INSTYTUT INFORMATYKI TEORETYCZNEJ I STOSOWANEJ
POLSKIEJ AKADEMII NAUK

Samo-Nadzorujące się Uczenie w Czasie Rzeczywistym dla Wykrywania Włamań w Bezpiecznym Internecie Rzeczy

MERT NAKIP

Rozprawa Doktorska przygotowana pod kierunkiem
PROF. DR. EROLA GELENBE

Gliwice, Polska
2023

Institute of Theoretical and Applied Informatics
Polish Academy of Sciences

Online Self-Supervised Learning Intrusion Detection Towards Secure Internet of Things

MERT NAKIP

Doctoral Dissertation prepared under the supervision of
PROF. DR. EROL GELENBE

Gliwice, Poland
2023

Abstract

Secure Internet of Things (IoT) systems are extremely difficult to achieve as most IoT devices are low-cost and low-maintenance devices with low computing power and human intervention to run complex security methods. Therefore, lightweight data-driven, especially Machine Learning (ML)-based, security methods have been developed particularly for IoT systems. On the other hand, these methods often learn offline from large data collected through extensive simulations, which may be time consuming and provide biased (misleading) data. This thesis investigates the open issues and ways to enable fully online and lightweight learning for ML-based intrusion detection paving the way towards secure IoT.

We first develop an Intrusion Detection System (IDS) that learns the normal traffic patterns of the IoT network and detects both malicious network traffic packets and compromised devices. This IDS is based on Deep Random Neural Network (DRNN) model combined with originally proposed traffic metrics and Statistical Whisker based Benign Classifier (SWBC). For each of malicious traffic detection and compromised device identification, we propose a set of original network traffic metrics that enable accurate recognition of Botnet traffic patterns and footprints of the attacker. In addition, we develop a new SWBC algorithm to classify traffic packets as benign and malicious by learning the classification criterion based on the DRNN outputs on the training data. We further present offline and quasi-online (incremental and sequential) learning algorithms for our IDS.

Subsequently, we evaluate the performance of our IDS with both offline and quasi-online learning algorithms for Botnet DDoS, DoS, and zero-day attacks on three public datasets. The results show the superior performance of our IDS with low computation time compared to well-known ML models. The results also reveal the potential of online learning for intrusion detection.

Finally, in order to enable fully online learning of ML-based IDS requiring no human intervention, we propose the novel Self-Supervised Intrusion Detection (SSID) framework. For the learning of utilized IDS, the SSID framework collects and labels traffic packets based only on the decisions of the IDS and their statistically measured trustworthiness. The SSID framework enables IDS to adapt time-varying characteristics of the network traffic quickly, eliminates the need for offline data collection, prevents human errors in data labeling, and avoids labor costs for model training and data collection through experiments. Therefore – as the experimental results on public datasets for malicious traffic and compromised device detection using well-known ML models also suggest – SSID is very useful and advantageous to develop an online learning ML-based IDS for IoT systems.

Acknowledgements

First and foremost, I would like to express my deepest gratitude and utmost respect to my supervisor, Prof. Erol Gelenbe. He has consistently provided unwavering support, invaluable guidance, and profound mentorship during my doctoral studies, and I have cherished every moment of collaborating with him, considering it a great honor. His advice and mentorship have also significantly contributed to my personal and professional growth. Furthermore, I would like to acknowledge that he has consistently been a true inspiration and role model, both as a wonderful person and a highly respected distinguished scientist.

I would also like to especially thank Prof. Tadeusz Czachórski for his endless support during my adaptation to my new work and social life in Poland and during my doctoral studies. I would like to extend my heartfelt appreciation to Prof. Cüneyt Güzeliş, Baran Can Gül, Onur Çopur and Godlove Kuaban for their friendship, delightful technical discussions, and exceptional cooperation. In addition, I would like to thank all my colleagues at IITiS, PAN for their warm smiles and helpfulness.

Note that I gratefully acknowledge that a major part of this thesis was supported by the European Commission H2020 Program through the IoTAC (Security By Design IoT Development and Certificate Framework with Front-end Access Control) Research and Innovation Action, under Grant Agreement No. 952684. The opinions expressed in this thesis are those of the author and do not necessarily reflect the views of the European Commission.

Also, I would like to thank my parents, Mehmet and Emine Nakıp, for the sacrifices they made for my education and personal development and for their endless support in all my choices. I would also like to express my endless love to my siblings, Melisa Nakıp and Melih Nakıp. I would also like to extend my gratitude to my family-in-law, whose support I have always felt.

Last but certainly not least, I want to convey my heartfelt and deepest gratitude to my beloved wife, Gizem Güler Nakıp for her endless love and support, and for being my companion through both the best and most challenging moments of the past nine years of my life.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

List of abbreviations (in alphabetic order)

Abbreviation	Definition
AADRNN	Auto-Associative Deep Random Neural Network
AAM	Auto-Associative Memory
ANOVA	Analysis of Variance
CDIS	Compromised Device Identification System
CNN	Convolutional Neural Network
DARPA	Defense Advanced Research Projects Agency
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
DRNN	Deep Random Neural Network
DT	Decision Tree
ELM	Extreme Learning Machine
FISTA	Fast Iterative Shrinkage-Thresholding Algorithm
IDS	Intrusion Detection System
IoT	Internet of Things
IP	Internet Protocol
ISSL	Incremental Semi-Supervised Learning
KL	Kullback-Leibler
KNN	K-Nearest Neighbour
Lasso	Least Absolute Shrinkage and Selector Operator
LR	Linear Regression
LSTM	Long-Short Term Memory
M2M	Machine-to-Machine
MitM	Man-in-the-Middle
ML	Machine Learning
MLP	Multi-Layer Perceptron
NB	Naive Bayes
RF	Random Forest
RNN	Random Neural Network
ROC	Receiver Operating Characteristic

(continue) List of abbreviations (in alphabetic order)

Abbreviation	Definition
SSID	Self-Supervised Intrusion Detection
SSID-AADRNN	Auto-Associative Deep Random Neural Network under the SSID framework
SSID-MLP	Multi-Layer Perceptron under the SSID framework
SVM	Support Vector Machine
SVM-OCC	Support Vector Machine - One Class Classifier
SWBC	Statistical Whisker based Benign Classifier
TNR	True Negative Rate
TPR	True Positive Rate

Chapter 4 (Section 4.2): IDS for malicious traffic detection

Symbol	Definition: i indicates the packet index since IDS has been started
H	The number of DRNN layers
C_h	The number of neuron clusters in layer h of DRNN
$N_{(c,h)}$	The number of identical neurons at cluster c of layer h
$k_{(c,h)}$	The internal state of each neuron at cluster c of layer h
$r_{(c,h)}$	An exponentially distributed random spiking interval
$q_{(c,h)}$	Probability that any neuron at cluster c of layer h is firing
p	Probability that a neuron that received trigger, immediately transmits a trigger to some other neuron in the same cluster
$\Lambda_{(c,h)}$	The input of each neuron at cluster c of layer h
$w^{(c,h)}_{(c',h-1)}$	The connection weight between any neuron at cluster $(c', h-1)$ to any neuron at (c, h).
λ^+	Rate of external Poisson flow of excitatory input spikes to any neuron in DRNN
λ^-	Rate of external Poisson flow of inhibitory input spikes to any neuron in DRNN
$\zeta(\cdot)$	Activation function of a DRNN neuron
M	Total number of network traffic metrics
$x_i^m \in [0,1]$	The $m-th$ metric extracted from network traffic for packet i
x_i	The vector of input metrics, $x_i = [x_i^1, \cdots x_i^m, \cdots, x_i^M]$
$\mathcal{D}_{\text{train}}$	Training dataset comprised of benign (non-attack) packets
$\mathcal{U}_m$	Set of unique non-numerical values in $\mathcal{D}_{\text{train}}$
P	Number of transmitted packets considered to calculate the first and second metrics
T	Length of time window to the past that is considered to calculate the third metric
$\hat{x}_i$	The output of AADRNN indicating the expected value of x_i in the absence of intrusion
$\hat{x}_{(i,h)}$	The output of layer h of AADRNN for packet i
W_h	Matrix of connection weights (including biases) between the layer $h-1$ and layer h
X	The matrix of metrics for normal traffic packets in the training dataset $\mathcal{D}_{\text{train}}$
$\hat{X}_h$	The output matrix of layer h for input X calculated with the learned weights
z_i^m	Absolute difference between x_i^m and $\hat{x}_i^m$ for metric m
w_m	Value of whisker for metric m that is learned from the available dataset $\mathcal{D}_{\text{train}}$
ψ_i	The number of (abnormal) metrics that are significantly different than the expected metrics
y_i	Binary output of IDS indicating whether the packet i is malicious
θ	Threshold on ψ_i to calculate binary decision y_i
Q_l^m	The lower quartile of the z_i^m values in $\mathcal{D}_{\text{train}}$
Q_u^m	The upper quartile of the z_i^m values in $\mathcal{D}_{\text{train}}$
μ_ψ	The mean of ψ_i in $\mathcal{D}_{\text{train}}$
σ_ψ	The standard deviation of ψ_i in $\mathcal{D}_{\text{train}}$
I	Interval between two successive parameter update during incremental learning
O_k	Operation matrix for incremental learning in window k
ρ_m	Absolute value of the Pearson coefficient for metric m
f_m	Normalized F-ratio for metric m
α_m	Overall importance score for metric m

Chapter 4 (Section 4.3): IDS for compromised device identification with sequential learning

Symbol	Definition: i indicates the node (or device) index within all nodes in the considered IoT network
$y_{i,k}$	Binary output of IDS indicating whether node i is compromised at time window k
$pk(t,s,d)$	Packet sent by source node s to destination d at time t, whose length is denoted by $\|pk(t,s,d)\|$
T	Fixed duration of each time window k
S	Set of all source nodes
D	Set of all destination nodes
$P_k^{s,d}$	Collection of the packets that have been sent from s to d in window k
$P_k^{S,d}$	Set of all packets arriving to node d in window k
$P_k^{s,D}$	Set of all packets sent by node s in window k
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
Ω_s	Maximum outgoing rate of each node s in bytes/second.
$x_{i,k}^m$	The $m-th$ metric extracted from network traffic in window k for node i
M	Total number of network traffic metrics which is equivalent to the number of inputs to the AADRNN
$x_{i,k}$	Vector of metrics $x_{i,k}=[x_{i,k}^1,\cdots,x_{i,k}^m,\cdots,x_{i,k}^M]$ for window k
H	The number of DRNN layers in AADRNN
$W_h^{i,k}$	Weight matrix connecting the clusters in layer $h-1$ to layer h obtained after the training in window k
$\hat{x}_{i,k}$	The output of AADRNN indicating the expected value of $x_{i,k}$ when node i is not compromised
$\hat{x}_{(i,k,h)}$	The output of layer h at time window k for node i
$\Psi_{i,k}$	Maximum of the absolute differences between actual and the expected metric values calculated by AADRNN
γ_i	Threshold for the binary decision of IDS for node i
$\hat{X}_{(i,k,h)}^{\text{train}}$	Matrix that collects the output vectors of layer h over all time windows $k'\in\{1,...,k\}$ when $y_{i,k'}=0$
ϕ_k^i	Ground truth for the infection level of node i at time window k
Θ	Threshold for the binary ground truth
v_k^i	Binary ground truth for the infection of node i at time window k

Chapter 5: Self-Supervised Intrusion Detection Framework

Symbol	Definition
M	Total number of network traffic metrics which is equivalent to the number of inputs to the IDS
x_i^m	The $m-th$ metric extracted from network traffic for packet i
x_i	The vector of input metrics, $x_i = [x_i^1, \cdots x_i^m, \cdots, x_i^M]$
y_i	Binary output of the IDS indicating whether the packet i is malicious
W	Number of learnable parameters in the ML model utilized in the IDS
$\hat{x}_i$	The output of ML-based AAM indicating the expected value of x_i in the absence of intrusion
Γ	Trust coefficient indicating the trust of SSID on the decisions of the IDS
Θ	Threshold on the trust coefficient (i.e. minimum desired value of Γ)
B_l	Batch of packets selected for learning
K	Minimum number of packets to be learned in each learning phase
I	Length of window in terms of the number of packets to calculate average decision
γ	Intrusion threshold
p_i^-	Probability of selecting packet i to use as a benign packet
p_i^+	Probability of selecting packet i to use as a malicious packet
q_i	Probability of rejecting packet i to use in training
C_{rep}	Factor of representativeness of the traffic packets learned by IDS
C_{gen}	Factor of generalization ability of IDS
S_l^{TT}	Set of inter-transmission times of all packets learned by IDS until the end of learning phase l, which has mean of $1/\lambda_l$
S_l^{PL}	Set of packet lengths times of all packets learned by IDS until the end of learning phase l, which has mean of $1/\lambda_o$
S_o^{TT}	Set of inter-transmission times of all normal packets observed during continuous detection, which has mean of $1/\mu_l$
S_o^{PL}	Set of packet lengths times of all normal packets observed during continuous detection, which has mean of $1/\mu_o$
D_{KL}^{TT}	KL-Divergence from S_l^{TT} to S_o^{TT}
D_{KL}^{PL}	KL-Divergence from S_l^{PL} to S_o^{PL}
$D_{KL-norm}^{TT}$	Normalized KL-Divergence from S_l^{TT} to S_o^{TT}
$D_{KL-norm}^{PL}$	Normalized KL-Divergence from S_l^{PL} to S_o^{PL}
Δ	Adequacy of the packets learned by IDS
κ	Knowledge of IDS obtained from the packets learned
$\mathcal{E}(l)$	Empirical error measured at the end of learning phase l

Chapter 1

Introduction

1.1 Motivation

The Internet is simply a network of interconnected computer systems, where nodes can communicate each other and send or receive data. In this way, it allows interconnected computer systems, i.e. nodes, to access services and exchange information through various platforms in the form of text, voice, image, video, etc. These systems communicate mainly using standardized protocol known as the Internet Protocol (IP) [1]. The IP was designed by the Defense Advanced Research Projects Agency (DARPA) for use in interconnected systems to manage data transmission over the Internet. Using IP, data is transmitted in the form of packet from a source to destination both of which are identified by fixed length addresses. An IP packet is comprised of a header and the data, where the header contains the necessary information, such as packet size and source-destination addresses, for successful transmission and reception of the packet.

A collection of interconnected nodes exchanging data to achieve a common goal often referred as a "networked system". While applications using the Internet are common examples for the networked systems, cloud computing, distributed systems (e.g. fog computing), and the Internet of Things (IoT) are other examples with increasing use and importance.

1.1.1 The Internet of Things

When a networked system is comprised of connected physical objects (or devices), which operate mostly with minimum or no human intervention and communicate using Machine-to-Machine (M2M) technologies [2, 3], it is called the Internet of Things. IoT objects are often embedded with actuators and sensors providing environmental data measurements to end users or execute automated commands [4]. In recent years, IoT-based systems have become one of the main foci of network research [5] and have increasingly large application areas, including healthcare, education, agriculture and safety, aiming to improve the quality of everyday life [6–9].

Considering the most fundamental components and security requirements of an IoT system, it is mainly modeled by a three layer architecture composed of “Perception Layer”, “Network Layer”, and “Application Layer” [10, 11]. These layers can be summarized as follows:

- The **perception layer** is considered as the first layer where the actuators and sensors are located to collect and process data of physical quantities [12]. When using some local or short-range networks, e.g. ad hoc network, cooperation between devices can also be performed at this layer [13]. In addition, as the sensor nodes have relatively simple and low-cost hardware, they are not sufficient to have complex cybersecurity systems [11].

- The **network layer** – also known as the "transmission layer" – is the second layer where packets carrying information collected via sensors are transmitted to other devices or a central data processing unit, such as a server or cloud. Although the communication security of networked systems is well researched, IoT communication is still plagued with attacks that eavesdrop on messages or create congestion due to the large number of low-cost communicating nodes.

- The **application layer** of IoT systems hosts user-end applications, such as smart homes, cities, and transportation [14, 15]. In this layer, while some services are provided for each application, the principal operations of the fundamental are also performed in this layer.

1.1.2 IoT Security

Since IoT devices are generally lightweight devices with limited resources (such as low computing power, low memory, and limited power supply), these devices are often

- struggling to perform preventive cybersecurity actions, such as encryption or authorization, due to their low computing power,
- vulnerable to attacks that flood system memory,
- severely affected by repeated operations and packet re-transmissions that may be the result of an attack.

In addition, most IoT devices have either software vulnerabilities or weak login credentials (or both) making them an easier target for attackers than user-enabled devices, such that 70% of all IoT devices are considered to be vulnerable. As a result of these vulnerabilities, it is estimated that various devices in an average smart home are targeted by $12,000$ attacks in a single week [16].

As the number of low-cost vulnerable IoT devices and networks containing such devices is increasing rapidly with expanding application areas and increasing use of data-driven systems, cybersecurity of networked systems has become one of the main concerns, such that cyberattacks on IoT devices are considered to be the primary concern by 33% of cybersecurity companies [16]. Therefore, it is crucial to enhance the cybersecurity of IoT networks in order to ensure their safe, trusted and seamless operation, and achieving secure networked and IoT systems has been one of the main research foci. On the other hand, while systemic approaches to improving the security of cyberphysical systems have been suggested [17, 18], it is difficult (if not totally impossible) to burden simple IoT devices with complex security functionalities [19].

1.2 Problem Statement and Thesis Contributions

The majority (approximately 52%) of IoT devices are low-cost and low-maintenance devices deployed in massive IoT networks [20]. Therefore, developing and implementing complex and advanced security methods, such as Machine Learning (ML)-based Intrusion Detection System (IDS), is a challenging task, especially for the following reasons: 1) The low computational powers of these devices are not sufficient to execute complex algorithms. 2) Data-driven algorithms (e.g. ML-based algorithms) require large amounts of data that are difficult to collect as they require considerable labor, high development costs, and long deployment time. 3) These algorithms are customized for the individual system or network to which they are applied, as their parameters are directly optimized for that system. As a result, when these algorithms need to be deployed for a new system, a significantly large amount of work has to be manually repeated.

