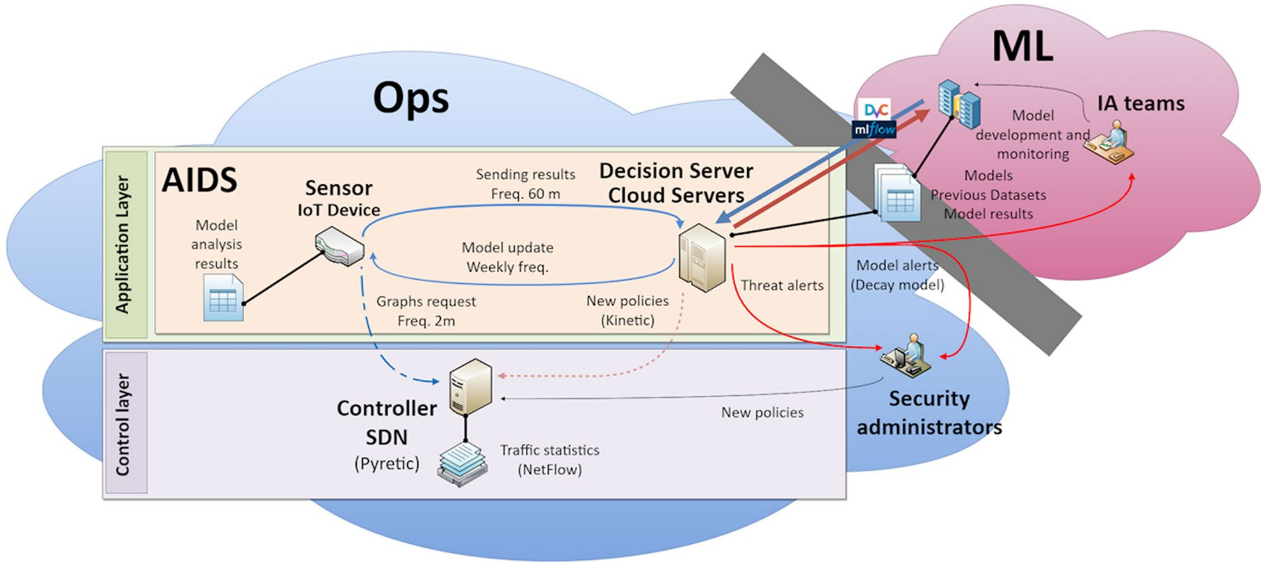


Fig. 7 Architecture and main characteristics of the proposed prototype for the detection of botnet attacks based on communication graphs

A subset of data from the available dataset was selected. We intended to have malicious traffic including different communication protocols with the C2 and types of attacks. In addition, they were selected for the relevance of the botnet itself. These files contain traffic from Mirai, Okiru, GagFyt, Hiajime, MushTik, and Hide &Seek malware. Mirai, and variants such as Okiru, and GagFyt (also known as Qbot or Bashlite) are still very relevant and trending botnets today. MushTik and GagFyt use a lighter IRC protocol. Hide &Seek and Hiajime incorporate the P2P protocol. Most of them have the capacity to carry out denial of service attacks, cryptocurrency mining, information theft, and antitampering.

The data preparation process was structured into three steps: (i) converting the original capture structure to the standard Comma-Separated Value (CSV) format, (ii) calculating the CGN from the CSV dataset and extracting the inputs to the model to be trained, and (iii) analysing and preparing data for training.

The source information was very comprehensive. The CGN was drawn with the appropriate features to identify botnets (Chowdhury et al. 2017; Daya et al. 2019; Venkatesh et al. 2015): IDM, ODM, IWM, OWM, CCM, BCM, LCM, and EVM. No major modifications were necessary once the CGNs were available. It was sufficient to normalize the features.

The second machine learning task consisted of defining and training the optimal model. We propose fully connected stacked autoencoders. This type of autoencoder tends to converge to a local minimum better than other deep autoencoders and facilitates the initialization strategy. The number of neurons in the input and output layers is equal to the number of features in the training data. The number of hidden layers, the number of neurons in each of them, and other hyperparameters were determined through experimentation. The configuration that achieved the best classification accuracy was considered optimal. Generally, a rectified linear unit (ReLU) activation function was used for hidden layers in deep learning models. We have preferred a Leaky ReLU activation function. Xu and Szegedy (2015) validated that leaky ReLU performs better than ReLU on classification problems. The bottleneck activation function was a traditional sigmoid. The initial weights were set by random initialization, and the optimization method was Adam. It converges quickly when dealing with sparse gradient problems. Batch normalization was applied to allow for much higher learning rates, to be less careful with initialization, and even to regularize (Ioffe and Weiqing 2015). The stacked autoencoder was trained only with normal behaviour, extracting its principal characteristics. The aim was to minimize the difference between the input and reconstruct its output. For this purpose, the mean squared error (see 1) was employed as a loss function.

$$\mathrm{MSE}\left(X,\hat{X}\right)=\frac{1}{N}\sum_{i=1}^{N}\left(\hat{x}_i-x_i\right)^2 \quad (1)$$

This difference will be an outlier when analysing the abnormal behaviour of a botnet device. The interquartile range (IQR) rule was applied to determine the outliers (see 2a). Therefore, the outliers were n times larger than the upper bound of the IQR (see 2b). We tried several values of n.

$$IQR = Q_3 - Q_1 \tag{2a}$$

$$Outlier(x) = \begin{cases} 1, & \text{if } x > Q_3 + (n * IQR) \text{or} \\ & x < Q_1 - (n * IQR) \\ 0, & \text{otherwise.} \end{cases} \tag{2b}$$

Training and validation of the models were performed in batches, recording the experiments on the mlflow platform.

