Online Self-Supervised Learning in Machine Learning Intrusion Detection for the Internet of Things

Mert Nakıp, *Student Member, IEEE*, and Erol Gelenbe *Fellow, IEEE*

***Abstract*—This paper proposes a novel Self-Supervised Intrusion Detection (SSID) framework, which enables a fully online Machine Learning (ML) based Intrusion Detection System (IDS) that requires no human intervention or prior off-line learning. The proposed framework analyzes and labels incoming traffic packets based only on the decisions of the IDS itself using an Auto-Associative Deep Random Neural Network, and on an online estimate of its statistically measured trustworthiness. The SSID framework enables IDS to adapt rapidly to time-varying characteristics of the network traffic, and eliminates the need for offline data collection. This approach avoids human errors in data labeling, and human labor and computational costs of model training and data collection. The approach is experimentally evaluated on public datasets and compared with well-known ML models, showing that this SSID framework is very useful and advantageous as an accurate and online learning ML-based IDS for IoT systems.**

***Index Terms*—Self-Supervised Learning, Random Neural Network (RNN), Auto-Associative Deep RNN, Botnet Attacks, Intrusion Detection, Machine Learning, Internet of Things**

I. Introduction

Botnet attacks can lead to thousands of infected devices [1] compromising the devices of victims and turning them into "bots" via malware [2], which in turn cause Distributed Denial-of-Service (DDoS) attacks. The *malicious* bots, i.e. compromised devices, can generate fraud information, cause data leaks, and spread malware. It is reported that 27.7% of all global website traffic in 2021 was generated by bots with malicious intent, and is growing with a 7.3% increase reported between 2018 and 2021 [3].

Botnet attacks severely challenge resource-constrained devices and Internet of Things (IoT) networks [4], as an attack propagates over the victim network increasing network congestion, power consumption, and processor and memory usage of IoT devices over time. Therefore, it is crucial to detect malicious network traffic and identify compromised IoT devices during an ongoing Botnet attack. While detecting malicious traffic allows reactive actions to alleviate the effects of the attack and stop it, identifying compromised IoT devices paves the way for preventive actions against the spread of malware and Botnet attack.

On the other hand, as the majority (approximately 52%) of IoT connections are to low cost and low maintenance devices deployed in massive IoT networks [5], developing and implementing complex and advanced security methods is challenging as well. To this end, early research [6] developed various types of lightweight Machine Learning (ML)-based Intrusion Detection Systems (IDS) for IoT networks, showing that machine ML based IDS anomaly detection is very promising in detecting zero-day attacks based on unknown intrusions that often target vulnerable devices and networks.

Since the decisions of anomaly-based IDS are highly dependent on the characteristics of the normal traffic used for parameter optimization (i.e. learning), accurate decisions become more difficult when the normal behavior of network traffic changes over time due to both internal and external influences. For example, new device(s) may be added to the IoT network causing a considerable change in aggregated network traffic. Therefore, anomaly-based IDS could greatly benefit from the ability to adapt in real time to time-varying characteristics of network traffic, ideally through sequential online learning [7], [8]. However, the effectiveness of online learning is often limited by its inability to collect and label sufficient data, as well as by the need to select appropriate time intervals for parameter updates.

In this paper a novel fully online Self-Supervised Intrusion Detection (SSID) framework is proposed. SSID learns from arriving traffic packets, measures the trustworthiness of the IDS, including its generalization ability and accuracy on traffic packets it uses for learning. It can then decide when to update the neural weights of its learning algorithm, keeping itself up-to-date with a high intrusion detection accuracy.

The SSID framework can be used with any anomaly-based IDS that requires parameter optimization, providing fully online self-supervised learning of parameters in parallel with real-time detection requiring no human intervention. It also eliminates the need for labeled or unlabeled offline data collection, and offline training or parameter optimization. Therefore, the proposed framework contrasts sharply with much of existing work [9]–[15] that has implemented self-supervised learning for intrusion detection, often utilizing offline (small-sized) labeled or unlabeled training data and pseudo-labeling. Accordingly, as its main advantages, the SSID framework

- Enables IDS to easily adapt time varying characteristics

This research has been supported by the European Commission H2020 Program through the IoTAC Research and Innovation Action, under Grant Agreement No. 952684.

M. Nakıp and E. Gelenbe are with Institute of Theoretical and Applied Informatics, Polish Academy of Sciences (PAN), Gliwice, Poland (e-mails: mnakip@iitis.pl, and seg@iitis.pl)

E. Gelenbe is also with Lab. I3S, Université Côte d'Azur, Nice, France, and Yaşar University, Izmir, Turkey

of the network traffic,
- Eliminates the need for offline data collection,
- Prevents human errors in data labeling (online or offline), and
- Avoids human labor and computational costs for model training and data collection through prior experiments.

We also evaluate the performance of the SSID framework for two tasks, malicious traffic detection and compromised device identification, using Deep Random Neural Network (DRNN), Multi-Layer Perceptron (MLP), and the state-of-the-art Compromised Device Identification System (CDIS) on Kitsune and Bot-IoT datasets. The results revealed that the intrusion detection systems trained under the SSID framework achieve considerably high performance compared to the same systems with offline and quasi-online (incremental and sequential) learning, although the IDS trained via SSID requires no offline dataset, external parameter optimization or human intervention.

The remainder of this paper is organized as follows: Section II reviews the related work on intrusion detection. Section III and Section IV respectively propose the SSID framework and present the methodology enabling the self-supervised learning for IDS. Section V evaluates the SSID framework for malicious traffic detection and compromised device identification on public datasets, and compares its performance against the state-of-the art methods. Finally, Section VI summarizes this paper and provides some insights for the future work.

II. Related Work

We now briefly review recent related work on intrusion detection in three categories of the work that: 1) detect malicious traffic during Botnet-based DDoS (in short Botnet) attacks, 2) identifies compromised network nodes, and 3) performs self-supervised learning for intrusion detection.

A. DDoS Botnet Attack Detection

In [16], Tuan et al. conducted a comparative study for performance evaluation of ML methods aiming to classify Botnet attack traffic. In this work, the authors evaluated the performances of Support Vector Machine (SVM), MLP, Decision Tree (DT), Naive Bayes (NB), and unsupervised ML methods (such as K-means clustering) on two datasets (including KDD'99) revealing that unsupervised ML methods achieve the best performance with 98% accuracy. In [17], Shao et al. created an ensemble of Hoeffding Tree and Random Forest (RF) models with online learning using both normal and attack traffic. In [18], Shafiq et al. developed a feature selection technique as a preprocessing algorithm for an ML-based botnet attack detector. This algorithm ranks features according to their Pearson correlation coefficients and greedily maximizes the detector's performance with respect to area under Receiver Operating Characteristic (ROC) curve in the Bot-IoT dataset. In [19], Doshi et al. developed an attack detection algorithm comprised of feature extraction from the network traffic and ML classifier. In the place of the ML classifier, the authors used each of K-Nearest Neighbour (KNN), SVM, DT, and MLP methods; then, they evaluated the performance of this algorithm on a dataset collected within the same work. Letteri et al. [20] developed an MLP based Mirai Botnet detector specialized for Software Defined Networks. The authors fed 5 metrics, including the used communication protocol, to MLP.

In [21], Banerjee and Samantaray performed experimental work to deploy a network of honeypots that attracts botnet attacks and to detect those attacks via ML methods, such as DT, NB, Gradient Boosting, and RF. In reference [22], McDermott et al. developed the Bidirectional LSTM-based method which is developed for packet-level botnet attack detection by performing text recognition on multiple features including source and destination IP addresses of a packet. In addition, Tzagkarakis et al. [23] developed a sparse representation framework with parameter tuning using only normal traffic for botnet attack detection.

Meidan et al. [24] developed an ML-based attack detection technique which is trained using only normal traffic and tested for Mirai and Bashlite botnet attacks on an IoT network with nine devices. The authors also published the data collected in this study under the name N-BaIoT dataset. In order to detect Botnet attack in N-BaIoT dataset, Htwe et al. [25] used Classification and Regression Trees with feature selection, and Sriram et al. [26] performed a comparative study using 7 different ML methods (including NB, KNN, and SVM). In Reference [27], Soe et al. developed a Botnet attack detection algorithm comprised of two sequential phases first to train utilized ML method and perform feature selection, then to perform attack detection. The authors used MLP and NB within this architecture, and they evaluated the performance on N-BaIoT dataset. In [28], Parra et al. developed a cloud based attack detection method using Convolutional Neural Network (CNN) for phishing and using Long-Short Term Memory (LSTM) for Botnet attacks. The authors evaluated the performance of this method also on the N-BaIoT dataset achieving 94.8% accuracy. CNN was also used by Liu et al. [29] with features that are processed by the triangle area maps based multivariate correlation analysis algorithm.