The main purpose of the present thesis is to research online learning for IDS towards the development of secure IoT systems. Our research aims to address the above issues and provide a lightweight, easy-to-implement algorithm for intrusion detection, which is one of the key security assurance methods (that shall be reviewed in Section 2.2). To this end, we propose a novel Self-Supervised Intrusion Detection (SSID) framework in order to enable fully online learning of ML-based IDS with no offline data or human intervention required. In addition, we develop an ML-based IDS which is considerably lightweight and based on original traffic metrics, new statistical decision-making algorithm and the well-known Deep Random Neural Network (DRNN). This IDS can learn either offline or online using small sized data and identify both malicious traffic and compromised (malware infected) network nodes.

Accordingly the main contributions of this thesis are as follows:

- The first, and the main, contribution of the present thesis is the development of a self-supervised learning framework for ML-based IDS, called Self-Supervised Intrusion Detection (namely, SSID) framework, that enables the fully online learning of the IDS parameters requiring no human intervention. As its main advantages, the SSID framework 1) enables IDS to easily adapt time varying characteristics of the network traffic, 2) eliminates the need for offline data collection, 3) prevents human errors in data labeling, and 4) avoids labor costs for model training and data collection through experiments.

- Within the development of the SSID framework, we statistically measure the trustworthiness of an ML-based IDS considering its generalization ability and the traffic packets that IDS learned. To this end, we present measures to estimate the generalization ability and the representativeness of the learned traffic.

- The rest of the contributions are towards the development of an ML-based IDS that can be trained offline or quasi-online. To this end, as the third contribution of this thesis, we use Deep Random Neural Network model to create an auto-associative memory on the network traffic patterns and combine it with originally determined traffic metrics and classification algorithm. In addition, we develop three learning procedures for the developed IDS for offline, sequential, and incremental learning.

- Subsequently, we develop a classification algorithm, called Statistical Whisker based Benign Classifier (SWBC), that identifies the malicious traffic comparing the actual traffic and the expected traffic estimated by auto-associative memory. In addition, we determine original metrics which are easy to calculate using only the header information of the traffic packets, and highly effective to analyse the impacts of Botnet attacks on the network traffic and to capture the signatures of an attacker.

- The last contribution of this thesis is the development of a new system to identify compromised IoT devices (bots) based only on the network traffic without requiring access the device status or message contents. Although it is crucial to identify compromised devices along with the malicious traffic to successfully prevent an attack from spreading or mitigate its impacts on the network, the compromised device identification has not yet been well-investigated in the literature.

1.3 Publications of the Author

This section lists the articles of the author which have been published or submitted for publication in peer-reviewed journals and conferences. These publications are given in two categories: 1) publications included in this thesis, and 2) publications published during doctoral studies of the author but not directly related to the content of this thesis.

1.3.1 Publications in This Thesis

Journal Papers:

- M. Nakıp and E. Gelenbe, “Fully Online Self-Supervised Learning Framework for Machine Learning based Intrusion Detection,” in *arXiv*, 2023, *Preprint*.
- E. Gelenbe and M. Nakıp, “Traffic Based Sequential Learning During Botnet Attacks to Identify Compromised IoT Devices,” in *IEEE Access*, vol. 10, pp. 126536-126549, 2022.

Conference Papers:

- E. Gelenbe and M. Nakıp, “Real-Time Cyberattack Detection with Offline and Online Learning,” *2023 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, London, United Kingdom, 2023, pp. 01-06.
- E. Gelenbe and M. Nakıp, “G-Networks Can Detect Different Types of Cyberattacks,” *2022 30th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Nice, France, 2022, pp. 9-16.
- M. Nakıp and E. Gelenbe, “Botnet Attack Detection with Incremental Online Learning,” *2021 Security in Computer and Information Sciences (EuroCybersec)*, Nice, France, 2022, pp. 51-59.
- M. Nakıp and E. Gelenbe, “MIRAI Botnet Attack Detection with Auto-Associative Dense Random Neural Network,” *2021 IEEE Global Communications Conference (GLOBECOM)*, Madrid, Spain, 2021, pp. 01-06.

1.3.2 Publications of the Author Published During His Doctoral Studies but Not Included in the Scope of This Thesis

Journal Papers:

- E. Gelenbe and M. Nakıp, “IoT Network Cybersecurity Assessment with the Associated Random Neural Network,” in *IEEE Access*, vol. 11, pp. 85501-85512, 2023.

- M. Nakıp, O. Çopur, E. Biyik and C. Güzeliş, “Renewable energy management in smart home environment via forecast embedded scheduling based on Recurrent Trend Predictive Neural Network,” in *Applied Energy*, vol. 340, pp. 121014, 2023.

- E. Gelenbe, M. Nakıp and T. Czachórski, “Improving Massive Access to IoT Gateways ,” in *Performance Evaluation*, vol. 157-158, pp. 102308, 2022.

- M. Nakıp, E. Çakan, V. Rodoplu and C. Güzeliş, “Dynamic Automatic Forecaster Selection via Artificial Neural Network Based Emulation to Enable Massive Access for the Internet of Things,” in *Journal of Network and Computer Applications*, vol. 201, pp. 103360, 2022.

- M. Nakıp, B. C. Gül, V. Rodoplu and C. Güzeliş, “Predictability of Internet of Things Traffic at the Medium Access Control Layer Against Information-Theoretic Bounds,” in *IEEE Access*, vol. 10, pp. 55602-55615, 2022.

- M. Nakıp, A. Helva, C. Güzeliş and V. Rodoplu, “MOSAL: A Subspace-Based Forecasting Algorithm for Throughput Maximization in IoT Networks,” in *IEEE Sensors Journal*, vol. 22, no. 24, pp. 24634-24646, 2022.

- A. K. Dayı, V. Rodoplu, M. Nakıp, B. Pehlivan and C. Güzeliş, “Multi-Channel Subset Iteration with Minimal Loss in Available Capacity (MC-SIMLAC) Algorithm for Joint Forecasting-Scheduling in the Internet of Things,” in *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 13, no. 5, pp. 68-95, 2022.

- M. Nakıp, K. Karakayali, C. Güzeliş and V. Rodoplu, “An End-to-End Trainable Feature Selection-Forecasting Architecture Targeted at the Internet of Things,” in *IEEE Access*, vol. 9, pp. 104011-104028, 2021.

- M. Nakıp, C. Güzeliş and O. Yıldız, “Recurrent Trend Predictive Neural Network for Multi-Sensor Fire Detection,” in *IEEE Access*, vol. 9, pp. 84204-84216, 2021.

Conference Papers:

- M. Nakıp, B. C. Gül and E. Gelenbe, “Decentralized Online Federated G-Network Learning for Lightweight Intrusion Detection,” *2023 31st International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Stony Brook, NY, USA, 2023, *In Press*.

- M. Nasereddin, M. Nakıp and E. Gelenbe, “Measurement Based Evaluation and Mitigation of Flood Attacks on a LAN Test-Bed,” *The 48th IEEE Conference on Local Computer Networks*, Florida, USA, 2023, *In Press*.

- E. Gelenbe, M. Nakıp and T. Czachórski, “Mitigating the Massive Access Problem in the Internet of Things,” *2021 Security in Computer and Information Sciences (EuroCybersec)*, Nice, France, 2022, pp. 118-132.

- O. Çopur, M. Nakıp, S. Scardapane and J. Slowack, “Engagement Detection with Multi-Task Training in E-Learning Environments,” *2022 Image Analysis and Processing (ICIAP)*, Lecce, Italy, 2022, pp. 411-422.

- M. Siavvas, E. Gelenbe, D. Tsoukalas, I. Kalouptsoglou, M. Mathioudaki, M. Nakıp, D. Kehag and D. Tzovaras, “The IoTAC Software Security-by-Design Platform: Concept, Challenges, and Preliminary Overview,” *2022 18th International Conference on the Design of Reliable Communication Networks (DRCN)*, Vilanova i la Geltrú, Spain, 2022, pp. 1-6.

- E. Gelenbe, M. Nakıp, D. Marek and T. Czachórski, “Diffusion Analysis Improves Scalability of IoT Networks to Mitigate the Massive Access Problem,” *2021 29th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Houston, TX, USA, 2021, pp. 1-8.

- M. Nakıp and E. Gelenbe, “Randomization of Data Generation Times Improves Performance of Predictive IoT Networks,” *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, New Orleans, LA, USA, 2021, pp. 350-355.

- M. Nakıp, A. Helva, C. Güzeliş and V. Rodoplu, “Subspace-Based Emulation of the Relationship Between Forecasting Error and Network Performance in Joint Forecasting-Scheduling for the Internet of Things,” *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, New Orleans, LA, USA, 2021, pp. 247-252.

- E. Çakan, A. Şahin, M. Nakıp and V. Rodoplu, “Multi-Layer Perceptron Decomposition Architecture for Mobile IoT Indoor Positioning,” *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)*, New Orleans, LA, USA, 2021, pp. 253-257.

- M. Nakıp, A. Asut, C. Kocabıyık and C. Güzeliş, “A Smart Home Demand Response System based on Artificial Neural Networks Augmented with Constraint Satisfaction Heuristic,” *2021 13th International Conference on Electrical and Electronics Engineering (ELECO)*, Bursa, Turkey, 2021, pp. 580-584.

- A. Saylam, N. Kelesoglu, R. O. Cikmazel, M. Nakıp and V. Rodoplu, “Dynamic Positioning Interval Based On Reciprocal Forecasting Error (DPI-RFE) Algorithm for Energy-Efficient Mobile IoT Indoor Positioning,” *2021 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Istanbul, Turkey, 2021, pp. 1-5.

- A. Saylam, R. O. Cikmazel, N. Kelesoglu, M. Nakıp and V. Rodoplu, “Energy-Efficient Indoor Positioning for Mobile Internet of Things Based on Artificial Intelligence,” *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Elazig, Turkey, 2021, pp. 1-6.

- M. Nakıp, O. Çopur and C. Güzeliş, “Comparative Study of Forecasting Models for COVID-19 Outbreak in Turkey,” *2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, Elazig, Turkey, 2021, pp. 1-6.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: In Chapter 2, we provide an overview of cybersecurity and intrusion detection. To this end, first, background on the most common cybersecurity breaches and security assurance methods is given. Then, recent work on security issues at each layer in the IoT system architecture is reviewed. Accordingly, this chapter provides the necessary background on cybersecurity in networked systems and highlights some major issues for secure IoT.

In Chapter 3, we address two main issues to improve the security of IoT networks through intrusion detection, where these issues are detection of cyberattacks and identification of compromised devices (or nodes) over an IoT network. This chapter provides a brief problem statement, discusses the use and importance of malicious (attack) traffic detection and compromised device identification, and comprehensively reviews the recent literature on each issue.

In Chapter 4, we develop a new IDS with offline and quasi-online auto-associative learning to both detect malicious traffic and identify compromised IoT devices. Accordingly, this section first reviews DRNN, which is the core ML model used in our IDS. In this chapter, in addition to offline, incremental and sequential learning algorithms, we determine nine original network traffic metrics and propose the SWBC algorithm. The performance of this IDS is evaluated for detecting Botnet attacks and different types of unknown (zero-day) cyberattacks on three publicly available datasets.

In Chapter 5, we propose the novel SSID framework designed to train any given IDS fully online with no human intervention. As a part of the SSID framework, we develop a self-supervised packet selection methodology based on an original statistical measurement of the IDS trustworthiness. We also evaluate the performance of the SSID framework using both the IDS developed in the previous chapter and well-known ML models for each of the malicious traffic detection and compromised device identification on 6 different attack types in two public datasets.

Chapter 6 summarizes this thesis emphasizing the contributions and the obtained results. In addition, it provides some insight into possible future work that may help to address remaining open issues and expand current work in new directions.

Chapter 2

Cybersecurity in Networked Systems

Security in information processing systems aims to ensure confidentiality, integrity, and availability of information against the unauthorized access, modification, and destruction during its storage, processing or transit [21]. That is, its main goal is to ensure the privacy, safety, and accessibility of critical information. On the other hand, when a security incident occurs, it can have hazardous consequences such as leakage, loss or corruption of private and critical data, or even ransom demand [22].

As we shall present some major breaches in the next section, the cybersecurity breaches often cause to loose confidentiality, integrity, or availability of information and critical data, which are are briefly defined as follows: **Confidentiality** refers to ensuring that only the authorized users are permitted to access information for any purpose. **Integrity** requires that information has not been tampered and is presented accurately. **Availability** means that information is accessible to authorized users whenever needed to process [23].

As the Internet consists of interconnected networked systems that provide or process information and become an essential part of daily personal or business activities, its security is extremely important. However, the openness and interconnectedness of the Internet have also made it vulnerable to various security threats, such as malware, phishing attacks, hacking, identity theft, and cyberterrorism. Therefore, it is crucial to ensure the security of the internet to prevent these threats, protect user and data privacy, and ensure comprehensive information security.

In order to achieve security in the internet and information processing systems, comprehensive measures have to be implemented. These measures include intrusion detection, authentication and access control, data encryption, and user education. In the remainder of this chapter, we review the security breaches, means and methods to assure security, and the common security issues in IoT.

2.1 Breaches of Cybersecurity in Networked Systems

Cybersecurity breaches can occur in networked systems that produce, process or store valuable assets and information, with significant consequences such as theft of sensitive information, financial loss, reputation damage, or disruption of services. Cybersecurity breaches can occur intentionally or unintentionally due to various factors, such as software or hardware vulnerabilities, malware, social engineering and hacking activities, or human error. In this section, we shall review some examples of cybersecurity breaches, especially common attack types, in networked systems.

2.1.1 Phishing Attacks

Phishing attacks are one of the most common type of attacks targeting the individual users and businesses as 83% of 573 investigated businesses identified phishing attacks in 2022 [24], and it is reported in [25] that financial losses directly from successful phishing attacks increased by 76% in 2022.

Phishing attacks are based on tricking users into taking “wrong actions” such as revealing sensitive information (e.g. usernames and passwords) and downloading malware. Although there are many different types of phishing attacks [26], the most common phishing attacks target either numerous victims at once (generally via email) or a specific victim through social engineering tactics such as impersonating a trusted entity or using fake login pages [27].

Email phishing usually means that an attacker sends thousands of fake emails, awaiting a response from a small percentage of victims and collecting personal information, account credentials, and money. Generally, the fake emails (including links, attachments, logos, and signatures that the email contains) are designed to imitate the actual emails from well-trusted organizations and to create a sense of urgency, so that the victims shall be less suspicious, more impulsive and more prone to error [28].

A more in-depth and the most effective (accounting over 90% of all phishing attacks [29]) type of phishing attack, called spear phishing, targets a specific person or organization rather than a large number of random victims. Since the attacker usually aims to gain access to sensitive data or administrative accounts, spear phishing attacks require the attacker to have detailed information about the targeted organization. Therefore, they are more difficult to detect than email phishing attacks.

2.1.2 Malware-based Attacks

Malware-based attacks, as the name suggests, are designed for compromising a vulnerable cybersystem via malicious software, i.e. malware. Malware such as viruses, trojans, worms, and ransomware can breach the networked system through security vulnerabilities (or sometimes links in phishing emails). As a result of the breach, the malware-based attack can result in severe consequences such as theft of confidential data, damage to systems, and even the victim device becoming the host of a new attack [30]. It is estimated that 2.8 billion malware-based attacks occurred worldwide in the first half of 2022 alone [31]. Additionally, in United Kingdom, 16% of businesses that detected a breach and/or attack are targeted by malware-based attacks, 4% of which is ransomware [24].

In ransomware attacks, the malware first gains access to the victim’s device, files and sensitive data. Then, the attacker demands a ransom (usually in cryptocurrency) to trade this data with the victim. Therefore, it is one of the most profitable, so one of the most popular, malware-based attacks [32]. In some cases, to obtain the sensitive information, attackers may target IoT devices (such as, IP cameras) and use their vulnerabilities.

Some malware-based attacks, such as worms and trojans, can pose a significant threat to IoT devices and networks. Due to the vulnerable architecture of low-cost interconnected IoT devices with default login credentials, worm attacks, in particular, can easily distribute malware across a network of thousands of devices and discover security vulnerabilities to cause hazardous damage to those devices. In June 2019, a worm malware, called Silex, bricked more than 2000 IoT devices in the first few hours [33, 34]. The Silex IoT-worm is specifically designed to target Linux-based devices (mostly, IoT devices) and permanently disable the operations of the victim device.

2.1.3 Denial-of-Service (DoS) and Distributed Denial-of-Service (DDoS) Attacks

Denial of Service (DoS) and Distributed Denial of Service (DDoS) are the other types of attacks that pose a significant threat to cyber systems. It can be said that the risks posed by these attacks are rapidly increasing, especially with the increase in the number of IoT devices with weak security. Indeed, in the second quarter of 2022, DDoS attacks annually occurred 109% more in the network layer and 72% more in the application layer [35].

Attacks that aim to disrupt the normal services of cyber system by overloading its resources are classified as DoS attacks. In DoS attacks, the attacker often sends too many requests to the victim to process, so that the victim can no longer perform its normal operations. By sending numerous requests, the attacker targets to either exhaust system resources (e.g. memory and storage) or flood the system's communication and processing capacity.

When a device is exposed to a DoS attack aimed at exhausting resources, it occupies all available computing resources of the victim device. Therefore, this attack often results in operational slowdowns, service delays, or complete system shutdown [36]. Moreover, when a device is exposed to a flooding DoS attack, its capacity is saturated by overwhelming amount of traffic received.