Operations

Mainly, the operational tasks were performed by the SDN and the AIDS prototype embedded in the application layer of edge-devices. The most important components are the controller, the monitoring sensor, and the decision server.

An SDN controller directs traffic according to forwarding policies; it is the core of an SDN. Any communication between applications and network devices must go through the controller, interacting with the devices using traditional OpenFlow-based protocols.

The monitoring sensor resides on all application layer edge devices and executes a set of periodic processes: (i) it invokes a simulated service from an SDN network controller that delivers behavioural data, transformed to CGN, to be evaluated. The sensor's machine learning model classifies this received input and the results are stored locally. (ii) The results to be transmitted are sent to the decision server, which avoids communication overhead, but is sent immediately when an attack is detected. This information is no longer retained in the sensor; and (iii) the most current version of the model is downloaded from the decision server.

The frequency of these processes depends on the characteristics of the network. The CGN summarizes the behaviour of devices on the network. The controller is responsible for creating and updating it. This process is not immediate, increasing the time and resources needed, depending on the number of nodes and the volume of network traffic. We have not studied the best strategy to minimize the impact on the controller. We have assumed a strategy where the controller obtains the CGN at startup and updates it for each node asynchronously and in parallel. Updating IDM, IWM, ODM and OWM metrics has no relevant resource requirements. However, centrality metrics have significant computational costs, and traffic from one node can affect the centrality of many nodes in the network. Periodic updating of these indicators reduces the required computational resources. On the other hand, lazy updating deteriorates the information available in the CGN. It should also be considered that edge devices do not require this information in real time to avoid network overload and excessive power consumption on the device itself. This same principle is also applicable to the communications that the devices carry out with the decision server. It is well known and empirically proven that it is more efficient to send a single message containing a set of information than to send it in different messages, even if the amount of data is larger. Finally, it is necessary to consider how the behaviour of the network evolves over time to determine the periodicity of the retraining of the model.

The IoT devices included in the dataset under study generated very low traffic when they were not infected. The Amazon Echo device generated more packets than the other devices, with an average of 229 flows per hour. However, it could reach 158 flows per minute when the network was affected by an Okiru attack. To detect attacks with the shortest delay, a frequency of two minutes was set to request its behaviour on the network from the controller. Therefore, thirty additional flows per hour were generated. The device sent the collected traffic information every hour. This limited the storage space required by the IoT device to 6 Kb. Finally, a weekly update of the model was planned for networks with little variation in their behaviour, considering the system alerts of model decay. Luo and Nagarajan (2018) proposes two-minute frequencies for traffic classification, with daily sending of results and model adjustment.

The decision server collects the results sent by the monitoring sensors and performs actions based on them. The server has the following features: (i) It receives predictions from sensors and forwards information about detected attacks to a dedicated service to mitigate or intercept them. Currently, only alerts are appended to the console of security administrators. However, dynamic security policies, written in the Pyretic language, could be automatically issued to the controller. (ii) Two types of alerts are emitted from the server to security administrators and the artificial intelligence (AI) teams. First, an alert is generated when an excessive number of attacks can be detected, which simply is caused by an intense attack or indicates that the model is decaying. Second, there is a low attack detection rate over a long period of time. It could be related to a low sensitivity of the model; (iii) randomly, it selects part of the normal behaviour of the network, which is combined with the learned dataset, discarding older content. The resulting file is the input to a new training cycle; thus, the model is fitted weekly to changes in the normal behaviour of the network. (iv) The new model is disseminated to the sensors when an accuracy of more than 90% is achieved. The AI teams are alerted when the model has not been updated.

Evaluation

In this section, we report the results of simulating the proposed architecture using the IoT-23 dataset. This is a collection of different network traffic from IoT devices, consisting of twenty-three scenarios. These are captures (pcap files) from infected IoT-edge devices. The final dataset was acquired by running the Zeek network analyser on the original pcap files and adding two new columns for traffic classification labels. The files contained twenty-three feature columns. The list and description of columns can be found in Garcia et al. (2020).

Table 2 describes the files selected for the experiments performed. The files were transformed into CGN using a process that generates the files with a row for each IP address and port, along with the features calculated in the CGN. The six files contain the prevalent IoT device-specific malware: Hide& Seek, Mushtik, linux-hiajime, Mirai, Okiru, and GagFyt. They contain flows produced over 24 hours, except for Hide& Seek and Mushtik, where traffic was tracked over a longer period. Mirai and GagFyt contain the highest number of packets, requiring 3.86 and 21.5 Gbytes of storage space, respectively. GafFyt is the most significant case, where the network flow is represented with 3.5 million records from the 271 million packets captured, and the storage space falls to 411 Mbytes.

Table 3 shows the time required for generating each of the CGNs. The total time is the sum of the time taken to render the graph, calculate each of the metrics associated with the nodes, and save the results in the output file. The table contains one column for each calculated metric. Rendering time depends on the number of Zeek flows and the number of devices and ports. Therefore, the minimum time is obtained in file 09_ 01.IoTMalware (156,104 flows in 3.76 s) and maximum time in 36_ 01.IoTMalware (13,645,107 flows in 2,171.02 s). The generation time, and the IDM, ODM, IWM and OWM metrics, are directly related to the number of devices and ports. The 36_ 01.IoTMalware file requires the maximum time in all these measurements (1,437.12 s, 62.14 s, 63.04 s, 62.16 s, and 64.02 s), while the minimum time is consumed in 60_ 01.IoTMalware file (0.58 s, 0.02 s, 0.02 s, 0.02 s, 0.02 s, 0.02 s). Centrality metrics depend not only on the number of devices and ports but also on the topology of the CGN. The 36_ 01.IoTMalware file requires the maximum time in all these metrics (261,550.12 s, 32,646.82 s, and 2,436.50 s), while the minimum time is spent in 60_ 01.IoTMalware file (4.05 s, 3.18 s, 2.01 s). The maximum total time is required to obtain 36_ 01.IoTMalware file (300,492.94 s) and minimum time is required for 60_ 01.IoTMalware file (87.16 s).

