IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 October 2022, accepted 26 November 2022, date of publication 2 December 2022, date of current version 7 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3226700

RESEARCH ARTICLE

Traffic Based Sequential Learning During Botnet Attacks to Identify Compromised IoT Devices

EROL GELENBE[1,2], (Life Fellow, IEEE), AND MERT NAKIP[1], (Student Member, IEEE)
[1]Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (PAN), 44100 Gliwice, Poland
[2]Laboratory I3S, Université Côte d'Azur, 06103 Nice, France

Corresponding authors: Erol Gelenbe (seg@iitis.pl) and Mert Nakıp (mnakip@iitis.pl)

This work was supported in part by the European Commission's H2020 Program through the H2020 Security By Design IoT Development and Certificate Framework with Front-end Access Control (IoTAC) Research and Innovation Action under Grant 952684.

ABSTRACT A novel online Compromised Device Identification System (CDIS) is presented to identify IoT devices and/or IP addresses that are compromised by a Botnet attack, within a set of sources and destinations that transmit packets. The method uses specific metrics that are selected for this purpose and which are easily extracted from network traffic, and trains itself online during normal operation with an Auto-Associative Dense Random Neural Network (AADRNN) using traffic metrics measured as traffic arrives. As it operates, the AADRNN is trained with auto-associative learning only using traffic that it estimates as being benign, without prior collection of different attack data. The experimental evaluation on publicly available Mirai Botnet attack data shows that CDIS achieves high performance with Balanced Accuracy of 97%, despite its low on-line training and execution time. Experimental comparisons show that the AADRNN with sequential (online) auto-associative learning, provides the best performance among six different state-of-the-art machine learning models. Thus CDIS can provide crucial effective information to prevent the spread of Botnet attacks in IoT networks having multiple devices and IP addresses.

INDEX TERMS Internet of Things (IoT), compromised device identification, random neural network, auto-associative deep random neural network, botnets, Mirai, attack detection and prevention.

I. INTRODUCTION

The number of Internet of Things (IoT) devices is increasing rapidly as the application of IoT expand, and [1] reported that 52% of all IoT devices will consist of low-cost and low-maintenance Massive IoT devices that perform a single task at a time and cannot run complex real-time algorithms to detect and prevent attacks. While systemic approaches to improving the security of cyberphysical systems have been suggested [2], [3], it is difficult (if not totally impossible) to burden simple IoT devices with complex security functionalities [4]. Thus IoT devices are often vulnerable to attackers [5], [6], [7], and common Denial of Service (DoS) accounts for 20% of all attacks against the IoT [8], in which an attacker or malicious device forwards superfluous requests to prevent its normal operation by usurping its limited resources with or without malware injection [9], [10].

Distributed DoS (DDoS) attacks can lead to thousands of compromised devices [11] through Botnet attacks where victim devices become compromised and turn into a "bot" via malware [12]. For instance, in 2016, a massive DDoS Botnet attack whose source code was later released under the name "Mirai", targeted Domain Name System (DNS) provider Dyn [13], rendering Netflix, Reddit, Spotify, and Twitter [14], [15] inaccessible, and gaining malicious access to the servers of leading cybersecurity companies from millions of different IP addresses [16]. The Mirai Botnet sends TCP SYN requests to the IP addresses of large numbers of IoT devices. When the victim device responds to a request, the attacker gains access to it using weak login credentials such as default usernames and passwords pre-installed during manufacture, and can install malware on the victim device, turning it into a compromised device (Bot). The Bot then generates traffic that

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

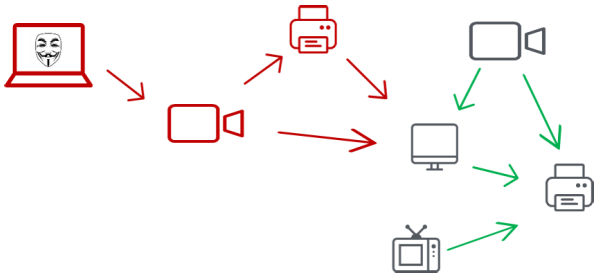


FIGURE 1. Example of a Botnet attack propagating over an IoT network, where the red laptop represents the initial attacker. Here, green arrows indicate benign traffic while red arrows indicate malicious traffic. When malware is injected, each device turns into a compromised device shown in red.

floods other servers and devices with meaningless requests. Thus the Botnet compromises new devices and propagates over the IoT network, as illustrated in Figure 1.

In addition to network-wide effects, Botnets significantly increase network congestion and power consumption, and processor and memory usage at the device level, hence posing challenges for resource-constrained devices [17]. Thus, given the nefarious impact of Botnets at both network and device level in Massive IoT networks, it is crucial to identify malicious packets and compromised IoT devices in real-time during an attack, so as to prevent the attack from spreading.

The remainder of the paper is organized as follows:

- Section II describes prior work and the novelty of this work as compared to the state-of-the-art.
- Section III describes the data used for validating and illustrating the results and introduces some related notation.
- Section IV presents the architectural design of CDIS, and the methodology that it uses, including the choice of traffic statistics.
- Section V compares the performance of CDIS among six different ML models, while also analyzing the effectiveness of the proposed network traffic metrics.
- Finally, in Section VI the main outcomes of this work are summarized, and some further directions for research are indicated.

II. RELATIONSHIP TO THE STATE-OF-THE-ART

We now review the relationship between this work and the state-of-the-art regarding the detection of Botnet traffic and of IoT devices that were compromised during a Botnet attack.

In work that addresses the detection of Botnet attacks, Antonakakis et al. have analyzed the characteristics of Botnet attacks [18], and in [13], Margolis et al. examine the capabilities and impact of Mirai, while in [19], Sinanovic et al. examine its source code. In addition, Ahmed et al. suggested that blockchains can be used to protect IoT devices against such attacks [20].

Recent research has used the following ML models to detect Botnet attacks: KNN, Support Vector Machine (SVM), Decision Trees (DT) and MLP in [21]; Classification and Regression Trees (CART) [22]; DT, Gradient Boosting and Random Forests [23]; and Logistic Regression [24]. Tuan et al. [25] conducted a comparative study of the performance of classification models and Neural Networks (NN). Neural networks were also used to detect Mirai Botnets in Software Defined Networks (SDN) by Letter et al. [26]. In 2020, Sriram et al. [27] used MLP with a deep architecture, while Soe et al. used NN and Naive Bayesian Models (NB) with a sequential architecture [28]. In addition McDermott et al. [29], developped a bidirectional LSTM-based text recognition model for packet-level detection. Another deep learning model, the Convolutional Neural Network (CNN), was used with feature transformation by Liu et al. [30] and combined with LSTM by Parra et al. [31]. Tzagkarakis et al. [32] detected Botnet attacks via a sparse representation framework with a large number of 115 inputs for which only normal traffic is used to select parameters, while recent work [33] developed a Mirai Botnet attack detector using the AADRNN with auto-associative learning.

Whereas the work reviewed in this paragraph aims at detecting Botnet attack traffic, the goal of the present paper is to identify compromised IoT devices or IP addresses that have received Botnet traffic or actually become "Bots" during a Botnet attack. The paper does not discuss the actions that a network may take, such as rerouting or blocking traffic [34], [35] after an attack is detected.

Other work has focused on detecting compromised IoT devices during Botnet attacks. Kumar and Lim. [36] have developed an optimization-based technique to detect Mirai-like bots by scanning the destination port numbers in packet headers; they analyze the subset of IoT packets to minimize the time it takes to detect compromised devices. Chatterjee et al. [37] develop an evidence theory based traffic flow analysis in IoT networks in order to detect malicious devices selecting the rarest set of traffic features, where the full set of features includes the transport layer protocol, number of reconnections and source/destination ports etc. In [38], Nguyen et al. focus on an anomaly detection technique for compromised devices using a combination of federated learning and language analysis for individual device types identified prior to anomaly detection. On the other hand, Abhishek et al. [39] detect compromised gateways rather than devices, monitoring the downlink channels in an IoT network. In the case of mobile devices, Taneja [40] detects compromised devices taking into account their location, so that a location change or unusual current location may hels to identify a device that is compromised.

In this paper, we develop a novel lightweight system called CDIS, so as to identify devices which have become Bots, which significantly differs from some past approaches aimed at detecting compromised IoT devices because:

- CDIS is only trained online, and only with normal traffic, so that the difficult collection of extensive attack data is no longer necessary, and biases that may be caused by the simulation of attacks are avoided, and
- CDIS only uses high-level packet information i.e. transmission times and packet lengths to calculate traffic

statistics. Therefore, it only needs access to the headers of traffic packets, and it is computationally light.

A. CONTRIBUTIONS OF THIS PAPER

This paper develops a novel *Compromised Device Identification System (CDIS)* to detect compromised devices or IP addresses in a network during an ongoing Botnet attack:

- CDIS learns sequentially from ongoing normal traffic, using only the packet streams that it recognizes as being benign, and uses an original choice of statistics regarding received and transmitted traffic, calculated only from packet lengths and transmission times.
- It uses a Machine Learning (ML) algorithm based on the Auto-Associative Dense Random Neural Network (AADRNN) [41] with online auto-associative learning that was initially designed for image recognition [42], and also used successfully to detect SYN attacks against IoT devices [43].

The high performance of the clasical Random Neural Network [44] with offline gradient-descent learning [45] to detect SYN attacks was shown earlier in [46], while the AADRNN's excellent performance with off-line training to detect MIRAI attacks was recently demonstrated in [33].

