IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 17, 2020, accepted March 30, 2020, date of publication April 15, 2020, date of current version May 1, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2988055*

# Artificial Neural Networks-Based Intrusion Detection System for Internet of Things Fog Nodes

**JESUS PACHECO[1], VICTOR H. BENITEZ[1], LUIS C. FÉLIX-HERRÁN[2], AND PRATIK SATAM[3]**

[1]Department of Industrial Engineering, Universidad de Sonora, Hermosillo 83000, Mexico
[2]School of Engineering and Sciences, Tecnológico de Monterrey, Hermosillo 83000, Mexico
[3]Department of Electrical and Computer Engineering, The University of Arizona, Tucson, AZ 85719, USA

Corresponding author: Victor H. Benitez (vbenitez@industrial.uson.mx)

This work was supported in part by the Consejo Nacional de Ciencia y Tecnología (CONACYT) through the Fondo de Cooperación Internacional en Ciencia y Tecnología del Conacyt (FONCICYT) Project entitled Cloud and Autonomic Computing Center, Universidad de Sonora, under Grant 296323. Authors are grateful to Tecnologico de Monterrey, Vicerrectory of Research and Technology Transfer for funding the publication cost of this research.

**ABSTRACT** The Internet of Things (IoT) represents a mean to share resources (memory, storage computational power, data, etc.) between computers and mobile devices, as well as buildings, wearable devices, electrical grids, and automobiles, just to name few. The IoT is leading to the development of advanced information services that will require large storage and computational power, as well as real-time processing capabilities. The integration of IoT with emerging technologies such as Fog Computing can complement these requirements with pervasive and cost-effective services capable of processing large-scale geo-distributed information. In any IoT application, communication availability is essential to deliver accurate and useful information, for instance, to take actions during dangerous situations, or to manage critical infrastructures. IoT components like gateways, also called Fog Nodes, face outstanding security challenges as the attack surface grows with the number of connected devices requesting communication services. These Fog nodes can be targeted by an attacker, preventing the nodes from delivering important information to the final users or to perform accurate automated actions. This paper introduces an Anomaly Behavior Analysis Methodology based on Artificial Neural Networks, to implement an adaptive Intrusion Detection System (IDS) capable of detecting when a Fog node has been compromised, and then take the required actions to ensure communication availability. The experimental results reveal that the proposed approach has the capability for characterizing the normal behavior of Fog Nodes despite its complexity due to the adaptive scheme, and also has the capability of detecting anomalies due to any kind of sources such as misuses, cyber-attacks or system glitches, with high detection rate and low false alarms.

**INDEX TERMS** Anomaly behavior, cyber security, fog computing, IoT, neural networks.

## I. INTRODUCTION

The growth in the use of mobile computing, social media technologies, cloud and pervasive computing, and the explosive growth and acceptance of Software as a Service (SaaS) has derived into the development of next-generation of Internet services that are pervasive and touch every aspect of modern life, as it is the case of the Internet of things. It is projected that there will be 75 billion IoT devices connected to the internet globally by 2025; making IoT technology a 7.5 trillion dollar market [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhui Yuan.

The advent of Fog computing has led the computation hosting services to be moved to the network edge, reducing the latency induced by communication. Fog computing allows IoT services to become the key technology for the development of smart cities enabling a revolution in the way business is done, health services are provided, critical infrastructure is managed, resident safety and security is maintained, education is provided, etc. [2], [3].

Although this use of Fog computing and IoT application has led to the growth of Smart Infrastructures, Smart Buildings and Smart Cities [3]–[5], it has also led to an increase in attack surfaces that attackers can target to exploit vulnerabilities. IoT usage has exposed devices and applications to

attackers at a scale like never before. IoT devices, designed to work in isolated environments, are now connected to a wider area network to satisfy particular requirements (e.g., remote administration requirements). This increases the attack surface of IoT systems, making them vulnerable to attacks that might lead to the delivery of inaccurate information to the end-users, resulting in catastrophic consequences when these users respond to this incorrect information, e.g., the Stuxnet attack [3], or face service outages wherein the user is unable to control his IoT device or Fog nodes [4]. The authors in [4] highlight the relevance of using a resilient Data Distribution algorithm to avoid data to be lost during connectivity outage periods caused by issues like regular maintenance, hardware constraints (e.g. buffer size), or cyber-attacks.

Diverse strategies about fog computing and IDS have been reported. Sohal *et al*. in [6] present a literature review of different network devices employed in Fog computing including routers, switches, and hubs. The authors present an IDS that make use of Virtual Honeypot Devices together with Markov models with the goal of identifying compromised edge devices in a fog environment. Another intrusion detection effort has been reported by Shafi *et al.* [7]. The authors developed a fog-assisted software design networking (SDN) solution through a computational arrangement with IoT network elements. The proposed system was able to identify attacks at the right time by employing four machine learning classifiers automatically detect attacks. Intrusion detection has been also possible by analyzing IDS log statistics in the fog nodes with a query-based strategy plus uncertainty tests to calculate the degree of a potential threat. This approach was tested for fog radio access networks (F-RANs) [8]. A different approach reported an IDS architecture for edge computing. To deal with the limitations in edge nodes, the solution handled a multilayer dominant and max-min fair (MDMMF) allocation of resources to improve IDS computational and storage efficiency [9].

A type of threat for fog computing who has gained importance is the Distributed Denial of Service (DDoS) attack. This type of invading agent illegally appropriates resources of the fog node. A solution is to modify a traditional IDS to generate a fog computing intrusion detection system (FC-IDS) framework. An *et al.* [10] proposed a hypergraph clustering model based on inferred decisions. Data mining work was carried out to study the type of link between the DDoS and the fog node under attack. With this knowledge and additional information provided by another fog node, a description of the attack could be obtained to respond with an adequate security action plan.

In this paper, we introduce a methodology to protect IoT Gateways and Fog devices against cyber-attacks. The aim of the proposed methodology is the assurance of Fog devices availability, despite the origin of abnormalities such as cyber-attacks, human errors and regular churn conditions, to name few. The benefit of applying the proposed ABA-IDS at fog level instead of applying it at cloud or end-devices level is the fact that Fog Nodes at the edge of the network, are necessary to communicate end-devices, which are constrained in memory, with the Cloud, where more sophisticated detection systems can be applied. Pacheco and Hariri in [11] presented an approach to develop a threat modeling methodology to recognize vulnerabilities in each of the four layers in IoT device architecture: devices, network, services and applications, and present countermeasures to mitigate each of the vulnerabilities. In [11] authors present a technique to detect anomaly behavior on compromised sensors. They developed a threat model that identifies attacks against end nodes, network, service and application layers. However, their approach is developed under the premise that the amount of sensors is limited; if this condition is not fulfilled, then data association is required in order to track signals of several classes. To deal with such scenario, fog computing and adaptive schemes based on machine learning are more appropriate. The adaptive properties of neural networks are incorporated to reinforce the ABA-IDS methodology proposed in [11], in order to address the data association requirement where a large amount of sensor is presented. In this paper, the methodology presented in [11] is extended to the design and development of an adaptive Anomaly Behavior Analysis Intrusion Detection System (ABA-IDS) using Artificial Neural Networks (ANN) [12], [13] to model the normal behavior of Fog and IoT devices. The performance of the approach was measured against attacks like the Replay, Flooding and DoS attacks on an IoT testbed, developed in the Center for Cloud and Autonomic Computing (CAC), at the University of Sonora. The results obtained demonstrate that the proposed ABA-IDS methodology can be used to deploy security methods capable of protecting the normal functionality of IoT Gateways and Fog devices. The approach was successfully able to detect known and unknown abnormalities such as cyber-attacks applied to IoT end nodes exhibiting high detection rate (up to 93%) with low false alarms (less than 3.3%) while introducing low overhead (up to 13% execution time overhead).

The rest of the paper is organized as follows. Section II offers the required information about basic concepts of fog computing, cyber security for the IoT, intrusion detection based on abnormal behavior, and the threat model applicability. Section III exposes the proposed security framework that can be used for IoT applications. Section IV focuses on the description of the ABA methodology. In section V, the experimental setup is described along with a brief discussion of the obtained results. Section VI concludes the paper summarizing the findings and providing potential research directions.