The generation of some centrality metrics has a high computational cost, such as BCM and CCM (Brandes 2001; Kang et al. 2011).The calculation of betweenness has been approximated (Brandes and Pich 2007). We used 1500 node samples (pivots) to estimate the betweenness values. Even so, the time required may not be affordable for the controller when the number of nodes is high

Table 2 Dataset used for experiments. Flows and packets captured during the trace time and storage space required

Dataset	Malware	Duration (hrs)	Packets	Pcap files (Mb)	Zeek flows	Zeek files (Mb)
01-01_IoTMaleware	Hide &Seek	112	1,686,000	140	1,008,749	126
03-01_IoTMaleware	Mushtik	36	496,000	56	156,104	21
09-01_IoTMaleware	linux-hiajime	24	6,437,000	472	6,378.294	849
35-01_IoTMaleware	Mirai	24	46,000,000	3686	10,447,796	1186
36-01_IoTMaleware	Okiru	24	13,000,000	992	13,645,107	1573
60-01_IoTMaleware	GagFyt	24	271,000,000	21,504	3,581,029	411

Table 3 Time (s) to generate the CGN, and metrics, related to the network flows described in the experiment dataset

Dataset	Malware	Render	IDM	ODM	IWM	OWM	CCM	BCM	EVM	Generate
01-01_IoTMaleware	Hide &Seek	27.37	0.11	0.12	0.17	0.17	12304.63	1535.87	9.72	15.27
03-01_IoTMaleware	Mushtik	3.76	0.02	0.02	0.02	0.02	39.97	220.47	2.41	3.13
09-01_IoTMaleware	linux-hiajime	240.89	2.01	1.60	1.98	1.59	101656.02	12688.70	232.42	306.34
35-01_IoTMaleware	Mirai	326.80	1.67	2.15	1.66	2.17	78904.09	9848.85	98.84	161.27
36-01_IoTMaleware	Okiru	2,171.02	62.14	63.04	62.16	64.02	261550.12	32646.82	2436.50	1437.12
60-01_IoTMaleware	GagFyt	77.26	0.02	0.02	0.02	0.02	4.05	3.18	2.01	0.58

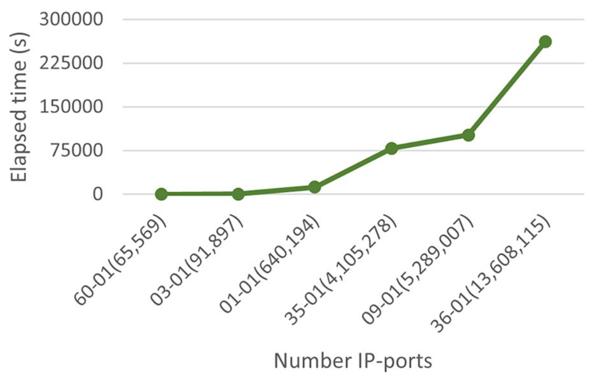


Fig. 8 Evolution of the elapsed time (s) to obtain the CCM metric in relation to the number of IP-ports in each IoT-23 dataset

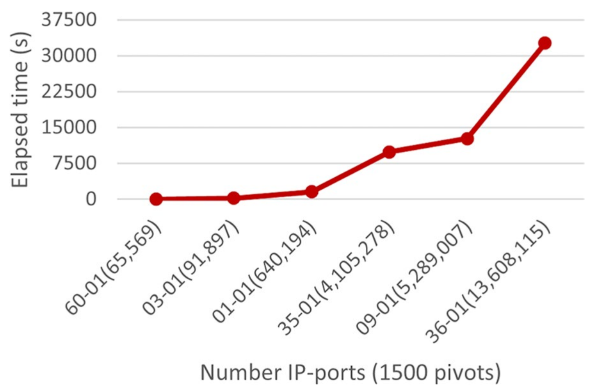


Fig. 9 Evolution of the elapsed time (s) to obtain the BCM metric in relation to the number of IP-ports in each IoT-23 dataset

(see Figs. 8 and 9). Growth is exponential as a function of the number of edges and arcs of the CGN. Both allow capturing the behaviour of P2P botnets, so a group of experiments was performed without them to evaluate the impact on classification accuracy.

Storage evaluation

We converted the original dataset in Zeek format to the CSV standard. Furthermore, we kept only normal traffic because the anomaly detection machine learning models are trained only on these flows. Next, we represented these flows as a CGN. The graph contains a node for each different combination of IP address and port that participates in a flow. Centrality features were calculated for each node, examining the relationships between them. The result were stored in a new CSV file, inserting a row for each node and a column for each feature. Therefore, the files contain a column for the IP address and another for the port, nine columns with normalized centrality metrics (IDM, ODM, IWM, OWM, CCM, BCM, LCM, in-EVM and out-EVM) and, finally, a label column. Due to the fact the PCAP captures included in the IoT-23 dataset, CCM was not finaly considered in this work because the same value is always obtained. It was calculated from the fraction of possible triangles that pass through that node and is zero for all the nodes of the graph.