We evaluate the performance of CDIS on the publicly available Kitsune Mirai Botnet attack dataset [47], [48] as well as the MedBIoT and Bot-IoT datasets. We also compare the performance of AADRNN with the following state-of-the-art Machine Learning (ML) models: Linear Regression (LR), Least Absolute Shrinkage and Selector Operator (Lasso), K-Nearest Neighbors Regressor (KNN), Multi-Layer Perceptron (MLP) and Long-Short-Term Memory (LSTM). Our results show that CDIS under AADRNN significantly outperforms all these other ML models. It successfully identifies compromised IP addresses by achieving high Sensitivity (98.7%) and Specificity (94.9%). In addition, the computation time of CDIS is shown to be highly acceptable for practical applications.

CDIS has the following contributions and advantages:

1) It is trained with the normal ongoing traffic which is collected during real-time operation. CDIS learns online and sequentially, in auto-associative mode, by **only** using the traffic that it identifies as being benign. Thus It does not require the collection of either prior attack or normal non-compromised traffic.
2) It identifies compromised IoT devices based on traffic statistics calculated using only high-level packet information such as packet lengths and transmission times. Therefore, it only processes packet headers with very low computational requirements.
3) It achieves high identification performance with low computational requirements, which is vital to quickly prevent the spread of Botnet attacks in large networks.

III. DATASET, GROUND TRUTH AND METRICS

We use data from the Mirai Botnet from the publicly available Kitsune dataset [47], [48] which contains 764, 137 packets cover a consecutive time period of roughly 7137 seconds (nearly 2 hours) with 107 distinct IP addresses that either sent or receive traffic and for each of which we will perform infection detection. We will denote by S the set of all sources nodes or devices, while D will represent the set of all destination node or devices in the dataset.

The dataset contains the **ground truth** regarding whether a packet is an attack packet or a normal non-attack packet. Thus, for a packet p in the dataset, $a(p)$ denotes the binary attack label for packet p in the dataset, with $a(p) = 1$ denoting an attack, and $a(p) = 0$ denoting a non-attack normal packet. Each packet also contains the date t at which it is sent, and the complete representation of packet p is:

$$p \equiv pk(t, s, d), \text{ where } s \text{ and } d \text{ are the} \times \text{source and destination nodes}. \quad (1)$$

We separate the packets into time windows of fixed duration T, so that the collection of packets that are sent by device s to any other device d in the network in the $k-th$ time window is:

$$P_k^{s,d} \equiv \{pk(t, s, d) : (k-1)T \leq t < kT\}, \quad (2)$$

and the set of all packets arriving to d in the $k-$th slot is:

$$P_k^{S,d} \equiv \sum_{s \in S} P_k^{s,d}. \quad (3)$$

The ground truth for the infection level is defined as the ratio of the number of attack packets to the total number of packets that arrive at device or node d in time window k:

$$\phi_k^i = \frac{\sum_{\{p \in \cup_{l=1}^k P_l^{S,i}\}} a(p)}{|\cup_{l=1}^k P_l^{S,i}|}, \quad (4)$$

and the binary estimate of the ground truth is then obtained from ϕ_k^i as:

$$v_k^i = \mathbf{1}[\phi_k^i \geq \Theta], \quad (5)$$

where $0 < \Theta < 1$ is a threshold on the infection level to compute the binary ground truth estimate. Note that Θ should be selected considering the desired sensitivity of the network regarding malicious packet transmission.

v_k^i is the variable that we use **to test how well our attack detection schemes are working**. In this paper it is **not used at all for learning** since we develop an online learning technique which does not rely on prior offline learning.

A. DEFINING THE TRAFFIC METRICS

Since the aim of the NTSC module in Figure 2 is to identify instances of the traffic that may contain infection regarding a device i, it is important to judiciously select the metrics to be extracted.

The traffic metrics in this paper are chosen to address Mirai Botnet attacks, and may not be useful for identifying

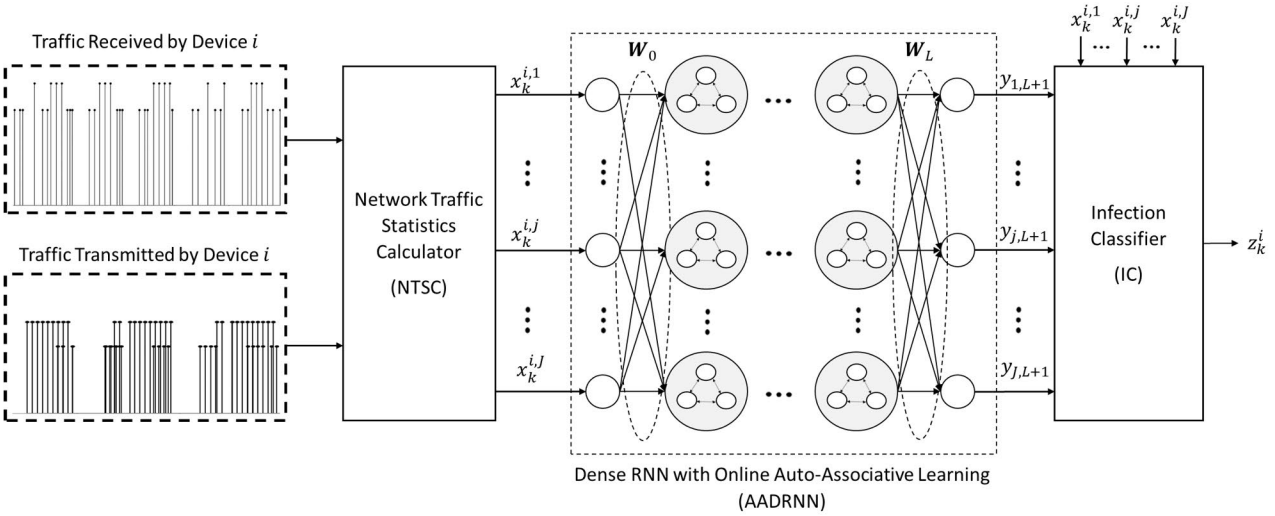


FIGURE 2. Compromised device identification system (CDIS) whose inputs are received and transmitted traffic flows of IoT device i, output is infection decision, and architecture consists of Network Traffic Statistics Calculator (NTSC), AADRNN and Infection Classifier (IC) modules.

compromised devices for other types of attacks, so that a CDIS for other attacks may require other metrics. Since Mirai attacks spread over the network by infecting IoT devices, when a device is compromised via malware, it will generate more packets with a larger amount of total traffic, so as to spread the attack over more nodes and overload the network. Previous work [33] had used and validated the following three network traffic statistics which are related to malicious packets in a Mirai attack:

- **Traffic Statistics 1**: The total size of the last P transmitted packets,
- **Traffic Statistics 2**: The average inter-transmission times of the packets over the last P packets,
- **Traffic Statistics 3**: Total number of packets that are transmitted in a time window with a duration of T.

Experimental results in [33] have shown that malicious packets are successfully detected using these three statistics during the Mirai attack.

However, since in this paper we wish to identify compromised devices (not just malicious packets), we develop a new set of metrics, or statistics, for both the traffic received and transmitted by each IoT device i, inspired from these three previous statistics. Indeed, in order to identify the sources of attacks, and the effect of the attacks, it is important to analyze the traffic received from each source individually, rather than the overall aggregated traffic received from all sources, since observing traffic from a compromised device can be an effective means of detecting the existence of an infection. Thus, we have selected some statistics to summarize the traffic sent by or received from each individual source.

Let $|pk(t, s, d)|$ denote the length of the packet in bytes. Packets sent by the set of nodes S have a maximum and minimum length L_S^M and L_S^m rin bytes, respectively, where the minimum may corresponds to a packet with just the header included and an empty data field. Each node s also has a maximum outgoing rate of θ_s in bytes/second.

The normalized statistics (or metrics) that are used for the traffic received or sent by node i within window k are as follows, where each normalized statistic takes a value between 0 and 1:

- **Received Traffic Statistics (RTS)1**: The normalized average size of packets received by device i from all the sources in time window k:

$$x_k^{i,1} = \frac{\sum_{p \in P_k^{S,i}} |p|}{\sum_{s \in S} L_s^M \times |P_k^{s,i}|} \tag{6}$$

- **RTS2**: The normalized maximum size of any packet received at node i from any of the sources in time window k:

$$x_k^{i,2} = \max_{p \in P_k^{S,i}} \frac{|p|}{L_S^M}. \tag{7}$$

  The use of L_S^M in RTS1 and RTS2 offers a normalization with respect to the maximum packet length. Note that large packets do not always suggest attacks: indeed, SYN attack packets may be quite short [43]. On the other hand, Denial of Service attacks that aim at creating congestion on links would have to be rather long.
- **RTS3**: The average number of packets received from all sources that have sent packets to i in time window k:

$$x_k^{i,3} = \frac{|P_k^{S,i}|}{\sum_{s \in S} 1[|P_k^{s,i}| > 0]}. \tag{8}$$

  Note that the denominator term in the above expression can be computed iteratively in a very efficient manner, so that $x_k^{i,3}$ is obtained directly from the terms in $x_{k-1}^{i,3}$.
- **RTS4**: The normalized maximum number of packets received from any single source in time window k:

$$x_k^{i,4} = \frac{\max_{s \in S} |\mathcal{P}_k^{s,i}|}{\max_{u:\ 1 \le u \le k} \left[\max_{s \in S} |\mathcal{P}_u^{s,i}| \right]}. \tag{9}$$

We now define the other traffic statistics that are important for detecting whether IoT device i is infected, and basically measure the total traffic in terms of both size and packet transmission rate from i to other nodes:

- **Transmitted Traffic Statistics (TTS) 1**: The normalized total amount of traffic transmitted by device i in time window k:

$$x_k^{i,5} = \frac{1}{\theta_i \times T} \sum_{d \in D} \sum_{p \in \mathcal{P}_k^{i,d}} |p| \ , \tag{10}$$

- **TTS2**: The normalized total number of packets that are transmitted by i in time window k:

$$x_k^{i,6} = \frac{L_i^m}{\theta_i \times T} \sum_{d \in D} |\mathcal{P}_k^{i,d}| \ , \tag{11}$$

where the use of L_i^m in TTS2 is due to the fact that the maximum number of packets that may be transmitted by node i in time T is $\frac{\theta_i \times T}{L_i^m}$.