B. Compromised Device Identification

Some recent work [30]–[35] focused on detecting compromised IoT devices during Botnet attacks, while Kumar et al. [30] detected Mirai-like bots scanning the destination port numbers in packet headers using an optimization-based technique for subsets of all IoT packets. Chatterjee et al. [31] identified malicious devices in IoT networks via evidence theory-based analysis. To this end, they analyzed traffic flows and selected the rarest features from a large number of communication features, including number of connections, transport layer protocol, and source/destination ports. In [36], in order to detect IoT botnet in an Industrial IoT network, Nguyen et al. developed a dynamic analysis technique utilizing various ML models, such as SVM, DT, and KNN, based on the features generated from the executable files. In Reference [37], Hristov and Trifonov developed a compromised device identification algorithm using wavelet transformation and Haar filter on the metrics indicating the processor, memory and network

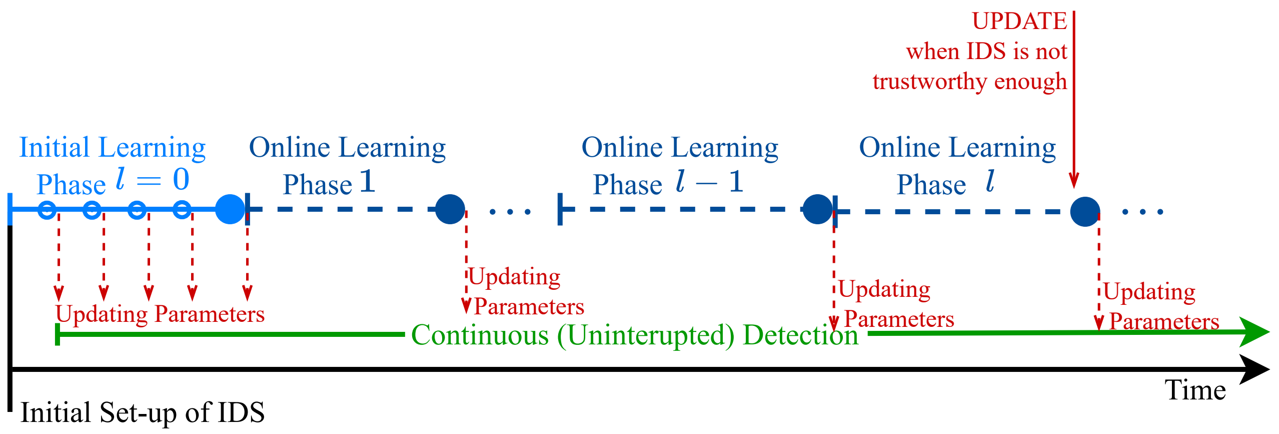


Fig. 1. Detection and learning processes of IDS within the Fully Online Self-Supervised Intrusion Detection (SSID) framework

interface card usage of an IoT device. In [35], Prokofiev et al. used Logistic Regression to determine if the source device is a bot based on 10 metrics regarding the traffic packets. The performance of logistic regression is tested for a botnet that spreads through brute-force attacks. Nguyen et al. [32] detected compromised devices by an anomaly detection technique combining federated learning with language analysis for individual device types identified prior to detection. In order to evaluate the performance of this technique, the authors collected a dataset by installing 33 IoT devices, 5 of which were malicious, and showed that detection performance is around 94$\%$ for positive and 99$\%$ for negative samples.

More differently, in Reference [33], Abhishek et al. detected not compromised devices but compromised gateways monitoring the downlink channels in an IoT network and performing binary hypothesis test. In [38], Trajanovski and Zhang developed a framework consisting of honeypots to identify the indicators of compromised devices and botnet attacks. Bahşi et al. in [39] addressed the scalability issues for ML-based Bot detection algorithms by minimizing the number of inputs of ML model via feature selection. In [34], for mobile IoT devices, Taneja proposed to detect compromised devices taking into account their location, such that if a location change or current location of an IoT device is classified as unusual behavior, the device is considered compromised.

C. *Self-Supervised Learning for Intrusion Detection*

Song and Kim [9] developed self-supervised learning algorithm for anomaly-based IDS in in-vehicle networks. In this algorithm, LSTM is used to generated noisy pseudo data as normal network traffic while Reduced Inception-ResNet is used to make binary classification and detect unknown (zero-day) attacks. Wang et al. [10] applied and adapted Bootstrap Your Own Latent (BYOL) self-supervised learning approach, which has been proposed in [40], for intrusion detection. For anomaly detection, Zhang et al. [11] developed deep adversarial training architecture by extending the well-known bidirectional Generative Adversarial Network (GAN) model. This architecture jointly learns from normal data and generated latent features. Caville et al. [13] developed a Graph Neural Network (GNN) based network-level IDS. This IDS was trained with a self-supervised learning approach using both positive and negative samples for offline training with an encoder graph created using an extended version of the well-known GraphSAGE framework. Wang et al. [15] developed an IDS with unsupervised learning combined with transformer based self-supervised masked context reconstruction, which improves the learning by magnifying the abnormal intrusion behaviours. Abououf et al. [14] developed a lightweight IDS architecture based on LSTM Auto Encoder (LSTM-AE) to perform detection on IoT nodes. This model is trained offline and unsupervised in an encoder-decoder architecture using a pre-collected dataset in the cloud.

In this work, we propose a novel learning framework, called SSID, for ML-based intrusion detection. In sharp contrast to existing studies, our SSID framework 1) eliminates the need for an additional generative model for training (although this is a common and well-known approach to self-supervised learning), 2) does not need training data collected offline (but can still use if available), and 3) paves the way for learning fully online on independent network nodes.

III. System Design of the Self-Supervised Intrusion Detection Framework

As the main contribution of this paper, we propose the novel SSID framework to enable fully online self-supervised learning of the parameters of IDS with no need for human intervention. This section now presents the system design of this framework and states the preliminaries and some assumptions. To this end, we first briefly present the detection process within the SSID framework (shown in Figure 1) and the general IDS structure. Next, we explain the learning process performed in our framework.

A. *Intrusion Detection Process*

As shown in Figure 1, within the SSID framework, there are two main operations performed, intrusion detection and learning. Intrusion detection is the main operation performed by IDS and is not modified by SSID. That is, intrusion detection (as an operation) is defined only by a particular

IDS algorithm used in SSID. Therefore, in this section, we may only present the IDS with a general structure with regard to the requirements of SSID. On the other hand, we can say that our SSID framework ensures that IDS makes accurate decisions by updating its parameters with online self-supervised learning, and it performs intrusion detection uninterruptedly and continuously.

We may note that regarding the communication (data transfer) between intrusion detection and online self-supervised learning processes in SSID, first, the parameters of IDS are updated for detection during the initial learning and at the end of each learning phase. In addition, the decisions made by IDS are provided to use in any learning phase since the data collection within both initial and online learning phases is performed in a self-supervised fashion which enables automatic labeling of the packet samples as benign or malicious.

The SSID framework does not consider a specific algorithm for IDS or have strict requirements for it, except that it is based on ML or some other function with learnable parameters and has a certain range of inputs and outputs. In addition, although the SSID framework can also be used with both anomaly and signature based detection algorithms, the anomaly based algorithms shall perform better as the real-time network traffic contains only the normal "benign" traffic until an attack occurs. Therefore, we now present the general structure of IDS shown in Figure 2 that is taken into account during our analysis.

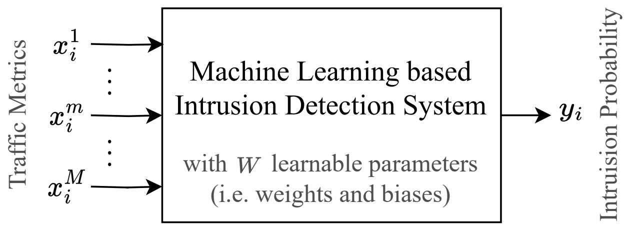

Traffic Metrics
x_i^1
x_i^m
x_i^M
Machine Learning based Intrusion Detection System
with W learnable parameters (i.e. weights and biases)
y_i
Intrusion Probability

Fig. 2. Overall structure of the IDS.

As shown in Figure 2, for each packet i (or bucket of packets), the IDS estimates the probability that packet i is malicious based on the provided traffic metrics. Accordingly, the input of the IDS is the vector of M metrics that are observed (or calculated) for the actual traffic packet i. This vector of M observed metrics is denoted by $x_i = [x_i^1, \ldots, x_i^m, \ldots, x_i^M]$, and $x_i \in [0,1]^M$. The output of the IDS, which is namely the intrusion decision and denoted by y_i, is the probability that packet i is malicious and takes value in $[0,1]$.

The structure of IDS is comprised of an ML model (which can also be considered a function with learnable parameters) with W parameters. Therefore, the ML-based IDS is a learned function that maps the metrics observed for the actual traffic packet i to the intrusion probability for that packet, i.e. $f : x_i \mapsto f(x_i)$ for $f(x_i) = y_i$, so that $f : [0,1]^M \rightarrow [0,1]$.

B. Online Self-Supervised Learning

In parallel with attack detection, our SSID framework provides online self-supervised learning of IDS parameters, as shown on the upper process line in Figure 1. As seen in that figure and Figure 3 which shows the learning process in SSID, the online self-supervised learning process starts with the initial learning phase (namely, $l = 0$) and continues with successive online learning phases.

Since network traffic characteristics may vary with time and substantially affect the IDS detection performance, it is crucial to update the parameters of the IDS onlineconcurrently with its real-time operation. The parameter updates performed by online learning ensure that the IDS is a trustworthy detector, by improving its performance and keeping it up-to-date with regard to the most recent traffic characteristics. Online learning also avoids having to collect and label large datasets for offline training, and thus it saves both time and resources. On the other hand, when ground-truth datasets are already available, the IDS can also be pre-trained, and its performance may be validated using traffic from other IoT networks.

In the remainder of this subsection, in order to clearly present the SSID framework, we shall only explain the main functionalities of both the initial and online learning phases through the block diagram of Figure 3. Then, Section IV presents the detailed comprehensive methodology that includes details regarding all the blocks in Figure 3.

1) Initial Learning: The initial learning phase in SSID can be considered a special case of the proposed methodology of self-supervised learning, which allows IDS to be used from its initial setup and updates the parameters of the IDS frequently achieving the desired performance gradually and quickly.