We compared both groups of files to evaluate if the number of records needed to represent the behaviour of the devices, and the storage in bytes, has decreased. Table 4 shows the details of the comparison for each file. The best reduction ratios are obtained for files 08-01-01_ IoTMalware (99.54% of rows and 99.70% of bytes), 60-01_ IoTMalware (97.46% of rows and 98.39% of bytes), and 44-01_ IoTMalware (95.73% of rows and 97.53% of bytes). However, rows increase in 4 files, with 07-01_ IoTMalware (−49.48%) achieving the worst ratio. The results reveal that storage space is saved in all cases, and in general, the number of rows decreases (see Fig. 10). Seventy-five percent of the files reduce the quantity of elements by more than 29.89%. However, it increases when the devices and ports involved in the communications are greater than the captured flows. This scenario occur in four of the twenty-three files evaluated (17%). On average, 59.75% of storage space is saved, and there are 33.19% fewer rows. (see Fig. 11).

We evaluated the storage space that the controller would need to store information on the behaviour of the network in the form of a communications graph. Table 5 details the storage space used for the experimental scenarios and the characteristics of the generated graphs. The highest space reduction is obtained in file 60_ 01.IoTMalware (from 411 to 7 Mbytes) because the generated graph is denser than in the rest of the files. However, almost no reduction is achieved in file 36_ 01.IoTMalware (from 1573 to 1512 Mbytes), as the network is very sparse.

Only one record is generated for each IP address-port, so the size of the file remains constant when the interaction between the same network devices grows thus increasing the percentage reduction of the output information with respect to the input flows.

Similar results to those previously achieved for normal traffic are obtained. The storage space reduction is 49%, and 46% in the rows of the CGN file. The number of nodes and arcs of the graphs confirms that they are sparse graphs and, in some cases, unconnected.

Classification experiments

A single input data scenario for the experiments was prepared from a selection of the IoT-23 dataset. This scenario contains traffic from different types of attacks produced by malware prevalent on IoT devices. Table 6 shows the selected files, the types of malware contained,

Table 4 Reduction (%) of rows, and storage bytes. Comparison between the original IoT-23 dataset and the generated CGNs

Dataset	Data flows		CGN		% Reduction	
	Rows	Bytes	Rows	Bytes	Rows	Bytes
44-01_IoTMalware	211	24,978	9	616	95.73%	97.53%
04-01_Honeypot	452	57,855	454	32,349	− 0.44%	44.09%
05-01_Honeypot	1374	170,507	944	66,922	31.30%	60.75%
20-01_IoTMalware	3193	388,802	628	45,833	80.33%	88.21%
21-01_IoTMalware	3272	398,067	1924	142,330	41.20%	64.24%
42-01_IoTMalware	4420	546,951	3099	226,308	29.89%	58.62%
08-01_IoTMalware	2181	271,260	10	824	99.54%	99.70%
34-01_IoTMalware	1923	223,957	293	20,869	84.76%	90.68%
03-01_IoTMalware	4536	534,951	5609	390,963	− 23.66%	26.92%
01-01_IoTMalware	469,275	53,974,498	441,334	32,239,514	5.95%	40.27%
60-01_IoTMalware	2476	284,405	63	4579	97.46%	98.39%
48-01_IoTMalware	3734	445,406	2319	159,938	37.90%	64.09%
49-01_IoTMalware	3665	443,332	3116	222,321	14.98%	49.85%
09-01_IoTMalware	22,548	2,787,317	18,639	1,333,986	17.34%	52.14%
35-01_IoTMalware	8,262,389	958,410,934	4,120,109	456,537,432	50.13%	52.37%
07-01_IoTMalware	75,955	9,676,495	113,538	7,911,788	− 49.48%	18.24%
36-01_IoTMalware	2663	306,067	1170	81,773	56.06%	73.28%
52-01_IoTMalware	1794	210,160	1369	94,072	23.69%	55.24%
33-01_IoTMalware	1,380,791	154,919,240	1,362,849	97,414,905	1.30%	37.12%
17-01_IoTMalware	31,438	3,933,107	25,206	1,834,122	19.82%	53.37%
39-01_IoTMalware	7337	870,739	8,534	613,082	− 16.72%	29.59%

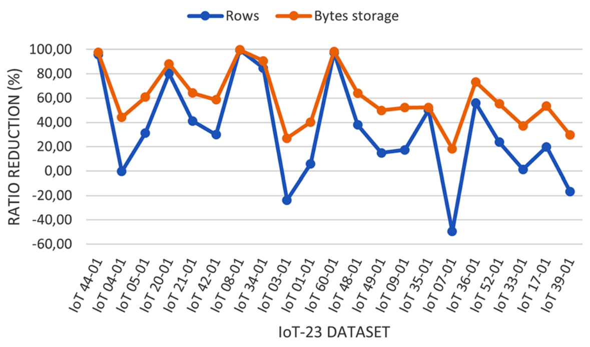


Fig. 10 Detail of the storage reduction for each file of the experiment. Comparison of rows and storage bytes between the original and the generated file

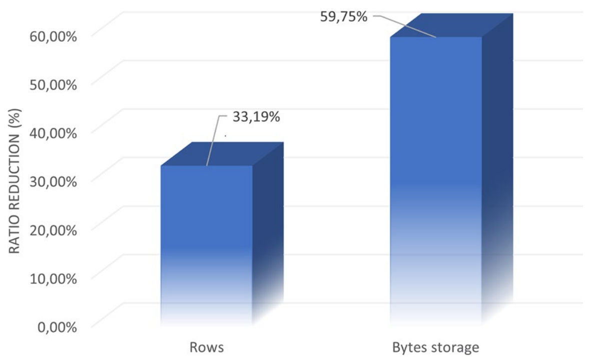


Fig. 11 Total storage reduction for the experiment dataset. Comparison of rows and storage bytes between the original and the generated dataset

and the quantity of devices and ports included in the CGNs. Most of the communications captured between devices and ports are in files 09_ 01.IoTMalware, 36_ 01.IoTMalware and 60_ 01.IoTMalware are malicious. However, benign communication is predominant in the remaining files.