## II. BACKGROUD

### A. FOG COMPUTING

Fog Computing technology extends the Cloud computing paradigm to the edge of the computational network, enabling a wide range of applications and services that exhibit lower latency, better awareness for location services, mobility, and elasticity [14]–[16]. Fog computing has been seen to be effective in supporting IoT applications that require predictable latency. For example, in [17] the authors described

an approach to secure fog-based systems under Byzantine attacks while enhancing the efficiency of data processing for IoT applications. Fog computing leverage IoT-based systems by providing the required mechanisms to ensure confidentiality, integrity, and availability (CIA) to the IoT infrastructure.

### B. IoT CYBER SECURITY

The IoT allows the operation and administration of a large variety and quantity of devices that are heterogeneous, by gathering and managing information as well as smart objects [18]. It represents interconnected systems and devices that comprise a large range of technologies including sensors, actuators, communication networks, etc. [19]. The heterogeneity of resources and dynamic utilization of services turns cybersecurity into a major problem because existing cybersecurity solutions are not necessarily appropriate for IoT-based systems due to [18], [19]: 1) IoT spreads ''internet'' through traditional networks, including the current Internet; 2) Most smart objects lack computational resources required to support complex security algorithms; 3) High interconnectivity in IoT devices, leads to multiple entry points that can be exploited to target the network; and 4) Shared IoT devices and services are prone to have different policies.

These issues are required to be tackled in order to build reliable IoT-based applications, where Confidentiality, Integrity, and Availability must be guaranteed. Therefore, there is a great research concern in developing novel security techniques that can secure and protect IoT applications and services [20].

### C. ANOMALY BEHAVIOR ANALYSIS

The growth of cloud computing and IoT have brought their own set of challenges in the form of increased attack surfaces and data security. Current cyber security solutions are not capable of stopping these threats in terms of their efficacy and scalability [17], [21]. In addition, there is a trend in increasing attack sophistication and speed of attack propagation as the internet has reached a global scale, making it possible to launch sophisticated attacks at little or no development costs in a few seconds to target entities across the globe [20]. To address this threat, there is a need to design Intrusion Detection Systems (IDS) that will be able to detect these sophisticated attacks before they cause significant damage to the target. There are two main methodologies to design IDSs: Signature based IDS and Anomaly based IDS [22]–[25]. Signature based IDS use known attack signatures to detect attacks, making them incapable of detecting new or modified attacks. Anomaly based IDS use modeling techniques like statistical modeling, machine learning, and deep learning to model the normal behavior of the system, making them capable of detecting not only known attacks but also new (zero day) or modified attacks.

The key feature of the anomaly detection approach is the capability of new attack detection. An anomaly-based IDS first defines a model of normal characteristics of the system through off-line training. Any activity outside this normal behavior is labeled to be abnormal behavior (caused due to potential attack or misconfiguration). Historically anomaly behavior analysis has been associated with high false positive rates. This drawback can be by performing a fine-grained behavior analysis while modeling the system behavior as shown by Satam *et al.* in [25].

### D. THREAT MODELING

Developing appropriate countermeasures to mitigate threats heavily depends on analyzing the system's vulnerabilities and the associated risks [26]. A threat model defines potential threats and correlates them with associated risks. This correlation helps in the analysis of glitches, as well as in the design of mitigation strategies plans before deploying the system. It also helps to prioritize what is required to be protected in case the solution is not feasible. A threat model is useful for detecting changes that need to be applied to an initial layout/architecture to minimize possible system threats. The general steps to create a threat model are: 1) Identification of potential assets and their associated threats; 2) rank the risks; 3) choose strategies to mitigate the threats; and 4) develop solutions based on the best possible strategies [26], [27]. The listed steps will be followed to study an IoT Fog node.

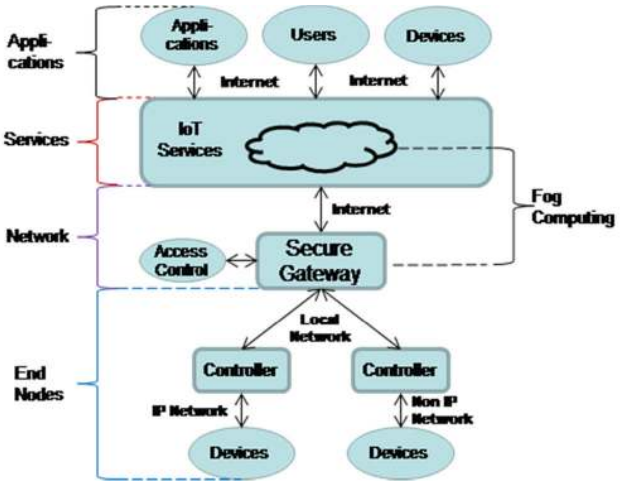



**FIGURE 1.** IoT framework defined in multiple layers [7].

## III. IoT SECURITY FRAMEWORK FOR SMART INFRASTRUCTURES

Several IoT frameworks and architectures can be utilized to establish a threat model and apply mitigation schemes [28]–[30]. Fig. 1 illustrates the general framework employed in this study. The framework was introduced and extensively explained in [11] and can be used for the development of security mechanisms in IoT-based systems. The framework contains four layers: end devices, Network, Services, and Applications. Fog computing is a key component for linking end devices layer with the service layer. Cyberattacks and other threats can influence the functionality in each level shown in Fig. 1. For each layer, risks are weighted

in considering target, impact, and effectiveness of known mitigation techniques.

In the first layer (perception layer), the information is taken by physical devices to identify the physical world or apply control to it [11]. The key components (targets) in this layer are sensors, actuators, and local controllers. Any attack targeting this layer will result in a loss of life, monetary loss or economic loss, and loss in service providers reputation. Mitigation mechanisms include lightweight encryption, authentication, IDS, anti-jamming, and behavior analysis.

The network layer is in charge of information exchange from/to final devices [30]. Communication technologies such as mobile communication networks, infrastructures for networking, protocols, and the Internet itself, constitute this layer. Network security is responsible for defending against cyber-attacks targeting infrastructures such as the Fog nodes, and information embedded in protocols. An attack on this layer can cause monetary loss, damage in reputation, and excessive energy consumption. Network mitigation methods include access control, anti-DoS, encryption, packet filtering, congestion control, anti-jamming, and behavior analysis IDS.

The services layer provides the required computational power by implementing Cloud services as well as Fog services [29], [30]. In this layer, the targets are confidential information, sensors and actuators, and monitor/control functions. An attack on this layer will cause a loss of safety, monetary loss, and information leakage. This layer can be secured by the implementation of encryption, access control, period identifiers, selective data disclosure, and behavior analysis.

The application layer presents customized services to end-users [30]. In this layer, data sharing is an important feature and consequently, cybersecurity must address privacy, access control, and data disclosure. The impacts could be in unauthorized access to data, disclosure of critical information and damage in reputation, and excessive energy consumption. Reported mitigation techniques include data encryption, and access control [31], [32].

### A. INTERNET OF THINGS TESTBED

The Fig. 2 depicts an overview of the IoT testbed at the CAC center at the University of Sonora. This testbed follows the architecture in Fig. 1 and can be split into the same four layers. In the testbed the components are sensors like temperature, current, and water flow; actuators like electric valves, fan, lights, door locks; and control units like PLC's, NI CompactRIO, and Arduino UNO as the end nodes. Amazon Web Services and Microsoft Azure over wired ethernet network and Wi-Fi network form the services and the network layers.