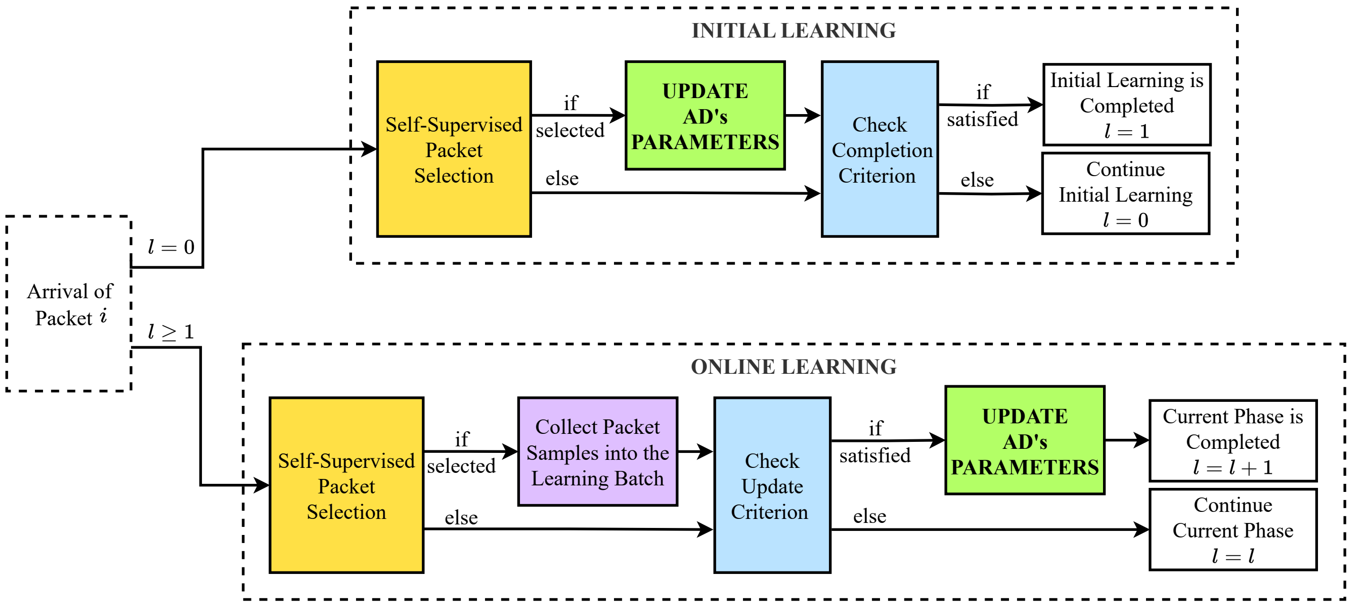
In detail, as shown in the top block of Figure 3, during the initial learning phase in SSID, the parameters of the IDS are updated (using any desired algorithm) for each packet that is selected for learning via our self-supervised packet selection methodology. Whether the parameters of the IDS are updated or not, SSID checks the trust based completion criterion of the initial learning phase $l = 0$ aiming to complete this phase as soon as the IDS is trained enough to make trustworthy decisions. To this end, it first calculates the trustworthiness of the IDS, namely the "trust coefficient" denoted by $\Gamma \in [0,1]$, which indicates the confidence of SSID in any decision made by the IDS. Since the IDS does not have any information about the network traffic patterns yet, SSID cannot judge the decisions of the IDS and starts the initial learning process with $\Gamma = 0$ meaning that there is no trust in the decisions of the IDS.

Subsequently, SSID checks the trust criterion to complete the initial learning phase $l = 0$ by measuring if SSID's trust in the IDS is greater than a threshold Θ, is the minimum desired trust level:

$$\text{if } \Gamma \geq \Theta, \text{ complete initial learning and set } l = 1 \quad (1)$$

Thus if $\Gamma \geq \Theta$, the initial learning phase has been completed, and the next packet will be considered for the first phase $l = 1$ of continuous on-line learning.

2) Online Learning: After the initial learning is completed, the parameters of IDS are updated via an online learning phase $l \geq 1$ when the trust of SSID in the IDS is unacceptably low. As the lower block in Figure 3 shows, the parameters of the



INITIAL LEARNING
Arrival of Packet i
$l = 0$
$l \geq 1$
Self-Supervised Packet Selection
if selected
else
UPDATE AD's PARAMETERS
Check Completion Criterion
if satisfied
else
Initial Learning is Completed $l = 1$
Continue Initial Learning $l = 0$
ONLINE LEARNING
Self-Supervised Packet Selection
if selected
else
Collect Packet Samples into the Learning Batch
Check Update Criterion
if satisfied
else
UPDATE AD's PARAMETERS
Current Phase is Completed $l = l + 1$
Continue Current Phase $l = l$

Fig. 3. Block diagram of the learning process in the SSID framework for online self-supervised learning of the parameters of IDS

IDS are updated for a collected batch of packets when the trustworthiness of the IDS is not acceptable anymore. When SSID is in the online learning phase $l \geq 1$, each packet i selected by our self-supervised packet selection method is collected into the batch of training packets, denoted by B^l.

Then, SSID checks the trust-based criterion to update the parameters of the IDS. Inversely with the initial learning phase, SSID now updates the parameters of the IDS if $\Gamma < \Theta$, at least K packets are collected for learning (i.e. $|B_l| \geq K$), and there is no attack detected by the IDS:

$$\text{if } \Gamma < \Theta \text{ and } |B^l| \geq K \text{ and } \frac{1}{I}\sum_{j=i-I+1}^{i} y_j \leq \gamma, \\ \text{update parameters and set } l = l + 1 \qquad (2)$$

where I is the number of packets to calculate the average of the intrusion decisions, γ is the intrusion threshold, and K is provided by the user considering properties of the network and learning algorithm. Limit of minimum K packets is added only to provide practical efficiency for training.

That is, SSID waits for a considerable decrease in the trustworthiness of the decisions of IDS to update the parameters since Γ is known to be already greater than Θ at the end of the initial learning phase $l = 0$. In this way, the learning is performed when it is essential.

On the other hand, if an intrusion is detected where the average output of the IDS is greater than γ, SSID clears the batch of collected packet samples, B^l:

$$\text{if } \frac{1}{I}\sum_{j=i-I}^{i} y_j > \gamma, \text{ empty } B^l \qquad (3)$$

With this cleanup, SSID aims to prevent the IDS from learning any false negative instances since false negative outputs are very likely to occur just before an attack is detected.

IV. Methodology of Self-Supervised Learning for Intrusion Detection

We now present our proposed methodology to train the utilized IDS in a self-supervised fashion enabling the fully online property of SSID. In other words, this section explains the details of the learning process in SSID, which are shown as subblocks in Figure 3. Recall that the learning process in SSID is independent of the learning (parameter update) algorithm and ML model used in the IDS; therefore, it can be used with any learning algorithm and any ML model that learns the normal network traffic.

A. Self-Supervised Packet Selection

As the first operation of the learning process in SSID, each packet i is decided to be used in learning the parameters of the IDS. We now present the methodology to select packets, which are observed during real-time detection, to be used in an upcoming learning phase. The packet selection is executed in a self-supervised manner that only considers the output of the IDS together with SSID's trust in it.

Let p_i^- and p_i^+ respectively be the probability of selecting packet i to be used as a *benign* or *malicious* packet sample in the training of IDS, and q_i be the probability of rejecting i to be used in training. That is, we select the packet i as the sample of a benign packet with probability p_i^- or that of an attack packet with probability p_i^+ to use it in training, or the packet i is not included in the training set with probability q_i. Also, recall that $y_i \in [0, 1]$ is the output of IDS for packet i.

Since we assume that there are no packet labeling mechanisms or labor to prepare packet data for learning, we select each packet i based on the output of IDS (which is the estimation of the probability of packet i being malicious) considering how trustworthy IDS is. Therefore, we shall also define a trust coefficient Γ to measure the trustworthiness of IDS at any time based on the representativeness of the packet samples that IDS learned until the end of the last learning phase and the generalization ability of IDS from these samples.

Accordingly, we start by defining p_i^+ as

$$p_i^+ \equiv \text{(trust in IDS) (est. prob. of packet } i \text{ being malicious)}$$

$$p_i^+ = \Gamma\, y_i \tag{4}$$

We further define p_i^- similarly to p_i^+:

$$p_i^- \equiv \text{(trust in IDS) (est. prob. of packet } i \text{ being normal)}$$

$$p_i^- = \Gamma\,(1 - y_i) \tag{5}$$

Subsequently, since

$$p_i^+ + p_i^- + q_i = 1, \tag{6}$$

the probability q_i of not selecting the packet i for training is:

$$\begin{aligned} q_i &= 1 - (p_i^+ + p_i^-) \\ &= 1 - \Gamma. \end{aligned} \tag{7}$$

Recall that SSID starts with $\Gamma = 0$ since the IDS does not have yet any information about the network traffic patterns at the initial learning phase. That is, the output of the IDS is calculated using the initially set parameter values (if available) and will not be able to achieve accurate detection for the particular traffic. In addition, for selecting the first packet, the parameters of the IDS are updated for the first time using $p_i^- = 1$, $p_i^+ = 0$, and $q_i = 0$. Thus, SSID selects the first packet to learn as a benign sample.

B. Trustworthiness of IDS

Now, we determine the trust coefficient Γ for the IDS in the SSID framework. Through this coefficient, we aim to include both the effects of changes in the normal behavior of network traffic over time and the generalization ability of the IDS into the packet selection model for learning.