We experimented with a machine learning model based on stacked autoencoders. Although it was an unsupervised model, we benefited from the fact that the input dataset was labelled, which allow us to use specific metrics from the supervised machine learning models. Classification performance was evaluated based on accuracy, precision, recall, and F1 score:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3a}$$

Table 5 Properties of the CGNs (nodes, arcs, and rows needed to represent them) compared with the flows contained in the files used for the experiments

Dataset	Malware	ZeekFlows	Zeek Files	CGN Files	Rows	IP addresses		IP addresses-ports	
			(Mb)	(Mb)	CGN file	Nodes	Arcs	Nodes	Arcs
01_01_IoTMaleware	Hide &Seek	1008749	126	70	640194	602833	620842	640194	620842
03_01_IoTMaleware	Mushtik	156104	21	10	91897	65010	65153	91897	81406
09_01_IoTMaleware	linux-hiajime	6378294	849	578	5289007	5220276	5221075	5289007	6369855
35_01_IoTMaleware	Mirai	10447796	1186	439	4170803	4104749	4170803	4105278	4375867
36_01_IoTMaleware	Okiru	13645107	1573	1512	13608115	13599603	13599713	13608115	13608335
60_01_IoTMaleware	GagFyt	3581029	411	7	65569	31	33	65569	65671

Table 6 Malware detected in each of the experiment files. Ports per device involved in the attacks

Dataset	Malware	Devices & ports	
		Benign	Malicious
01-01_IoTMaleware	Hide &Seek	441,334	210,748
03-01_IoTMaleware	Mushtik	534,951	88,586
09-01_IoTMaleware	linux-hiajime	18,639	5,276,049
35-01_IoTMaleware	Mirai	4,120,109	65,546
36-01_IoTMaleware	Okiru	1170	13,607,153
60-01_IoTMaleware	GagFyt	63	65,515

$$Precision = \frac{TP}{TP + FP} \tag{3b}$$

$$Recall = \frac{TP}{TP + FN} \tag{3c}$$

$$F1score = \frac{2 * TP}{(2 * TP) + FP + FN} \tag{3d}$$

where true positives (TP) are the normal behaviour items correctly classified as normal; false-positives (FP) are the malicious behaviour samples that are misclassified as benign; true negatives (TN) are the malicious behaviour elements that are correctly classified as attacks; and false-negatives (FN) are the abnormal behaviour items that are misclassified as normal.

We followed three different lines of research, all consisting of 360 test cases with different combinations of hyperparameters. A Cartesian search was used to find the best one. Table 7 shows the combined hyperparameters. Tests were performed with topologies with different depths and numbers of units per layer. Additionally, we evaluated the impact on the results of the number of bottleneck units and variations in the magnitude of the IQR. Moreover, different numbers of epochs and batch sizes were used.

First, we trained the model with all the features obtained from the CGN. We discarded the CCM feature during the preparation of data before training. In the following, we repeated the training without those metrics, omitting CCM and BCM to estimate their impact on classification. Finally, we trained the model with the input dataset based on network flows. The best results are shown in Table 8. The best results have been achieved with a 2-layer topology (7 and 5 units) with a bottleneck of 3 units. The IQR factor chosen has very little effect on the result, and other metrics will be explored in future studies.

Table 7 Hyperparameters and combined values for the Cartesian search of the best configuration of the trained ML model

Topology	Bottleneck	IQR factor	Epochs	Batch size
Units L1: 7 L2: 5	1	0.01	10	32
Units L1: 8 L2: 7 L3: 6 L4: 5	2	0.1	20	64
Units L1: 15 L2: 10 L3: 7 L4: 5	3	0.5		128
	4	1.5		
		3		

Discussion

The discussion of our findings in this article is presented in this section.

The stacked autoencoder-based model trained with CGN detects botnet attacks with high accuracy.

It requires less storage for training than a traditional flows-based model. The maximum space required is determined by the number of devices in the network, regardless of how connections are made between them. Therefore, the maximum space can be easily calculated and helps to manage the controller's storage resources, avoiding problems of lack of space and mitigating the

Table 8 Configuration and evaluation metrics of the best results obtained for the three lines tested in the experiments

Test	Hyperparameters	Accuracy	Precision	Recall	F1-score
CGN	Units L1: 7 L2: 5	Train: 94.29%			
	Bottleneck: 3	Test Benign:	98.09%	94.81%	96,42%
	IQR factor: 0.01	Test Malicious:	98.16%	99.59%	98.87%
	Epochs: 10	Test: 98.16%	98.13%	97.20%	97.65%
	Batch size: 32				
CGN w/o BCM, LCM, CCM	Units L1: 7 L2: 5	Train: 92.8%			
	Bottleneck: 3	Test Benign:	99.76%	89.05%	94.10%
	IQR factor: 0.01	Test Malicious:	95.52%	99.86%	97.64%
	Epochs: 20	Test: 96.7%	97.64%	94.46%	95.87%
	Batch size: 32				
Flows	Units L1: 8 L2: 7 L3: 6 L4: 5	Train: 73.50%			
	Bottleneck: 3	Test Benign:	27.76%	66.09%	39.24%
	IQR factor: 0.01	Test Malicious:	97.52%	88.02%	92.53%
	Epochs: 20	Test: 86.73%	62.64%	77.46%	65.88%
	Batch size: 32				

negative impact of increasing the density of connections in the network.