DDoS attacks are variants of DoS attacks in which malicious traffic is generated by multiple sources, each acting similarly to the source of a DoS attack. DDoS attacks usually originate from a network of malware-infected, compromised devices called bots. Therefore, DDoS attacks cannot be blocked by blacklisting an individual source of the attack, as with DoS attacks.

In a DDoS attack, each of the originating bots generates meaningless and superfluous traffic (or requests). While the individual impact of each DDoS bot is less overwhelming than the source of a DoS attack, the cumulative impact of the network of bots – namely the botnet – is more hazardous and can result in long service interruptions [37].

Furthermore, the Botnet DDoS attacks are also one of the most common and dangerous malware-based attacks for the IoT networks. The Botnet is a network of devices infected by a malware which turns the victim device into a bot. Although these bots may be used to perform a variety of tasks, including spreading spam, stealing data, mining cryptocurrency, and conducting phishing scams, it is most commonly used to perform DDoS attacks on IoT networks.

2.1.4 Eavesdropping and Modification Attacks

Eavesdropping and modification attacks refer to incidents when attacker gains access to the communication between two devices. Depending on the purpose and type of an attack, the attacker can intercept, delete, or modify the network traffic. One of the most active form of these attacks is the Man-in-the-Middle (MitM) attack, as 95% of Hypertext Transfer Protocol Secure (HTTPS) servers are estimated to be vulnerable against this attack [38].

The basic idea behind the MitM attack is that the attacker intercepts communication between the two communicating nodes [39]. If the attack is successful, the attacker will act as a proxy so that the nodes will believe there is direct communication between them. This way, all messages in the conversation will be available for the attacker to read, modify or even delete. The attacker in MitM can obtain valuable secret data such as conversations within the organization and login credentials of users.

2.1.5 Unauthorized Access and Insider Threats

Unauthorized access to the system assets, such as devices, databases and control centres, is another major issues for the cyber systems. Insider threats are one of the main sources of unauthorized access breaches for the organizations as the employees or contractors intentionally or unintentionally compromise network security. Unauthorized access incidents often result in data leaks or malware injection. These incidents, which increased by 35% in 2021, are one of the leading causes of data breaches for organizations [40].

In summary, cybersecurity breaches in networked systems can have severe consequences. Detecting and preventing those breaches through methods and means is crucial to minimize their impact and assure the security of cybersystems.

2.2 Methods and Means to Assure the Security of Cybersystems

Assuring the security of cybersystems involves developing and implementing various methods and means to protect these systems and ensure their confidentiality, integrity, and availability [23]. These methods and means include both technical (software and hardware) and organizational development tools. Although the development tools, such as training employees, developing incident response plans, and conducting cybersecurity audits have positive effects on raising awareness and preventing human errors [41, 42], we now focus on the technical means to be developed and implemented to minimize the risk of security breaches in cybersystems. These methods and means are intrusion detection systems, authentication and access control mechanisms, and cryptography techniques. It should be noted that although these methods are necessary for cybersecurity, none of them alone is sufficient to ensure the security of a cyber system against all breaches.

2.2.1 Intrusion (Attack) Detection

Intrusion (or attack) detection is one of the methods to assure the security of a cybersystem, e.g. an IoT network. It is a software application that monitors the cybersystem regarding the network traffic and/or the states of the nodes (e.g. devices and gateways) in order to identify malicious activities and actual threats.

IDSs are often used as part of protection mechanisms against the cyberattacks, such as Intrusion Detection and Prevention Systems, in heterogeneous networks, especially those involving IoT devices, which are extremely vulnerable and difficult to protect against various cybersecurity breaches [43–45]. As most of them are also reported by NIST [46, 47], IDSs are specially needed for the following actions:

- Detecting security breaches, in particular cyberattacks, that are not prevented by other methods which shall be reviewed in the rest of this section
- Reporting malicious activities and possible threats immediately to a network management system
- Identifying and deterring individuals or machines (e.g. compromised devices) that pose a threat to the rest of the system, and

- Controlling the quality of existing security policies.

Accordingly, the decisions of IDS can further be used to mitigate the impacts of cyberattacks [48–50] or to optimize the system for maximum security along with high Quality-of-Experience and Quality-of-Service [51]. On the other hand, the absence of IDS or the use of poor-performing IDS can have serious consequences, such as leaking private data, service interruptions, and damage to software and hardware.

In order to enable a quick and accurate response, the IDS generally operate in either device (i.e. host) or network level. The host-based IDS analyses the incoming and outgoing traffic on the network interfaces of the considered device, while the network-based IDS is located to observe and analyze the traffic of all devices in the network (or sub-network) [52]. Therefore, host-based intrusion detection is capable of precisely detecting attacks that directly target or originate from the device hosting the detector, and network-based intrusion detection can identify attacks originating from both inside and outside the network.

Both host-based and network-based IDS can be further classified according to their methodologies used as signature-based and anomaly-based detection [53]:

Signature-based IDS aims to detect cyberattacks whose signature patterns are known by the technique [54]. To this end, an IDS compares the actual traffic patterns against the signature patterns stored in the database for the known attacks. If the actual traffic patterns match with any type of attack known in-advance, the detector raise an alarm and report the incident of intrusion [55]. This type of IDS is clearly limited for detecting zero-day attacks which are unknown by the techniques and no stored signatures are available.

Some examples of works that study signature-based IDS are as follows: Syed et al. [56] used three different supervised machine learning models to detect DoS attacks targeting devices using the MQTT protocol, with metrics calculated from traffic characteristics such as source/destination addresses and port numbers. Gelenbe et al. [57] developed a signaling storm detection and mitigation method which analyses the signaling transitions and time-outs due to inactivity. Tan et al. [58] converted traffic patterns into images and used computer vision with Earth Mover's Distance to detect DoS attacks in cybersystem.

Anomaly-based IDS usually learns the normal traffic patterns of devices or networks, i.e. the behaviour of devices when there is no intrusion. Then, if the actual traffic pattern deviates significantly from the learned patterns, it is classified as an anomaly corresponding to an intrusion [59].

In the last two decades there are plenty of works developing anomaly-based intrusion detection techniques to assure the security of cybersystems [60]. For example, in earlier research [61], Guangzhi et al. performed anomaly-based detection of DDoS, worm, and spam attacks by analyzing the states of system resources and network protocols. Tan et al. [62] developed a method using multivariate correlation analysis for DoS attack detection. Hindy et al. [63] recently used auto-encoder neural network to develop an anomaly-based detection system for zero-day attacks.

In Chapter 3, we shall examine in more detail the work on both signature and anomaly-based intrusion detection specifically targeting IoT devices and networks.

2.2.2 Authentication & Access Control

Authentication and access control together aim to ensure that only verified and authorized users can access the permitted assets of the cybersystem. To this end, a user is first authenticated by verifying its identity through various factors, such as username and password, biometrics, and token. It is then allowed to access the assets for which it is authorized to read, modify, or delete. The authentication and access control mechanisms may prevent cybersystems against some security breaches, such as phishing, brute force, eavesdropping and MitM attacks as well as the unauthorized access and insider threats [64, 65]. Cybersystems that contain IoT devices mainly use password-based, token-based, or multi-factor authentication mechanisms [66].

Password-based authentication is widely used and well investigated mechanism combining a unique username and a password which consists of letters, numbers and special characters. The combination of username and password is stored in the authentication database, and a user is only allowed to perform requested actions when it provides the right combination. Although this mechanism has been extensively studied and formalized in the last three decades [67, 68], some recent works focused on improving the security of and adapting password-based authentication for IoT devices and networks. In [69], Renuka et al. developed a password-based authentication scheme for IoT networks in which any pair of IoT nodes can authenticate each other and the mobile user holds only one secret key. Recently for IoT-enabled smart cities, Meshram et al. [70] developed an authentication protocol using a chaotic map and a smart card that identifies an incorrect password.

Token-based authentication is in two types: 1) One-time password is stored in the authentication database and compared with the one provided by the user. 2) A hardware device, e.g. smart card, stores a token that will be used to authorize the user for its requests. Due to their ease of use and implementation, token-based authentication mechanisms are well-accepted [66]. Recent research investigate the token-based authentication mechanisms specialized for IoT (or heterogeneous) networks [71, 72]. For example, Aman et al. [73] enhanced token-based authentication for IoT devices through a dynamic energy quality exchange method that aims to consume less energy when a lower level of security is acceptable. Dammak et al. [74] developed a token-based authentication mechanism designed to be lightweight and suitable for IoT devices in order to reduce energy consumption and computational requirements for authentication of the devices. Amin et al. [75] developed a token-based authentication protocol using smart card for distributed cloud environment containing IoT devices.

Multi-factor authentication is based on the combination of two or more factors, such as passwords, tokens, and biometrics. Due to its versatility, multi-factor authentication has increasing popularity to be used for securing cybersystems, including IoT networks [76]. Statistics show that the percentage of accounts using multi-factor authentication was increased from 28% to 78% between 2017 and 2021 [77]. Recently, Noura et al. [78] developed multi-factor authentication mechanism combining a cryptographic protocol with communication protocol features targeting to achieve high security under limited computational resources of IoT devices. In addition, for IoT cloud-based environment, Alsahlani et al. [79] developed a lightweight multi-factor authentication and authorization mechanism based on three factors which are passwords, smart cards, and biometrics.

2.2.3 Cryptography

Cryptography is another technique to secure the communication against the unintended read and process of the information. It is generally used to encrypt and decrypt data during its transmission between nodes [80]. In this way, cryptography techniques aim to ensure information security aspects such that the data is protected against unauthorized accesses and changes, so eavesdropping and modification attacks. On the other hand, implementation of cryptography requires high computational power, and it is still vulnerable against cryptographic attacks, such as brute force or ciphertext attacks.

The most common types of cryptography techniques are categorized as Symmetric Encryption, Asymmetric Encryption, Cryptographic Hash Functions, and Quantum Cryptography [81]. In Symmetric Encryption, as known as **Secret Key Cryptography**, a single key is generated both encryption and decryption of the data, so both the transmitter and the receiver must have the key. The distribution of the key is one of the biggest challenges for this category of techniques [82]. Asymmetric Encryption, as known as **Public Key Cryptography**, proposed by Diffie and Hellman [83] and requires a pair of keys to first encrypt then decrypt data. One of these keys, namely public key which can be freely distributed, is used for encryption while the private key is kept as a secret and used for decryption. The public and private keys are mathematically related; however, constructing private key from the public key is computationally infeasible. **Cryptographic Hash Functions** are algorithms that generate a fixed-length hash code from data of any size; such that it is computationally impossible to reconstruct original data from the hash code. Finally, **Quantum Cryptography** uses quantum mechanics to securely communicate between two nodes, often based on quantum key distribution. Quantum Key Distribution allows a key to be shared over a public communication medium without the risk of eavesdropping.

2.3 Security Issues in IoT Devices and Networks

We now review the security and privacy issues in IoT systems. According to a study by HP [19], as an average of 25 vulnerabilities were detected per device, 70% of IoT devices are vulnerable to attacks due to poor password security and lack of encryption and access permission methods. Moreover, while only less than 10% of IoT devices are reported to be secure, the most common attacks are caused by hackers themselves, malware attacks, and Denial of Service (DoS) attacks, with 65%, 23% and 20% respectively [84].

Recall – from Section 1.1.1 – that an IoT system can be represented through Perception, Network and Application layers [85]. Each of these layers has a different structure, different features, and special technologies [86–89]; therefore, the security issues for these layers are also different. Accordingly, in this section, we review security issues and some recommended solutions in recent literature for each of the layers in three-layer architecture of IoT.

2.3.1 Perception Layer

This layer is directly connected to the physical world and contains devices with limited resources. Therefore, the majority of security issues occur in this layer are directly related to the vulnerable IoT devices. Unauthorized access to passive tags, replay attacks, and false data injection attacks are security issues that are often reported to occur in this layer [90–93].

Unauthorized Access problem often occurs in systems with a large number of RFID modules and no strong authentication protocols [92]. In such systems, an unauthorized person (an attacker) can access modules and modify or delete records [86]. Examples of studies aimed at preventing unauthorized access: Gharooni et al. [94] proposed a mutual authentication model based on random number generation. Mujahid et al. [95] used recursive hashing to authenticate tags. Also, a method called Arbitrator is developed in [96, 97], which consists of reader authentication and channel jamming modules. For IoT networks, Luo et al. [98] developed a communication protocol using symmetric key cryptography.

Replay Attacks on IoT networks, communication is interrupted by an attacker to resend or delay the message [90, 99]. The replay attack is quite similar to the MitM attack that occurs at the network layer. For protection against replay attacks, Kim [100] proposed a mechanism that adds timestamps and nonce options to packages. In recent years, Malik et al. [101] has proposed a feature space for the Support Vector Machine (SVM) to detect higher-order harmonic distortions introduced by modeling replay attacks.

False Data Injection attacks target cyber-physical systems (such as IoT networks) to compromise sensor readings and corrupt collected data causing miscalculations and operations. In recent years, these attacks are among the most critical and damaging perception layer attacks that have occurred, especially in smart grids [102, 103]. In order to detect False Data Injection attacks in smart grids: Han et al. [104] developed a detection methods using graph neural network, Tan et al. [105] developed lightweight detection algorithm using a multi-objective optimization and singular value decomposition especially for DC micro grids, and Lin et al. [106] developed a edge-based federated learning framework which enables individuals and separate data owners to contribute to improve attack detection.

2.3.2 Network Layer

This layer contains the communication medium and creates the connection between IoT devices and the network [15], so that it can be said that the main security problems of this layer are mostly caused by attackers exploiting security vulnerabilities in the communication network. Although there are numerous attack types targeting this layer, such as Routing, Data Transit, Storage, Exploit, Sink Hole and Sleep Deprivation, and two of the most common threats are DDoS and MitM attacks [107].

In **Denial-of-Service** attacks, attackers aim to prevent the normal functioning of a device (or system) by usurping the limited resources of device [108]. To this end, an attacker or malicious device attacks the target device by forwarding superfluous requests [109].

When a network attack occurs following the same techniques as DoS attacks, but affects more devices through botnets (i.e. networks of compromised devices), it is called a **Distributed Denial-of-Service** attack [110]. Due to the low-security measures implemented in IoT devices, they are often more easily compromised compared to user-enabled devices (such as computers and mobile phones). Therefore, one can say that IoT devices are the most likely potential victims of DDoS attacks while they are the nodes where these attacks can most easily be originated.

In the last decade, it has been observed that DDoS attacks such as TCP SYN, PUSH and ACK, ICMP Flood, UDP Flood, Smurf, DNS Flood and HTTP Flood have often occurred in IoT networks. These DDoS attacks originate from devices compromised by malware, the most popular and recent examples being Mirai, Tsunami, BASHLITE, LUABOT, Remaiten, NewAidra, Torii, and Meris [111, 112]. The recent research well-investigated these attacks and malware types to understand their characteristics and impacts [113]. For example, several authors have done extensive work to understand the characteristics [114], traffic patterns [115], source code [116], and the capabilities and impacts [117] of the Mirai Botnet attacks.

There are also significantly large number of works to develop defense mechanisms against the DDoS attacks in IoT networks, where this mechanism consists of multiple phases such as detection, prevention and mitigation [118–120]. As one can also see from the recent examples [121–123] of Botnet attack mitigation methods developed using ML and implemented for IoT networks, successful detection of both attack traffic and bots (compromised IoT devices) is a prerequisite for effective mitigation actions. Therefore, in this thesis, we propose a novel self-supervised learning framework for ML-based IDS which are also investigated and developed within this thesis using both offline and online learning algorithm. In addition, Chapter 3 shall review both device and network level attack detection, including the identification of compromised devices, for IoT networks.

Furthermore, the basic idea behind the **Man-in-the-Middle** attack is that the attacker intercepts communication between two nodes [39]. If the attack is successful, the attacker will act as a proxy so that the nodes will believe there is direct communication between them. In this way, all messages in the conversation will be available for the attacker to read, modify or even delete.

To address MitM attack in IoT networks, Li et al. [124] investigated an MitM attack with its impacts and consequences on an IoT-Fog network. Kang et al. [125] developed a routing mechanism for IoT networks to prevent the network against MitM attacks. Anthi et al. [126] developed a ML-based intrusion detection system with supervised learning to classify different attacks, including MitM and DoS in a smart home IoT network. Lahmadi et al. [127] developed MitM attack detection algorithm using auto-encoder neural network together with a Convolutional Neural Network (CNN)-based classifier. To this end, they compared Long-Short Term Memory (LSTM) and Temporal CNN for auto-encoder and used CNN combined Random Forest (RF) for attack classification.

2.3.3 Application Layer

Since this layer connects the end-user to the network via application, it is vulnerable to attacks targeting both applications and users as well as some of the network layer attacks (e.g. DoS attack) of the IoT system. Usually the cyberattacks that take advantage from the human errors target the application layer of IoT, such as phishing attacks, data leaks, and malicious code attacks [15, 90, 91].

In **Phishing Attacks**, the attacker aims to compromise the system and gain access to the victim's personal data through counterfeit communication over channels such as e-mail, voice, text message or website [128, 129].

Recent studies have mostly focused on the detection of phishing attacks developing ML-based approaches [130]: In order to determine if a given website is a fake website created for malicious phishing activity, Liu and Lee [131] used a CNN model that processes a screenshot of the website, and Yang et al. [132] combined CNN and LSTM for processing multidimensional features based on URL and the HTML code of the website. For the same purpose, Gandotra and Gupta [133] first determined a set of features related to the presence of the website, its URL, and its HTML source; then they used these features with six different ML models including Decision Tree (DT), K-Nearest Neighbour (KNN), and SVM.