To this end, we first define the factor of "representativeness", denoted by C_{rep}, for the traffic packets that are learned by the IDS. The representativeness factor C_{rep} takes a value in the range of $[0, 1]$ and measures how much the packets used for learning (during all of the past learning phases) represent the total observed traffic. In addition, we define the factor of "generalization ability", denoted by C_{gen}, of the IDS. The generalization factor C_{gen} takes a value in the range of $[0, 1]$ and is calculated only at the end of each parameter update since it is the only time when the parameters of the IDS are updated. These two factors shall respectively be presented in Section IV-C and Section IV-D.

Accordingly, in order to evaluate the trustworthiness of the intrusion decisions, we determine the trust coefficient Γ by combining the representativeness of packets learned with the generalization ability of the IDS simply as the multiplication of C_{rep} and C_{gen}:

$$\Gamma = C_{rep}\, C_{gen} \tag{8}$$

In this way, Γ simultaneously measures how much the IDS is able to learn and generalize from provided traffic packets and how much these packets reflect actual traffic patterns. That is, through this trust coefficient, we evaluate how much information the IDS can generalize from the traffic packets provided to make decisions for the upcoming traffic.

C. Representativeness of the Traffic that is Learned

In order to calculate the representativeness of the packet traffic used during the earlier learning phases, we compare the learned traffic with the total observed traffic through Kullback-Leibler (KL)-Divergence [41]. Therefore, there are two sets of traffic packets for comparison, the packets used in the previous learning phases up to and including l (where l is the latest completed learning phase) and the normal packets that are observed by IDS during continuous detection.

During this comparison, we assume that the packet traffic consists of two main properties, inter-transmission time (TT) and the packet length (PL) since these properties can be considered as the basis of traffic metrics, which are the inputs of the IDS. We further assume that packet arrivals – any sample collected from the network traffic – has a Poisson distribution so that the inter-transmission time TT is an Exponentially-distributed random variable. The packet length PL is also assumed to be an Exponentially-distributed random variable because the header length is considerably larger than the message length for the majority of IoT applications. In addition, TT and PL are considered to be independent. On the other hand, for particular applications, these assumptions and the traffic model can be changed and the below methodology can easily be adapted for the new traffic model with a new set of assumptions.

Furthermore, let S_l^{TT} and S_l^{PL} respectively denote the sets of the inter-transmission times and lengths of all packets learned at the end of l, and S_o^{TT} and S_o^{PL} respectively denote the same of all normal packets observed during continuous detection. In addition, according to our assumptions, S_l^{TT} and S_o^{TT} have exponential distributions with means of $1/\lambda_l$ and $1/\lambda_o$ while S_l^{PL} and S_l^{TT} also have exponential distributions with means of $1/\mu_l$ and $1/\mu_o$.

1) KL-Divergence for Inter-Transmission Times: For the set of inter-transmission times, $D_{KL}(S_o^{TT}||S_l^{TT})$ is KL-Divergence from S_l^{TT} to S_o^{TT} measuring the information gain achieved if S_o^{TT} would be used instead of S_l^{TT} which has been used during the learning phases of SSID. Note that small KL-Divergence means low information gain, and $D_{KL}(S_o^{TT}||S_l^{TT}) = 0$ shows that S_o^{TT} and S_l^{TT} provide the same amount of information. Accordingly, using the definition of KL-Divergence [41], we first calculate $D_{KL}(S_o^{TT}||S_l^{TT})$, which can shortly be denoted by D_{KL}^{TT}, as

$$\begin{aligned} D_{KL}^{TT} &= \int_{-\infty}^{\infty} f(x;\lambda_o) log(\frac{f(x;\lambda_o)}{f(x;\lambda_l)})\, dx \\ &= \mathbb{E}_{f(x;\lambda_o)} \left[log(\frac{f(x;\lambda_o)}{f(x;\lambda_l)}) \right] \\ &= \mathbb{E}_{f(x;\lambda_o)} \left[log(\frac{\lambda_o}{\lambda_l}) - x(\lambda_o - \lambda_l) \right] \end{aligned} \tag{9}$$

where $f(x;\lambda_o)$ and $f(x;\lambda_l)$ denote the probability distribution functions of S_o^{TT} and S_l^{TT} respectively with parameters λ_o and λ_l. This leads to the result of

$$D_{KL}^{TT} = log(\frac{\lambda_o}{\lambda_l}) - \frac{(\lambda_o - \lambda_l)}{\lambda_o} \tag{10}$$

2) KL-Divergence for Packet Lengths: Similarly with transmission times, for the set of packet lengths, $D_{KL}(S_o^{PL}||S_l^{PL})$

is KL-Divergence from S_l^{PL} to S_o^{PL}, which is shortly denoted by D_{KL}^{PL}, and is calculated as

$$\begin{aligned} D_{KL}^{PL} &= \int_{-\infty}^{\infty} f(x;\mu_o) log(\frac{f(x;\mu_o)}{f(x;\mu_l)})\, dx \qquad (11) \\ &= \mathbb{E}_{f(x;\mu_o)} \big[log(\frac{f(x;\mu_o)}{f(x;\mu_l)}) \big] \\ &= \mathbb{E}_{f(x;\mu_o)} \big[log(\frac{\mu_o}{\mu_l}) - x(\mu_o - \mu_l) \big] \end{aligned}$$

where $f(x;\mu_o)$ and $f(x;\mu_l)$ denote the probability distribution functions of S_o^{PL} and S_l^{PL} respectively with parameters μ_o and μ_l. This results in:

$$D_{KL}^{PL} = log(\frac{\mu_o}{\mu_l}) - \frac{(\mu_o - \mu_l)}{\mu_o} \qquad (12)$$

3) Representativeness Factor based on Normalized KL-Divergence: For both transmission times and packet lengths, we now obtained the KL-Divergence between the set of observed packets and the set of packets learned; that is, we measure the representativeness of the packets that have already been used to train the IDS. However, the KL-Divergence cannot directly be used as a representativeness factor because of the following reasons: 1) It has no upper bound but the representativeness factor $C_{rep} \in [0,1]$. 2) KL-Divergence is decreasing function of the similarity between two sets but we need an increasing function of that as the name "representativeness" suggests. 3) This factor should be the combination of D_{KL}^{TT} and D_{KL}^{PL}.

Therefore, in order to obtain the representativeness factor, we first normalize each of D_{KL}^{TT} and D_{KL}^{PL} as

$$D_{KL-norm}^{TT} = e^{-D_{KL}^{TT}}, \qquad (13)$$

$$D_{KL-norm}^{PL} = e^{-D_{KL}^{PL}}. \qquad (14)$$

which solve the issues 1) and 2) stated above. Each of these normalized divergence measures can also be written in terms of only the traffic parameters:

$$\begin{aligned} D_{KL-norm}^{PL} &= e^{-\big[log(\frac{\lambda_o}{\lambda_l}) - \frac{(\lambda_o - \lambda_l)}{\lambda_o} \big]} \\ &= \Big[\frac{\lambda_l}{\lambda_o}\, e^{-\frac{(\lambda_l - \lambda_o)}{\lambda_o}} \Big] \qquad (15) \end{aligned}$$

Similarly,

$$\begin{aligned} D_{KL-norm}^{PL} &= e^{-\big[log(\frac{\mu_o}{\mu_l}) - \frac{(\mu_o - \mu_l)}{\mu_o} \big]} \\ &= \Big[\frac{\mu_l}{\mu_o}\, e^{-\frac{(\mu_l - \mu_o)}{\mu_o}} \Big] \qquad (16) \end{aligned}$$

Then, we combine $D_{KL-norm}^{TT}$ and $D_{KL-norm}^{PL}$ into the "representativeness factor" C_{rep} as

$$C_{rep} = c_1 D_{KL-norm}^{TT} + c_2 D_{KL-norm}^{PL} \qquad (17)$$

where $c_1 \leq 1$ and $c_2 \leq 1$ are positive constants that satisfy $c_1 + c_2 = 1$.

In order to weigh transmission times and packet lengths equally, we take $c_1 = c_2 = 0.5$. That is, we take their average:

$$\begin{aligned} C_{rep} &= \frac{1}{2}\big[D_{KL-norm}^{TT} + D_{KL-norm}^{PL} \big] \qquad (18) \\ &= \frac{1}{2}\big[e^{-D_{KL}^{TT}} + e^{-D_{KL}^{PL}} \big] \end{aligned}$$

We can rewrite C_{rep} only in terms of the traffic parameters using (15) and (16):

$$C_{rep} = \frac{1}{2}\Big[\frac{\lambda_l}{\lambda_o}\, e^{-\frac{(\lambda_l - \lambda_o)}{\lambda_o}} + \frac{\mu_l}{\mu_o}\, e^{-\frac{(\mu_l - \mu_o)}{\mu_o}} \Big] \qquad (19)$$

D. Generalization Ability of IDS

As stated above, we consider the generalization ability of the IDS as one of two factors that define the trustworthiness of intrusion decisions. To this end, the aim of this subsection is to determine the generalization ability of the IDS in simple terms to make its computation as easy as possible using the available measures during the execution of SSID. Accordingly, we start with the basic definition of generalization [42]:

$$\text{Generalization} \equiv \text{Data + Knowledge}$$

stating that the generalization depends on the "Data", which is denoted by Δ and refers to the adequacy of the packet samples that are used for learning, and the "Knowledge", which is denoted by κ and refers to the knowledge of the IDS obtained from packets learned. Therefore, we define the generalization factor C_{gen} as

$$C_{gen} = c_3\, \Delta + c_4\, \kappa \qquad (20)$$

where c_3 and c_4 are positive constants such that $c_3, c_4 \leq 1$, and $c_3 + c_4 = 1$.