The shown characteristics are considered as a minimum to deploy the proposed system with acceptable overhead. In the case of resources constrained devices, other methodologies such as rule-based approaches can be applied as discussed in [11]. To demonstrate the methodology described in this paper, a raspberry pi3 model B [33] configured as a fog node with internet access will be used. The node is a key

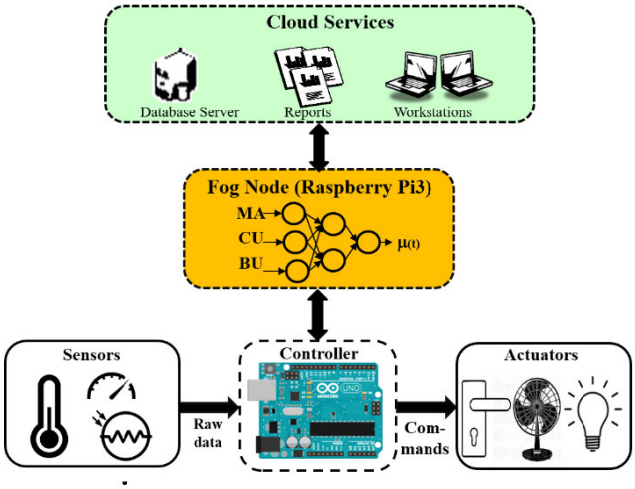



**FIGURE 2.** IoT testbed overview.

component in the IoT testbed, it contains 4 ARM Cortex-A53 cores, with 1.2GHz, 1GB LPDDR2 (900 MHz) ram memory, and 32 GB storage. It works under Raspbian lite (Debian) for ease of configuration.

### B. IDENTIFICATION OF ATTACK SURFACE

Systems can be compromised by deploying cyberattacks inside the operating ecosystem or by launching an attack from an outside location [34]. Both scenarios will make use of the system's resources, methods, and data to initiate the attack. In this research, the security of an IoT application is considered with respect to the local and public networks [35]. Local networks include controllers and devices, communications and gateways, while public networks include IoT services and applications. From Fig. 1 an attack surface can be derived as shown in Table 1.

**TABLE 1.** Attack surface for IoT architecture.

| Location | Attack surface |
|---|---|
| Local Network (Insider attack) | Device attacks another Device<br>Device attacks a Controller<br>Controller attacks the Gateway<br>User attacks the Gateway |
| Public Network (Outsider attack) | User affects IoT services<br>Service affects another service<br>Application affects a service<br>IoT device affects a service |

This work focuses on the security of a Fog node (Gateway) implemented on a Raspberry Pi 3 configured to perform communications between other Fog nodes and IoT subsystems such as the Smart Water Testbed introduced in [35]. Fog nodes security is crucial to develop trustworthy IoT applications and services, providing resiliency and preventing cyber threats to be disseminated among other IoT subsystems. In the context of this study, trustworthy service is defined as the one capable of performing self-protection against cyberattacks (self-protect), that can operate normally meeting required

performance goals regardless of operational conditions (self-optimization), and can update its configuration to comply with new requirements (self-configuration).

## IV. ANOMALY BEHAVIOR ANALYSIS METHODOLOGY

ABA aims at modeling the usual behavior of a system, such that it is able to identify any abnormal behavior i.e. an attack on the target system that it is modeling [5], [36]. The proposed methodology focuses on the availability of the secure gateway (see Fig. 1) to recognize potential threats that can affect its functionality, preventing it to deliver the information where required. The modeling of the Fog node is carried out by foot-printing features like system memory, CPU usage, hardware configuration, etc. Fig. 3 depicts the ABA deployment methodology.

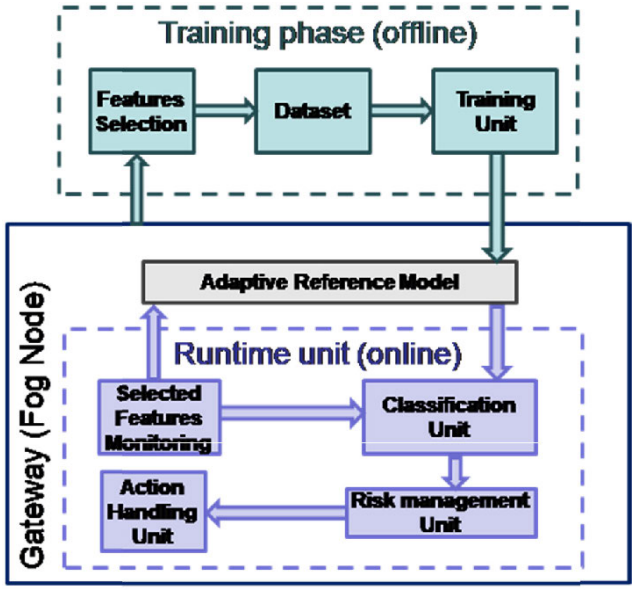



**FIGURE 3.** Anomaly behavior analysis deployment methodology.

The algorithm that explains the procedure depicted in Fig. 3 is presented in Table 2.

**TABLE 2.** Anomaly behavior analysis general algorithm.

| Anomaly behavior analysis Algorithm |
|---|
| **Input:** /proc directory |
| **Output:** Alert and rank for abnormality |
| **while** true **do** |
| read files in root/proc directory |
| extract features for Buffer (BU), CPU (CU), and memory (MA) |
| **if** (compare(features) != normal) **then** |
| trigger alert for abnormality |
| rank the abnormality severity |
| execute the required corrective activities |
| **end if** |
| **end while** |

### A. TRAINING PHASE (OFFLINE)

The training phase in the ABA methodology is carried out offline and is used to characterize the normal behavior of the Fog node. In what follows, the steps of the training phase are explained.

#### 1) FEATURES SELECTION

The correlation of 260 system variables or features was verified using the Pearson product-moment correlation coefficient technique [37]. The results show that 11 features are sufficient to describe the node normal behavior, these features are: 1) available memory (AM), 2) buffers utilization (BU), 3) CPU utilization (CU), 4) sockets (SO), 5) processes (PO), 6) process running (PR), 7) Active Connections (AC), 8) WLAN Reception (WR), 9) WLAN Transmission (WT), 10) Ethernet Reception (ER), and 11) Ethernet Transmission (ET). These features will constitute the dataset after being collected. The same features are inspected online as they will be used to build the reference model (off-line) and later compared with the on-line model.

#### 2) DATASET

In the offline stage, the IoT testbed was used to create the training dataset. For each inspection, the information (features) is stored in a MySQL database [38] which will be used to train the ANN-based model. Legitimate commands were executed on the testbed to collect the data for the original feature set (e.g. Open_Actuator_1, Read_Sensor_1, etc.). On completion of retrieving all the information for a specific command, the next instruction is processed. Those steps are repeated for all available commands until the incoming traffic shows similarity, meaning that the command has been fully processed. The universe should be $U = N \cup A$ for all data in the dataset, where N represents the normal behavior and A represents abnormality. However, Equation (1) shows a more precise description for the described method.

$$U = N \cup A + N^u \tag{1}$$

where $N^u$ is the non-classified normal traffic. The probability of getting false positives (false alerts) will rise as $N^u$ increases. Therefore, the accurateness of the reference model will strongly depend on the quality and quantity of information in the dataset.

#### 3) TRAINING UNIT

The training unit is the knowledge builder of the behavioral analysis. Required features (recall the features selection module), stored in the dataset, are internally taken from the system to perform the offline training of an Artificial Neural Networks (ANNs) cluster that will be formally defined in subsection B. Table 3 displays the algorithm's steps to train one ANN.

By following the steps listed in Table 3, a cluster of ANNs was tuned and used as the reference model. The next step is to calibrate the ANNs to predict the trend in the extracted features, this task is performed taking runtime information.