The volume of data that the controller sends to the devices is small, consisting only of control plane information. The information that the edge-IoT devices share with the decision server is also related to the control plane, so the IDS does not affect the confidentiality of the data plane.

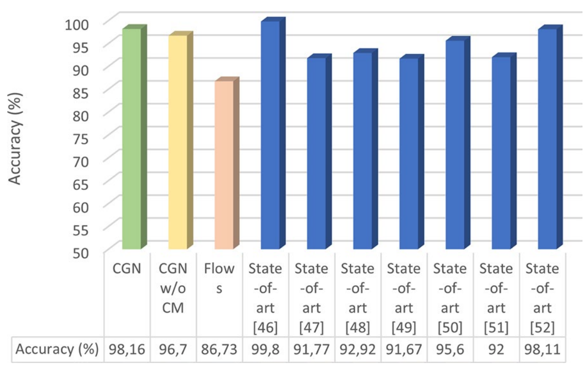
The model has a high classification performance with a simple topology consisting of few hidden layers and neurons in each layer. The model requires low memory due to the small number of parameters (255). Metrics are effective in representing the behaviour of network traffic. A few features are sufficient to represent it. Therefore, a deep topology is not necessary to obtain high accuracy. Additionally, since autoencoders tend to overfit, better results are achieved with shallow networks and fewer epochs.

The main drawback is the cost of computing some centrality metrics, which can be mitigated by omitting them from the model. The BCM, CCM and LCM centrality metrics, which are intended to capture the behaviour of P2P attacks, have been excluded due to their computing overhead which makes their calculation unaffordable in a real-world controller. The accuracy and precision of the classification were slightly reduced when evaluating samples not previously seen by the model. We have not studied in depth the impact of discarding metrics. General comparative experiments have been conducted. A specific experimentation plan is required to determine the impact and explore other centrality metrics.

Topologies used on data flows show very low precision in classifying normal traffic. Deeper topologies are required to improve the results.

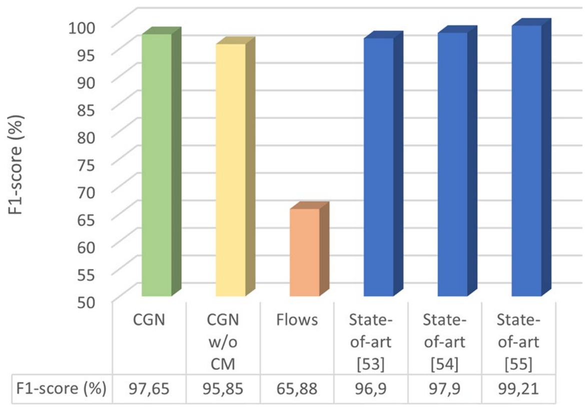
Finally, the proposed IDS was compared with other state-of-the-art works (Saba et al. 2021; Babu and Reddy 2020; Tian et al. 2021; Lin and Huang 2020; Nguyen et al. 2019; Liu et al. 2021; Shafee et al. 2020; Anthi et al. 2019; Huong et al. 2021; Zhao et al. 2020). The particular reason for choosing them for comparison is that they perform traffic classification of edge-IoT devices with different classifiers, and in general, it is performed on the device itself. Therefore, the effectiveness of the proposed IDS is compared to the state of the art considering accuracy and F1 score.

Figure 12 presents the comparison of the accuracy of the three proposed models with the works (Saba et al. 2021; Babu and Reddy 2020; Tian et al. 2021; Lin and Huang 2020; Nguyen et al. 2019; Liu et al. 2021; Shafee et al. 2020). Saba et al. (2021) details a model that combines genetic algorithms, support vector machines and decision trees, obtaining an accuracy of 99.80%. Tian et al. (2021) used a denoising autoencoder, achieving an accuracy of 92.92%. Lin and Huang (2020), Nguyen et al. (2019), Liu et al. (2021) utilized various types of recurrent neural networks with results of 91.67%, 95.6% and 92%, respectively. Finally, Shafee et al. (2020) proposes several models, where 98.11% is the highest accuracy.



	CGN	CGN w/o CM	Flows	State-of-art [46]	State-of-art [47]	State-of-art [48]	State-of-art [49]	State-of-art [50]	State-of-art [51]	State-of-art [52]
Accuracy (%)	98,16	96,7	86,73	99,8	91,77	92,92	91,67	95,6	92	98,11

Fig. 12 Comparison of the accuracy obtained in the experiments with the state of the art



	CGN	CGN w/o CM	Flows	State-of-art [53]	State-of-art [54]	State-of-art [55]
F1-score (%)	97,65	95,85	65,88	96,9	97,9	99,21

Fig. 13 Comparison of the F1-score obtained in the experiments with the state of the art

Figure 13 shows the comparison of the F1-score of our models with the works (Anthi et al. 2019; Huong et al. 2021; Zhao et al. 2020). Anthi et al. (2019) described a decision tree, reaching an F1-score of 96.9%. Huong et al. (2021) proposed a model that combines a variational autoencoder, and a recurrent neural network, achieving an F1-score of 97.9%. Zhao et al. (2020) used a recurrent neural network with results of 99.21%.

Our work obtains similar results to current state-of-the-art works with storage in reduction space.

Conclusions

In this article, we propose to represent the behaviour of IoT devices in an SDN using CGNs. It has been shown that CGNs, compared to traditional data, reduce the required amount of storage. Furthermore, CGNs simplify the typification of device-to-device traffic, and an implementation with few hidden layers and neurons can identify devices with anomalous behaviour. We have taken advantage of the low memory and reduction in computing resources to train a stacked autoencoder-based machine learning model that learns the normal communication pattern of network devices. This enables behaviour analysis to be conducted at the edge thereby distributing the computational effort associated with detecting botnet attacks. Experiments demonstrate that the accuracy and precision are comparable to those currently achieved in the literature. The next step is to execute an experimentation plan oriented toward exploring centrality metrics and their effects on classification accuracy with the focus on integrating them into an SDN controller.