1) Data Adequacy (Δ): We evaluate the adequacy of the packet samples that are used for learning with respect to the number of learnable parameters in the IDS. Although there is no hard rule for determining the adequacy of the learning data (i.e., the number of training samples required) for a given ML model, most studies have shown its relationship to the total number of learnable parameters in the model and taken the minimum number of required training samples as a multiple of the number of parameters [43].

Therefore, we first define the counterpart of Δ (namely the inadequacy of data), denoted by $\tilde{\Delta}$, as the ratio of the number of learnable parameters in the IDS to the total number of packet samples used for learning up to and including learning phase l:

$$\tilde{\Delta} = \min\Big(\frac{W}{\sum_{k=0}^{l} |B^k|}, 1 \Big) \qquad (21)$$

where $\sum_{k=0}^{l} |B^k|$ is the total number of packet samples that are sequentially used to learn model parameters until the end of learning phase l. Clearly, $\tilde{\Delta}$ takes value in $[0,1]$. While W is a constant number and $|B^l| \geq 1$ for any learning phase l (in which a learning is performed), $\lim_{l\to\infty}(\tilde{\Delta}) = 0$. In addition, $\tilde{\Delta} = 1$ when $\sum_{k=0}^{l} |B^k| \leq W$.

We can then define the adequacy of learning data as

$$\Delta = 1 - \tilde{\Delta} = \Big[1 - \min\Big(\frac{W}{\sum_{k=0}^{l} |B^k|}, 1 \Big) \Big] \qquad (22)$$

As a result ($\lim_{l\to\infty}(\Delta) = 1$), and as expected, the adequacy of the aggregated data consisting of packet samples used in the learning stages increases over time. Also, recall and note that we already include the representativeness of these packet samples directly in the trust coefficient of the IDS.

2) Knowledge (κ): We consider knowledge to be the measure of the ML model's expected performance for the upcoming traffic packets. Subsequently, we measure the knowledge (i.e. expected performance) of the IDS based on its performance on the packet samples used for learning and on the online available validation data.

To this end, in this paper, we consider the worst-case scenario when there is no validation data available. Let $\mathcal{E}(l)$ denote the empirical error measured at the end of learning phase l on both packet samples learned and validation data (if available), such that $0 \leq \mathcal{E}(l) \leq 1$. We then define the knowledge κ as the counterpart of the exponentially weighted moving average of empirical errors for all learning phases up to and including the l–th phase:

$$\kappa = 1 - \sum_{k=0}^{l} (\frac{1}{2})^{(l-k+1)}\, \mathcal{E}(k) \tag{23}$$

where the multiplier is set as $1/2$ to keep the value of κ in $[0, 1]$. That is, if the empirical training error decreases with the successive learning phases (i.e. $\mathcal{E}(l)$ is the decreasing function of l), the knowledge of the IDS increases converging to its maximum.

In practice, at the end of each learning phase l, κ can easily be updated using only its previous value and the empirical error $\mathcal{E}(l)$ as

$$\kappa \leftarrow \frac{1}{2} - \frac{1}{2}\Big[\mathcal{E}(l) - \kappa\Big] \tag{24}$$

3) Generalization Factor: We now easily calculate the generalization factor C_{gen} combining the "data adequacy" Δ (22) and "knowledge" κ (23) using the definition of the generalization factor (20):

$$C_{gen} = \tag{25}$$
$$c_3\Big[1 - \min(\frac{W}{\sum_{k=0}^{l}|B^k|}, 1)\Big] + c_4\Big[1 - \sum_{k=0}^{l} (\frac{1}{2})^{(l-k+1)}\, \mathcal{E}(k)\Big]$$

We particularly set $c_3 = c_4 = 0.5$ representing that the data and knowledge are equally important for generalization:

$$C_{gen} = \tag{26}$$
$$1 - \frac{\min\big(W/\sum_{k=0}^{l}|B^k|, 1\big) + \sum_{k=0}^{l}(1/2)^{(l-k+1)}\,\mathcal{E}(k)}{2}$$

V. Results

We now evaluate the performance of SSID framework for two different intrusion detection tasks to identify malicious traffic packets and compromised devices.

A. IDS Used in the SSID Framework

We first present the structure of IDS that we used within the SSID framework during the performance evaluation. This particular IDS structure is displayed in Figure 4, which is mainly comprised of an ML model and a decision maker component. The input of this IDS is the vector of M network traffic metrics, $x_i = [x_i^m, \dots, x_i^m, \dots, x_i^M]$, and the output, y_i is a decision indicating the probability of intrusion corresponding to current traffic.

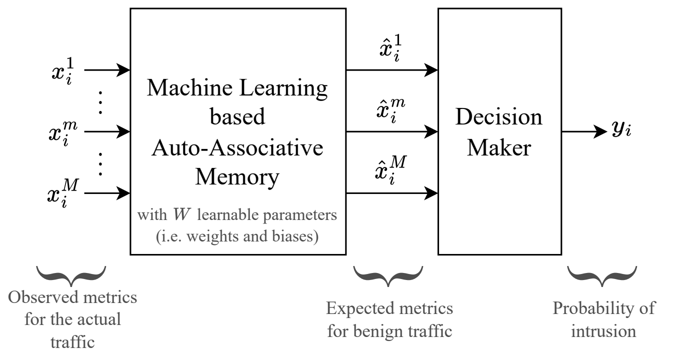

x_i^1
x_i^m
x_i^M
Machine Learning based Auto-Associative Memory
with W learnable parameters (i.e. weights and biases)
$\hat{x}_i^1$
$\hat{x}_i^m$
$\hat{x}_i^M$
Decision Maker
y_i
Observed metrics for the actual traffic
Expected metrics for benign traffic
Probability of intrusion

Fig. 4. Particular structure of IDS used within the SSID framework during performance evaluation

An IDS structure that can learn only from normal traffic when no attack traffic is available in the network may provide higher performance under self-supervised learning. Therefore, in the IDS structure shown in Figure 4, the ML model is used to create an Auto-Associative Memory (AAM) that is used to reconstruct benign traffic metrics – which are the expected metrics according to the norm of the actual traffic learned by the AAM – from observed metrics, which may be the indicators of malicious traffic. The vector of expected metrics, which is the output of ML-based AAM, for packet i is denoted by $\hat{x}_i = [\hat{x}_i^1, \dots, \hat{x}_i^m, \dots, \hat{x}_i^M]$. In order words, the ML-based AAM is a learned function that maps the noisy or disordered metrics to the normal metrics, i.e. $f_{aam} : x_i \mapsto f_{aam}(x_i)$ for $f_{aam}(x_i) = \hat{x}_i$, so that $f_{aam} : [0,1]^M \to [0,1]^M$.

Within this IDS, based on the output of ML-based AAM, the decision maker measures the deviation of the actual metrics x_i from the expected metrics $\hat{x}_i$. The main criterion for this decision maker is that it requires no human intervention or parameter settings based on offline data.

Furthermore, since we use an anomaly-based algorithm that learns only the benign traffic, we take the learning error as the mean of estimated attack probabilities for packets in learning batch B^l:

$$\mathcal{E}(l) = \frac{1}{|B^l|}\sum_{i=1}^{|B^l|} y_i. \tag{27}$$

We should note that since the purpose of this section is to evaluate the impact of the SSID framework on the performance of an IDS, we use well-known and state-of-the-art methods for the specific implementation of the IDS as they are. These methods are mainly based on two ML models, Deep Random Neural Network (DRNN) and Multi-Layer Perceptron (MLP).

1) Traffic Metrics and Decision Maker: **For malicious traffic detection**, we implement the metrics and the simple decision maker of the IDS developed in [44]. Accordingly, in order to capture the signatures of Botnet attacks especially Mirai, as the inputs of AAM, we use $M = 3$ normalized metrics that measure the total size and average inter-transmission times of the last 500 packets and the number of packets in the last 100 seconds. In addition, the output of the IDS is calculated as

$$y_i = \frac{1}{M} \sum_{m=1}^{M} |x_i^m - \hat{x}_i^m|, \tag{28}$$

For compromised device identification, we implement CDIS developed in [8] for each IP address in the considered network. This CDIS works in time windows of length 10 seconds. In each time window, there are $M = 6$ normalized metrics as the inputs of the CDIS implemented for IP address i, which measure the average size and number of packets received from a single source, maximum size and number of packets received from any single source, and total size and number of packets transmitted in this window. Then, for each window, the output of CIDS is calculated as

$$y_i = \max_{m \in \{1,\dots,M\}} (|x_i^m - \hat{x}_i^m|) \tag{29}$$

2) AAM Using the Deep Random Neural Network: The IDS architecture shown in Figure 4 uses the DRNN [45], [46], which is an extension of the RNN [47], [48] with dense feedback loops between clustered neuronal somas, and an overall feed-forward structure between layers of dense clusters as the ML model. Due to its auto-associative learming we call it the Auto-Associative DRNN (AADRNN) based AAM. Earlier research has shown that a DRNN-based IDS has a lightweight architecture and offers high accuracy when used with unsupervised auto-assiciative training with normal (benign) traffic [7], [8], [44], [49]–[51]. The DRNN-based IDS was also evaluated with offline [44], incremental [7] and sequential [8] learning to detect malicious traffic and compromised devices during Botnet attacks. G-Networks [52], which generalize the RNN, and the simple RNN itself were also used with offline learning to detect zero-day [51] and SYN DoS [53] attacks.