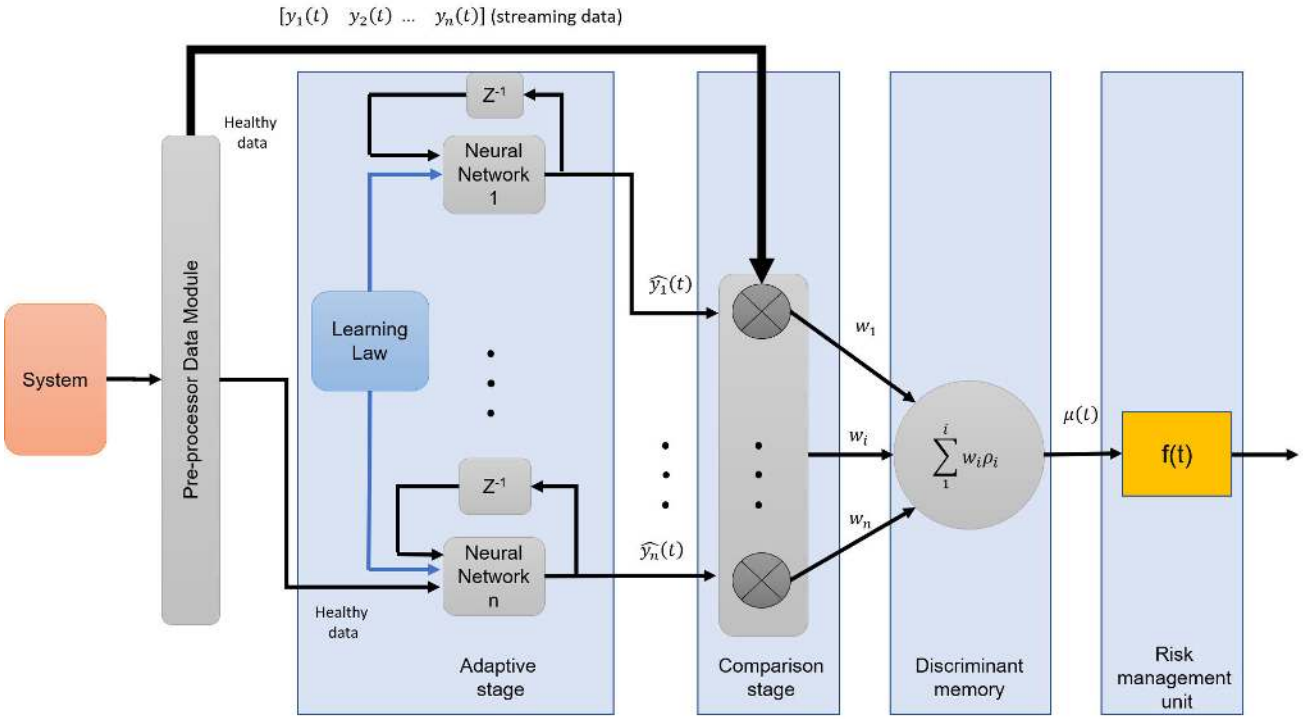



**FIGURE 4.** Anomaly behavior analysis deployment methodology.

**TABLE 3.** NN training process: prediction of series $\hat{Y}$ (t) given D past values of y(t).

| **Neural network training Process** |
|---|
| **Input:** Raw data features |
| **Output:** Prediction of $\hat{Y}$ |
| **Step 1**. Preprocessing: |
| • Select variable of interest to monitor; |
| • Scale raw data with the min - max criteria; |
| • Randomly divide up the target data: training (70%), validation (15%) and testing (15%). |
| **Step 2.** Neural network architecture: |
| • Select the number of hidden neurons; |
| • Select the number of tapped delays D. |
| **Step 3.** Training: |
| • Apply the Levenberg Marquardt learning law to update neural parameters to reduce the error; |
| • Validate and test the neural network output $\hat{Y}$. |

### B. REFERENCE MODEL

The reference model relies on the adaptive properties of the ANNs, whose properties are well known and described in the literature [39]. Accurate detection of attacks and failures is crucial for the discrimination of normal vs abnormal operations [40]. The scheme developed in this paper leans on an ANN Cluster (ANN-C) architecture as shown in Fig. 4. The proposed ANN-C is architected upon a) an adaptive stage, b) a comparison stage, c) a discriminant memory, and d) a risk management unit.

#### 1) ANN MODELS (NNI)

Neural Networks are frequently employed for time-series prediction in non-deterministic scenarios, they are configured to calculate future values $\{y_{N+1}, y_{N+2}, \ldots\}$ given a time-series represented by N values $\{y_1, y_2, y_N\}$. The adaptive model is tuned by the training unit using healthy information (as defined in Section V) using a nonlinear autoregressive model (NAR) [12], [13], as shown in Equation (2) to be integrated with the architecture as depicted in Fig. 4.

$$\hat{y} = f(y(t-1), \ldots, y(t-d)), \tag{2}$$

where d is the feedback unit represented as *Delay n* in Fig. 4.

#### 2) COMPARISON STAGE

To determine the amount of drift between normal behavior $y_i\,(t)$ and the NNi output, a residual signal is generated [40].

The comparison module is designed to obtain the residuals $\rho_i\,(t)$ defined as

$$\rho_i(t) = y_i(t) - \hat{y}_i(t), \tag{3}$$

where $y_i\,(t)$ are the data generated by the system operation and $\hat{y}_i\,(t)$ are the data predicted by the ith-NN module. The following elements (Discriminant Memory and Risk Management Unit) are used for the runtime unit (see Fig. 3). In what follows, the runtime unit is described.

### C. RUNTIME UNIT

The runtime unit (Fig. 3) is in charge of the behavioral classification of the system (normal or abnormal), as well as

of ranking the impact of an abnormality. Once an abnormality is detected and ranked, the required mitigation mechanism is applied.

#### 1) ONLINE MONITORING

A key ability of the ABA is monitoring in runtime. The system's information was monitored from files in /proc directory (Debian Operating System). The data was retrieved with a daemon that running independently, and automatically overcomes from crashes. The runtime unit monitors the features (see Equation (4)) whose output is fed into the classification unit to build the model to be used in runtime.

$$y = \begin{bmatrix} AM & \cdots & ET \end{bmatrix}. \tag{4}$$

#### 2) CLASSIFICATION UNIT

Using the residuals, the following function is proposed:

$$\mu(t) = \sum_{i}^{N} w_i \rho_i(t). \tag{5}$$

which evaluates the contribution of each residual obtained after all comparisons. As can be seen, Equation (5) describes a Discriminant Memory (DM). The DM characterizes the drift in the normal behavior of the system. Equation (5) is parametrized by weighting values $w_i$ which are computed online considering the normalized version of the LMS cost criteria as follows:

$$w_i = \frac{1}{\sigma^2 N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2, \tag{6}$$

where $w_i$ are positive values that weigh the contribution of the residuals to the function $\mu$(t). Is important to remark that in the absence of a fault condition $\rho_i(t) = 0$, the output of the DM is only due to noise which does not affect the rule mechanism.

#### 3) RISK MANAGEMENT UNIT

This unit chooses the appropriate mitigation method and prioritizes the actions to be taken. This unit provides a label that will be used to take the required action if an alert is triggered. It maps the output of the classification unit into an alert code represented by a label, which is forwarded to the action handling unit. Equation (7) shows the definition for this unit

$$f(t) = \begin{cases} Label_1 & if \quad \mu(t) \leq \tau_1 \\ Label_2 & if \quad \tau_1 < \mu(t) \leq \tau_2 \\ \vdots & \vdots \quad \vdots \\ Label_N & if \quad \mu(t) \geq \tau_{N-1}. \end{cases} \tag{7}$$

where $\mu(t)$ is the residual defined in (5); $\tau$ is a threshold value selected by the user; $\text{Label}_\text{N}$ is the categorical data associated with the events and the actions to be taken by the Action Handling Unit.

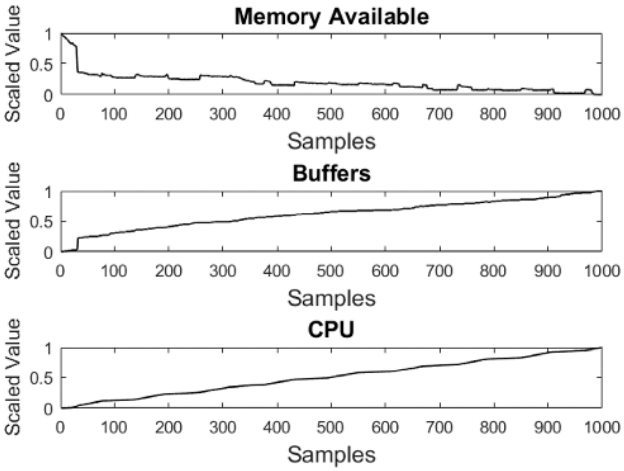



**FIGURE 5.** Healthy data from process.

#### 4) ACTION HANDLING UNIT

This unit implements the actions requested by the risk management unit. Table 4 shows the possible measures implemented by this unit. The issue may persist, for instance, a malicious entity could trigger Event1 code each time the connection is renewed. The Action Handling Unit employs a log file to keep a record of each error, including its timestamp. Before enforcing any protective policy, the log file is reviewed looking for the periodicity of a given error. If the period is less than 24 hours, the unit will handle it as Event3.