Specifically for the IoT networks: Bustio-Martínez et al. [134] detected phishing URL attacks achieving 99.57% accuracy using the RF model with feature selection. Gopal et al. [135] developed a detection and mitigation mechanism against the phishing URL attacks over an IoT network. In this mechanism, a deep neural network is combined with an auto-encoder architecture to learn – in a supervised fashion – the classification of legitimate and phishing websites.

Data Leak refers to the transmission of data and/or the circulation of confidential information by unauthorized users [136]. Due to vulnerabilities in controlling users' access and weak authentication mechanisms, data leakage can occur as a result of an accidental or deliberate attack [137].

Due to small number of security means that applied in IoT networks, they are highly vulnerable against the intrusions resulting in data leakage. In order to address this issue, Yu et al. [138] developed a model, called Adaptive Feature Graph Update, to detect the data leakages through a feature graph in an IoT system while Lu et al. [139] developed a data sharing architecture based on permissioned blockchain for federated learning especially in Industrial IoT.

In **Malicious Code Attacks**, the attacker can target the vulnerabilities of the victim device or system by injecting malicious code, thereby gaining control of the system or causing serious damage [86]. In Reference [140], Wei and Qiu detected different types of malicious code by monitoring the runtime of tasks on an IoT device. In addition ML techniques are often used to detect malware with dynamic analysis: CNN is combined with a 2-dimensional version of the extracted features in each of [141] and [142]; SVM, DT and KNN classified the malware types based on system behavior in [143]; and recurrent neural network is used to make decisions on the malicious file based on machine activity features in [144].

Chapter 3

Intrusion Detection in the Internet of Things Networks

3.1 Intrusion Detection as a Means to Enhance Cybersecurity

As presented in Section 2.2, intrusion (or attack) detection is one of the methods to assure the security of a cybersystem, e.g. an IoT network. It is a software application that monitors the cybersystem regarding the network traffic and/or the states of the nodes (e.g. devices and gateways) in order to identify malicious activities and actual threats. When a malicious activity is detected, the IDS reports it immediately to a network management system.

To enable a quick and accurate response, IDS generally operates in either device (i.e. host) or network level. The host-based IDS analyses the incoming and outgoing traffic on the network interfaces of the considered device, while the network-based IDS is located to observe and analyse the traffic of all devices in the network (or sub-network) [52]. Therefore, host-based intrusion detection is capable of precisely detecting attacks that directly target or originate from the device hosting the detector, and network-based intrusion detection can identify attacks originating from inside or outside the network.

Recently, considering both device and network level, ML-based intrusion detection is capable of detecting malicious activities very accurately (for example, the intrusion detection methodology that we shall present in this thesis), and successful IDS enables network management systems to take early actions and respond attacks before the damage occurs. Therefore, one can say that IDS is a very important component to enhance the cybersecurity of the networked system.

3.2 IoT Intrusion Detection

IoT devices are the target of various attacks that differ from user-enabled devices due to their limited resources and specific network settings. Therefore, we may say that different approaches are required to successfully detect attacks in IoT networks. We now aim to review the detection techniques targeting the attacks in the network layer of IoT (which are given in Section 2.3.2). To this end, we present some recent studies whose foci are on detecting DoS and Botnet DDoS, two of the most popular attacks targeting IoT devices. We also present the works that developed techniques to detect multiple types of attacks simultaneously using a single software.

3.2.1 Denial of Service (DoS) Attacks

Brun et al. in [145] developed an attack detection algorithm that analyzes network traffic flowing through the IoT gateway in a smart home environment. This algorithm detects TCP SYN and UDP flood attacks based on specific metrics for each attack type using the DRNN model trained via Stochastic Gradient Descent. In [146], Evmorfos et al. investigated the performances of LSTM and Random Neural Network (RNN) models for detecting TCP SYN flood attack via time series prediction approach. The experimental results of this work revealed the superior performance of RNN over LSTM for TCP SYN flood attack detection.

In [147], Rathore and Park developed an Extreme Learning Machine (ELM)-based attack detection algorithm operating edge devices to perform distributed detection. The algorithm developed in this work combine ELM with semi-supervised fuzzy c-means method for clustering unlabelled samples, the performance of this algorithm has been evaluated on NSL-KDD dataset which includes samples of DoS attacks. Iqbal et al. in reference [148] used DT classifier to detect DoS attacks in the NSL-KDD dataset. During the training phase of DT, it is combined with feature selection based on well-known four search algorithms Genetic Algorithm, Particle Swarm Optimization, Best First, and Rank Search.

In addition, DT and SVM were used with supervised learning to detect DoS attacks, including Black Hole and Flood, on Wireless Sensor Networks by Al-issa et al. in [149]. Performance evaluation showed that DT outperformed SVM and successfully detected Black Hole attacks while having difficulty detecting Flood attacks. In [150], Wankhede and Kshirsagar performed experimental study to detect DoS attacks. In this work, RF and Multi-Layer Perceptron (MLP) methods are used with supervised learning to perform binary classification.

3.2.2 Botnet based Distributed Denial-of-Service (DDoS) Attacks

DDoS attacks, including various type of Botnet attacks, are one of the most common types of attacks with $61,880,000$ incidents reported by SAM in 2021 [151]. Therefore detection of DDoS attacks has been very actively studied for the last decade.

There are also plenty of research studies aiming to detect Botnet attacks which are very common way of executing DDoS attacks. Botnet – command and control – attack occurrence reportedly increased by 64% from Q2 to Q3 and 124% from Q3 to Q4 of 2021 [152, 153].

In reference [154], 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 SVM, MLP, 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 [155], Shao et al. created an ensemble of Hoeffding Tree and RF models with online learning using both normal and attack traffic. In [156], 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 [157], 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 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. [158] developed an MLP based Mirai Botnet detector specialized for Software Defined Networks. The authors fed 5 metrics, including the used communication protocol, to the MLP.

In [159], Banerjee and Samantaray performed an experimental work to deploy network of honeypots to attacks botnet attacks and to detect attacks via ML methods, such as DT, NB, Gradient Boosting, and RF. In reference [160], 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. [161] detected botnet attacks via sparse representation framework with a large number of inputs (115) for which only normal traffic is used to tune parameters.

Meidan et al. [162] 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. [163] used Classification and Regression Trees with feature selection, and Sriram et al. [164] performed a comparative study using 7 different ML methods (including NB, KNN, and SVM). In Reference [165], 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 [166], Parra et al. developed a cloud based attack detection method using CNN for phishing and using 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. [167] with features that are processed by the triangle area maps based multivariate correlation analysis algorithm.

3.2.3 Different – New – Types of Unknown (Zero-day) Attacks

Furthermore, although the Botnet attacks are one of the most common attacks targetting IoT networks, they are not alone (as mentioned in Section 2.1) to raise a concern about the security of an IoT network in different layers. Therefore, along with the highly accurate detection of Botnet (or any specific type of) attack, it is important to detect different types of unknown attacks simultaneously with acceptable accuracy using a single IDS that does not depend on the signatures of a specific attack type. To this end, as an early example, Moradi and Zulkernine [168] used MLP with supervised learning to classify two different attack types SYN Flood and Satan achieving about 90% test accuracy.

Recently, in [169], Catillo et al. developed an autoencoder based anomaly detection method which is trained using both benign and malicious data in a semi-supervised fashion to classify anomalies caused by an intrusion. The authors evaluated the performance of this method on a public dataset that contains data for various attacks, such as DoS, DDoS, and Botnet. Two studies are used RNN with supervised gradient-based learning to detect different attacks: Huma et al. [170] used DRNN with fully connected layers to perform multi-class classification on DS2OS and UNSW-NB15 datasets, while Latif et al. [171] used RNN with feed-forward architecture to perform binary classification (in other words, anomaly detection) on DS2OS dataset.

The authors of some works [172–176] have evaluated the performance of their techniques on NSL-KDD dataset: In [172], Sarker performed an extensive comparative and experimental study for anomaly detection and multi-class classification of attacks using ML models. To this end, Sarker compared the performances of ten ML models, including NB, KNN, SVM and DT on two public datasets UNSW-NB15 and NSL-KDD. All models trained in a supervised fashion using samples corresponding both benign and all types of attacks, and RF is said to be achieve the best performance. Subba et al. [173] used MLP with supervised learning to developed a binary intrusion classifier. In [174], Zhang et al. used a Deep Belief Network model to perform intrusion detection. This model is trained using both benign and malicious samples, and the hyper-parameters (e.g. number of hidden layers) of this model are determined via a genetic algorithm prior to the training of the model. In [175], Yin et al. trained a recurrent neural network model with supervised gradient learning to minimize cross entropy for each of the binary and multi-class classification of different attacks. Kunang et al. [176] used MLP with supervised learning and hyper-parameter tuning for binary and multi-class classification of different attacks in two different datasets including NSL-KDD. To this end, MLP is first trained as the encoder part of a deep auto-encoder model and then an output layer is added and parameters updated for each classification task.

All of the works reviewed above used machine learning models with supervised learning; that is, the techniques developed in these works require training samples for both benign and malicious situations. On the other hand, for a real IoT network, it is difficult or often impossible to identify all types of attacks that could target the network and to collect data for these attacks [177]. Therefore, it can be said that although the supervised techniques are promising in terms of the performance, it is crucial to develop an unsupervised technique that can detect different types of – unknown – attacks (i.e. zero-day attacks) [178]. While the technique developed in this thesis fulfills this task, there are also the following examples of works that develop unsupervised intrusion detectors: Alom and Taha in [179] combined K-means clustering with each of the auto-encoder neural network and Restricted Boltzmann Machine, and evaluated the performance of the combinations on NSL-KDD dataset achieving maximum of 92.12% accuracy. Choi et al. [180] compared performance of the basic auto-encoder neural network with its denoising, stacked, and variational versions on NSL-KDD dataset. Their results revealed that the basic auto-encoder outperforms other auto-encoder versions with 91.7% accuracy.

3.3 Compromised Device (Bot) Identification

Various types of Botnet attacks, such as DDoS and brute force attacks, can lead to thousands of infected devices [110] compromising victim devices and turning them into a "bot" via malware [181]. These *malicious* bots can generate fraud information, cause data leaks, and spread a malware. It is reported that 27.7% of all global website traffic in 2021 was generated by bots with malicious intent, and this number is growing rapidly with a 7.3% change from 2018 to 2021 [182].

For instance, in 2016, a massive DDoS Botnet attack that mainly targets IoT devices and whose source code was later released under the name "Mirai", targeted Domain Name System (DNS) provider Dyn [117], rendering Netflix, Reddit, Spotify, and Twitter [183, 184] inaccessible, and gaining malicious access to the servers of leading cybersecurity companies from millions of different IP addresses [185]. Mirai and its variants can still be considered among the most influential and hazardous Botnet – DDoS – attacks on IoT devices and networks in recent years [186].

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 (namely bot). The bot then generates traffic that floods other servers and devices with meaningless requests and/or compromises them.

The Botnet attack compromises new devices and propagates over the IoT network, as illustrated in Figure 3.1. Therefore, one may say that Botnet attacks significantly increase network congestion, power consumption, and processor and memory usage at the device level, hence posing challenges for resource-constrained devices and IoT networks [187].

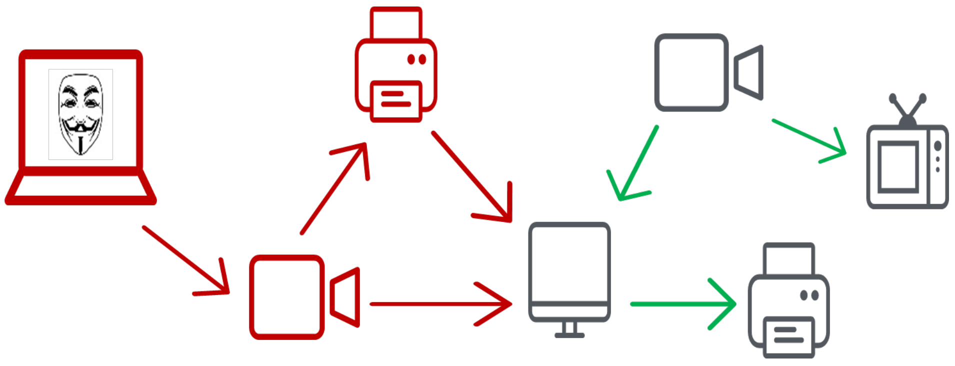

Figure 3.1: Example of the spread of a malware during an ongoing Botnet attack

Given the nefarious impact of Botnet attacks at both network and device levels in IoT networks, it is crucial to identify compromised IoT devices along with the malicious network traffic during an ongoing Botnet attack. While detecting malicious traffic allows reactive actions to alleviate the effects of the attack and (maybe) stop it, identifying the compromised IoT devices paves the way for preventive actions against the spread of malware and Botnet attack.

Some recent works [188–193] focused on detecting compromised IoT devices during Botnet attacks. In Reference [188], Kumar et al. developed an optimization-based technique to detect Mirai-like bots by scanning the destination port numbers in packet headers. In this technique, they analysed the subset of IoT packets to minimize the delay due to the detection of compromised devices. Chatterjee et al. [189] developed 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. In [194], 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 [195], Hristov and Trifonov developed a compromised device identification algorithm using wavelet transformation and Haar filter on the metrics indicating the processor, memory and network interface card usage of an IoT device. In [193], 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. In Reference [190], Nguyen et al. developed an anomaly detection technique to detect compromised devices using a combination of federated learning and language analysis for individual device types identified prior to anomaly 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 [191], Abhishek et al. detected not compromised devices but compromised gateways monitoring the downlink channels in an IoT network and performing binary hypothesis test. In [196], Trajanovski and Zhang developed a framework consisting of honeypots to identify the indicators of compromised devices and botnet attacks. Bahşi et al. in [197] addressed the scalability issues for ML-based Bot detection algorithms by minimizing the number of inputs of ML model via feature selection. In [192], 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.

Chapter 4

Intrusion Detection System with Offline and Quasi-Online Learning

In this chapter, we develop an IDS based on DRNN model with both offline and quasi-online learning. To enable this IDS to learn benign traffic patterns, we create an Auto-Associative memory using DRNN, called Auto-Associative DRNN (AADRNN). For AADRNN based IDS, we develop two training algorithms for offline and quasi-online learning, and we develop simple and original decision making algorithm which is capable of classifying network traffic as either malicious or benign based on only the output of AADRNN. In this chapter, we also show that such IDS can be extended and successfully used for different tasks such as Botnet attack detection, simultaneous detection of different types of attacks, and identification of compromised IoT devices (namely, bots).

The proposed IDS has the following advantages:

1. It is trained only with normal traffic (i.e. its training does not require any attack 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.

2. Single IDS that learned legitimate (benign) traffic can simultaneously provide accurate detection for various types of attacks on an IoT network.

3. The proposed IDS can be trained in parallel to its real-time operation (when it provides detection results) using the ongoing legitimate network traffic.

Section 4.1 of this chapter reviews DRNN, which is the special case of RNN and used as the core ML model used in our IDS. Section 4.2 presents the methodology of our IDS with offline learning algorithm. The IDS with both offline and incremental quasi-online learning is evaluated to detect malicious packets generated by a spreading Botnet attack and those generated simultaneously by various types of attacks. Section 4.3 presents the IDS with quasi-online sequential learning for compromised device identification. Finally, Section 4.4 evaluates the overall performance of IDS for different tasks.

4.1 Review of Deep Random Neural Network

In this chapter, we use RNN with feed-forward architecture and deep learning, called DRNN as the core ML model of our IDS. The RNN model, which is proposed by Gelenbe [198], is a recurrent spiking neural network model with random firing of neurons according to an exponential distribution of excitation rates. In this section, we briefly review the RNN and its special case DRNN in order to introduce the terminologies about DRNN and make our IDS to be understood completely.

The DRNN model that we use is structured in H fully connected feed-forward layers each of which with C_h-neuron clusters of densely coupled neuronal cells (namely, neuron) of RNN. Each cluster c at layer h, denoted by (c,h), has total $N_{(c,h)}$ identical neurons. Since all of the $N_{(c,h)}$ neurons in a given cluster c are statistically identical, we index them directly by (c,h).

As the main property of the RNN, each neuron at cluster (c,h) has an internal state of $k_{(c,h)}(t) \geqslant 0$ at any time t. If $k_{(c,h)}(t)$ at any time t is strictly positive, then a neuron at cluster (c,h) fires a spike (sends a "trigger") after an exponentially distributed random interval of parameter $r_{(c,h)}$ to the set of other neurons in the same cluster (c,h) each of which is selected by probability of $1/N_{(c,h)}$.

We define the probability that any neuron at cluster (c,h) is firing (sending a "trigger"), denoted by $q_{(c,h)}$, as

$$q_{(c,h)} \equiv \lim_{t\to\infty} Prob[k_{(c,h)}(t) > 0]. \tag{4.1}$$

This trigger has one of the following effects on the receiving neuron:

- This trigger causes immediate transmission of another trigger by the receiving neuron to some other neuron in the cluster with probability p, so the internal state of the receiving neuron decreases by -1.
- The receiving neuron absorbs the arriving trigger with probability $(1-p)$ and it's internal state increases by $+1$.

In addition, when a neuron at cluster (c,h) fires (with probability $q_{(c,h)}$), the internal state of this neuron $k_{(c,h)}(t)$ drops by -1.

From all neurons at the previous layer $h-1$, all neurons at cluster (c,h) will receive the following inhibitory input spikes, denoted by $\Lambda_{(c,h)}$:

$$\Lambda_{(c,h)} = \sum_{c'=1}^{C_{h-1}} w_{(c',h-1)}^{(c,h)}\, q_{(c',h-1)}, \tag{4.2}$$

where $w_{(c',h-1)}^{(c,h)} \geqslant 0$ is the connection weight between any neuron at cluster $(c',h-1)$ to any neuron at cluster (c,h). In addition to the triggers, each neuron in a cluster receives an external Poisson flow of excitatory spikes at rate λ^+ and a Poisson flow of inhibitory spikes at rate λ^-. An arriving excitatory spike increases $k_{(c,h)}(t)$ by $+1$. On the other hand, if $k_{(c,h)}(t) > 0$, the arrival of an inhibitory spike decreases $k_{(c,h)}(t)$ by -1.