In the DRNN-based IDS, we use a DRNN model consists of M layers. The hidden layers (the first $M - 1$ layers) contains M neural clusters, the output layer is comprised of M linear neurons. Each DRNN cluster has the following activation function, which is unique to DRNN model:

$$\begin{aligned} \zeta(\Lambda) &= \frac{p\,(r + \lambda^+) + \lambda^- + \Lambda}{2\,[\lambda^- + \Lambda]} \\ &- \sqrt{\Big(\frac{p\,(r + \lambda^+) + \lambda^- + \Lambda}{2\,[\lambda^- + \Lambda]}\Big)^2 - \frac{\lambda^+}{\lambda^- + \Lambda}}\,, \end{aligned} \tag{30}$$

where Λ is the input of the given cluster, p is the probability that any neuron received trigger transmits a trigger to some other neuron, and λ^+ and λ^- are respectively the rates of external Poisson flows of excitatory and inhibitory input spikes to any neuron.

At any learning stage l, we perform learning using the packet samples collected in B_l. For the particular structure of IDS, during the initial learning stage of SSID (i.e. $l = 0$), we create an AAM from the DRNN model trained via learning algorithm, which has been developed in [45], [46] and used for IDS in [44]. During the online learning stage of SSID, we use the specifically designed algorithms for each of the AADRNN-based IDS and CDIS. These are respectively incremental learning algorithm for IDS presented in [7] and sequential learning algorithm for CDIS presented in [8].

3) AAM using Multi-Layer Perceptron: During the performance evaluation of the SSID framework, we also use MLP, which is one of the most popular feed-forward neural networks used for various tasks such as signal processing, forecasting, anomaly detection, etc. As also reviewed in Section II, various works [16], [19], [20], [27] used MLP to develop different IDS methods.

Similar to the DRNN, the MLP model that we use is also comprised of M layers with M neurons each. Each neuron has sigmoid activation function as

$$\zeta(\Lambda) = \frac{1}{1 + e^{-\Lambda}}, \tag{31}$$

where Λ is an input to the neural activation.

In both the initial and online learning stages, the parameters of the MLP are updated using the state-of-the-art optimizer Adam [54]. In each online learning phase, incremental learning is applied by starting parameter optimization from the connection weight values already in use at the beginning of this phase.

B. Parameter Settings for SSID

We set the parameters of SSID as follows: $\Theta = 0.95$, $I = 10$, $K = 100$, and $\gamma = 0.25$. That is, SSID aims to keep the trust in the IDS above 0.95 while it considers a packet as malicious if the output of the IDS is above 0.25. In addition, we want to update parameters using at least $K = 100$ packets for computational efficiency.

C. Performance Evaluation for Malicious Traffic Detection

We first evaluate the performance of SSID for malicious traffic detection during Mirai Botnet attack. To this end, we use the well-known **Kitsune** dataset [55], [56], which contains $764,137$ packet transmissions of both normal and attack traffic cover a consecutive time period of roughly 7137 seconds.

Figure 5 displays the ROC curve, where the x-axis of this figure is plotted in logarithmic scale. We see that AADRNN-based IDS trained under our novel SSID framework achieves significantly high TPR above 0.995 even for very low FPR about 10^{-5}.

In more detail, in Figure 6, we present the predictions and Γ of SSID with respect to time. This figure reveals an important fact that while the IDS is completely indecisive at the beginning, SSID framework enables it to learn the normal traffic very quickly. As a result, SSID makes significantly low false alarms although it learns – fully online – during real-time operation based only on its own decision using

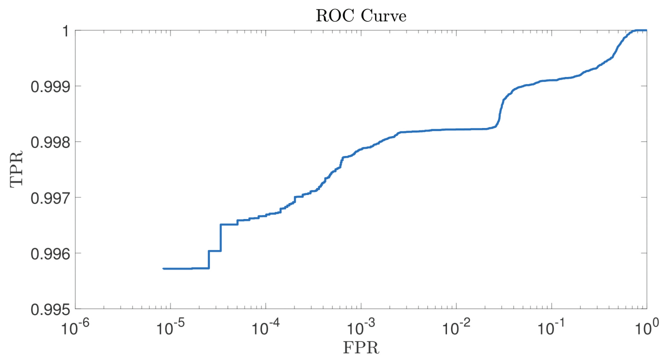


Fig. 5. ROC curve for the performance of AADRNN-based IDS under the SSID framework for malicious traffic detection

no external (offline collected) dataset. We also see that Γ accurately reflects the trustworthiness of decisions made by AADRNN. In addition, although Γ slightly decreases as a result of random packet selection, especially after attack starts, the parameters of AADRNN are not updated by SSID as the traffic is detected as malicious.

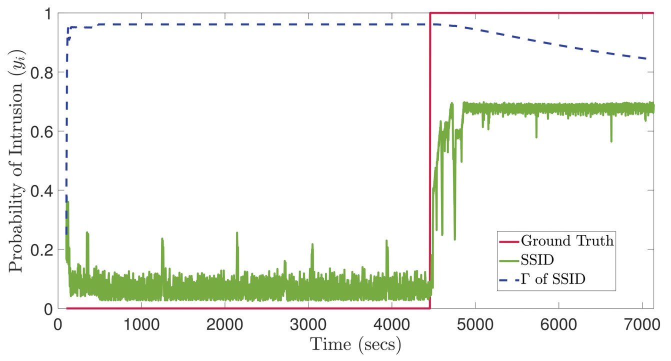


Fig. 6. Predictions of SSID and the value of trust coefficient Γ with respect to time

1) Comparison with Incremental and Offline Learning: We further compare the performance of AADRNN under SSID with the performance of AADRNN with incremental and offline learning. All methods with offline learning are trained using the first $83,000$ benign traffic packets while the AADRNN with incremental learning is trained periodically for the window of 750 packets using AADRNN's own decision, where the first 750 packets received are assumed to be normal packets during the cold-start of the network.

Figure 7 displays the performancs of SSID and the AADRNN, with incremental and offline learning. The results in this figure first reveal that the fulyy online trained AADRNN using the SSID framework achieves competitive results with the AADRNN which is trained offline using approximately $83,000$ packets. We also see that the SSID significantly outperforms the AADRNN with incremental learning with respect to all performance metrics. Also note that the SSID learned from a total of $4,161$ packets while also conducting real-time attack detection.

In contrast with offline and incremental learning, SSID framework assumes only that the first packet is known to be

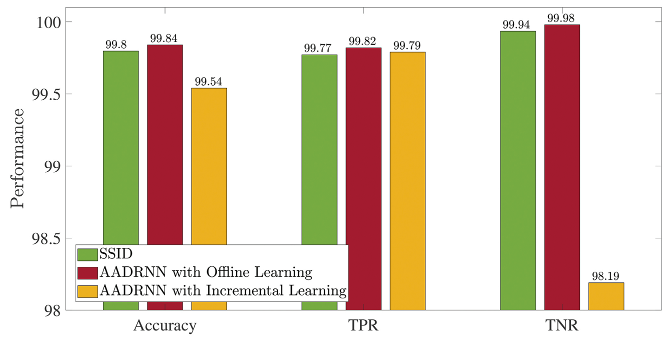


Fig. 7. Performance comparison between the AADRNN under SSID and the AADRNN with incremental and offline learning

benign so the duration of cold-start equals the transmission of a single traffic packet. That is, using no offline dataset or requiring no cold-start, the SSID framework is able train an ML-based IDS to achieve considerably high performance which is highly competitive against the ML models trained on significantly large dataset.

2) A Different ML Model – MLP – under the SSID Framework: In order to further analyze the impact of the proposed SSID framework on the performance of a different ML model, we evaluate the performance of the well-known MLP under the SSID framework (called SSID-MLP) and compare it with the performance of MLP with offline and incremental learning, respectively. The results of this performance evaluation is presented in Figure 8.

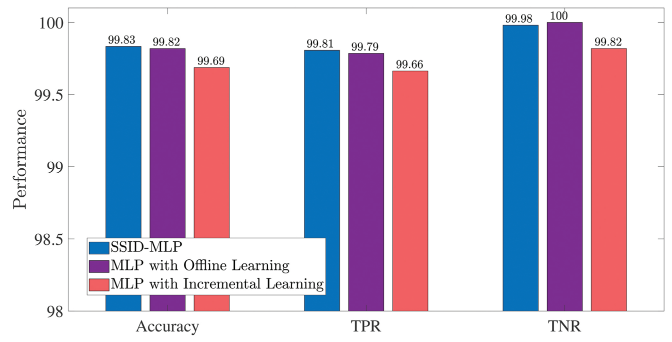


Fig. 8. Performance comparison between the SSID-MLP and the MLP with incremental and offline learning

Figure 8 shows that SSID-MLP achieves slightly higher Accuracy and TPR than MLP with offline learning, although MLP with offline learning raises no false alarms at all (i.e. with 100% TNR). Moreover, we see that SSID-MLP significantly outperforms the MLP model that is also trained via incremental learning periodically for every 750 packets based on its own output.