**TABLE 4.** Actions to be taken by the action handling unit.

| Label | Error description | Action |
|---|---|---|
| NONE | No threat detected | No action is required |
| Event1 | Threat Level 1. Small changes in one parameter | Reset connections |
| Event2 | Threat Level 2. Changes in two parameters | Reset the system |
| Event3 | Threat Level 3. Significant changes in one parameter or changes in three parameters | Change system configuration, request for authentication |

## V. EXPERIMENTS AND RESULTS

### A. EXPERIMENTAL SETUP

As we are targeting availability, three variables (MA, BU, and CU) were used to test the performance of the ANN-C. Under DoS or flooding attacks (which target systems availability), the most affected components are memory, processing capacity, and internal communication [41], therefore, memory availability, buffer utilization, and CPU utilization are critical variables when seeking for availability. The k-index is designated to identify each variable type as shown in Table 5. The k-index is used as a variable selector when algorithm 1 is applied (see Table 3).

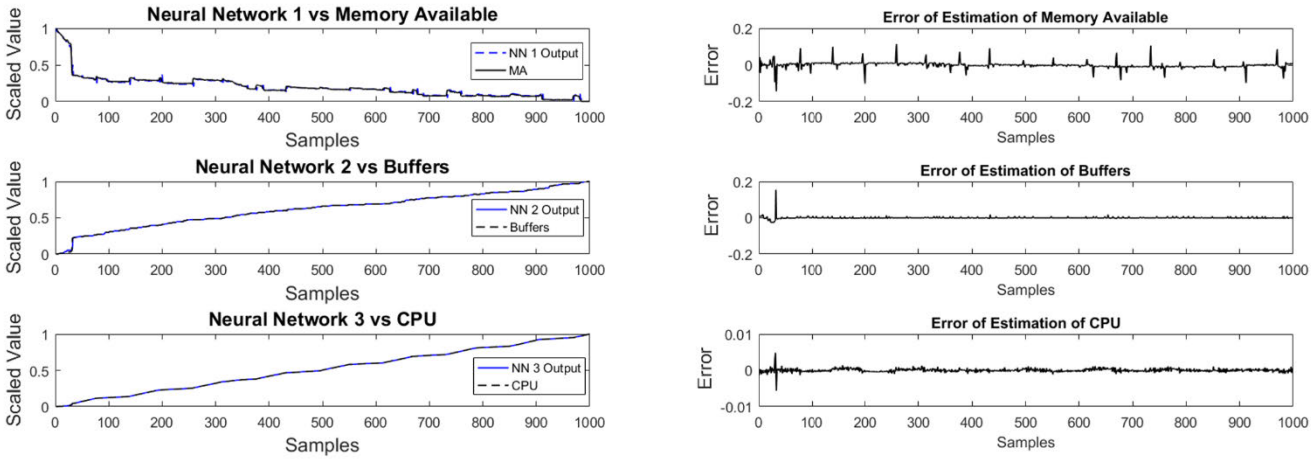



**FIGURE 6.** In the first column, NN trained for MA, BU and CPU are depicted. In the second column the respective training error are shown.

**TABLE 5.** Variable selection (k-index).

| k-index | Variable |
|---|---|
| 1 | Memory Available (MA) |
| 2 | Buffers (BU) |
| 3 | CPU Utilization (CU) |

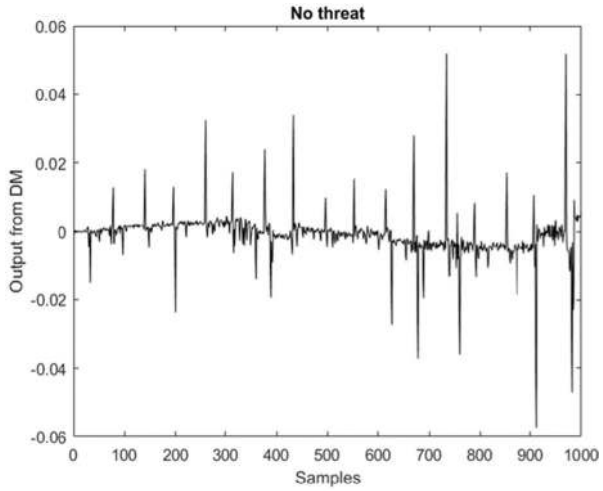



**FIGURE 7.** Output from discriminant memory due to process noise when the system is operating under normal condition.

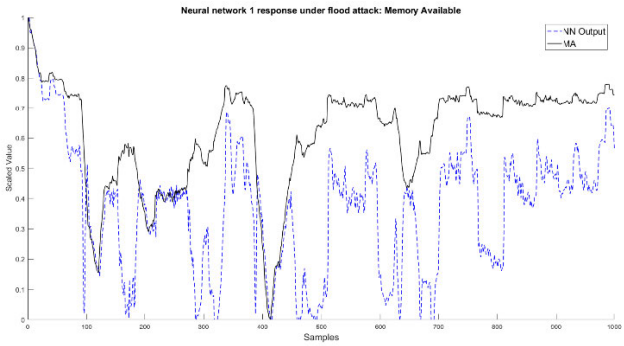



**FIGURE 8.** NN response under flood attack: memory available.

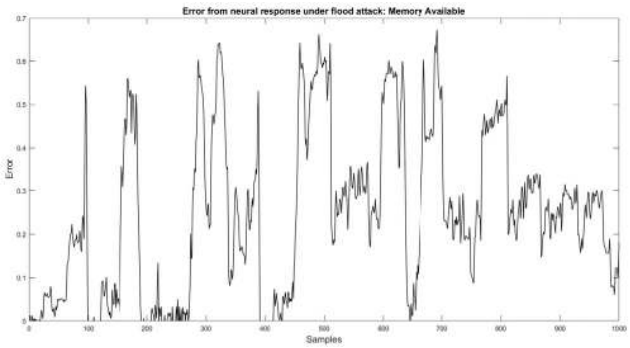



**FIGURE 9.** Error from NN response under flood attack: memory available.

The experimental setup consists of two phases: 1) offline phase, to train the system, and 2) online phase to test the NN-ABA-IDS. In the offline phase, three neural networks were trained, each one for a variable k. The NN was trained on 1000 samples collected while the system was operating normally. Parameter for NN are as follows: 10 neurons in the hidden layer; 2 delay units; one linear output neuron; the activation function g(t) is a sigmoid symmetric function designed as:

$$g(t) = \frac{2}{1 + e^{-2t}} - 1, \tag{8}$$

Raw data is scaled to the [0 1] space by means of the min-max method as follows

$$\overline{x} = \frac{\text{x-min}(x)}{max(x) - min(x)} \tag{9}$$

where $\overline{x}$ is the scaled value of x. Pre-processed data is shown in Fig. 5, where healthy data is generated under normal operative condition of the sensor, with no attacks nor anomaly behavior affecting the system and it constitutes the baseline data for normal operation.