For simplicity, let focus on the cluster (c,h) and let use n instead of $N_{(c,h)}$, r instead of $r_{(c,h)}$, and Λ instead of $\Lambda_{(c,h)}$. Accordingly, the probability that any neuron at cluster (c,h) is excited, namely $q_{(c,h)}$, is given in [199, 200] as

$$q_{(c,h)} = \frac{\lambda^+ + \frac{r\, q_{(c,h)}\,(n-1)\,(1-p)}{n} S_{(c,h)}}{r + \lambda^- + \Lambda + \frac{r\, q_{(c,h)}\,(n-1)\,p}{n} S_{(c,h)}}, \tag{4.3}$$

where

$$S_{(c,h)} = \sum_{i=0}^{\infty} \left[\frac{q_{(c,h)}\, p\, (n-1)}{n} \right]^i, \quad \text{and} \quad r \geqslant \Lambda \tag{4.4}$$

hence:

$$q_{(c,h)} = \frac{\lambda^+ + \frac{r\, q_{(c,h)}\,(n-1)\,(1-p)}{n - q_{(c,h)}\, p\,(n-1)}}{r + \lambda^- + \Lambda + \frac{r\, q_{(c,h)}\,(n-1)\,p}{n - q_{(c,h)}\, p\,(n-1)}} . \tag{4.5}$$

The analysis can be simplified for large n, to obtain:

$$q_{(c,h)} \approx \frac{\lambda^+ + \frac{r\, q_{(c,h)}\,(1-p)}{1 - q_{(c,h)}\, p}}{r + \lambda^- + \Lambda + \frac{r\, q_{(c,h)}\,p}{1 - q_{(c,h)}\, p}} , \tag{4.6}$$

resulting in the second degree polynomial in $q_{(c,h)}$ given by:

$$0 = q_{(c,h)}^2\,(\lambda^- + \Lambda) - q_{(c,h)}[p\,(r + \lambda^+) + \lambda^- + \Lambda] + \lambda^+ . \tag{4.7}$$

Since $q_{(c,h)}$ is a probability, we obtain the following positive solution and consider it to be the activation function, denoted by $\zeta(\Lambda)$ of our DRNN model for input Λ to cluster (c,h):

$$\zeta(\Lambda) = q_{(c,h)} = \frac{p\,(r+\lambda^+) + \lambda^- + \Lambda}{2\,[\lambda^- + \Lambda]} - \sqrt{\left(\frac{p\,(r+\lambda^+) + \lambda^- + \Lambda}{2\,[\lambda^- + \Lambda]}\right)^2 - \frac{\lambda^+}{\lambda^- + \Lambda}} , \tag{4.8}$$

where – recall that – the input $\Lambda = \sum_{c'=1}^{C_{h-1}} w_{(c',h-1)}^{(c,h)}\, q_{(c',h-1)}$ for the cluster (c,h). It is also provided that $\lambda^+\,(1-p) < \lambda^- + p\,r + \Lambda$, and the condition $\lambda^+ < \lambda^- + p\,r$ is sufficient.

4.2 Malicious Traffic Detection with Offline and Quasi-Online Learning

This section describes our IDS, which is shown in Figure 4.1, based on AADRNN with an offline and incremental learning algorithm. This IDS is composed of three modules, namely Metric Extraction and Preprocessing, AADRNN, and Statistical Whisker based Benign Classification (which is the decision maker). These modules respectively calculates original metrics of the actual network traffic and prepares them to be processed by AADRNN, calculates metrics for expected legitimate traffic, and compares the metrics for expected and actual traffic to make a decision on the probability that the current traffic is malicious.

4.2.1 Metric Extraction and Preprocessing

In order to capture the patterns of IoT traffic that AADRNN can learn and IDS observe the footprints of attacks, we extract metrics via the Metric Extraction and Preprocessing module. This module first calculates the vector of M metrics for each packet (or burst of traffic) i, denoted by $x_i = [x_i^1, \ldots, x_i^m, \ldots, x_i^M]$. Note that each of these metrics, namely x_i^m, defined in the design phase of IDS – prior to the training or execution – considering the characterizations of both traffic and targeted attacks.

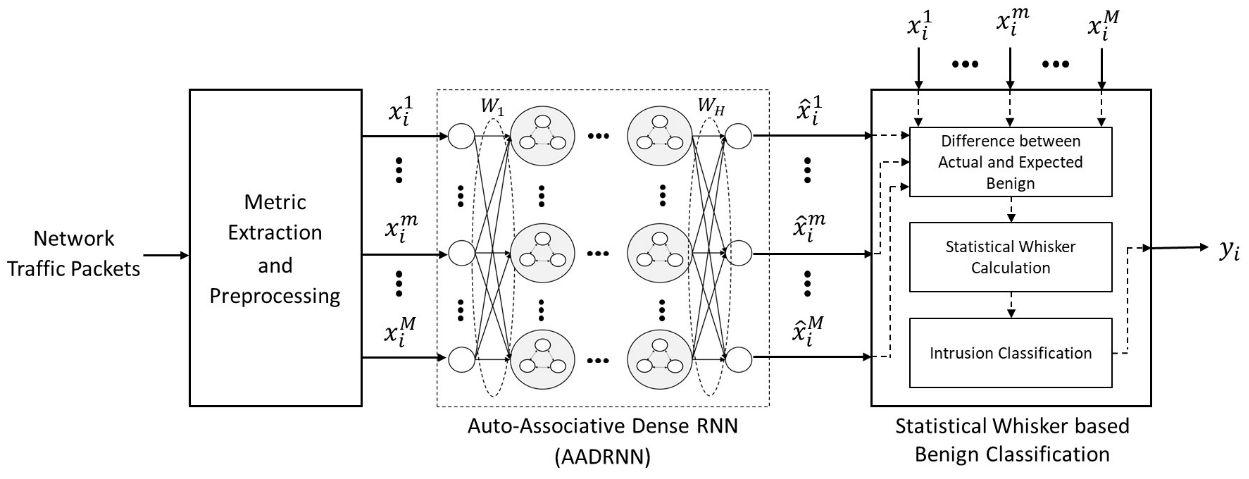
Network Traffic Packets
Metric Extraction and Preprocessing
x_i^1
x_i^m
x_i^M
W_1
W_H
$\hat{x}_i^1$
$\hat{x}_i^m$
$\hat{x}_i^M$
Auto-Associative Dense RNN (AADRNN)
Difference between Actual and Expected Benign
Statistical Whisker Calculation
Intrusion Classification
Statistical Whisker based Benign Classification
y_i

Figure 4.1: Architecture of the DRNN based attack detector with its three modules: Metric Extraction and Preprocessing, AADRNN, and Statistical Whisker based Benign Classification

Preprocessing Metrics:

This module preprocesses (via simple/lightweight operations) the extracted metrics in order to prepare them for the input of AADRNN. To this end, if the set includes non-numerical (categorical) metrics, they are first encoded into numerical features. For each non-numerical metric m, the possible set of unique values, denoted by $\mathcal{U}_m$, is determined from the available dataset $\mathcal{D}_{\text{train}}$ (which shall also be used for offline learning). Then, for each unique value $u \in \mathcal{U}_m$, a positive integer in $[1, |\mathcal{U}_m|]$ is assigned. As soon as each non-numerical metric is converted to numerical, the value of the numerical metric, assigned into x_i^m, is normalized by min-max scaling as

$$x_i^m \leftarrow \frac{x_i^m - \min\limits_{i \in \mathcal{D}_{\text{train}}} x_i^m}{\max\limits_{i \in \mathcal{D}_{\text{train}}} x_i^m - \min\limits_{i \in \mathcal{D}_{\text{train}}} x_i^m} \tag{4.9}$$

Determination of the Metrics for Mirai Botnet Attacks:

We now explain the metrics that are originally defined aiming to detect Mirai Botnet attack. Recall that Mirai is a type of attack that spreads to IoT devices over the network. In addition, every device infected by Mirai generates more traffic than usual, causing a DDoS in the network, and the traffic pattern of that device (e.g. the sizes or the transmission intervals of packets) shows different characteristics. Thus, we intuitively know that, when a device is affected by the Mirai attack, it will try to increase the total size of the traffic to overload the network via generating more packets.

Accordingly, in order to point out the most important footprints of the Mirai attacks, we define the following metrics:

- **Metric 1**: The total size of the last P transmitted packets,
- **Metric 2**: The average inter-transmission times of the packets over the last P packets, (The inter-transmission time of a packet is the time passed between the transmission of this packet and that of the previous packet that is generated by the same source.)

- **Metric 3**: Total number of packets that are transmitted in a time window with a duration of T.

These metrics are specifically defined to represent network traffic in a way that the differences between attack and normal traffic become more visible while they can be calculated using only the header information of the traffic packets. Therefore, these metrics can be easily calculated without the need for any sensitive or device-specific information, thus preventing IDS from making biased decisions, remaining anonymous regarding packet content and communicating devices, and suitable for real-time operation on lightweight systems. We shall also measure the effectiveness of these metrics in Section 4.4.2 on the detection of Mirai attacks.

4.2.2 Auto-Associative Memory with DRNN (AADRNN)

As shown in Figure 4.1, the central part of our IDS utilizes an auto-associative memory that is created by a trained DRNN model. During the real-time operation of the IDS, from the metrics x_i, this model calculates the metrics $\hat{x}_i$ representing the expected x_i value when i is benign. To this end, i.e. to create an auto-associative memory on the benign network traffic, we train the DRNN model to learn the data collected during the normal operation of the IoT network with no intrusion attempt.

Structure of DRNN Used:

As the methodology of RNN and its special case DRNN is already reviewed in Section 4.1, we now present the particular structure of DRNN which is used in this work. In the IDS, we use a DRNN model which consists of H layers (the hidden and output layers) and a rectangular structure with equal units at each layer. To this end, each hidden layer $h \in \{1, \ldots, H-1\}$ is comprised of M (which is equal to the the number of inputs of DRNN) clusters of RNN cells, and the output layer is comprised of the same number of linear neurons. We let W_h denote the $[(M+1) \times M]$ matrix of connection weights (including biases) between the layer $h-1$ and layer h for $h \in \{1, \ldots, H\}$; that is, W_h is the multiplier for the inputs of layer h. In addition, $\zeta(\cdot)$ denote the activation function of a DRNN cluster. Note that we obtain the AADRNN as this DRNN model learned the benign traffic to create auto-associative memory.

Accordingly, in real-time operation, the forward pass of AADRNN model for the given input vector x_i is computed as:

$$\hat{x}_{(i,1)} = \zeta([x_i, 1]\, W_1) \tag{4.10}$$

$$\hat{x}_{(i,h)} = \zeta([\hat{x}_{(i,h-1)}, 1]\, W_h) \qquad \forall h \in \{2, \ldots, H-1\}, \tag{4.11}$$

$$\hat{x}_i = [\hat{x}_{(i,H-1)}, 1]\, W_H, \tag{4.12}$$

where $\hat{x}_{(i,h)}$ is the output of layer h for packet i, and the term $[x_i, 1]$ or $[\hat{x}_{(i,h)}, 1]$ indicates that 1 is added to the input of each layer as a multiplier of the bias.

Offline Learning Algorithm:

In order create an auto-associative memory that can retrieve expected normal metrics from the metrics of either normal or malicious traffic, we train the DRNN using only normal traffic data by our offline learning algorithm which is based on the semi-supervised algorithm presented in [200]. During the learning process, DRNN learns to retrieve the metrics of benign traffic from their noisy versions.

First, let X be the matrix of metrics for normal traffic packets in the training dataset $\mathcal{D}_{\text{train}}$, and $\hat{X}_h$ be the matrix of the corresponding outputs of layer h calculated with the learned weights as given in (4.10)-(4.12):

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_{|\mathcal{D}_{\text{train}}|} \end{bmatrix}, \quad \hat{X}_H = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_i \\ \vdots \\ \hat{x}_{|\mathcal{D}_{\text{train}}|} \end{bmatrix}, \quad \text{and} \quad \hat{X}_h = \begin{bmatrix} \hat{x}_{(1,h)} \\ \vdots \\ \hat{x}_{(i,h)} \\ \vdots \\ \hat{x}_{(|\mathcal{D}_{\text{train}}|,h)} \end{bmatrix} \quad \forall h \in \{1, \dots, H-1\} \tag{4.13}$$

Then, within the offline learning algorithm, for each layer $h \in \{1, \dots, H\}$, we compute W_h as

$$W_h = \underset{\{W:\, W \geqslant 0\}}{\operatorname{argmin}} \left(\left|\left| [adj(\zeta(\hat{X}_{h-1} W_R)), \mathbf{1}_{(|\mathcal{D}_{\text{train}}| \times 1)}]\, W - \hat{X}_{h-1} \right|\right|_{L_2}^2 + ||W||_{L_1} \right), \tag{4.14}$$

where we take $\hat{X}_0 = X$, $\mathbf{1}_{(|\mathcal{D}_{\text{train}}| \times 1)}$ is a column vector of ones with length $|\mathcal{D}_{\text{train}}|$, and the $(M \times M)$ weight matrix W_R is randomly generated with elements in the range $[0, 1]$. On the other hand, $adj(A)$ is the linear mapping of the elements of matrix A into the range $[0, 1]$ then applies the z-score (standard score), and adds a positive constant to remove negativity.

In our particular design of offline learning algorithm, we use Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [201] to solve the minimization problem (4.14) of cost function with L1 regularization. To this end, FISTA iterates the operations reviewed in (4.16) for a constant number of times. After its iterations are completed, we finally normalize each resulting weight matrix W_h:

$$W_h \leftarrow 0.1 \frac{W_h}{\max(\hat{X}_h)}\,. \tag{4.15}$$

<u>Review of FISTA operations for a single iteration t :</u>

Let $A = [adj(\zeta(\hat{X}_{h-1} W_R)), \mathbf{1}_{(|\mathcal{D}_{\text{train}}| \times 1)}]$, and a constant $\alpha = 0.001$

$\mu_0 = 0$, and W and $Z_1$are initialized as zero matrices (4.16)

$$\mu_t = \frac{1 + \sqrt{1 + 4\mu_{t-1}^2}}{2}, \quad \xi_t = \frac{1 - \mu_t}{\mu_{t+1}}$$

$$\begin{aligned} Z_{t+1} &= prox_{(\alpha/\beta)||\cdot||_1}(W - \frac{1}{\beta}\nabla||AW - \hat{X}_{h-1}||_2^2) \\ &= prox_{(\alpha/\beta)||\cdot||_1}(W - \frac{1}{\beta}[2A^T(AW - \hat{X}_{h-1})]) \end{aligned}$$

$$W \leftarrow \max\Big[0,\ (1 - \xi_t) Z_{t+1} + \xi_t Z_t\Big]$$

where, we added the last step to the original FISTA in order to always obtain positive weights. In addition,

$$prox_{(\alpha/\beta)||\cdot||_1}(B) = U, \ \ U_i = \max\Big[0,\ |B_i| - \frac{\alpha}{\beta}\Big]\, sign(B_i),$$

β is the maximum eigenvalue of the argument B, and $sign(B_i)$ returns $+1$ if $B_i > 0$, -1 if $B_i < 0$, and 0 if $B_i = 0$.

4.2.3 Statistical Whisker based Benign (Non-Attack) Classification

In order to classify the traffic features as attack or benign, we now design a classifier, which compares the output of AADRNN ($\hat{x}_i$) with the metrics of actual traffic (x_i) based on the statistical whisker calculated from learning dataset. That is, by this classifier, we aim whether the metrics of the actual traffic are significantly different than the metrics expected under normal traffic.

Decision Making:

During the decision making for each packet i in real-time operation of IDS, we first calculate the absolute difference between x_i^m and $\hat{x}_i^m$ for each metric m:

$$z_i^m = |x_i^m - \hat{x}_i^m| \qquad \forall m \in \{1, \dots, M\} \tag{4.17}$$

Then, the total number of metrics with abnormal values, denoted by ψ_i, (i.e. the number of metrics that are significantly different than the expected metrics) is computed:

$$\psi_i = \sum_{m \in \{1, \dots, M\}} \mathbf{1}(z_i^m > w_m), \tag{4.18}$$

where w_m is the value of whisker that is calculated from the available dataset. Note that $\mathbf{1}(\Xi) = 1$ if the statement Ξ is true, and $\mathbf{1}(\Xi) = 0$ otherwise. That is, in (4.18), the metric m is considered to have an abnormal value indicative of an attack when $z_i^m > w_m$.

Finally, the packet i is considered an attack if there are more than θ features with abnormal values:

$$y_i = \mathbf{1}(\psi_i > \theta). \tag{4.19}$$

Determination of Statistical Whisker:

Using the training data, $\mathcal{D}_{\text{train}}$, which consists of only benign traffic features, we determine the values of θ and w_m for each metric m. To this end, for each metric m, the value of the absolute difference z_i^m is computed for all $i \in \mathcal{D}_{\text{train}}$ using (4.17). Then, the lower quartile Q_l^m and upper quartile Q_u^m of $\{z_i^m\}_{i \in \mathcal{D}_{\text{train}}}$ are calculated as

$$\mathcal{D}_{\text{lower}}^m \equiv \{i : z_i^m < \text{median}(\{z_j^m\}_{j \in \mathcal{D}_{\text{train}}})\}, \; \forall m \in \{1, \dots, M\} \tag{4.20}$$

$$Q_l^m = \text{median}(\{z_i^m\}_{i \in \mathcal{D}_{\text{lower}}^m}), \; \forall m \in \{1, \dots, M\}, \tag{4.21}$$

and

$$\mathcal{D}_{\text{upper}}^m \equiv \{i : z_i^m > \text{median}(\{z_j^m\}_{j \in \mathcal{D}_{\text{train}}})\}, \; \forall m \in \{1, \dots, M\} \tag{4.22}$$

$$Q_u^m = \text{median}(\{z_i^m\}_{i \in \mathcal{D}_{\text{upper}}^m}), \; \forall m \in \{1, \dots, M\} \tag{4.23}$$

Using Q_l^m and Q_u^m, the whisker w_m for the upper quartile is calculated as

$$w_m = Q_u^m + \frac{3}{2}(Q_u^m - Q_l^m) \qquad \forall m \in \{1, \dots, M\} \tag{4.24}$$

Since the training data contains only benign traffic, θ must be selected to classify training samples as benign traffic. Meanwhile, we should also consider that the training data may include some outlier samples.