3) Comparison of Different ML Models: We further compare the performance of AADRNN under SSID (called SSID-AADRNN for clarity) and SSID-MLP with those of some well-known ML models, including KNN and Lasso with offline learning. Figure 9 displays the performance of all

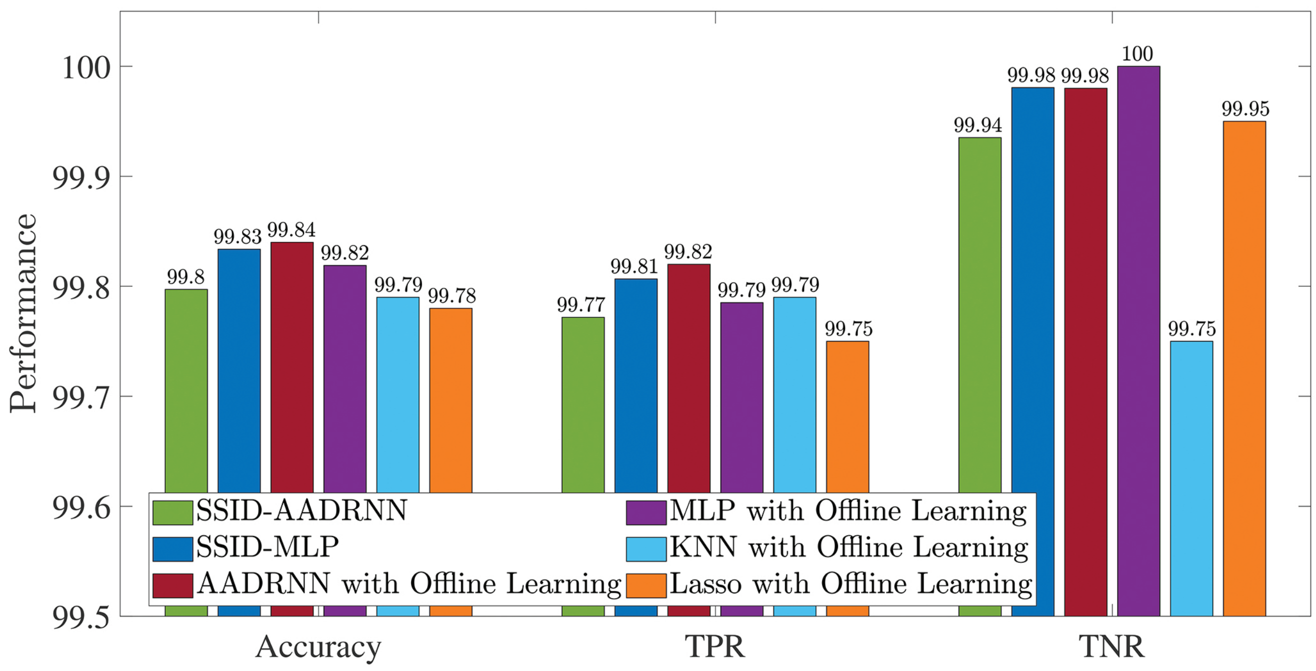


Fig. 9. Performance comparison between the ML models under the SSID framework and those with offline learning

compared models with respect to Accuracy, TPR and TNR. The results in this figure show that SSID-MLP achieves the second-best performance with respect to all performance metrics. In addition, both SSID-MLP and SSID-AADRNN achieve highly competitive results with the offline trained ML models, while the SSID framework completely eliminates the need for data collection and labeling.

D. Performance Evaluation for Compromised Device Identification

We now evaluate the performance of CDIS [8] under the SSID framework, in short SSID-CDIS, on six (6) different attacks, from the two (2) distinct datasets Kitsune [56] and Bot-IoT [57]. For each dataset, the performance of SSID-CDIS is compared with the original CDIS technique with sequential learning. Using the same methods as in [8], compromised device identification is performed for a 10 seconds long time window. The performance is evlaued using the Balanced Accuracy [58].

Figure 10 displays the performance of SSID-CDIS and its comparison with CDIS to identify compromised IP addresses, for each of the Mirai Botnet and SYN DoS attacks in the Kitsune dataset. Specifically it shows that while the SSID framework provides the same performance as sequential learning to identify compromised devices during a Mirai Botnet attack, it significantly improves the overall performance of CDIS during a SYN DoS attack. The box plot on the right of Figure 10 shows that SSID-CDIS achieves 100% median balanced accuracy when there is only one outlier IP address with around 85% accuracy. On the other hand, the sequentially

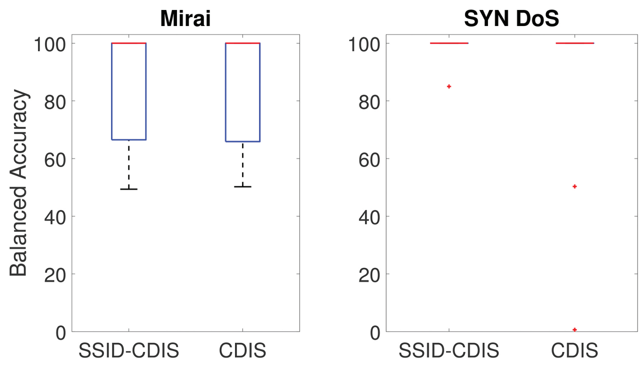


Fig. 10. Performance comparison of the CDIS trained under the SSID framework with that under sequential learning on Kitsune dataset

trained CDIS has two outlier IP addresses with performances of 50% and 1%, respectively.

Figure 11 exhibits the performance of the SSID-CDIS system, and compares it with CDIS, to identify compromised IP addresses during DDoS and DoS attacks, using different communication protocols available in the Bot-IoT dataset. These results show that the SSID framework achieves higher identification performance, as compared to the use of CDIS sequential learning for the majority of attack types.

Starting with the box plot displayed at the far left of this figure, we observe the following results:

1) For the DDoS HTTP attack, the overall performance is almost the same for SSID and CDIS with sequential learning. However, as expected, performance varies slightly for individual IP addresses.

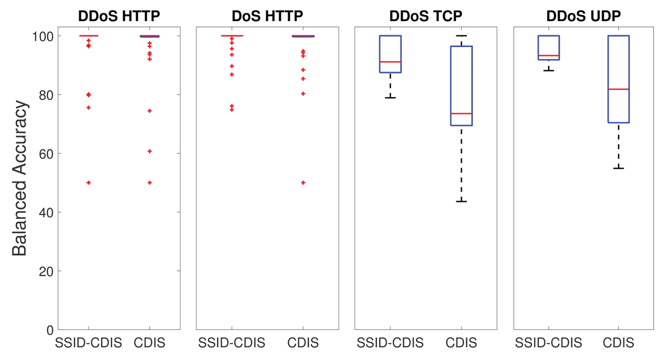


Fig. 11. Performance comparison of the CDIS trained under the SSID framework with that under sequential learning on Bot-IoT dataset

2) For the DoS HTTP attack, using SSID improved the performance by 2% on average with a minimum of 75% balanced accuracy.
3) For the DDoS TCP attack, SSID significantly improved the median accuracy by 18%, where SSID-CDIS achieves 91% median accuracy. In addition, while the balance accuracy of CDIS with sequential learning is below 80% (with a minimum of 49%) for 9 out of 13 unique IP addresses, the balance accuracy of SSID-CDIS is equal to 79% for only 2 IP addresses and above 80% for the rest.
4) Similar to the results for the DDoS TCP attack, SSID was seen to provide significant performance improvement to identify compromised devices during a DDoS UDP attack. The median accuracy increased by 11%, achieving above 88% balanced accuracy for all IP addresses.

VI. Conclusions and Further Work

This paper has proposed a novel Self-Supervised Intrusion Detection (SSID) framework which is designed to train any given IDS (whose parameters are calculated using the network traffic) **fully online** with no need for human intervention or prior offline training. The SSID framework comprises two successive learning stages, namely initial learning and online learning. Initial learning aims to quickly adapt the IDS parameters to the network where the IDS is deployed. Online learning aims to update the parameters whenever it is required to ensure high detection accuracy of the IDS.

During the real-time operation of the IDS, in parallel to detection, the SSID framework performs the following main tasks:

- It continually estimates the trustworthiness of intrusion decisions to identify normal and malicious traffic. It also measures the ability of the IDS to learn and generalize from data provided by SSID and the extent to which this data can represent the current online network traffic patterns.
- In order to provide training data for the IDS, the SSID framework selects and labels network traffic packets in a self-supervised manner based only on the decisions of IDS, and on the trust of SSID with regard to those decisions.
- The SSID framework determines when the IDS parameters need to be updated, based on the trustworthiness of the IDS, the selected training packets, and the latest state of network security.

Thus the proposed SSID framework eliminates the need for offline data collection, it prevents human errors in data labeling avoiding labor and computational costs for model training and data collection through experiments. Its most important advantage is in terms of performance, and it enables IDS to easily adapt to the time varying characteristics of the network traffic.

We also evaluated the performance of the SSID framework for two tasks: malicious traffic detection and compromised device identification to enhance the security of an IoT network. For malicious traffic detection, two different ML models, DRNN and MLP, have been deployed with the SSID framework and tested on the Kitsune dataset. The results we obtain reveal that the ML models trained under the SSID framework without offline training also achieve considerably high performance compared to the same models with offline and incremental learning.

For compromised device identification, the performance of the state-of-the-art CDIS has been tested under sequential learning and the SSID framework on data from six (6) different cyberattacks provided by the two publically available Kitsune and Bot-IoT datasets. The results show that the use of SSID significantly improves the performance of CDIS for the majority of cases considered.

Future work will evaluate the use of SSID for adapting a pre-trained IDS for use across different networks whose traffic has not be learned *a priori*, which seems to be a promising approach for fast, self-supervised, and successful adaptation of the IDS parameters of various networks. It would also be interesting to examine security assurance methods targeting distributed systems that combine the SSID framework with Federated Learning and attack prevention or mitigation algorithms. It seems that a successful integration of the SSID framework with Federated Learning may provide secure, distributed and self-supervised online learning for collaborative systems.

References

[1] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: classification and state-of-the-art," *Computer networks*, no. 5, pp. 643–666, 2004.