IV. THE COMPROMISED DEVICE IDENTIFICATION SYSTEM (CDIS)

In this section, we present our Compromised Device Identification System CDIS based on the AADRNN whose architecture for IoT device i is shown in Figure 2. The AADRNN [41] is an extension of the Random Neural Network Model [44], which (in addition to excitatory and inhibitory spikes), incorporates soma-to-soma triggering [49], which generalizes the RNN and retains its "product form" solution. The power of this model was recently confirmed in extensive tests with conventional supervised off-line learning for a wide range of network cyberattacks [50], based on the proven aproximation capability of these models [51].

In our approach, a distinct instance of the CDIS is installed on each device i to determine if that device is compromised. The inputs of the CDIS are extracted from the received and transmitted traffic flows for the device (or IP Address) i, and the output is a binary infection decision z_k^i for device i.

CDIS is composed of the Network Traffic Statistics Calculator (NTSC), AADRNN, and Infection Classifier (IC) modules. The Traffic Statistics Calculator is already detailed in Section III-A so that we now focus on the AADRNN and IC Modules. Note that Table 1 summarizes the symbols in order of appearance in this paper.

In CDIS of Figure 2, the NTSC module extracts the values $x_k^{i,j}$ of the distinct metrics $1 \leq j \leq J$ from the packets received or sent in time window k by device i. We will define the vector of metrics for node i at window k:

$$x_k^i = [x_k^{i,1}, \ \ldots \ x_k^{i,J}], \tag{12}$$

and the ordered sequence of vector of metrics collected from window 1 up to and including window k:

$$X_k^i = (x_1^i, \ \ldots \ x_k^i). \tag{13}$$

We now detail the weights of the AADRNN used by CDIS in Section VI and use $W^{i,k}$ to denote the generic form of the whole matrix of weights for device i **after** input x_k^i has been used for learning

Thus $W^{i,k}$ is composed of the weight matrices connecting the clusters of neurons in layer l to layer $l+1$: $W_0^{i,k}$ denotes the weight matrix connecting the inputs (from the traffic metrics) to the first layer, while $W_L^{i,k}$ connects the last $L-th$ layer to the outputs $[y_{1,L+1}, \ \ldots \ , y_{J,L+1}]$.

Now let $\zeta(.)$ be the $J-$vector activation function for the AADRNN defined in Section VI. We can iteratively define the outputs of the AADRNN's $l-th$ layer, $1 \leq L$, for the overall weight matrix $W^{i,k}$ obtained after x_k^i, the $k-th$ $J-$vector input, has been processed. Let:

$$\begin{aligned} y_l(x_k^i, W^{i,k}) &= [y_{1,l}(x_k^i, W^{i,k}), \ldots, y_{j,l}(x_k^i, W^{i,k})], \\ &\quad \textit{where each } 0 \leq y_{j,l}(x_k^i, W^{i,k}) \leq 1, . \\ &\quad \textit{Then } y_1(x_k^i, W^{i,k}) = \zeta(x_k^i, W_0^{i,k}), \\ y_l(x_k^i, W^{i,k}) &= \zeta(y_{l-1}(x_k^i, W^{i,k}), W_{l-1}^{i,k}) \end{aligned} \tag{14}$$

and the whole AADRNN′s output is :

$$y_L(x_k^i, W^{i,k}) = \zeta(y_{L-1}(x_k^i, W^{i,k}), W_{L-1}^{i,k}), \tag{15}$$

and finally : $y_{L+1} = y_L . W_L^i$. (16)

Thus each $y_{j,L+1}, \ 1 \leq j \leq J$ is a function:

$$y_{j,L+1} \equiv y_{j,L+1}(x_k^i, W^{i,k}), \tag{17}$$

of the corresponding input x_k^i and of the overall weight matrix $W^{i,k}$ after each time window k.

A. THE INFECTION CLASSIFIER (IC)

Using the previously defined quantities, we now compute an output error that is needed to classify the inputs of the successive windows at step k as being "normal" or of attack nature. This is done by computing the maximum of all the differences between the elements of the input vector x_k^i and the elements of the output vector:

$$\Psi_k^i = \max_{j \in \{1, \ \ldots \ , J\}} |x^{i,j} - y_{j,L+1}(x_k^i, W^{i,k-1})|. \tag{18}$$

We then use a specific threshold value $0 < \gamma_i < 1$ for each of the devices or nodes, so as to provide a binary decision of the form:

$$z_k^i = 1 \ (attack) \ if \ \Psi_k^i \geq \gamma_i, \ z_k^i = 0 \ otherwise. \tag{19}$$

Since we are carrying out online learning, without prior off-line training using the ground truth, the outputs z_k^i not only provide decisions, but they also allow us to operate the on-line auto-associative algorithm given below.

B. ONLINE LEARNING

Since we use on-line learning, the $W_l^{i,k}$ matrices may be updated after each successive input x_k^i, so that the sequence of weight matrices are updated after each subsequent input, as follows. However, we can only train with "normal" (non-attack) values of x_k^i so that:

- If $z_k^i = 1$, i.e. x_k^i is estimated to contain an attack, then do not update the weights, i.e. $W_l^{i,k} \leftarrow W_l^{i,k-1}, \ 1 \leq l \leq L$,

TABLE 1. List of symbols in order of appearance.

Symbol	Definition: i indicates the device number
J	Total number of input statistics and number of inputs to the AADRNN
T	Fixed duration of each time window k
$x_k^{i,j} \in [0,1]$	The $j-th$ statistic extracted from network traffic in window k of device i
x_k^i	Input statistic vector $x_k^i = [x_k^{i,1}, \ldots, x_k^{i,J}]$ for window k
X_k^i	Sequence $X_k^i = (x_1^i, \ldots, x_k^i)$ of the first k input vectors
y_{L+1}^j	Output of AADRNN for the $j-th$ statistic with input x_k^i
y_{L+1}	Output vector of AADRNN with input x_k^i
z_k^i	Binary infection decision for the infection of device i at time window k
L	Number of layers of AADRNN
$W_l^{i,k}$	Weight matrix connecting the clusters in layer l to layer $l+1$ obtained after x_k^i has been processed for training
F_l	Optimization function specialized to layer l of AADRNN
$P_k^{s,d}$	Collection of the packets that have been sent from s to d in window k
S	Set of all possible source nodes
D	Set of all possible destination nodes
L_S^M	Maximum length of a packet sent by the set of nodes S
L_S^m	Minimum length of a packet sent by the set of nodes S
θ_s	Maximum outgoing packet rate at node or device s
Θ_s	Maximum incoming packet rate at node or device s
$q_{j,l}$	State of the cell in the $j-th$ cluster of layer l of the AADRNN corresponding to device i
$r_{j,l}$	Total firing rate of a cell in the $j-th$ cluster of layer l of the AADRNN corresponding to device i
n	Number of cells in each cluster of AADRNN
$\lambda_{j,l}$	Total inhibitory input spike rate to a cell in the $j-th$ cluster of layer l of AADRNN
Λ_{jl}	Total excitatory input spike rate to a cell in the $j-th$ cluster of layer l of AADRNN
$w^-_{j',l,j}$	Inhibitory weight from a cell in cluster j' in layer l to a cell in cluster j in layer $l+1$
$\zeta(\cdot)$	Activation function of each cell in AADRNN
ψ_k^i	Maximum of the absolute differences between actual and te expected values calculated by AADRNN
γ_i	Threshold for the binary decision of CDIS for device i
ϕ_k^s	Ground truth for the infection level of device s at time window k
Θ	Threshold for the binary ground truth
v_k^i	Binary ground truth for the infection of device i at time window k

where $W_l^{i,k-1}$ is the value, prior to the processing of input x_k^i, of the weight matrix that connects the $l-th$ layer to the inhibitory inputs at layer $l+1$.

- Otherwise, if $z_k^i = 0$ (non-attack) then

$$W_l^{i,k} \leftarrow F_l\big(W_l^{i,k-1}, x_k^i, y_l(x_k, W^{i,k-1})\big), \qquad (20)$$

using the optimization function detailed in Section IV-C. which is specialized to layer l as F_l.

C. OPTIMIZATION ALGORITHM

The "online auto-associative learning" for the AADRNN uses the fast training algorithm from previous work [33], [43]. It adapts to the naturally time-varying characteristics of network traffic, and the CDIS will update its parameters automatically as a function of the traffic it encounters as it operates. Here $W_l^{i,k-1}$ is $l-th$ layer output connection matrix, just before the CDIS processes input x_k^i with the semi-supervised algorithm of [42]. We then use the following approach only if $z_k^i = 0$:

Computation: For each layer $l \in \{0, \ldots, L\}$, compute $W_l^{i,k}$ using the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [52]:

$$W_l^{i,k} = \arg \min_{\{W:\, W \geq 0\}} \times \Big[\, ||adj(\zeta(y_{l-1}(x_k^i, W^{i,k-1}), W_R)W - y_l^i(x_k^i, W^{i,k-1})||_{L_2}^2 + ||W||_{L_1} \, \Big], \qquad (21)$$

where the $J \times J$ weight matrix W_R is randomly generated with elements in the range [0, 1]. On the other hand, $adj(B)$ is the linear mapping of the elements of matrix B into the range [0, 1] then applies the z-score (standard score), and adds a positive constant to remove negativity.