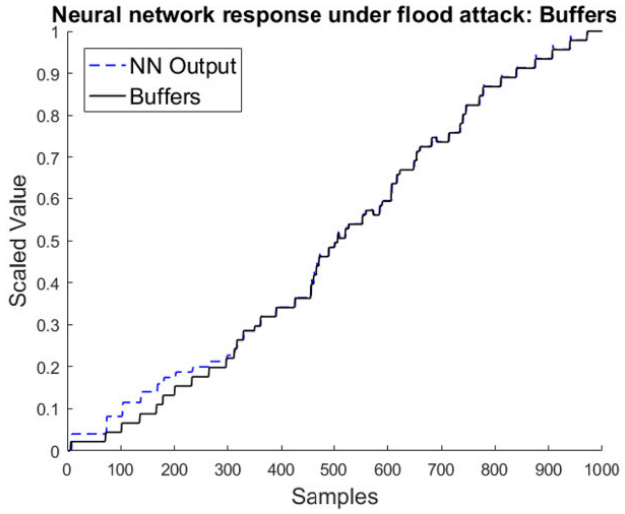



**FIGURE 10.** NN response under flood attack: buffers.

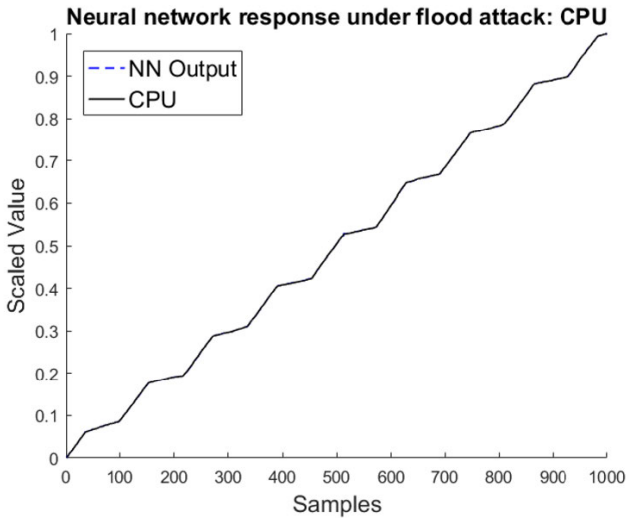



**FIGURE 12.** NN response under flood attack: CPU.

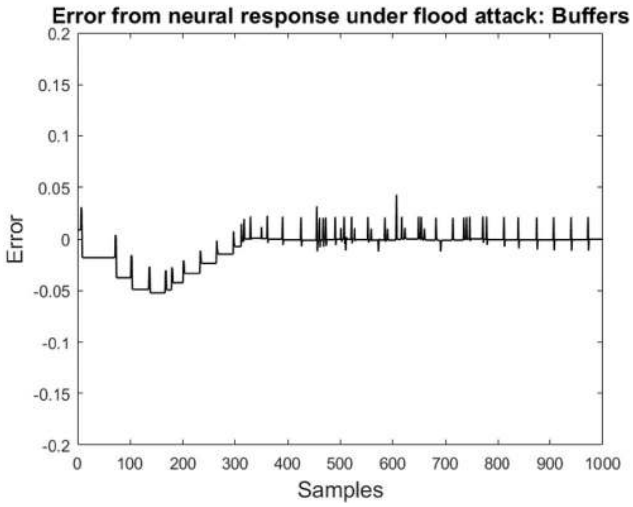



**FIGURE 11.** Error from NN response under flood attack: buffers.

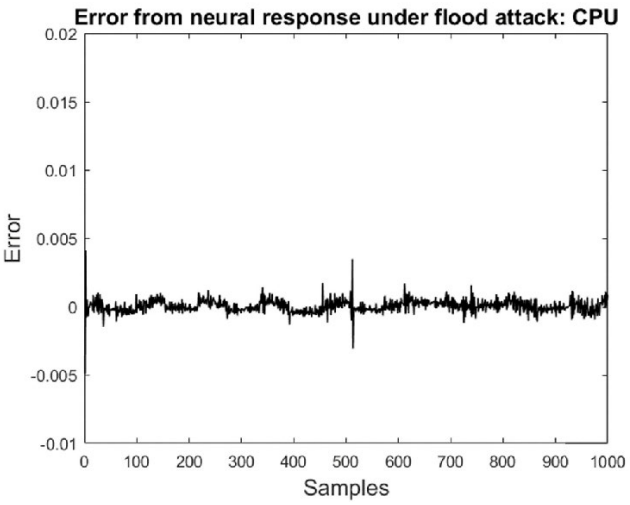



**FIGURE 13.** Error from NN response under flood attack: CPU.

### *B. OFFLINE TRAINING PHASE*

The k-index is used to train the NN for each variable. In the design of the NN three layers are taking in consideration a) input layer, which is feed by the previous values of data $\{y_{t-1}, y_{t-2}, \cdots, y_{t-d}\}$; b) output layer, which give the estimation value of the neural network $\hat{y}_t$; and c) hidden layer, which process data between input and output layers.

In Fig. 6, the ANN estimated output is displayed; they were trained using the Levenberg-Marquardt backpropagation algorithm. As can be seen, it exhibits an error close to zero, which means that the ANN-C is capable of tracking the behavior of the system.

### *C. ONLINE PHASE*

A threat level (TL) was proposed to detect the severity from attacks. The discriminant memory and risk management unit implemented by Equations (4) to (7) worked together to generate an attack severity profile. The threshold $\tau$, for the rule mechanism, was selected as follows: 1 for $\tau < 0.3$, 2 for $0.3 \leq \tau \leq 0.6$, and 3 for $\tau > 0.6$. Several tests were applied in order to evaluate the ANN Based IDS for IoT Fog Nodes performance.

#### 1) NO ATTACK

The first test was to investigate the performance of the approach under normal operational conditions (no attack condition was applied). As depicted in Fig. 7, only noise of the overall process was present at the output of the ABA-IDS.

#### 2) FLOODING ATTACK

A flooding attack constituting a large stream of packets aiming to fill the target memory [42], was performed on the system reducing legitimate packet delivery from 90%