[2] D. Goodin, "100,000-strong Botnet built on router 0-day could strike at any time," *Ars Technica*, December 2017. [Online]. Available: https://arstechnica.com/information-technology/2017/12/100000-strong-botnet-built-on-router-0-day-could-strike-at-any-time/

[3] Imperva, "2022 Imperva Bad Bot Report," p. 1–37, 2022. [Online]. Available: https://www.imperva.com/resources/resource-library/reports/bad-bot-report/

[4] 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," *arXiv preprint arXiv:2104.09041*, 2021.

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

[6] H. Liu and B. Lang, "Machine learning and deep learning methods for intrusion detection systems: A survey," *applied sciences*, vol. 9, no. 20, p. 4396, 2019.

[7] M. Nakip and E. Gelenbe, "Botnet attack detection with incremental online learning," in *Security in Computer and Information Sciences: Second International Symposium, EuroCybersec 2021, Nice, France, October 25–26, 2021, Revised Selected Papers*. Springer, 2022, pp. 51–60.

[8] E. Gelenbe and M. Nakıp, "Traffic based sequential learning during botnet attacks to identify compromised IoT devices," *IEEE Access*, vol. 10, pp. 126 536–126 549, 2022.

[9] H. M. Song and H. K. Kim, "Self-supervised anomaly detection for in-vehicle network using noised pseudo normal data," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 2, pp. 1098–1108, 2021.

[10] Z. Wang, Z. Li, J. Wang, and D. Li, "Network intrusion detection model based on improved byol self-supervised learning," *Security and Communication Networks*, vol. 2021, pp. 1–23, 2021.

[11] X. Zhang, J. Mu, X. Zhang, H. Liu, L. Zong, and Y. Li, "Deep anomaly detection with self-supervised learning and adversarial training," *Pattern Recognition*, vol. 121, p. 108234, 2022.

[12] H. Kye, M. Kim, and M. Kwon, "Hierarchical detection of network anomalies: A self-supervised learning approach," *IEEE Signal Processing Letters*, vol. 29, pp. 1908–1912, 2022.

[13] E. Caville, W. W. Lo, S. Layeghy, and M. Portmann, "Anomal-e: A self-supervised network intrusion detection system based on graph neural networks," *Knowledge-Based Systems*, vol. 258, p. 110030, 2022.

[14] M. Abououf, R. Mizouni, S. Singh, H. Otrok, and E. Damiani, "Self-supervised online and lightweight anomaly and event detection for iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 24, pp. 25 285–25 299, 2022.

[15] W. Wang, S. Jian, Y. Tan, Q. Wu, and C. Huang, "Robust unsupervised network intrusion detection with self-supervised masked context reconstruction," *Computers & Security*, vol. 128, p. 103131, 2023.

[16] T. A. Tuan, H. V. Long, R. Kumar, I. Priyadarshini, N. T. K. Son *et al.*, "Performance evaluation of botnet ddos attack detection using machine learning," *Evolutionary Intelligence*, pp. 1–12, 2019.

[17] Z. Shao, S. Yuan, and Y. Wang, "Adaptive online learning for IoT botnet detection," *Information Sciences*, vol. 574, pp. 84–95, 2021.

[18] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani, "Corrauc: A malicious bot-iot traffic detection method in IoT network using machine-learning techniques," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3242–3254, 2021.

[19] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.

[20] I. Letteri, M. Del Rosso, P. Caianiello, and D. Cassioli, "Performance of botnet detection by neural networks in software-defined networks," in *ITASEC*, 2018.

[21] M. Banerjee and S. Samantaray, "Network traffic analysis based IoT botnet detection using honeynet data applying classification techniques," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 17, no. 8, 2019.

[22] C. D. McDermott, F. Majdani, and A. V. Petrovski, "Botnet detection in the Internet of Things using deep learning approaches," in *2018 international joint conference on neural networks (IJCNN)*. IEEE, 2018, pp. 1–8.

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

[24] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[25] C. S. Htwe, Y. M. Thant, and M. M. S. Thwin, "Botnets attack detection using machine learning approach for IoT environment," in *Journal of Physics: Conference Series*, vol. 1646, no. 1. IOP Publishing, 2020, p. 012101.

[26] S. Sriram, R. Vinayakumar, M. Alazab, and K. Soman, "Network flow based IoT botnet attack detection using deep learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2020, pp. 189–194.

[27] 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, 2020.

[28] G. D. L. T. Parra, P. Rad, K.-K. R. Choo, and N. Beebe, "Detecting Internet of Things attacks using distributed deep learning," *Journal of Network and Computer Applications*, vol. 163, p. 102662, 2020.

[29] J. Liu, S. Liu, and S. Zhang, "Detection of IoT botnet based on deep learning," in *2019 Chinese Control Conference (CCC)*. IEEE, 2019, pp. 8381–8385.

[30] A. Kumar and T. J. Lim, "Early detection of mirai-like iot bots in large-scale networks through sub-sampled packet traffic analysis," in *Future of Information and Communication Conference*. Springer, 2019, pp. 847–867.

[31] M. Chatterjee, A. S. Namin, and P. Datta, "Evidence fusion for malicious bot detection in iot," in *2018 IEEE International Conference on Big Data (Big Data)*, 2018, pp. 4545–4548.

[32] 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 *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 756–767.

[33] N. V. Abhishek, T. J. Lim, B. Sikdar, and A. Tandon, "An intrusion detection system for detecting compromised gateways in clustered iot networks," in *2018 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. IEEE, 2018, pp. 1–6.

[34] M. Taneja, "An analytics framework to detect compromised iot devices using mobility behavior," in *2013 International Conference on ICT Convergence (ICTC)*, 2013, pp. 38–43.

[35] A. O. Prokofiev, Y. S. Smirnova, and V. A. Surov, "A method to detect internet of things botnets," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*. IEEE, 2018, pp. 105–108.

[36] T. N. Nguyen, Q.-D. Ngo, H.-T. Nguyen, and G. L. Nguyen, "An advanced computing approach for iot-botnet detection in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 8298–8306, 2022.

[37] A. Hristov and R. Trifonov, "A model for identification of compromised devices as a result of cyberattack on iot devices," in *2021 International Conference on Information Technologies (InfoTech)*, 2021, pp. 1–4.

[38] T. Trajanovski and N. Zhang, "An automated and comprehensive framework for iot botnet detection and analysis (iot-bda)," *IEEE Access*, vol. 9, pp. 124 360–124 383, 2021.

[39] H. Bahşi, S. Nõmm, and F. B. La Torre, "Dimensionality reduction for machine learning based iot botnet detection," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2018, pp. 1857–1862.

[40] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in neural information processing systems*, vol. 33, pp. 21 271–21 284, 2020.

[41] S. Kullback and R. A. Leibler, "On information and sufficiency," *The annals of mathematical statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[42] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," *Advanced Lectures on Machine Learning: ML Summer Schools 2003, Canberra, Australia, February 2-14, 2003, Tübingen, Germany, August 4-16, 2003, Revised Lectures*, pp. 169–207, 2004.

[43] A. Alwosheel, S. van Cranenburgh, and C. G. Chorus, "Is your dataset big enough? sample size requirements when using artificial neural networks for discrete choice analysis," *Journal of choice modelling*, vol. 28, pp. 167–182, 2018.

[44] M. Nakip and E. Gelenbe, "Mirai botnet attack detection with auto-associative dense random neural network," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 01–06.

[45] E. Gelenbe and Y. Yin, "Deep learning with random neural networks: I," in *2016 International Joint Conference on Neural Networks (IJCNN)*, 2016, pp. 1633–1638.

[46] E. Gelenbe and Y. Yin, "Deep learning with dense random neural networks: II," in *International Conference on Man–Machine Interactions*. Springer, 2017, pp. 3–18.

[47] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, no. 4, pp. 502–510, 1989.

[48] ——, "Learning in the recurrent random neural network," *Neural Computation*, vol. 5, no. 1, pp. 154–164, 1993.

[49] O. Brun et al., "Deep learning with dense random neural networks for detecting attacks against IoT-connected home environments: I," in *International ISCIS Cyber-Security Workshop*. Springer, Cham, 2018, pp. 79–89.

[50] O. Brun, Y. Yin, and E. Gelenbe, "Deep learning with dense random neural network for detecting attacks against iot-connected home environments: II," *Procedia Computer Science*, vol. 134, pp. 458–463, 2018.

[51] E. Gelenbe and M. Nakıp, “G-networks can detect different types of cyberattacks,” in *2022 30th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. IEEE, 2022, pp. 9–16.

[52] E. Gelenbe, “G-networks: a unifying model for neural and queueing networks,” *Annals of Operations Research*, vol. 48, no. 5, pp. 433–461, 1994.

[53] S. Evmorfos et al., “Neural network architectures for the detection of syn flood attacks in IoT systems,” in *Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, 2020, pp. 1–4.

[54] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[55] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: An ensemble of autoencoders for online network intrusion detection,” in *The Network and Distributed System Security Symposium (NDSS) 2018*, 2018.

[56] “Kitsune Network Attack Dataset,” August 2020. [Online]. Available: https://www.kaggle.com/ymirsky/network-attack-dataset-kitsune

[57] 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 Generation Computer Systems*, vol. 100, pp. 779–796, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18327687

[58] K. H. Brodersen, C. S. Ong, K. E. Stephan, and J. M. Buhmann, “The balanced accuracy and its posterior distribution,” in *2010 20th international conference on pattern recognition*. IEEE, 2010, pp. 3121–3124.