FISTA: After FISTA [52] is performed for 700 iterations, we finally normalize the resulting weight matrix $W_l^{i,k}$:

$$W_l^{i,k} \leftarrow 0.1 \frac{W_l^{i,k}}{\max\big[y_l(x_k^i, W_l^{i,k})\big]}. \qquad (22)$$

D. PARAMETER SETTINGS FOR CDIS

We first set $T = 10$ seconds to have a significant number of time windows (approximately 712) each of which contains significant number of packets. Then, for normalization,

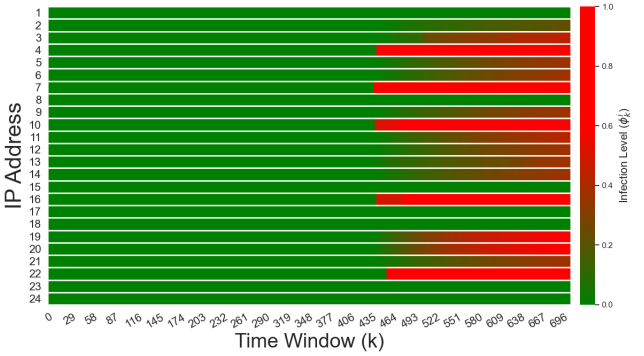


FIGURE 3. Ground truth value of the level of infection (ϕ_k^i) for individual IP addresses over 712 windows.

we set $\tau = 0.1$ seconds and $M = 85$ bytes based on our observations on the dataset.

In order to be able to present clear and detailed analysis, in the first part of this section, we consider only the IP addresses in "192.168" network within the experimental setup of the used Mirai Botnet dataset [47]. The ordered list of IP addresses in the network are: [192.168.1.252, 192.168.2.1, 192.168.2.101, 192.168.2.103, 192.168.2.104, 192.168.2.105, 192.168.2.107, 192.168.2.108, 192.168.2.109, 192.168.2.110, 192.168.2.111, 192.168.2.112, 192.168.2.113, 192.168.2.115, 192.168.2.117, 192.168.2.118, 192.168.2.119, 192.168.2.120, 192.168.2.121, 192.168.2.122, 192.168.2.126, 192.168.2.196, 192.168.2.255, 192.168.4.1]. Note that, during our experiments, we do not consider any information about the characteristics of the devices to which the IP addresses belong; that is, we treat all IP addresses as equivalent (or as individual IoT devices).

Figure 3 displays the ground truth values of the level of infection (ϕ_k^i) for these IP addresses (shown with local indexes) over 712 windows. One may see that the infection level gets significantly high for the 4th, 7th, 10th, 16th, 19th, 20th and 22nd IP addresses while the infection level remains around 0 for only the 1st, 8th, 15th, 17th, 18th, 23rd and 24th IP Addresses.

Based on the data in Figure 3, we can consider that an IP address is compromised if infection level ϕ_k^i is at least 0.5 (i.e. at least 50% of transmitted packets are malicious).

Let us note that the 50% threshold may be too high in the case of particularly dangerous or stealthy attacks, where the presence of 10% or 20% of attacking packets may result in a significant compromise of the system.

However, we set $\Theta = 0.5$ to have a represenative case, and calculate the binary ground truth of compromised devices via using (5). The resulting ground truth data contains 1494 positive (compromised) samples and 15594 negative samples in total over all of these 24 IP addresses over all of the 712 windows. In addition, using the binary ground truth for $\Theta = 0.5$, we observe that the 4th, 7th, 10th, 16th, 19th, 20th, and 22nd IP addresses become compromised over time.

E. HYPERPARAMETER SETTINGS FOR ML MODELS

In order to compare the performance of CDIS under AADRNN with other ML models, we replace the AADRNN in our architecture in Figure 2 with each of the LR, Lasso, KNN, MLP, and LSTM models. At each time window k, all these models are sequentially trained with the same input $\mathbf{X}_k^i$ and output $\mathbf{Y}_k^i$ matrices as for AADRNN in Section IV-C.

Thus, in the remainder of this subsection presents the parameter selections for each of these models as well as AADRNN.

1) AADRNN

The number of hidden layers of the AADRNN is $L = 3$; then, $n = LJ$ for total number of statistics $J = 6$, and $p = 1/n$. We also set $r = 1$ and $\Lambda_{j,l} = \lambda_{j,l} = 0.005$.

2) LR AND LASSO

We use two different linear ML models, LR and Lasso, in place of the AADRNN, and selected the most simple LR technique to create a baseline performance. Also, Lasso is used to observe the effects of feature selection on the cumulative performance since it is a linear model which shrinks irrelevant statistics values to zero, and we search for the best value of the L1 term multiplier between 0.1 and 1 in increments of 0.1, setting the value of this multiplier to 0.2. Moreover, we implement both LR and Lasso using the scikit-learn library [53] on Python.

3) K-NEAREST NEIGHBOURS REGRESSOR (KNN)

Early research [21] showed that the KNN achieves highly competitive results for detection of Botnet attacks, thus the KNN is one of the methods that we have implemented in scikit-learn and compared against the AADRNN. In each window k tthe number of neighbors in KNN is set to $\min(k-1, J)$, since the number of samples used for training equals $k-1$ and KNN requires at least as many neighbors as the number of samples.

4) MULTI-LAYER PERCEPTRON (MLP)

A feed-forward MLP with two hidden layers with $J = 6$ neurons, followed by an output layer was used, with a sigmoidal activation function and $J = 6$ neurons in the output layer. Both training and execution was performed using Keras on Python, where the MLP is trained via the Adam optimizer for 50 epochs at each time window.

5) LONG-SHORT TERM MEMORY (LSTM)

Lastly, we use LSTM with a single LSTM layer, $J = 6$ units, and three fully connected layers including the output layer which is comprised of $J = 6$ neurons, and sigmoidal activation througout. Training and execution of is also performed using Keras on Python, and trained via Adam optimizer for 100 epochs at each time window k.

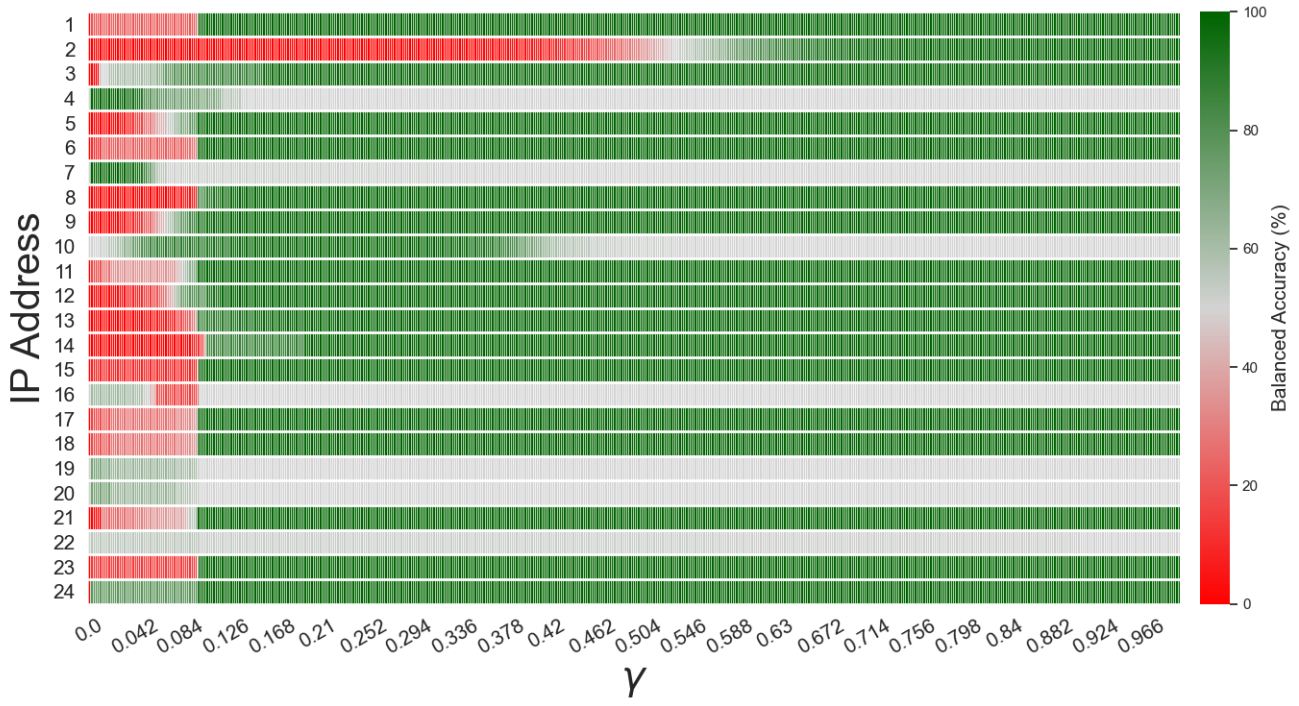


FIGURE 4. Balanced accuracy during the search for the best value of γ_i in CDIS with online training for each IP address $i \in \{1, \ldots, 24\}$.