In the future, we plan to integrate the AIDS prototype into a real SDN to explore how to minimize the time elapsed from the start of an attack until actions are applied to intercept it. Decreasing this delay to move closer to a real-time response would improve the network security and the ability to mitigate the consequences of attacks. Furthermore, we hope to assign a different class to each type of attack and thus automatically generate type-specific dynamic security policies.

Abbreviations

AIDS	Anomaly IDS
BCM	Node betweenness
C2	Command and control
CCM	Clustering coefficient
CGN	Communications graphs of network
CSV	Comma-separated value
DNS	Domain name server
DVC	Data version control
EVM	Eigenvector centrality
FN	False negative
FP	False positive
IDM	In degree
IDS	Intrusion detection systems
IoT	Internet of Things
IP	Internet protocol
IQR	Interquartile range
IWM	In weight degree
LCM	Node closeness
MLOps:	Operationalization management
ODM	Out degree
OWM	Out weight degree
P2P	Peer to peer
ReLU	Rectified linear unit
SDN	Software-defined network
TN	True negative
TP	True positive

Acknowledgements
Not applicable.

Author contributions
All authors read and approved the final manuscript.

Funding
Not applicable. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Availability of data and materials
Dataset used in paper is publicly available in Garcia et al. (2020).

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Inetum España S.A., C/ María de Portugal, 9 - 11, Building 1, 28050 Madrid, Spain. [2]Department of Computer Engineering, Polytechnic School, University of Alcala, Madrid-Barcelona Road Km 33.6. Alcala de Henares, 28871 Madrid, Spain.