Therefore, we determine θ to classify the majority but not all of the training samples as benign traffic, and we set the value of θ to the mean of ψ_i (i.e. the number of abnormal metrics) plus the standard deviation of ψ_i in the training data:

$$\theta = \mu_\psi + 2\sigma_\psi, \tag{4.25}$$

where

$$\mu_\psi = \frac{\sum_{i \in \mathcal{D}_{\text{train}}} \psi_i}{|\mathcal{D}_{\text{train}}|}, \text{ and } \sigma_\psi = \sqrt{\frac{\sum_{i \in \mathcal{D}_{\text{train}}} (\psi_i - \mu_\psi)^2}{|\mathcal{D}_{\text{train}}|}} \tag{4.26}$$

4.2.4 Incremental Learning for Malicious Traffic Detection

For malicious traffic detection, we revise the classification module and the learning algorithm of the IDS presented in Section 4.2. That is, we use the AADRNN structure presented in Section 4.2.2 with the metrics proposed in Section 4.2.1. For the classification module, as large enough training set is not available for offline learning, we use a simpler classifier:

$$y_i = \mathbf{1}\left(\frac{1}{M} \sum_{m=1}^{M} |x_i^m - \hat{x}_i^m| > \theta\right). \tag{4.27}$$

In this section, we present the incremental training of AADRNN for malicious traffic detection. Using incremental quasi-online learning, we will now enable the use of an attack detector without requiring the offline collection of benign traffic as well. To this end, we develop an Incremental Semi-Supervised Learning (ISSL) algorithm which combines offline semi-supervised learning algorithm developed in [200] with sequential learning algorithm developed in [202] and is comprised of initialization and incremental quasi-online learning stages.

The ISSL algorithm is executed at the end of the transmission of every I packets; that is, ISSL works on packet transmission windows of length I. The end of the last packet transmission window k is associated with the transmission of packet i, such that i is multiple of I.

The initialization stage is considered as the cold-start of the proposed IDS. Thus, the transmission of the first I packets are known to be benign packets since the network is assumed to be working in cold-start for the first I packets. Using these packets, the connection weights of AADRNN, $\{W_h\}_{h \in \{1,\dots,H\}}$, are initialized. To this end, at the end of the first window, $k = 1$, i.e. after the transmission of the first I packets via the semi-supervised learning algorithm presented in Section 4.2.2.

At the end of each transmission window $k > 1$, only the connection weights of the output layer of AADRNN, W_H, are updated via the second stage of ISSL to learn the metrics of the benign packets within window k. In order to detail this incremental learning process, let define the operation matrix O_k, which is initialized as the inverse of the Gram matrix:

$$O_1 \equiv \left[(\hat{X}_k^{\text{train}})^T \hat{X}_k^{\text{train}} \right]^{-1} \tag{4.28}$$

where $\hat{X}_k^{train}$ is the matrix of metrics for benign packets within the last I packets which are determined based on the output of IDS as

$$\hat{X}_k^{\text{train}} = \left\{ \hat{x}_j : y_j = 0, \forall j \in \{i - I + 1, \dots, i\} \right\}. \tag{4.29}$$

Note that if all of the last I packets are detected as malicious, no training will be performed until the end of the first window with normal traffic.

If there is at least one benign packet in $\hat{X}_k^{\text{train}}$, we first compute the value of O_k for the current window k as

$$O_k = O_{k-1} - O_{k-1}(\hat{X}_{k,H-1}^{\text{train}})^T \Big[I + \hat{X}_{k,H-1}^{\text{train}} O_{k-1} (\hat{X}_{k,H-1}^{\text{train}})^T\Big]^{-1} \hat{X}_{k,H-1}^{\text{train}} O_{k-1}. \tag{4.30}$$

Then, we update W_H as

$$W_H \leftarrow W_H + O_k(\hat{X}_{k,H-1}^{\text{train}})^T (X_k - \hat{X}_{k,H-1}^{\text{train}} W_H) \tag{4.31}$$

4.3 Compromised Device Identification System with Sequential Quasi-Online Learning

We further develop an IDS with sequential quasi-online learning in order to enable the training of the IDS in parallel to its real-time operation with very low human intervention for compromised device (i.e. bot) identification in an IoT network. In order to secure an IoT network during an ongoing DDoS (especially, Botnet) attack, both malicious traffic detection and compromised device identification tasks are crucial as they pave the way to prevent botnet attack from spreading over the network.

The IDS with quasi-online learning ability uses the traffic that is collected in real-time operation of the network and identified as being benign. Thus, it does not require the collection of either prior attack or normal (benign) traffic, except during the cold-start of the network. The sequential quasi-online learning enables IDS to adapt the naturally time-varying characteristics of the network traffic by updating the parameters of AADRNN automatically and periodically as a function of the traffic it encounters.

Figure 4.2 displays the design of the IDS enhanced with sequential quasi-online learning for the identification of a compromised IoT device i, which is called Compromised Device Identification System (CDIS). Different than the IDS for malicious traffic detection explained in the previous section, it now has a sequential semi-supervised learning algorithm that updates the parameters of AADRNN periodically at the end of every time window k, and both the metric extraction and decision maker (called, infection classifier) modules are reoriented to identify compromised devices. On the other hand, the core ML model remains the same as AADRNN; therefore, this section details only the traffic metrics and sequential learning algorithms for compromised device identification.

In our approach, a distinct instance of the CDIS, which is shown in Figure 4.2, is installed on each device i to determine if that device is compromised. The CDIS makes decision on a basis of time window during which it observes traffic packets and calculate statistical metrics. At each time window k, 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 $y_{i,k}$ for device i.

4.3.1 Defining the Traffic Metrics

Since the aim of the Metric Extraction and Preprocessing module in Figure 4.2 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. To this end, we now present a new original set of metrics for both the traffic received and transmitted by each IoT device i, inspired from the three metrics that are defined for malicious traffic

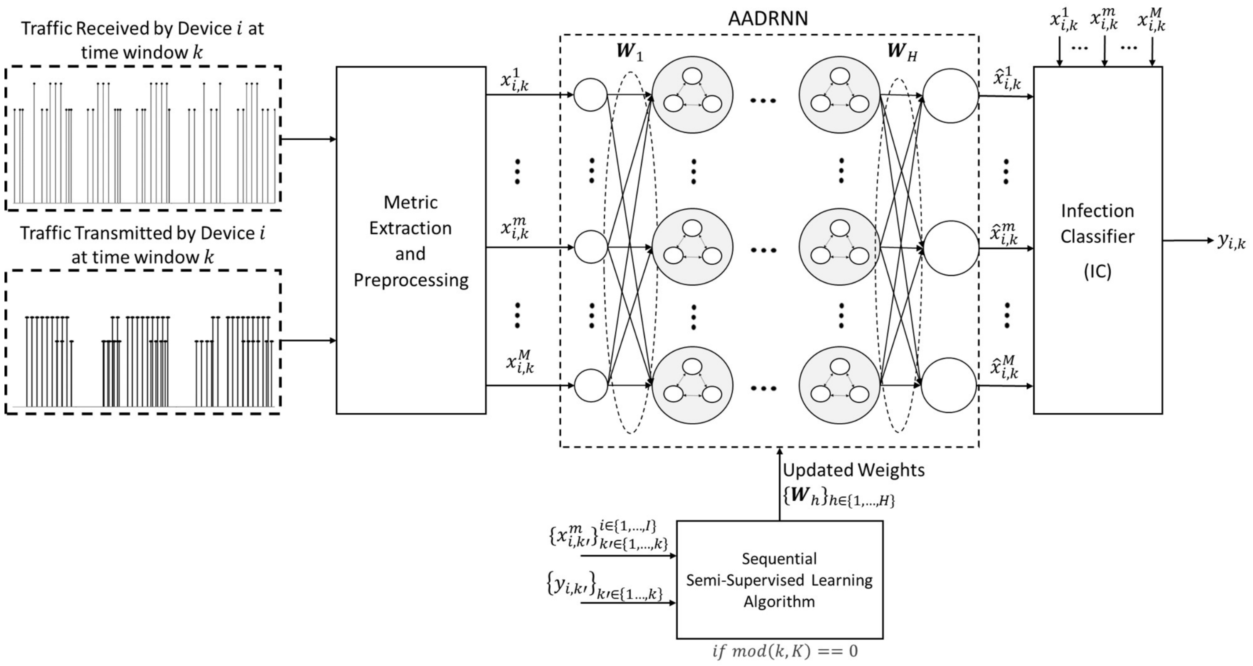
Traffic Received by Device i at time window k
Traffic Transmitted by Device i at time window k
Metric Extraction and Preprocessing
$x_{i,k}^1$
$x_{i,k}^m$
$x_{i,k}^M$
AADRNN
W_1
W_H
$\hat{x}_{i,k}^1$
$\hat{x}_{i,k}^m$
$\hat{x}_{i,k}^M$
Infection Classifier (IC)
$y_{i,k}$
Updated Weights $\{W_h\}_{h\in\{1,\dots,H\}}$
$\{x_{i,k'}^m\}_{k'\in\{1,\dots,k\}}^{i\in\{1,\dots,I\}}$
$\{y_{i,k'}\}_{k'\in\{1\dots,k\}}$
Sequential Semi-Supervised Learning Algorithm
$if\ mod(k,K) == 0$

Figure 4.2: The design of the CDIS with sequential quasi-online learning exemplified for the task of compromised device identification

detection in Section 4.2.1. These new metrics are also chosen to address Botnet attacks (especially, that caused by Mirai malware) and to capture the effects of an attack on the network traffic.

Recall that since Botnet 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. Indeed, in order to identify the sources of attacks (and the effect of attack), it is important to analyze the traffic received from each source individually, rather than the overall aggregated traffic received from all sources. In this way, observing the traffic from a compromised device can be an effective means of detecting the existence of an infection. Thus, we have selected some statistics (metrics) to summarize the traffic sent or received by each individual device.

First, $pk(t, s, d)$ be the packet sent by source device s to destination d at time t, and $|pk(t, s, d)|$ denote the length of this packet in bytes. We also let T be the duration of a time window over which the metrics will be computed. S denotes the set of all source devices while D denotes the set of all destination devices. Accordingly, the collection of the packets that have been sent by device s to any other device d the network in time window k, denoted by $P_k^{s,d}$, is computed as

$$P_k^{s,d} \equiv \{pk(t, s, d) : \ (k-1)T \leqslant t < kT\}, \tag{4.32}$$

and the set of all packets arriving to d in window k is:

$$P_k^{S,d} \equiv \sum_{s\in S} P_k^{s,d} \,. \tag{4.33}$$

Also let packets sent by the set of nodes S have a maximum and minimum length L_S^M and L_S^m in 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 Ω_s in bytes/second.

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

- **Received Traffic Metric 1 (RTM1)**: The normalized average size of packets received by device i from all the sources in time window k:

$$x^1_{i,k} = \frac{\sum_{p \in P^{S,i}_k} |p|}{\sum_{s \in S} L^M_s \times |P^{s,i}_k|} \tag{4.34}$$

- **RTM2**: The normalized maximum size of any packet received at node i from any of the sources in time window k:

$$x^2_{i,k} = \max_{p \in P^{S,i}_k} \frac{|p|}{L^M_S} \,. \tag{4.35}$$

  The use of L^M_S 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 [145]. On the other hand, DoS attacks that aim at creating congestion on links would have to be rather long.

- **RTM3**: The average number of packets received from all sources that have sent packets to i in time window k:

$$x^3_{i,k} = \frac{|P^{S,i}_k|}{\sum_{s \in S} \mathbf{1}[|P^{s,i}_k| > 0]} \,. \tag{4.36}$$

  Note that the denominator term in the above expression can be computed iteratively in a very efficient manner, so that $x^3_{i,k}$ is obtained directly from the terms in $x^3_{i,k-1}$.

- **RTM4**: The normalized maximum number of packets received from any single source in time window k:

$$x^4_{i,k} = \frac{\max_{s \in S} |\mathcal{P}^{s,i}_k|}{\max_{u:\, 1 \leqslant u \leqslant k} [\, \max_{s \in S} |\mathcal{P}^{s,i}_u| \,]} \,. \tag{4.37}$$

We now define the other traffic metrics 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 devices:

- **Transmitted Traffic Metric 1 (TTM1)**: The normalized total amount of traffic transmitted by device i in time window k:

$$x^5_{i,k} = \frac{1}{\Omega_i \times T} \sum_{d \in D} \sum_{p \in \mathcal{P}^{i,d}_k} |p| \ , \tag{4.38}$$

- **TTM2**: The normalized total number of packets that are transmitted by i in time window k:

$$x^6_{i,k} = \frac{L^m_i}{\Omega_i \times T} \sum_{d \in D} |\mathcal{P}^{i,d}_k| \ , \tag{4.39}$$

  where the use of L^m_i in TTM2 is due to the fact that the maximum number of packets that may be transmitted by node i in time T is $\frac{\Omega_i \times T}{L^m_i}$.

4.3.2 Structure of DRNN Used

We use the same DRNN structure given in Section 4.2.2 to create an auto-associative memory. Recall that this DRNN is comprised of H layers with M units at each layer, where M is the number of metrics (i.e. inputs of DRNN). The units in each hidden layer $h \in \{1, \dots, H-1\}$ are the clusters of RNN cells while the units of the output layer is linear neurons. At each time window k, each layer h has the matrix W_h^k of input weights which are sequentially learned to create an auto-associative memory.

Let $x_{i,k}$ be the input vector of the AADRNN (i.e. the vector of the metrics) for device i at window k such that $x_{i,k} = [x_{i,k}^1, \dots, x_{i,k}^6]$, and $\hat{x}_{(i,k,h)}$ be the output of layer h at time window k when the network input is $x_{i,k}$ and the connection weights are $\{W_h^{i,k}\}_{h\in\{1,\dots,H\}}$. We also let $\hat{x}_{i,k}$ denote the output of AADRNN, i.e. $\hat{x}_{i,k} = \hat{x}_{(i,k,H)}$. Accordingly, the forward pass of the AADRNN in CDIS is as follows:

$$\hat{x}_{(i,k,1)} = \zeta([x_{i,k},\, 1]\, W_1^{i,k}), \tag{4.40}$$

$$\hat{x}_{(i,k,h)} = \zeta([\hat{x}_{(i,k,h-1)}, 1]\, W_h^{i,k}) \qquad \forall h \in \{2, \dots, H-1\}, \tag{4.41}$$

$$\hat{x}_{i,k} = [\hat{x}_{(i,k,H-1)}, 1]\, W_H^{i,k}\ . \tag{4.42}$$

4.3.3 The Infection Classification

In order to classify the analyzed device i at time window k as compromised or uncompromised, we now compute the deviation of the actual metrics from the expected metrics calculated by AADRNN. This is done by computing the maximum of all the differences between the elements of the input vector $x_{i,k}$ and the elements of the output vector $\hat{x}_{i,k}$:

$$\Psi_{i,k} = \max_{m\in\{1,\dots,M\}} \big(\, |x_{i,k}^m - \hat{x}_{i,k}^m|\, \big). \tag{4.43}$$

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

$$y_{i,k} = \begin{cases} 1 \text{ (compromised)}, & \text{if } \Psi_k^i \geqslant \gamma_i \\ 0 \text{ (uncompromised)}, & \text{otherwise} \end{cases} \tag{4.44}$$

Since we are carrying out quasi-online learning, without prior offline training using the ground truth, the output $y_{i,k}$ not only provides decisions, but it also allows us to operate the sequential quasi-online auto-associative algorithm given below.

4.3.4 Sequential Quasi-Online Learning

As we use sequential learning to create an auto-associative memory from DRNN (namely, AADRNN), each weight matrix $W_h^{i,k}$ is updated based on only the benign traffic at the end of each time window k after input $x_{i,k}$ has been processed. At the end of each window k, if $y_{i,k} = 1$, i.e. $x_{i,k}$ is estimated to contain an attack, then we do not update the weights, i.e. $W_h^{i,k} \leftarrow W_h^{i,k-1}$, $1 \leqslant h \leqslant H$.