F. PERFORMANCE EVALUATION METRICS

The dataset being used is unbalanced due to the nature of the problem. Therefore our main metric of "Balanced Accuracy" [54], has been revised to handle IP addresses that are never compromised as well as those that may be compromised. To this end, for each IP address i, we sum the number of True Positive, True Negative, False Positive, and False Negative cases in each window for each i to obtain:

$$TP_i = \sum_k 1[v_k^i = z_k^i = 1], \; TN_i = \sum_k 1[v_k^i = z_k^i = 0],$$
$$FP_i = \sum_k 1[v_k^i = 0 \;\&\; z_k^i = 1],$$
$$FN_i = \sum_k 1[v_k^i = 1 \;\&\; z_k^i = 0].$$

Then, for each IP address i, the *Revised Balanced Accuracy* BA_i is:

$$BA_i = \frac{1}{2}\left(\frac{TP_i}{TP_i + FN_i} + \frac{TN_i}{TN_i + FP_i}\right), \quad if \; \sum_k v_k^i > 0,$$
$$= \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}, \quad otherwise. \qquad (23)$$

Thus BA_i is "Balanced Accuracy" if the IP address i is compromised in any time window k, while it is simply "Accuracy" if the IP address i is never compromised.

In addition to the Balanced Accuracy, we also use other well-known metrics: Sensitivity (True Positive Rate), Specificity (True Negative Rate), geometric mean of Sensitivity and Specificity (G-Mean), and Matthews Correlation Coefficient (MCC), which are displayed as percentages. Note that Sensitivity, G-Mean, and MCC metrics can only be presented for IP addresses which are compromised in at least one time window. Since some IP addresses are never compromised, as seen in Figure 3, these three metrics cannot be presented for uncompromised addresses.

V. PERFORMANCE EVALUATION RESULTS

For each IP address i in the considered network, in order to achieve the highest performance of the CDIS, we first analyze and select the best value of γ_i. To this end, the Balanced Accuracy performance of CDIS of IP address i is measured for all γ_i values increasing in 0.002 intervals from 0 to 1. The measured performance displayed in Figure 4, where the performance range is shown with colors which range from red to green, also reveals that the Balanced Accuracy performance of CDIS (using AADRNN) under the best value of γ_i is very close to 100% (shown with dark green) for all IP addresses except for the 16th, 19th, 20th, and 22nd IP addresses, whose performances are around 62%, 71%, 72% and 56% respectively under the best value of γ_i.

Furthermore, Figure 4 shows that the Balanced Accuracy performance of CDIS is acceptably high for various γ_i values around the best value; hence, one may say that the performance of CDIS is highly robust with respect to the choice of γ_i. On the other hand, the best value of γ_i is considerably different for each IP address i, and is best for successive IP addresses as follows: 0.102, 0.888, 0.998, 0.002, 0.114, 0.102, 0.002, 0.186, 0.156, 0.212, 0.102, 0.132, 0.282, 0.208, 0.102, 0.002, 0.102, 0.102, 0.002, 0.002, 0.122, 0.098, 0.102, and 0.102. Note that if several values of γ_i achieve the best performance, the smallest value is selected.

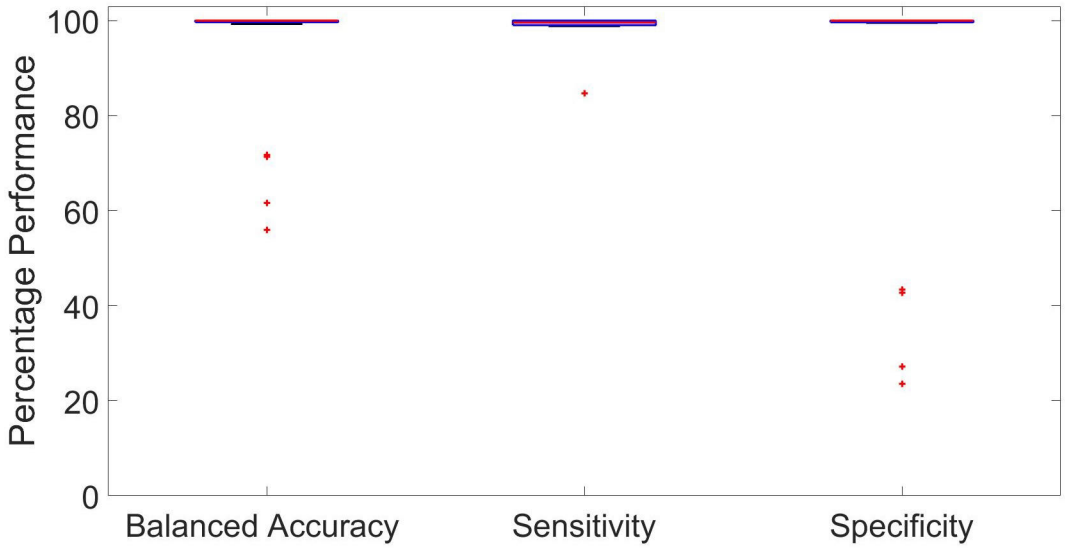


FIGURE 5. Box plot of the balanced accuracy, sensitivity, specificity, G-mean and MCC performances over considered 24 IP addresses.

TABLE 2. Average percentage performance of CDIS under different ML models over IP addresses.

ML Models	Balanced Accuracy	Sensitivity	Specificity
AADRNN	94.1	97.6	89
LR	85.6	44	87.5
Lasso	94.8	76	96.6
KNN	86.9	46.5	89.4
MLP	86	70	80.8
LSTM	85.8	73.9	79.2

Using the best values of γ_i's, we evaluate the performance of CDIS under AADRNN with respect to Balanced Accuracy, Sensitivity, and Specificity. Figure 5 displays the box plot of this performance evaluation over the considered IP addresses. Recall that Sensitivity is only presented for IP addresses that are compromised in at least one window.

The Balanced Accuracy, Sensitivity and Specificity show that the median performance of CDIS is almost 100%. However there are four IP addresses for which the measured performances are the outliers. The Balanced Accuracy for the outliers are 72%, 71%, 62%, and 56%.While searching for the best value of γ_i, we observed that these are the 16th, 19th, 20th, and 22nd IP addresses in Figure 4.

These results also reveal that CDIS is able to successfully detect infection for all IP addresses (minimum Sensitivity is 85%) but suffers from low Specificity (i.e. high false alarm rate) for outlier IP addresses. Indeed, we observe that the reason for the low specificity of the IP addresses with the outlier CDIS performance is that their traffic statistics do not indicate infection, and two of these IP addresses (19th and 20th) do not receive traffic but only transmit, so that the indicators RTS 1-4 are zero for all time windows.

A. PERFORMANCE OF CDIS UNDER DIFFERENT ML MODELS

The performance of CDIS under AADRNN is compared with that under each of LR, Lasso, KNN, MLP, and LSTM, where the best value of decision threshold is selected via exhaustive search for each ML model. The comparison of the performances with respect to Balanced Accuracy, Sensitivity, and Specificity are presented in Table 2. The numerical results in this table are presented as the average of each measurement over the IP addresses considered. For example, the Balanced Accuracy is first calculated for each IP address; then, the average of Balanced Accuracy is computed over all IP addresses.

The results in Table 2 show that CDIS is able to achieve highly acceptable performances under various ML models although some models lack the balance between Sensitivity and Specificity. On the other hand, the best Balanced Accuracy performance of CDIS is observed under AADRNN, which achieves the most balanced performance between Sensitivity and Specificity. It also appears that linear models (LR, Lasso and KNN) achieve high Specificity but LR and KNN have significantly low Sensitivity. In addition, the Sensitivity of linear models and the Specificity of MLP and LSTM are

TABLE 3. Training and execution times of CDIS under different ML models (in milliseconds).

ML Models	Training Time		Execution Time	
	Mean	Standard Deviation	Mean	Standard Deviation
AA-Dense RNN	52.6	15	0.3	0.4
LR	0.3	0.5	0.1	0.2
Lasso	75.1	109.9	0.1	0.3
KNN	0.5	0.5	0.4	0.5
MLP	297.9	195.3	34.2	76.7
LSTM	2219.2	1613.3	34.2	189.5

significantly low. That is, majority of linear models cannot properly detect compromised IP addresses, while MLP and LSTM cause a high rate of false positive alarms.

Furthermore, Table 3 displays the average and standard deviation, in milliseconds, of each of the training and execution times over all IP addresses and all time windows for each ML model. Training and execution times are measured in Python using the CPU of a PC with 32 GB of ram and AMD Ryzen 7 3.70 GHz processor. Also, recall (from Section IV-E) that each of the neural network models considered (i.e. AADRNN, MLP, and LSTM) has three hidden layers with six neurons each. In addition to its three hidden layers, LSTM neural network has also an LSTM layer with six units.

The results in Table 3 show that:

1) The mean training time of AADRNN is lower than Lasso, MLP and LSTM, with very low standard deviation of 15 ms. However, all of the AADRNN, Lasso, MLP and LSTM models require significantly more training time than LR and KNN.
2) Considering both the mean and standard deviation of the execution time, AADRNN, LR, Lasso and KNN are competitive with each other, but are much faster than MLP and LSTM.

B. PERFORMANCE OF CDIS FOR 107 DISTINCT IP ADDRESSES

We now evaluate the performance of CDIS under AADRNN for the extended IoT network scenario where all 107 unique IP addresses provided in the [47] dataset are addressed instead of only considering the IPs in "192.168" network. To this end, Figure 6(top) displays the average performance of CDIS over IP addresses with respect to each of Balanced Accuracy, Sensitivity and Specificity, and Figure 6 (bottom) displays the box plot of the performance of CDIS over IP addresses with respect to the same metrics.