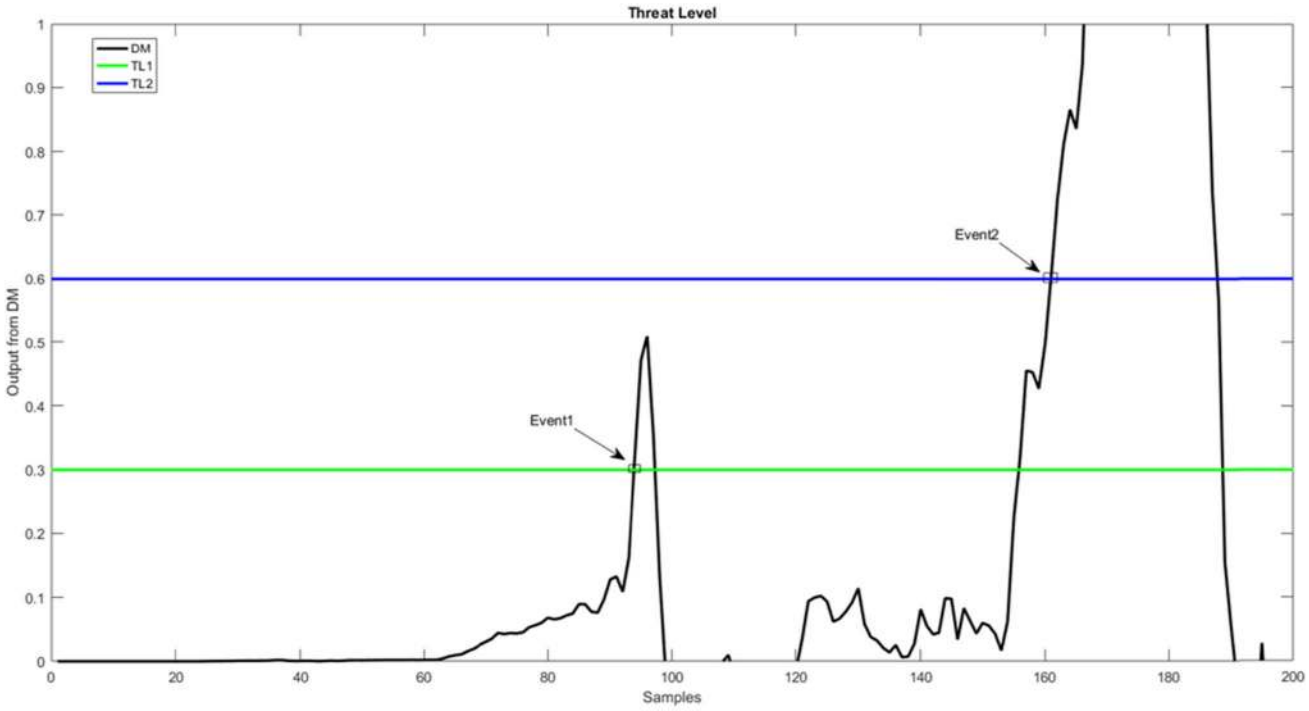



**FIGURE 14.** Two events detected under flood attack.

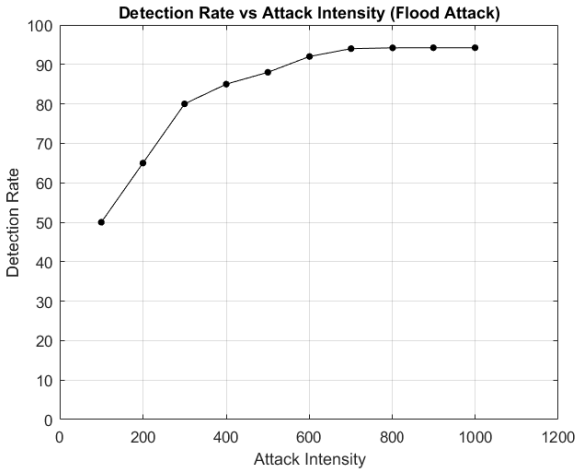



**FIGURE 15.** Detection rate vs attack intensity for Flooding attack.

to less than 40 Fig. 8 shows the response of memory available variable and Fig. 9 shows the error of the neural network.

The response of Buffers variable is shown in Fig. 10 and 11. It is remarkable that the respective NN trained for Buffer can deal with the flooding attack (Fig. 10) with high accuracy, which is verified by the performance of the error shown in Fig. 11. The CPU utilization response of the NN is depicted in Fig. 12. In this case, the NN fits with an error very close to zero (Fig. 13). From the results depicted in Fig. 8 to Fig. 13, one can figure out that the error of the neural network contributes in different ways to the performance of each variable. It is possible to establish that the performance of the NN trained for AM variable is worst that the performance of BU and CPU. This differences in the error (residuals) are processed by the Discriminant Memory in Equation (5), whose output was evaluated as indicate Table 5.

In Fig. 14, the Threat Level was obtained when the flood attack was simulated in the process. According to Fig. 14, two events are detected when the threshold is reached. The TL with $\tau = 0.3$ is tagged as *Event1* by the rule mechanism triggered by Equation (7), which indicates that a minor threat is detected. As can be seen in Fig. 14, the dynamics of the discriminant memory can track the behavior of the threat. If TL overpasses the value $\tau = 0.6$, a new event is triggered and labeled as *Event2*.

### 3) ATTACK INTENSITY VS DETECTION RATE AND OVERHEAD

The ABA-IDS approach was tested under different intensities for attacks. Flooding attack is established as a control subject, sending 100 to 1000 packets per second. It is assumed that for intensities under 100 packets per second, the node is not compromised, as it can handle such traffic. Fig. 15 shows how different intensities affect the detection rate. As can be seen, the detection rate is satisfactory (more than 90%) for intensities of 600+ packets per second. The node was operated for 24 hours under no attack (in an isolated environment) to verify false positives. After 86400 trials, the system indicated 2803 alerts (3.24% false positives).

**TABLE 6.** Evaluation metrics.

| Metric | Value |
|---|---|
| A | 97.51 % |
| P | 98.4 % |
| R | 98.9 % |

The system overhead is another important parameter to be considered when implementing intrusion detection. To verify this parameter, the experiments were executed without the IDS and then using it, inspecting three features: 1) time to execute commands (time overhead); 2) memory consumption; and 3) CPU usage. To inspect time overhead, 1 to 10 commands were sent in the same request. In the worst-case scenario for the time overhead, the proposed approach consumes 0.3 milliseconds, from 2.2 without IDS to 2.5 milliseconds running the IDS, which represents 13% in time overhead.

With 0.3 milliseconds overhead, the end-user will not be able to notice a delay in the issued commands, however, it will notice how the system is capable of operating even under network unstable circumstances. The overhead in memory represents 0.8%, and the CPU overhead is about 0.05%, which means that the approach can be considered as lightweight for the fog node. Notice that the given overhead is only possible due to the node specifications discussed in section 3. A.

Finally, metrics such as Accuracy, Precision, and Recall were used to evaluate the performance of the proposed approach

$$A = \frac{TP + TN}{TP + TN + FP + FN}$$
$$P = \frac{TP}{TP + FP}$$
$$R = \frac{TP}{TP + FN} \tag{10}$$

## VI. CONCLUSION

In this paper, an Intrusion Detection System was introduced. The system is based on the Anomaly Behavior Analysis Methodology (ABA-IDS) which is in turn powered by a cluster of Artificial Neural Networks. We demonstrated how to apply a methodology based on the ABA-IDS to secure and protect a fog node integrated into the IoT realm, ensuring availability. The proposed methodology includes the use of a profile based on features extracted from the node and fed to Artificial Neural Networks, configured to accurately characterize the normal operations of the node. The proposed approach showed to be effective in detecting both known and unknown attacks with high detection rates (more than 90%) and low false-positive alerts (less than 3.3%), also having insignificant overhead in terms of execution time, memory and CPU utilization. It is important to emphasize that the proposed methodology is meant to assure the availability of the fog node, providing resiliency to the overall process of IoT applications that make use of Fog computing technology.

## REFERENCES

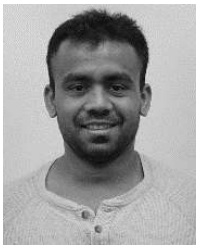
[1] M. Al-Bahri, A. Yankovsky, A. Borodin, and R. Kirichek, "Testbed for identify IoT-devices based on digital object architecture," in *Proc. Conf. Internet Things Smart Spaces*, 2018, pp. 129–137.
[2] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[3] D. Kushner, "The real story of stuxnet," *IEEE Spectr.*, vol. 50, no. 3, pp. 48–53, Mar. 2013.
[4] J. L. Pérez, A. Gutierrez-Torre, J. L. Berral, and D. Carrera, "A resilient and distributed near real-time traffic forecasting application for fog computing environments," *Future Gener. Comput. Syst.*, vol. 87, pp. 198–212, Oct. 2018.
[5] J. Pacheco and S. Hariri, "IoT security framework for smart cyber infrastructures," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 242–247.
[6] A. S. Sohal, R. Sandhu, S. K. Sood, and V. Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," *Comput. Secur.*, vol. 74, pp. 340–354, May 2018.
[7] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network," *IEEE Access*, vol. 6, pp. 73713–73723, 2018.
[8] X. An, X. Lu, L. Yang, X. Zhou, and F. Lin, "Node state monitoring scheme in fog radio access networks for intrusion detection," *IEEE Access*, vol. 7, pp. 21879–21888, 2019.
[9] F. Lin, Y. Zhou, X. An, I. You, and K.-K.-R. Choo, "Fair resource allocation in an intrusion-detection system for edge computing: Ensuring the security of Internet of Things devices," *IEEE Consum. Electron. Mag.*, vol. 7, no. 6, pp. 45–50, Nov. 2018.
[10] X. An, J. Su, X. Lü, and F. Lin, "Hypergraph clustering model-based association analysis of DDOS attacks in fog computing intrusion detection system," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, p. 249, Dec. 2018.
[11] J. Pacheco and S. Hariri, "Anomaly behavior analysis for IoT sensors," *Trans. Emerg. Telecommun. Technol.*, vol. 29, no. 4, p. e3188, Apr. 2018.
[12] J. M. Caswell, "A nonlinear autoregressive approach to statistical prediction of disturbance storm time geomagnetic fluctuations using solar data," *J. Signal Inf. Process.*, vol. 5, no. 2, pp. 42–53, 2014.
[13] F. Pereira, F. Bezerra, S. Junior, J. Santos, I. Chabu, G. Souza, F. Micerino, and S. Nabeta, "Nonlinear autoregressive neural network models for prediction of transformer oil-dissolved gas concentrations," *Energies*, vol. 11, no. 7, p. 1691, 2018.
[14] A. Hegyi, H. Flinck, I. Ketyko, P. Kuure, C. Nemes, and L. Pinter, "Application orchestration in mobile edge cloud: Placing of IoT applications to the edge," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 230–235.
[15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile cloud Comput. (MCC)*, 2012, p. 13.
[16] C. A. Garcia-Perez and P. Merino, "Enabling low latency services on LTE networks," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 248–255.
[17] J.-W. Xu, K. Ota, M.-X. Dong, A.-F. Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," *Frontiers Inf. Technol. Electron. Eng.*, vol. 19, no. 12, pp. 1546–1557, Dec. 2018.
[18] Q. Yaseen, F. AlBalas, Y. Jararweh, and M. Al-Ayyoub, "A fog computing based system for selective forwarding detection in mobile wireless sensor networks," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 256–262.
[19] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of Things security and forensics: Challenges and opportunities," *Future Gener. Comput. Syst.*, vol. 78, pp. 544–546, Jan. 2018.
[20] M. Ammar, G. Russello, and B. Crispo, "Internet of Things: A survey on the security of IoT frameworks," *J. Inf. Secur. Appl.*, vol. 38, pp. 8–27, Feb. 2018.
[21] S. Ali, T. Al Balushi, Z. Nadir, and O. K. Hussain, *Risk Management for CPS Security*. Cham, Switzerland: Springer, 2018, pp. 11–33.
[22] O. Can and O. K. Sahingoz, "A survey of intrusion detection systems in wireless sensor networks," in *Proc. 6th Int. Conf. Modeling Simulation, Appl. Optim. (ICMSAO)*, 2015, pp. 1–6.
[23] H. Alipour, Y. B. Al-Nashif, P. Satam, and S. Hariri, "Wireless anomaly detection based on IEEE 802.11 behavior analysis," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 10, pp. 2158–2170, Oct. 2015.