If $y_{i,k} = 0$ (i.e. i is estimated to be not compromised), we solve the following problem via FISTA similarly with the offline learning in Section 4.2.2:

$$W_h^{i,k} = \underset{\{W:\, W\geqslant 0\}}{\operatorname{argmin}} \left(\left|\left| [adj(\zeta(\hat{X}_{(i,k,h-1)}^{\text{train}} W_R\,)), \mathbf{1}_{k\times 1}]\, W - \hat{X}_{(i,k,h-1)}^{\text{train}} \right|\right|_{L_2}^2 + ||W||_{L_1} \right), \tag{4.45}$$

where $\hat{X}^{\text{train}}_{(i,k,h)}$ is the matrix that collects the output vectors of layer h over all time windows $k' \in \{1, \dots, k\}$ when $y_{i,k'} = 0$:

$$\hat{X}^{\text{train}}_{(i,k,h)} = \{\hat{x}_{(i,k',h)} : y_{i,k'} = 0, \forall k' \in \{1, \dots, k\}\}, \tag{4.46}$$

and

$$\hat{X}^{\text{train}}_{(i,k,0)} = \{x_{(i,k')} : y_{i,k'} = 0, \forall k' \in \{1, \dots, k\}\}. \tag{4.47}$$

In practice, when $y_{i,k} = 0$, we can easily compute $\hat{X}^{\text{train}}_{(i,k,h)}$ based on only its previous value:

$$\hat{X}^{\text{train}}_{(i,k,h)} = \begin{bmatrix} \hat{X}^{\text{train}}_{(i,k-1,h)} \\ \hat{x}_{(i,k,h)} \end{bmatrix} \tag{4.48}$$

After FISTA is performed for a predefined number of iterations to solve (4.45), we normalize the resulting weight matrix $W^{i,k}_h$:

$$W^{i,k}_h \leftarrow 0.1 \frac{W^{i,k}_h}{\max\left(\hat{X}^{\text{train}}_{(i,k,h)}\right)} . \tag{4.49}$$

4.4 Overall Performance Evaluations

4.4.1 Experimental Setup

Datasets:

During our experimental study, we use various datasets in order to evaluate the performance of our IDS for three different scenarios for detecting malicious botnet traffic, various attack types simultaneously, or compromised device (bot) identification:

- For malicious traffic detection and compromised device identification during Mirai Botnet attack, we use the data from the publicly available **Kitsune** dataset [203, 204]. This dataset contains $764,137$ packet transmissions cover a consecutive time period of roughly 7137 seconds (nearly 2 hours) including both normal and attack traffic. There are 107 distinct IP addresses that either sent or receive traffic and for each of which we will perform infection detection.

- In order to evaluate the performance of our IDS for detecting malicious traffic generated by various types of attacks, we use **KDD Cup'99** dataset [205]. This dataset includes both benign and attack (intrusion) traffic. It contains three different subsets: the training set, the smaller training set reduced to 10% of the total training set, and the test set. These subsets are respectively comprised of "$4,898,431$", "$494,021$", and "$311,029$" samples with 41 features related to network traffic.

- During the performance evaluation of IDS for identifying compromised devices (i.e. CDIS), in addition to the Mirai Botnet data from Kitsune dataset, we also use SYN DoS data from the Kitsune dataset, which contains $2,771,276$ packets transmitted in about 53 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 [206]. 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.

Hardware:

All experiments are performed on a workstation with 32 Gb RAM and an AMD 3.7 GHz (Ryzen 7 3700X) processor. In addition, all models and algorithms are implemented in Python.

Techniques Used for Comparison:

In this section, we compare the performance of AADRNN-based IDS with the following state-of-the-art ML models: Linear Regression (LR), Least Absolute Shrinkage and Selector Operator (Lasso), KNN, MLP and LSTM. Unless otherwise specified in each experiment, these models are used in place of the AADRNN in our IDS as follows:

- We first selected the **Simple Threshold** method as the simplest benchmark for malicious traffic detection. In this method, we apply threshold on the mean of the metric values, and we use this model as a complete IDS, not in place of the AADRNN, unlike the (below) ML models. In order to achieve the best performance of the Simple Threshold, we search for the best value of the threshold on the test set which includes both normal and attack traffic.
- We then use two different linear ML models, **Linear Regression (LR)** and **Least Absolute Shrinkage and Selector Operator (Lasso)**. We 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. Moreover, we implement both LR and Lasso using the scikit-learn library [207] on Python.
- Early research [157] showed that the **K-Nearest Neighbours Regressor (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. Since KNN requires at least as many neighbors as the number of samples, the number of neighbors in KNN is set to the minimum of the number of metrics M and the number of training samples.
- A feed-forward **Multi-Layer Perceptron (MLP)** with two hidden layers with M neurons, followed by an output layer also with M neurons is used. All nuerons in MLP have sigmoidal activation function. Both training and execution was performed using Keras on Python, where the MLP is trained via the Adam optimizer.
- Lastly, we use **Long-Short Term Memory (LSTM)** with a single LSTM layer – M units – and three fully connected layers including the output layer which is comprised of M neurons, and sigmoidal activation function used for all neurons. Training and execution of is also performed using Keras on Python and trained via Adam optimizer.

4.4.2 Malicious Traffic Detection with Offline Learning

We first evaluate the performance of the IDS with offline learning proposed in Section 4.2. To this end, in this subsection, we start with analyzing the importance of proposed metrics and effects of using separated metrics on the performance of the IDS. Then, we show the performance of the IDS for varying value of decision threshold θ; as a result, the best value of θ is obtained. During the performance evaluation for detecting Botnet attack packets, we set $p = 0.05$, $r = 0.001$, $\lambda^+ = \lambda^- = 0.1$, $P = 500$ packets, and $T = 100$ seconds.

Analysis on the Importance and Effects of Metric Candidates:

In order to analyze the importance of each feature candidate for the detection of Mirai botnet attacks in the Kitsune dataset, we first perform the following analysis. For each metric, we first compute Pearson correlation coefficient [208] between that metric and the attack label. This coefficient measures the strength and the direction of the linear relationship between the considered metric and the attack label. Since we desire to measure only the importance of "Metric m" for the detection of attack, we need only the strength of the relationship so we let ρ_m denote the absolute value of the coefficient for Metric m.

We also compute the other coefficients as the F-ratios [209] that are calculated via the Analysis of Variance (ANOVA) method. The value of the F-ratio corresponding to metric m measures the statistical significance of that metric for the decision of attack. We let f_m denote the normalized F-ratio for metric m in the range $[0, 1]$.

We then calculate the overall importance of the metrics as the combination of Pearson correlation coefficient and ANOVA:

$$\alpha_m = \frac{\rho_m + f_m}{\sum_{m\in\{1,\dots,M\}} (\rho_m + f_m)} \qquad \forall m \in \{1, \dots, M\}. \tag{4.50}$$

In Figure 4.3, we present the value of α_m as well as the values of ρ_m and f_m for each Metric m, $m \in \{1, 2, 3\}$. In this figure, we see that the importance of each of Metrics 1 and 3 is higher than that of the Metric 2 with respect to each of α_m, ρ_m and f_m. In addition, the values of α_1, ρ_1 are close to those of α_3, ρ_3 although there is a significant gap between f_1 and f_3.

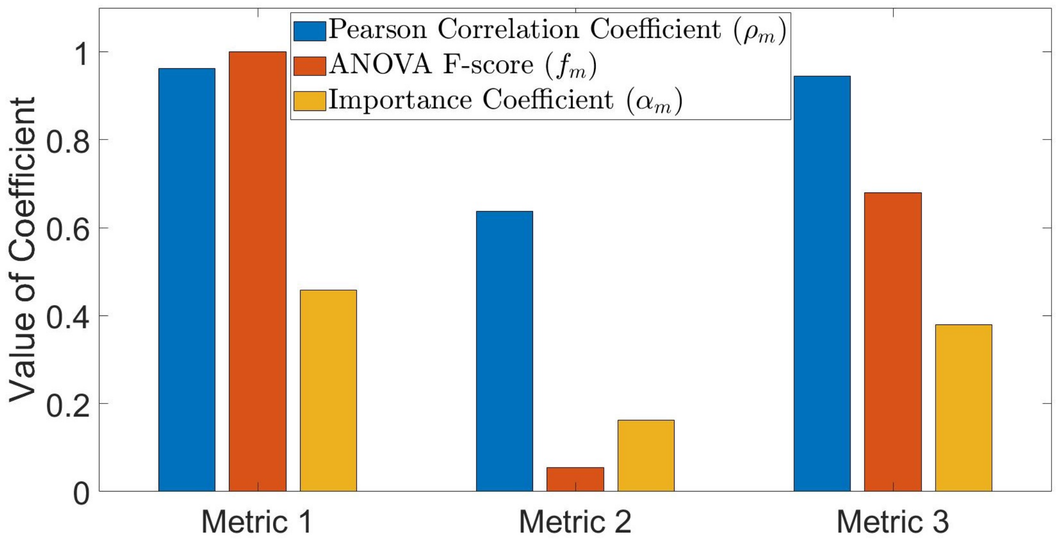

Figure 4.3: Pearson correlation coefficient ρ_m, normalized coefficient by ANOVA f_m and importance coefficient α_m for each Metric m, $m \in \{1, 2, 3\}$.

Furthermore, we evaluate the performance of the AADRNN-based IDS using either individual metrics or all metrics at the same time to measure the effects of metrics on the performance of the IDS. Figure 4.4

shows the performance of the proposed IDS method with the selection of different metrics, as well as the combination of all metrics. We see that AADRNN achieves the highest accuracy at 99.84% when we use all of Metric 1, Metric 2, and Metric 3, as we do for the performance evaluations in the rest of this section.

The high detection performance result shows that the developed AADRNN is able to classify normal and malicious traffic although it has been trained with only normal traffic. Moreover, our results show that the accuracy of the AADRNN is more than 95% under the selection of any metric. In addition, we observe a close relationship between the importance coefficients of metrics in Figure 4.3 and the performance of the AADRNN in Figure 4.4.

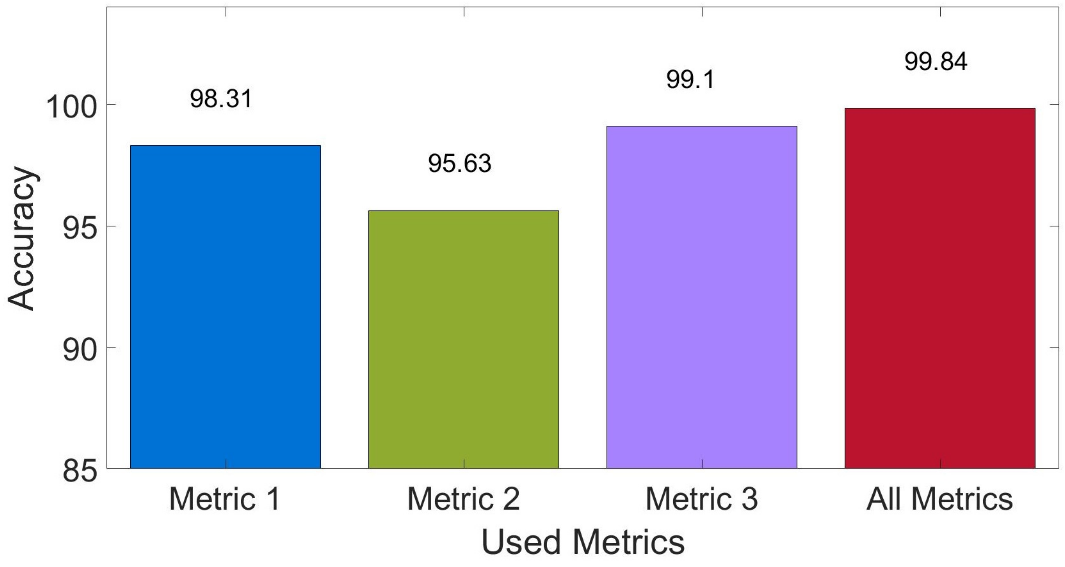

Figure 4.4: Performance of the AADRNN-based IDS under each of Metric 1, Metric 2, Metric 3, and the α_m weighted combination of all metrics.

Selection of the Value of Threshold θ:

We now analyze the performance of the AADRNN with respect to the value of threshold θ. Figure 4.5 presents the True Positive and True Negative percentages with respect to the increasing value of θ from 0 to 0.5 with 0.01 increments.

In Figure 4.5, we see that the AADRNN is highly robust with respect to $\theta \in [0.01, 0.25]$. Thus, in the practical usage of the proposed method, we may select any value of θ in the range $[0.01, 0.25]$ without a significant performance loss. In addition, in this range the AADRNN is fair in detecting both attack and normal traffic, and it achieves high performance for both.

Comparison of the AADRNN's Performance with KNN and Lasso:

Let us now evaluate the attack detection performance of the AADRNN in more detail and compare it with the Simple Thresholding, Lasso, and KNN methods, where both the Lasso and KNN are trained as auto-associative memories.

In Table 4.1, we present the comparison of the detection methods with respect to each of the accuracy and percentages of true positive, false negative, true negative and false positive. The detection methods in

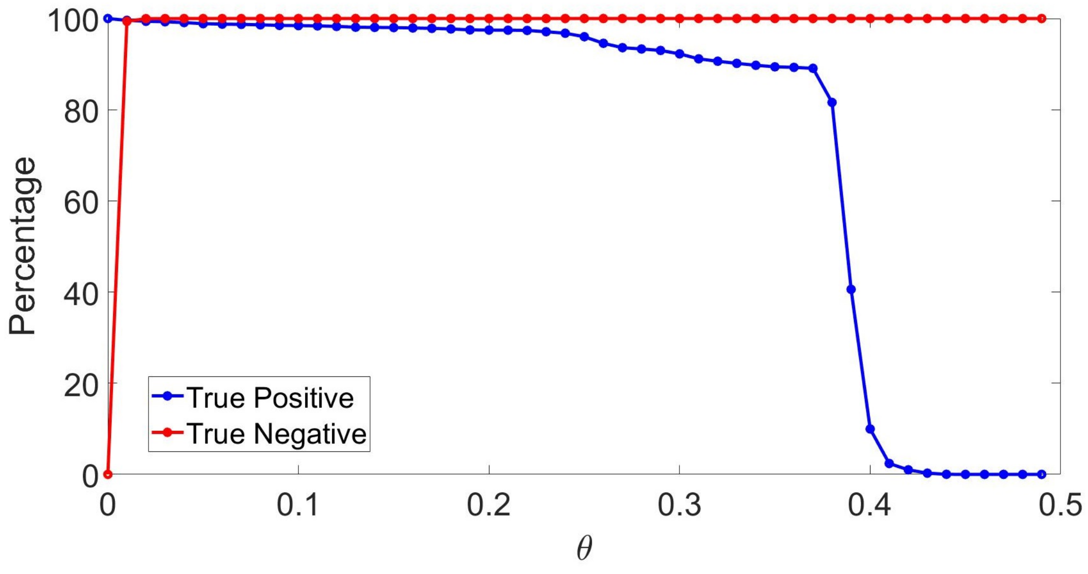

Figure 4.5: True Postive and True Negative percentages of the AADRNN for the increasing value of θ.

Table 4.1: Comparison of attack detection methods with respect to accuracy as well as each of the true positive, false negative, true negative and false positive percentages

Attack Detection Methods	Accuracy	True Positive	False Negative	True Negative	False Positive
AADRNN	99.84	99.82	0.18	99.98	0.02
KNN	99.79	99.79	0.21	99.75	0.25
Lasso	99.78	99.75	0.25	99.95	0.05
Simple Thresholding	93.18	93.09	6.94	93.63	6.37

this table are placed in descending order with respect to their accuracy. Our results show that AADRNN attack detection significantly outperforms the other methods with respect to accuracy, achieving 99.82% true positive and 99.98% true negative percentages. In addition, we see that the auto-associative networks achieve much higher accuracy than Simple Thresholding. Among all methods, the Lasso obtains the true negative percentage closest to the AADRNN, and significantly higher than KNN and Simple Thresholding.

Computation Time:

Figure 4.6 shows the training time of each of the AADRNN, KNN, and Lasso models, where the training is performed for 70% of the normal traffic ($83,138$ samples). While the attack detector may be trained offline in real-life usage, the training time is not a major issue as long as it is acceptable. In the same figure, we see that the training time of the AADRNN is less than 0.1 secs which is highly acceptable. In addition, the training time of the AADRNN is significantly less than that of KNN; however, it is higher than that of the Lasso method.

Figure 4.7 shows the execution time of each of the AADRNN, KNN, and Lasso models for the classifi-

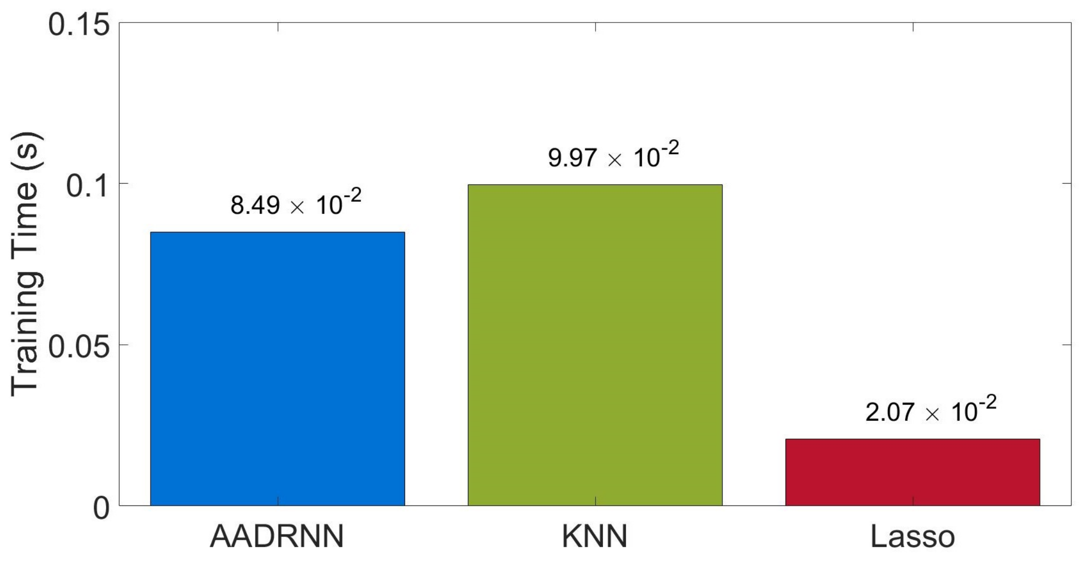

Figure 4.6: Training times of the different attack detection methods.