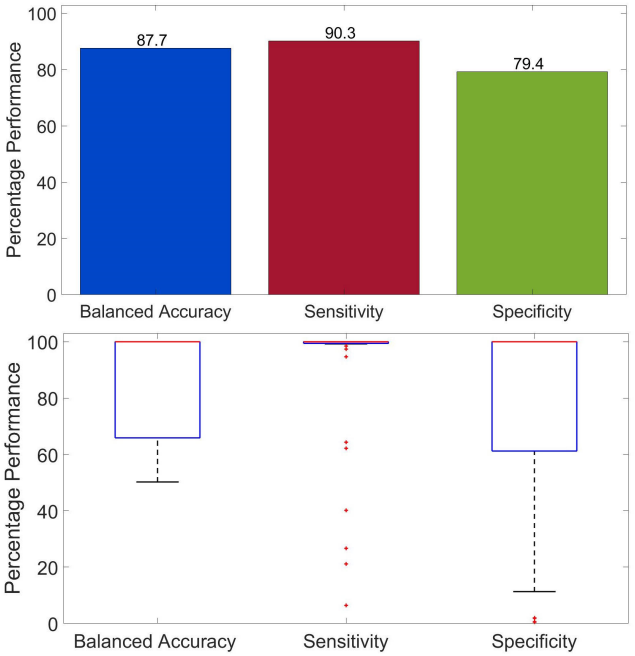


FIGURE 6. Bar graph (top) of the average performance and box plot (bottom) of the performance of CDIS over all (107) IP addresses in the Kitsune Mirai dataset.

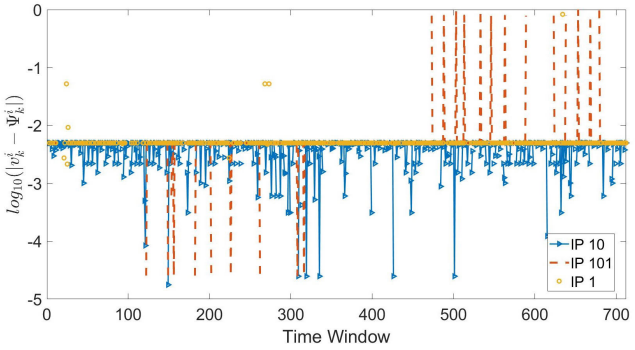


FIGURE 7. In this figure, the logarithmic prediction error $\log_{10}(|v_k^i - \Psi_k^i|)$, $1 \le k \le 712$ of CDIS with AADRN, is plotted versus the time slot k for three IP addresses: the 1st, 10th, and 101st. Only three IP addresses were chosen to clearly visualize the prediction errors that may be affected by online training.

In the top of Figure 6 we see that CDIS under AADRNN provides an average 88% Balanced Accuracy for this complex network structure with 107 unique IP addresses, while both average Sensitivity is 90% and Specificity is 79%.

More detailed results in Figure 6(bottom) reveal that the Balanced Accuracy performance of CDIS is above 92% for 2/3 of IP addresses and above 50% for all IP addresses. That is, the Balanced Accuracy performance of CDIS is between 50% and 92% for only 36% of IPs. It is also seen that both median Sensitivity and median Specificity are 100%; however, the number of node with lower Specificity is high compared to Sensitivity. On these results, we also observed that Sensitivity is above 90% for 85% of the IP addresses that are compromised at least in one time window, while Specificity is above 90% for 67% of all IP addresses. On the other hand, for only 4 IPs, the Sensitivity is below 40%.

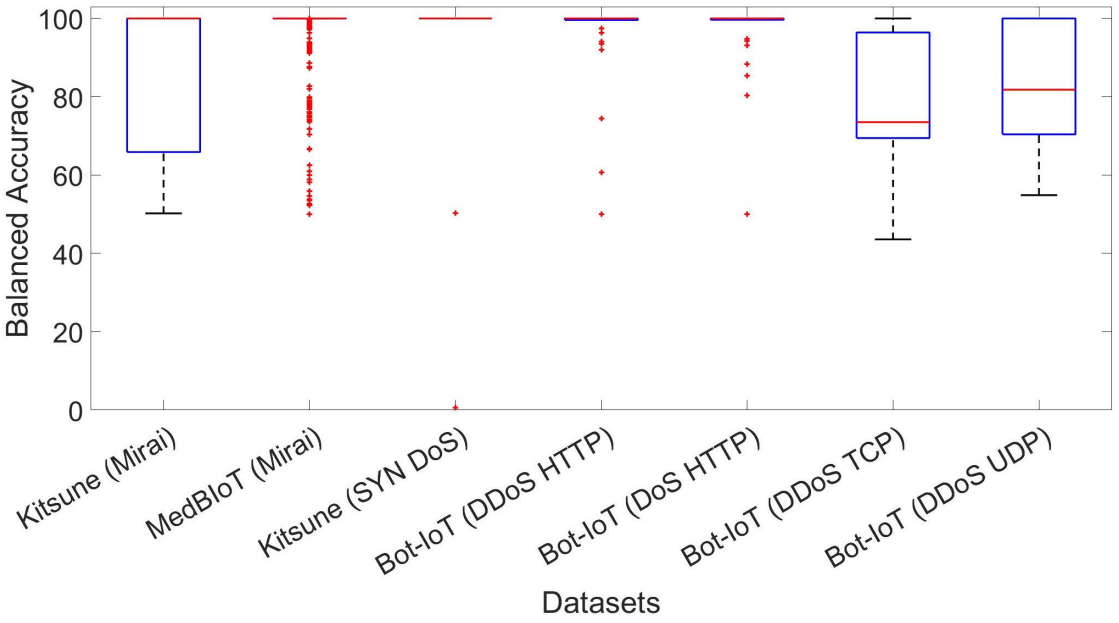


FIGURE 8. Performance of CDIS using the parameters set for Mirai is evaluated for other type of attacks on various datasets.

Furthermore, in Figure 7, we plot the logarithmic prediction error of CDIS with AADRNN, defined as $\log_{10}(|v_k^i - \Psi_k^i|)$, $1 \leq k \leq 712$, versus the slot k for three IP addresses: the 1st, 10th, and 101st. Our purpose is to present the prediction errors of the online training clearly. Indeed, the results on the 10th and 101st IP addresses show that CDIS achieves lower prediction errors for normal non-attack traffic after $k = 100$. On the other hand, the online training does not appear to reduce the accuracy for the 1st IP address, which may be because this address was never compromised as shown by the ground truth in Figure 3.

C. CDIS WITH DIFFERENT DATASETS

Although this paper mainly focuses on identifying the compromised IoT devices during a Mirai Botnet attack, the proposed CDIS architecture can also be used for different types of DDoS or DoS attacks, in which the malware spreads over the devices. However, the proposed network statistics may or may not be effective while implementing our CDIS for DDoS attacks other than Mirai. Accordingly, achieving a high performance for various types of DDoS attacks may require to define and use a much larger set of statistics. In this section, we now evaluate the performance of CDIS for various types of DDoS attacks provided in three different datasets: Kitsune [47], [48], MedBIoT [55], and Bot-IoT [56].

In addition to the Mirai Botnet data which is used during the performance evaluation in Section V, we now use SYN DoS data from the Kitsune dataset, which contains $2,771,276$ packets transmitted in about 53 minutes. Next, we evaluate the performance of CDIS on Mirai attack in the MedBIoT dataset. For this dataset, we merged the files of attack and normal traffic preserving the actual time stamps resulted in $5,727,929$ packets transmitted in about 30 minutes. We also present our results for DDoS attacks using HTTP, TCP and UDP protocols as well as a DoS attack using HTTP protocol from the Bot-IoT dataset. In Bot-IoT, there are $19,826$ packets transmitted in 42 minutes for the DDoS HTTP, $19,548,235$ packets in 40 minutes for DDoS TCP, $18,965,736$ packets in 47 minutes for DDoS UDP, and $29,762$ packets in 49 minutes for DoS HTTP. During the performance evaluation for the datasets except DDoS TCP and DDoS UDP, we use the same data processing and parameter settings which are described in Sections III, IV-D and IV-E. For DDoS TCP and DDoS UDP, we set $\tau = 0.01$ and $M = 190$.

Figure 8 displays the balanced accuracy results of the performance evaluation of CDIS for the Kitsune, MedBIoT and Bot-IoT datasets. First of all, these results show that the proposed CDIS is able to very successfully identify compromised devices during Mirai Botnet attacks, with a median Balanced Accuracy of 100% for both Kitsune and MedBIoT. Recall that the Balanced Accuracy is above 92% for 2/3 of unique IP addresses. For MedBIoT dataset, the Balanced Accuracy performance is above 85% for 90% of all IP addresses.

The results in Figure 8 also show that CDIS can achieve high performance for DDoS and DoS attacks. On the other hand, the results for DDoS and DoS attacks, especially those use TCP and UDP protocols, are significantly lower than those for Mirai attacks. The inferior performance is mainly because the traffic statistics are defined considering the Mirai Botnet attacks. For each of the DDoS HTTP and DoS HTTP, the median Balanced Accuracy performance is above 100%. We may also see that the performance of CDIS slightly lower for DDoS attacks on the network traffic using TCP or UDP protocols. Since CDIS parameters have already been adapted for DDoS TCP and DDoS UDP datasets, our results show that communication protocols are effective on the identification

performance and can be evaluated to determine more specific statistics.

D. FURTHER REMARKS ON THE RESULTS

We first evaluated the performance of CDIS on Kitsune Mirai attack dataset for two different network setups with 24 and 107 unique IP addresses. We also presented the performance evaluation for 7 attack data in three different datasets, namely Kitsune, MedBIoT and Bot-IoT. The experimental results show that:

- The CDIS achieves high performance (94% Balanced Accuracy) with low computation time for both execution and online training.
- The CDIS under AADRNN outperforms the other models (LR, Lasso, KNN, MLP, LSTM) by a significant margin, while its computation time is very competitive with that of the fastest (simplest) models.
- The CDIS can be used not only for Mirai, but also for various types of DDoS attacks where malware spreads over IoT devices. However, some attack types and/or communication protocols may require customization of traffic statistics and parameter settings of CDIS.