Received: 7 March 2023 Accepted: 14 June 2023
Published online: 03 December 2023

References
Ahmed U, Raza I, Hussain SA, Syed A, Amjad A, Muddesar I (2015) Modelling cyber security for software-defined networks those grow strong when exposed to threats. J Reliable Intell Environ 1:123–146
Alaba FA, Othman M, Hashem IAT, Alotaibi F (2017) Internet of things security: a survey. J Netw Comput Appl 88:10–28
Anthi E, Williams L, Słowińska M, Theodorakopoulos G, Burnap P (2019) A supervised intrusion detection system for smart home iot devices. IEEE Internet Things J 6(5):9042–9053
Babu MJ, Reddy AR (2020) Sh-ids: specification heuristics based intrusion detection system for iot networks. Wireless Pers Commun 112:2023–2045
Bank D, Koenigstein N, Giryes R (2020) Autoencoders. arXiv:2003.05991
Benzekki K, El Fergougui A, Elbelrhiti Elalaoui A (2016) Software-defined networking (sdn): a survey. Secur Comm Netw 9:5803–5833
Brandes U (2001) A faster algorithm for betweenness centrality. J Math Sociol 25:163–177
Brandes U, Pich C (2007) Centrality estimation in large networks. Int J Bifurc Chaos 17(7):2303–2318
Check Point: Check Point Software's 2023 Cyber Security Report (2023). https://pages.checkpoint.com/cyber-security-report-2023.html Accessed 20 Feb 2023
Choi H, Lee H, Lee H, Kim H(2007) Botnet detection by monitoring group activities in dns traffic. In: 7th IEEE international conference on computer and information technology (CIT 2007), pp 715–720
Chowdhury S, Khanzadeh M, Akula R (2017) Botnet detection using graph-based feature clustering. J Big Data 4:14
Daya AA, Salahuddin M, Limam N, Boutaba R (2019) A graph-based machine learning approach for bot detection. arXiv
Douceur JR (2002) The sybil attack. In: Springer (ed.) International workshop on peer-to-peer systems. Lecture notes in computer science: 2002; Heidelberg, vol 2429
Garcia S, Parmisano A, Erquiaga MJ (2020) IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) . https://www.stratosphereips.org/datasets-iot23 Accessed 10 Feb 2022
Geetha A, Sreenath N (2016) Byzantine attacks and its security measures in mobile adhoc networks. Int J Comput Commun Instrum Eng (IJCCIE) 3(1):42–47
Hafeez I, Antikainen M, Tarkoma S (2019) Protecting iot-environments against traffic analysis attacks with traffic morphing. In: 2019 IEEE international conference on pervasive computing and communications workshops (PerCom Workshops), pp 196– 201
Horrow S, Sardana A (2012) Identity management framework for cloud based internet of things. In: Proceedings of the first international conference on security of internet of things (SecurIT '12), pp 200– 203
Hu YC, Perrig A, Johnson DB (2003) Packet leashes: a defense against wormhole attacks in wireless networks. In: IEEE INFOCOM 2003. Twenty-second annual joint conference of the IEEE computer and communications societies (IEEE Cat. No.03CH37428), vol 3, pp 1976– 1986
Hung-Jen Liao L, Chun-Hung RL, Ying-Chih L, Kuang-Yuan T (2013) Intrusion detection system: a comprehensive review. J Netw Comput Appl 36(1):16–24
Huong T, Bac T, Long D, Luong T, Dan N, Quang L, Cong L, Thang B, Tran K (2021) Detecting cyberattacks using anomaly detection in industrial control systems: a federated learning approach. Comput Ind 132:103509
Ioffe S, Weiqing S(2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv
Jindal K, Dalal S, Sharma KK(2014) Analyzing spoofing attacks in wireless networks. In: 2014 fourth international conference on advanced computing & communication technologies, pp 398– 402
Kang U, Papadimitriou S, Sun J, Tong H (2011) Centralities in large networks: Algorithms and observations, pp 119– 130
Karim A, Salleh R, Shiraz M, Shah S, Awan I, Anuar N (2014) Botnet detection techniques: review, future trends, and issues. J Zhejiang Univ Sci C 15:943–983
Khraisat A, Gondal I, Vamplew P, Kamruzzaman J (2019) Survey of intrusion detection systems: techniques, datasets and challenges. Cybersecurity 2(1):1–22
Kibirige G, Sanga C (2015) A survey on detection of sinkhole attack in wireless sensor network. Int J Comput Sci Inf Secur 13:1–9
Limarunothai R, Munlin MA (2015) Trends and challenges of botnet architectures and detection techniques. J Inf Syst Telecommun 5(1):51–57
Lin K, Huang W(2020) Using federated learning on malware classification. In: 2020 22nd international conference on advanced communication technology (ICACT), pp 585– 589
Liu Y, Garg S, Nie J, Zhang Y, Xiong Z, Kang J, Hossain M (2021) Deep anomaly detection for time-series data in industrial iot: a communication-efficient on-device federated learning approach. IEEE Internet Things J 8(8):6348–6358
Lu Z, Lu X, Wang W, Wang C (2010) eview and evaluation of security threats on the communication networks in the smart grid. In: 2010 Military Communications Conference, pp. 1830– 1835
Luo T, Nagarajan SG (2018) Distributed anomaly detection using autoencoder neural networks in wsn for iot. In: 2018 IEEE International Conference on Communications (ICC), pp. 1– 6
Malladi S, Alves-Foss J, Heckendorn RB (2002) On preventing replay attacks on security protocols. Department of Computer Science University of Idaho
Mendes LDP, Aloi J, Pimenta TC(2019) Analysis of iot botnet architectures and recent defense proposals. In: 2019 31st international conference on microelectronics (ICM), pp 186– 189
Mirsky Y, Doitshman T, Elovici Y, Shabtai A (2018) Kitsune: an ensemble of autoencoders for online network intrusion detection. arXiv:1802.09089, pp 665–674
Murray SN, Walsh BP, Kelliher D, O'Sullivan DTJ (2014) Multi-variable optimization of thermal energy efficiency retrofitting of buildings using static modelling and genetic algorithms–a case study. Build Environ 75:98–107
Nguyen TD, Marchal S, Miettinen H M andFereidooni Asokan N, Sadeghi AR (2019) DÏot: A federated self-learning anomaly detection system for iot. In: International conference on distributed computing systems, pp 756– 767
Niyaz Q, Weiqing S, Javaid AY (2017) A deep learning based ddos detection system in software-defined networking (sdn). EAI Endorsed Trans Secur Saf 4:2
Pawar MV, Anuradha J (2015) Network security and types of attacks in network. Procedia Comput Sci 48:503–506
Rumelhart DE, Hinton GE, Williams RJ (1986) Learning internal representations by error propagation. In: Parallel distributed processing: explorations in the microstructure of cognition pp 318–362
Saba T, Sadad T, Rehman A, Mehmood Z, Javaid Q (2021) Intrusion detection system through advance machine learning for the internet of things networks. IT Prof 23(2):58–64
Sarica AK, Angin P (2020) Explainable security in sdn-based iot networks. Sensors 20(24):7326
Sengupta T, De, S, Banerjee I (2021) A closeness centrality based p2p botnet detection approach using deep learning. In: 12th international conference on computing communication and networking technologies (ICCCNT), pp 1– 7
Shafee A, Baza M, Talbert DA, Fouda MM, Nabil M, Mahmoud M (2020) Mimic learning to generate a shareable network intrusion detection model. In: 2020 IEEE 17th annual consumer communications networking conference (CCNC), pp 1– 6
Shinan K, Alsubhi K, Alzahrani A, Ashraf MU (2021) Machine learning-based botnet detection in software-defined network: A systematic review. Symmetry 13(5):866

Silva S, Silva R, Pinto R, Salles R (2013) Botnets: a survey. Comput Netw 57:378–403

Tanenbaum A, Wetherall D (2011) Computer Networks, 5th edn. Pearson, Boston

Thakkar A, Lohiya R (2019) Review on machine learning and deep learning perspectives of ids for iot: recent updates, security issues, and challenges. Arch Computat Methods Eng 28:3211–3243

Tian P, Chen Z, Yu W, Liao W (2021) Towards asynchronous federated learning based threat detection: a dc-adam approach. Comput Secur 108:102344

Venkatesh B, Choudhury SH, Nagaraja S (2015) Botspot: fast graph based identification of structured p2p bots. J Comput Virol Hack Tech 11:247–261

Xu B, Szegedy C (2015) Empirical evaluation of rectified activations in convolution network. arXiv:1505.00853

Zeidanloo HR, Manaf AA (2009) Botnet command and control mechanisms. In: 2009 second international conference on computer and electrical engineering, pp 564– 568

Zhao R, Yin Y, Shi Y, Xue Z (2020) Intelligent intrusion detection based on federated learning aided long short-term memory. Phys Commun 42:101157

Zhao K, Ge L(2013) A survey on the internet of things security. In: 2013 ninth international conference on computational intelligence and security, pp 663– 667

Zhou W, Jia Y, Peng A, Zhang Y, Liu P (2019) The effect of iot new features on security and privacy: new threats, existing solutions, and challenges yet to be solved. IEEE Internet Things J 6(2):1606–1616

Zhou C, Paffenroth R(2017) Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD international conference, pp 665– 674

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.