[24] P. Satam, H. Alipour, Y. Al-Nashif, and S. Hariri, "Anomaly behavior analysis of DNS protocol," *J. Internet Serv. Inf. Secur.*, vol. 5, no. 4, pp. 85–97, 2015.
[25] P. Satam, S. Satam, and S. Hariri, "Bluetooth intrusion detection system (BIDS)," in *Proc. IEEE/ACS 15th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2018, pp. 1–7.
[26] S. Fayssal, S. Hariri, and Y. Al-Nashif, "Anomaly-based behavior analysis of wireless network security," in *Proc. 4th Annu. Int. Conf. Mobile Ubiquitous Syst., Netw. Services (MobiQuitous)*, 2007, pp. 1–8.
[27] 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. Int. Symp. Netw., Comput. Commun. (ISNCC)*, May 2016, pp. 1–6.
[28] R. Schlegel, S. Obermeier, and J. Schneider, "Structured system threat modeling and mitigation analysis for industrial automation systems," in *Proc. IEEE 13th Int. Conf. Ind. Informat. (INDIN)*, Jul. 2015, pp. 197–203.
[29] H. Tran-Dang and D.-S. Kim, "An information framework for Internet of Things services in physical Internet," *IEEE Access*, vol. 6, pp. 43967–43977, 2018.
[30] J. Guth, U. Breitenbucher, M. Falkenthal, F. Leymann, and L. Reinfurt, "Comparison of IoT platform architectures: A field study based on a reference architecture," in *Proc. Cloudification Internet Things (CIoT)*, Nov. 2016, pp. 1–6.
[31] V. V. Gadde, H. Awano, and M. Ikeda, "An encryption-authentication unified A/D conversion scheme for IoT sensor nodes," in *Proc. IEEE Asian Solid-State Circuits Conf. (A-SSCC)*, Nov. 2018, pp. 123–126.
[32] B. Daddala, H. Wang, and A. Y. Javaid, "Design and implementation of a customized encryption algorithm for authentication and secure communication between devices," in *Proc. IEEE Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2017, pp. 258–262.
[33] J. P. Mitchell, "Performance, management, and monitoring of 68 node raspberry pi 3 education cluster: Big orange bramble (BOB)," Univ. Tennessee, Knoxville, TN, USA, Tech. Rep. ECE-599, Aug. 2016.
[34] Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann, and W. Rosenstiel, "Attack surface modeling and assessment for penetration testing of IoT system designs," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2018, pp. 177–181.
[35] J. Pacheco, D. Ibarra, A. Vijay, and S. Hariri, "IoT security framework for smart water system," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct. 2017, pp. 1285–1292.
[36] G. Orsini, D. Bade, and W. Lamersdorf, "CloudAware: A context-adaptive middleware for mobile edge and cloud computing applications," in *Proc. IEEE 1st Int. Workshops Found. Appl. Self Syst. (FASW)*, Sep. 2016, pp. 216–221.
[37] S. Chawla, M. Sachdeva, and S. Behal, "Discrimination of DDoS attacks and flash events using Pearson's product moment correlation method," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 10, pp. 382–389, 2016.
[38] L. Welling and L. Thomson, *PHP and MySQL Web Development*, 4th ed. New York, NY, USA: Pearson, 2016.
[39] M. R. Mohammadi, S. A. Sadrossadat, M. G. Mortazavi, and B. Nouri, "A brief review over neural network modeling techniques," in *Proc. IEEE Int. Conf. Power, Control, Signals Instrum. Eng. (ICPCSI)*, Sep. 2017, pp. 54–57.
[40] J. A. Ruz-Hernandez, E. N. Sanchez, and D. A. Suarez, "Neural networks-based scheme for fault diagnosis in fossil electric power plants," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Jul. 2005, pp. 1740–1745.
[41] J. Rodriguez, "Detecting and mitigating denial of service attacks," U.S. Patent 12/983 179, Jul. 5, 2012.
[42] S. B. I. Shah, M. Anbar, A. Al-Ani, and A. K. Al-Ani, *Hybridizing Entropy Based Mechanism With Adaptive Threshold Algorithm to Detect RA Flooding Attack in IPv6 Networks*. Singapore: Springer, 2019, pp. 315–323.

**JESUS PACHECO** received the bachelor's degree in electronics engineering with a focus on digital systems and the M.Sc. degree in computer science from the Tecnológico de Hermosillo and the Ph.D. degree from the Electrical and Computer Engineering Department, The University of Arizona. He is currently a Full Professor with the Industrial Engineering Department, Universidad de Sonora, and the Director of the NSF Center for Cloud and Autonomic Computing. His research interests include cyber security for the Internet of Things and cyber-physical systems.

**VICTOR H. BENITEZ** received the B.E. degree in electronics from the Universidad de Guadalajara, Mexico, in 1999, and the M.Sc. and Ph.D. degrees in electrical engineering from the Advanced Studies and Research Center, CINVESTAV-IPN, in 2002 and 2009, respectively. He has worked in the manufacturing industry related to the manufacture of printed circuits boards, and in the area of testing engineering. He is currently a Full Professor of mechatronics with the Industrial Engineering Department, Universidad de Sonora. His research interests include myoelectric control systems, neural networks applied to control electromechanical systems, and embedded systems applied to the Internet of Things.

**LUIS C. FÉLIX-HERRÁN** received the B.Eng. degree, in 2001, the M.S. degree in 2006, and the Eng.Sc.D. degree, in 2011. He is currently a Professor with the School of Engineering and Sciences, Tecnólogico de Monterrey. In addition to his teaching activities, he conducts a research on modeling and control of linear and non-linear systems as well as educational innovation in undergraduate engineering programs.



**PRATIK SATAM** received the bachelor's degree in electronics and telecommunications engineering from Mumbai University, in 2013, and the master's and Ph.D. degrees from the Department of Electrical and Computer Engineering, The University of Arizona, in 2015 and 2019, respectively. He is currently a Research Assistant Professor with The University of Arizona. His research interests include network security, the IoT security, cloud security, and application so machine learning towards cyber security.