VI. CONCLUSION

In order to identify compromised devices and/or IP addresses in an IoT network as ongoing traffic flows through the system, we have developed a novel "Compromised Device Identification System (CDIS)", which analyzes the received and transmitted traffic by each individual device. Based on inter-packet transmission times and packet lengths that are measured from the traffic flow and taken in successive time windows, it determines whether the device is compromised using ML. The AADRNN is used as the ML tool, and is trained via online auto-associative learning using only normal traffic that is available during the system's normal real-time operation. Thus, it is important to state that CDIS does not require prior offline collection of any attack or normal traffic.

The performance of CDIS is tested on a publicly available Kitsune Mirai Botnet attack dataset for two different network setups, as well as on the MedBIoT and Bot-IoT datasets. As the experimental results suggest, the proposed CDIS provides highly accurate results, and it may pave the way to prevent Botnet attacks from spreading over the devices in an IoT network by blocking, or dropping, the outflow of traffic from IoT devices or IP addresses that have been identified as being compromised and turned into Bots.

Thus future work could also study the dynamics related to the CDIS system, from the instant when the attack begins, to the instant of detection, and finally to the time when the Botnet outflows can be blocked or dropped, so as to determine the overall time delay needed to actually stop the Botnet from spreading. Such studies can also examine the possibility of identifying the sources of the Botnet and carrying out preventive blocking/dropping of traffic at the sources themselves as was suggested in some earlier work [57]. Another important issue that is worth considering, is the energy consumption consequences of the AD itself [58], [59] as well as of the mitigation actions that are taken.

The proposed CDIS does not consider the known direct relationships between devices, such as their connectivity patterns. Thus in future work, it will be interesting to investigate the effects of each compromised device on other devices in an IoT network not just from the arriving packets at each node but also from any available known information concerning the connectivity between IP addresses, network nodes and devices.

APPENDIX: DENSE RANDOM NEURAL NETWORK WITH ONLINE AUTO-ASSOCIATIVE LEARNING (AADRNN)

The AADRNN structure introduced in [42], which was adapted to the design of CDIS, has L hidden layers, with J clusters in each hidden layer as shown in Figure 2. Each cluster consists of n statistically identical, probabilistically interconnected cells. As shown in Figure 2, the input to the AADRNN is the collection of network statistics, x_k^i, calculated by the NTSC module for each device or port i for each successive time window $k-1$, and the corresponding AADRNN output is denoted by $y_{L+1}(x_k^i, W^{i,k})$, for current window k.

Since cells in a given cluster are identical, for the time being we denote the state of each cell in layer l and cluster j by $q_{j,l}$, with total firing rate denoted by $r_{j,l}$, and it receives external excitatory arrivals of spikes at rate $\Lambda_{j,l}$. Each cell j also receives inhibitory inputs from a corresponding cell in each cluster j' belonging to the previous layer $l-1$, with inhibitory weight $w^-_{j',l-1,j}$, $j' \in \{1, \ldots J\}$. Thus for any cell in cluster (j,l) at layer $l \in \{2, \ldots, L\}$, the cell's total external inhibitory input is:

$$\lambda_{j,l} = \lambda^-_{j,l} + \sum_{j'=1}^{J} q_{j',l-1} \times w^-_{j',l-1,j}, \tag{24}$$

where $\lambda^-_{j,l}$ is an additional external inhibitory spiking rate into cell or neuron (j,l), and the sum of the (j,l)-neuron's outgoing inhibitory weights is:

$$w_{j,l} = \sum_{j'=1}^{J} w^-[j,l,j'], \quad hence\ r_{j,l} = w_{j,l} + r,\ r \geq 0, \tag{25}$$

where r is the firing rate of the soma-to-soma interactions that provoke the joint firing of any cell in the same cluster with probability $\frac{p}{n}$ where n is the number of cells in each cluster. Thus, we have:

$$q_{j,l} = \frac{\Lambda_{j,l} + rq_{j,l}(n-1)\sum_{v=0}^{\infty}[\frac{q_{j,l}p(n-1)}{n}]^v \frac{1-p}{n}}{r_{j,l} + \lambda_{j,l} + rq(n-1)\sum_{v=0}^{\infty}[\frac{q_{j,l}p(n-1)}{n}]^v \frac{p}{n}}, \tag{26}$$

which reduces to:

$$q_{j,l} = \frac{\Lambda_{j,l} + \frac{rq_{j,l}(n-1)(1-p)}{n-q_{j,l}p(n-1)}}{r_{j,l} + \lambda_{j,l} + \frac{rq_{j,l}p(n-1)}{n-q_{j,l}p(n-1)}}. \tag{27}$$

We are interested in solutions of the above expression which are probabilities, and first seek the maximum value $q_{j,l} = 1$, which is attained for the maximum value of $\Lambda_{j,l}$. Thus we can write:

$$\begin{aligned}\Lambda_{j,l} &\leq r_{j,l} + \lambda_{j,l} - \frac{r(n-1)(1-2p)}{n - p(n-1)},\\ &\leq r_{j,l} + \lambda_{j,l} - \frac{r(n-1)}{n},\\ &\leq w_{j,l} + \lambda_{j,l} + \frac{r}{n}. \end{aligned} \tag{28}$$

We notice that th expression for $q_{j,l}$ is a second degree equation in $q_{j,l}$ and for large n it becomes:

$$q_{j,l}^2 p\lambda_{j,l} - q_{j,l}[\lambda_{j,l} + p(\Lambda_{j,l} + r)] + \Lambda_{j,l} = 0,$$

whose solution is the activation function $\zeta(.)$ for each cell in each cluster of the AADRNN:

$$\begin{aligned}\zeta_{j,l} = max\Big[1, \frac{\lambda_{j,l} + p(\Lambda_{j,l} + r)}{2p\lambda_{j,l}}\\ \pm \frac{\sqrt{[\lambda_{j,l} + p(\Lambda_{j,l} + r)]^2 - 4p\lambda_{j,l}\Lambda_{j,l}}}{2p\lambda_{j,l}}\Big]. \end{aligned} \tag{29}$$

REFERENCES

[1] Cisco. (Mar. 2020). *Cisco Annual Internet Report (2018–2023)*. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] G. Matta, S. Chlup, A. M. Shaaban, C. Schmittner, A. Pinzenöhler, E. Szalai, and M. Tauber, "Risk management and standard compliance for cyber-physical systems of systems," *Infocommun. J.*, vol. 13, no. 2, pp. 32–39, 2021, doi: 10.36244/ICJ.2021.2.5.

[3] S. Maksuti, M. Zsilak, M. Tauber, and J. Delsing, "Security and autonomic management in system of systems," *Infocommun. J.*, vol. 13, no. 3, pp. 66–75, 2021, doi: 10.36244/ICJ.2021.3.7.

[4] *Hp Study Reveals 70 Percent of Internet of Things Devices Vulnerable to Attack*. Accessed: Jan. 25. 25. [Online]. Available: https://www.hp.com/us-en/hp-news/press-release.html%3Fid=1744676

[5] M. Frustaci, P. Pace, G. Aloi, and G. Fortino, "Evaluating critical security issues of the IoT world: Present and future challenges," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2483–2495, Aug. 2018.

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

[7] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in Internet-of-Things," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1250–1258, Oct. 2017.

[8] S. Benzarti, B. Triki, and O. Korbaa, "A survey on attacks in Internet of Things based networks," in *Proc. Int. Conf. Eng. MIS (ICEMIS)*, May 2017, pp. 1–7.

[9] CISA. *Understanding Denial-of-Service Attacks*. Accessed: Dec. 4, 2022. [Online]. Available: https://us-cert.cisa.gov/ncas/tips/ST04-015

[10] G. Carl, G. Kesidis, R. R. Brooks, and S. Rai, "Denial-of-service attack-detection techniques," *IEEE Internet Comput.*, vol. 10, no. 1, pp. 82–89, Jan. 2006.

[11] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Comput. Netw.*, vol. 44, no. 5, pp. 643–666, Apr. 2004.

[12] D. Goodin. (Dec. 2017). *100,000-Strong Botnet Built on Router 0-Day Could Strike at Any Time*. Ars Technica. [Online]. Available: https://arstechnica.com/information-technology/2017/12/100000-strong-botnet-built-on-router-0-day-could-strike-at-any-time/

[13] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the mirai botnet," in *Proc. Int. Conf. Softw. Secur. Assurance (ICSSA)*, Jul. 2017, pp. 6–12.

[14] J. Biggs. (Oct. 2018). *Hackers Release Source Code for a Powerful DDoS App Called Mirai*. TechCrunch. [Online]. Available: https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/

[15] R. Hackett, "Why a hacker dumped code behind colossal website-trampling Botnet," Tech. Rep., Oct. 2016. [Online]. Available: https://finance.yahoo.com/news/why-hacker-dumped-code-behind-145847907.html

[16] N. Statt. (Oct. 2016). *How an Army of Vulnerable Gadgets Took Down the Web Today*. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[17] B. Tushir, H. Sehgal, R. Nair, B. Dezfouli, and Y. Liu, "The impact of DoS attacks onResource-constrained IoT devices: A study on the Mirai attack," 2021, *arXiv:2104.09041*.

[18] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the Mirai botnet," in *Proc. 26th USENIX Secur. Symp.*, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[19] H. Sinanovic and S. Mrdovic, "Analysis of Mirai malicious software," in *Proc. 25th Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, Sep. 2017, pp. 1–5.

[20] Z. Ahmed, S. M. Danish, H. K. Qureshi, and M. Lestas, "Protecting IoTs from Mirai botnet attacks using blockchains," in *Proc. IEEE 24th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, Sep. 2019, pp. 1–6.

[21] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer Internet of Things devices," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2018, pp. 29–35.

[22] C. S. Htwe, Y. M. Thant, and M. M. S. Thwin, "Botnets attack detection using machine learning approach for IoT environment," *J. Phys., Conf. Ser.*, vol. 1646, no. 1, Sep. 2020, Art. no. 012101.

[23] M. Banerjee and S. Samantaray, "Network traffic analysis based iot botnet detection using honeynet data applying classification techniques," *Int. J. Comput. Sci. Inf. Secur.*, vol. 17, no. 8, pp. 61–66, 2019.

[24] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect Internet of Things botnets," in *Proc. IEEE Conf. Russian Young Researchers Electr. Electron. Eng. (EIConRus)*, Jan. 2018, pp. 105–108.

[25] T. A. Tuan, "Performance evaluation of botnet DDoS attack detection using machine learning," *Evol. Intell.*, vol. 13, pp. 283–294, Nov. 2019.

[26] I. Letteri, M. D. Rosso, P. Caianiello, and D. Cassioli, "Performance of botnet detection by neural networks in software-defined networks," in *Proc. ITASEC*, 2018, pp. 1–10.

[27] S. Sriram, R. Vinayakumar, M. Alazab, and S. Kp, "Network flow based IoT botnet attack detection using deep learning," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 189–194.

[28] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Machine learning-based IoT-botnet attack detection with sequential architecture," *Sensors*, vol. 20, no. 16, p. 4372, Aug. 2020.

[29] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet detection in the Internet of Things using deep learning approaches," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[30] J. Liu, S. Liu, and S. Zhang, "Detection of IoT botnet based on deep learning," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 8381–8385.

[31] G. D. L. T. Parra, P. Rad, K.-K.-R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *J. Netw. Comput. Appl.*, vol. 163, Aug. 2020, Art. no. 102662.

[32] C. Tzagkarakis, N. Petroulakis, and S. Ioannidis, "Botnet attack detection at the IoT edge based on sparse representation," in *Proc. Global IoT Summit (GIoTS)*, Jun. 2019, pp. 1–6.

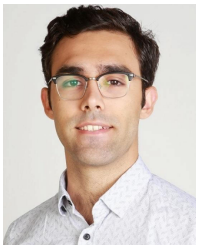
[33] M. Nakip and E. Gelenbe, "MIRAI botnet attack detection with auto-associative dense random neural network," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2021, pp. 1–6.

[34] E. Gelenbe and E. C.-H. Ngai, "Adaptive QoS routing for significant events in wireless sensor networks," in *Proc. 5th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst.*, Sep. 2008, pp. 410–415.

[35] E. Gelenbe, J. Domanska, P. Frohlich, M. P. Nowak, and S. Nowak, "Self-aware networks that optimize security, QoS, and energy," *Proc. IEEE*, vol. 108, no. 7, pp. 1150–1167, Jul. 2020.

[36] A. Kumar and T. J. Lim, "Early detection of Mirai-like IoT bots in large-scale networks through sub-sampled packet traffic analysis," in *Proc. Future Inf. Commun. Conf.* Cham, Switzerland: Springer, 2019, pp. 847–867.
[37] M. Chatterjee, A. S. Namin, and P. Datta, "Evidence fusion for malicious bot detection in IoT," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 4545–4548.
[38] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "DÏoT: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 756–767.
[39] N. V. Abhishek, T. J. Lim, B. Sikdar, and A. Tandon, "An intrusion detection system for detecting compromised gateways in clustered IoT networks," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Quality Rel. (CQR)*, May 2018, pp. 1–6.
[40] M. Taneja, "An analytics framework to detect compromised IoT devices using mobility behavior," in *Proc. Int. Conf. ICT Converg. (ICTC)*, Oct. 2013, pp. 38–43.
[41] E. Gelenbe and Y. Yin, "Deep learning with random neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2016, pp. 1633–1638.
[42] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks," in *Proc. Int. Conf. Man–Mach. Interact.* Cham, Switzerland: Springer, 2017, pp. 3–18.
[43] O. Brun, Y. Yin, E. Gelenbe, Y. M. Kadioglu, J. Augusto-Gonzalez, and M. Ramos, "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments," in *Proc. Int. ISCIS Secur. Workshop*. Cham, Switzerland: Springer, 2018, pp. 79–89.
[44] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Comput.*, vol. 1, no. 4, pp. 502–510, 1989.
[45] E. Gelenbe, "Learning in the recurrent random neural network," *Neural Comput.*, vol. 5, no. 1, pp. 154–164, 1993.
[46] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "Neural network architectures for the detection of SYN flood attacks in IoT systems," in *Proc. 13th ACM Int. Conf. Pervasive Technol. Rel. Assistive Environ.*, Jun. 2020, pp. 1–4.
[47] *Kitsune Network Attack Dataset*. Accessed: Dec. 4, 2022. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune
[48] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2018, pp. 1–15.
[49] E. Gelenbe, "G-networks with triggered customer movement," *J. Appl. Probab.*, vol. 30, no. 3, pp. 742–748, Sep. 1993.
[50] E. Gelenbe and Nakıp, "G-networks that detect different types of cyber-attacks," in *Proc. 30th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Aug. 2022, pp. 1–8.
[51] E. Gelenbe, Z.-W. Mao, and Y.-D. Li, "Function approximation with spiked random networks," *IEEE Trans. Neural Netw.*, vol. 10, no. 1, pp. 3–9, Jan. 1999.
[52] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[53] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12 no. 10, pp. 2825–2830, 2012.
[54] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, "The balanced accuracy and its posterior distribution," in *Proc. 20th Int. Conf. Pattern Recognit.*, Aug. 2010, pp. 3121–3124.
[55] A. Guerra-Manzanares, J. Medina-Galindo, H. Bahsi, and S. Nõmm, "MedBIoT: Generation of an IoT botnet dataset in a medium-sized IoT network," in *Proc. 6th Int. Conf. Inf. Syst. Secur. Privacy*, 2020, pp. 207–218.
[56] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687
[57] E. Gelenbe and G. Loukas, "A self-aware approach to denial of service defence," *Comput. Netw.*, vol. 51, no. 5, pp. 1299–1314, Apr. 2007.
[58] B. Pernici, M. Aiello, J. vom Brocke, B. Donnellan, E. Gelenbe, and M. Kretsis, "What IS can do for environmental sustainability: A report from CAiSE'11 panel on green and sustainable IS," *Commun. Assoc. Inf. Syst.*, vol. 30, no. 1, p. 18, 2012.
[59] P. Fröhlich, E. Gelenbe, J. Fiołka, J. Checinski, M. Nowak, and Z. Filus, "Smart SDN management of fog services to optimize QoS and energy," *Sensors*, vol. 21, no. 9, p. 3105, Apr. 2021, doi: 10.3390/s21093105.

EROL GELENBE (Life Fellow, IEEE) received the B.S. degree from Middle East Technical University, Turkey, the M.S. and Ph.D. degrees in electrical engineering from the Polytechnic Institute (now Tandon School of New York University), and the D.Sc. degree in mathematical sciences from Sorbonne University, Paris. He is currently a Professor at the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences. He is also an Honorary Professor at the University of Electronic Science and Technology of China, a Research Professor at Yaşar University, Bornova, Turkey, and a Researcher at the I3S CNRS Laboratory, University Côte d'Azur, Nice, France. His prior positions include Chairs at the Universities of Liège, Paris-Saclay and Paris-Descartes, NJIT, Duke University, UCF, and Imperial College. He invented mathematical models to optimize computer and network performance, including diffusion approximations, G-networks, and the random neural network and its machine learning algorithms, and routing techniques to optimize multi-party Internet communications. He is ranked among the top 25 all-time doctoral supervisors by the Mathematics Genealogy Project for graduating over 90 Ph.D. He also works on methods that maximize performance, cybersecurity, and minimize energy consumption in computer systems and networks. Recognized for his broad research impact, he was elected as a fellow of ACM, IFIP, RSS, IET, the National Academy of Technologies of France, and the Science Academy of Turkey, a Foreign Fellow of the Royal Academy of Science, Arts and Letters of Belgium and the Science Academy of Poland, and an Honorary Fellow of the Hungarian Academy of Sciences and the Islamic Academy of Sciences. He was awarded the Honours of Commander of Merit of Italy, in 2005, and of France, in 2019, Knight of the Légion d'Honneur, in 2014, and Commander of the Order of the Crown of Belgium, in 2022.



MERT NAKIP (Student Member, IEEE) received the B.Sc. (Hons.) and M.Sc. degrees in electrical and electronics engineering from Yaşar University, Izmir, Turkey, in 2018 and 2020, respectively. He is currently pursuing the Ph.D. degree with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences. His M.Sc. thesis focused on the application of machine learning methods to the IoT. He is also a Research Assistant with the Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland. He was supported by the National Graduate Scholarship Program of Turkish Scientific and Technological Research Council (TÜBİTAK) 2210C in high-priority technological. He participates in European Commission H2020 Program under the IoTAC Research and Innovation Action as a Researcher. His design of a multi-sensor fire detector via machine learning methods was ranked #1 nationally at the Industry-Focused Undergraduate Graduation Projects Competition organized by TÜBİTAK.