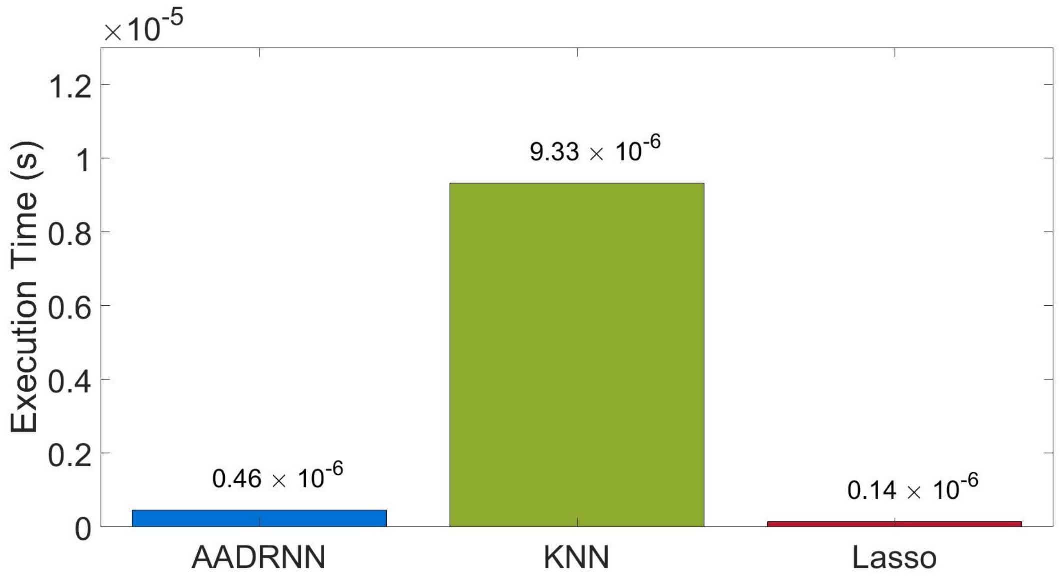

Figure 4.7: Execution times of the different attack detection methods.

cation, evaluated per single traffic packet. We see that the execution time of the AADRNN detector is around 0.5 μ secs. While that of all other methods is less than 10 μ secs, the KNN's execution time is quite high, and Lasso's execution time is the shortest. This shows that the AADRNN and Lasso detectors are suitable for use in real-time attack detection.

4.4.3 Malicious Traffic Detection with Incremental Learning

Since we are still considering the Mirai Botnet attack, we use the following parameter settings: $p = 0.05$, $r = 0.001$, $\lambda^{+} = \lambda^{-} = 0.1$, $P = 500$ packets, and $T = 100$ seconds

Selecting Interval of Incremental Learning:

First, we evaluate the performance of the proposed IDS for varying number of training packets I (which is the period of incremental learning) between 100 and 1000. In this way, we shall also select the best value of I and set it for the rest of this section.

Figure 4.8 presents the average classification accuracy (over all packets) for each value of $I \in \{100, 250, 500, 750, 1000\}$. The results in Fig. 4.8 show that AADRNN with incremental learning achieves its best performance for $I = 750$ packets, where the average accuracy equals 99.54. In addition, one may see that AADRNN achieves acceptable accuracy for all I.

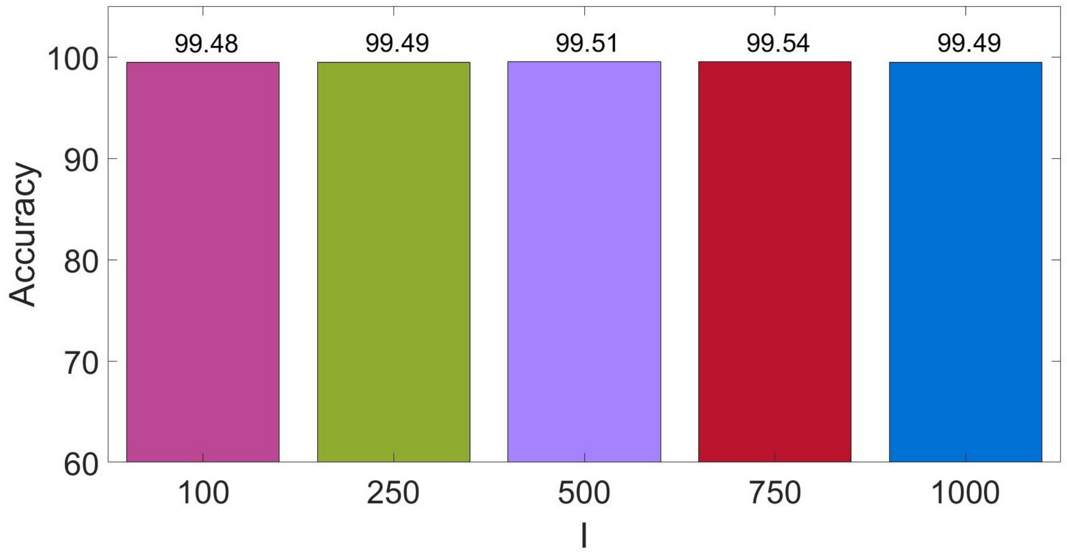

Figure 4.8: Average accuracy of AADRNN attack detector with incremental quasi-online learning for different values of $I \in \{100, 250, 500, 750, 1000\}$

Performance Comparison Against Offline Learning:

We then compare the performance of AADRNN based IDS under incremental quasi-online learning with that under offline learning whose performance shown in Section 4.4.2. Note that since the superior performance of AADRNN against some other ML techniques for Mirai attacks has already been shown in the results of a previous subsection, we do not compare the performance of AADRNN with other ML techniques in this section.

Figure 4.9 presents the percentage Accuracy, TNR, and TPR for AADRNN under "Incremental Learning" and that under "Offline Learning", where the offline training is performed using 70% of benign traffic as used in Section 4.4.2. We see that the accuracy of AADRNN with incremental learning is slightly lower than that with offline learning; however, offline learning uses a significantly large number of collected packets to train AADRNN. Moreover, the TPR results in this figure show that AADRNN with incremental learning performs very close to AADRNN with offline learning for detecting malicious packets, while the performance gap is more significant for TNR.

As an additional experiment, in Figure 4.10, we present the comparison of incremental and offline training for the same cold-start duration which is spent for the collection of $I = 750$ packets. When the number

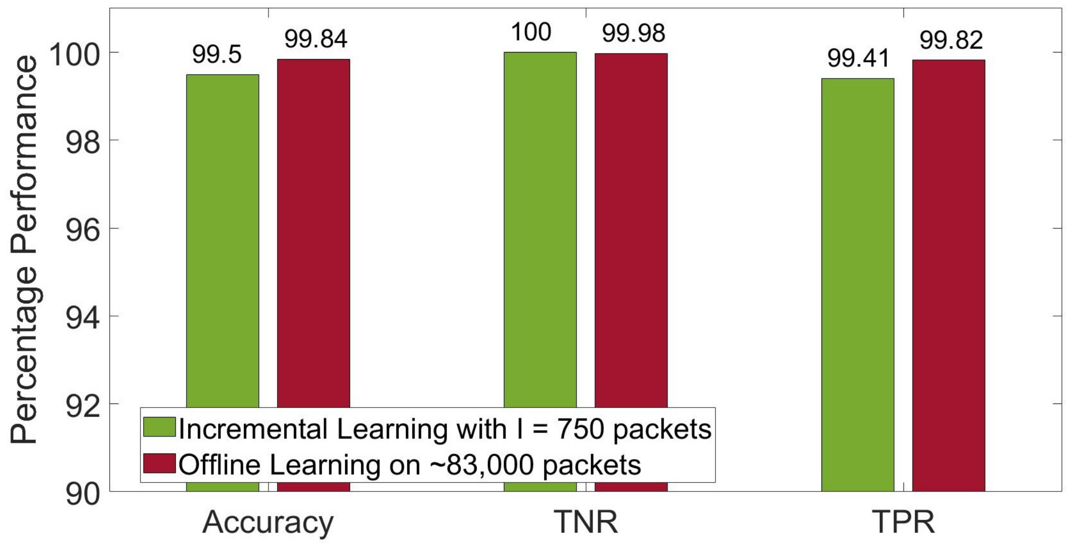

Figure 4.9: Performance comparison between AADRNN based IDS under Incremental Learning with $I = 750$ packets and that under Offline Learning with about $83,000$ packets

of training packets is decreased from $85,000$ to 750, we see that the performance of AADRNN significantly decreases resulting in inferior performance to AADRNN with incremental learning.

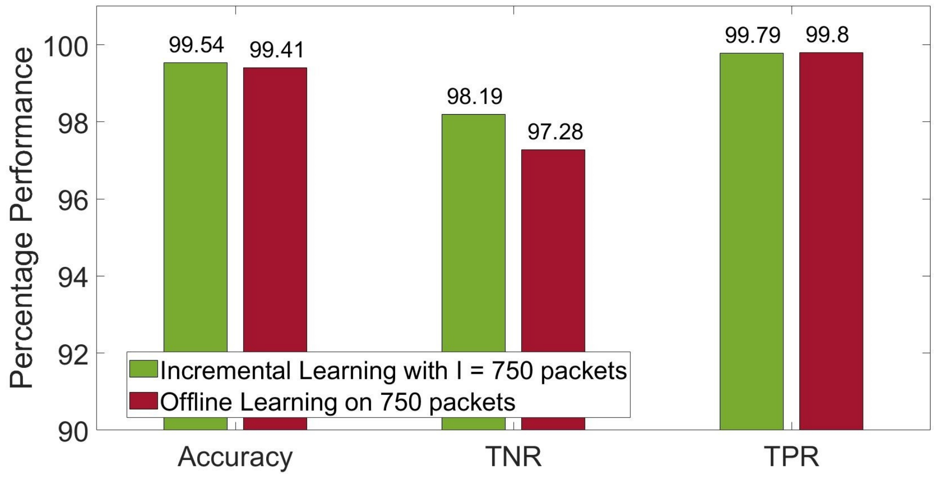

Figure 4.10: Performance comparison between AADRNN based IDS under Incremental Learning with $I = 750$ packets and that under Offline Learning with 750 packets

Computation Time:

Finally, for the proposed method, Table 4.2 presents the execution time (i.e. time elapsed) for making a decision on a single packet as well as the initialization and incremental update stages of the training algorithm for $I = 750$.

The results in this table first show that the execution time of AADRNN is very low and acceptable for real-time attack detection. Also, we see that the initialization and incremental learning of our method take $15\ ms$ and $4.3\ ms$, respectively. As observed in the evaluated dataset, $4.3\ ms$ is slightly less than the minimum measured time for transmission of 22 packets; that is, the parameters of AADRNN will be updated until the transmission of the $22nd$ packet after the incremental learning phase has begun.

Table 4.2: Training and execution times of the proposed attack detection method with incremental learning

Training Time	Initialization	15 $msecs$
(for $I = 750$)	Incremental Update	4.3 $msecs$
Execution Time		0.11 $msecs$

4.4.4 Simultaneous Detection of Various Types of Attacks with Offline Learning IDS

We compare the performance of AADRNN against the unsupervised state-of-the-art one class classification technique based on the Support Vector Machine - One Class Classifier (SVM-OCC). In addition, we compare the performance of AADRNN with that of several supervised machine learning techniques using the same dataset, namely the LR, KNN, DT and RF, which are detailed in [210]. During the performance evaluation for detecting various types of attacks simultaneously, we set $p = 0.01$, $r = 1$, $\lambda^+ = \lambda^- = 0.005$. Note that we do not measure our own metrics for this evaluation task as KDD dataset provides 41 input features for detection.

In Figure 4.11, the performance is presented with respect to Accuracy, True Negative Rate (TNR), True Positive Rate (TPR), Precision, and F1 Score on the KDD dataset as a whole, showing that AADRNN achieves 93% accuracy with high TNR (95.7%) and TPR (92.3%). In this results, the structure of AADRNN is comprised of three layers with 41 neurons each resulting in a structure of $41 - 41 - 41$. According to these results, AADRNN successfully classifies both benign and attack traffic as a whole for all the test data and attack types in the KDD dataset, although it has been trained only with a small benign traffic dataset.

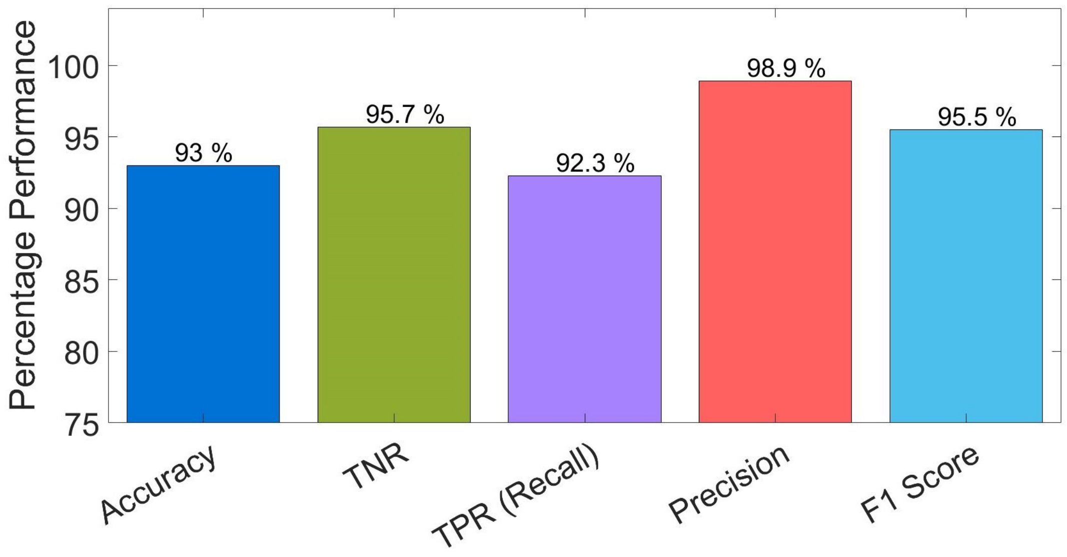

Figure 4.11: Performance of the $41 - 41 - 41$ cluster three layer AADRNN with respect to Accuracy, TNR, TPR, Precision and F1 Score metrics for the KDD dataset taken as a whole for all attack types

As indicated earlier, the KDD Cup'99 dataset contains a wide variety of attack types, so that the performance of the ADRNN can differ for different attack types. To this end, Figure 4.12 displays the performance of AADRNN for each individual type of attack in the test set. The results show that the prediction accuracy is less than or equal to 50% for 7 out of 37 attack types while it is above 98% for 21 out of 37 attack types.

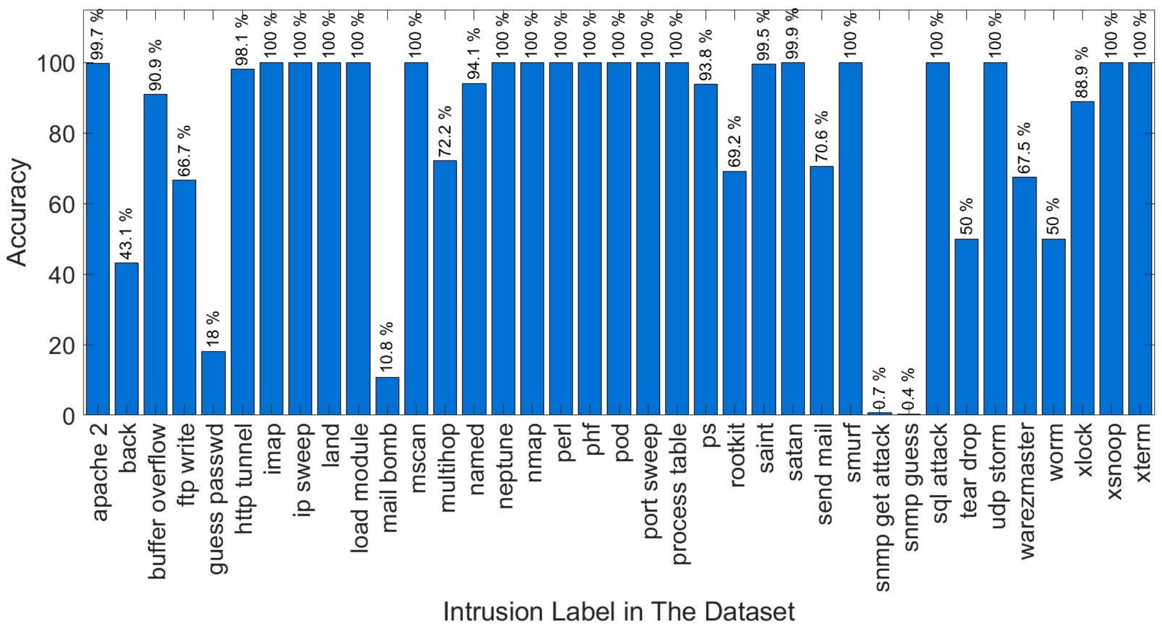

Figure 4.12: Performance of the $41 - 20 - 41$ cluster three layer AADRNN with respect to Accuracy, TNR, TPR, Precision and F1 Score metrics for each attack type in KDD dataset.

Results for the AADRNN with 41-20-41 Structure

During our experiments for detecting various attack types learning only from the normal traffic, we observed that the overall performance of AADRNN can be improved for the 37 attack types available in KDD Cup'99. To this end, we revise the structure of AADRNN as $41 - 20 - 41$ clusters in three layers.

In Figure 4.13, we summarize the performance of the three layer $41 - 20 - 41$ cluster AADRNN for each of the individual attack types, and again we see a somewhat higher performance than that the earlier network structure (with $41-41-41$ clusters) that is shown in Figures 4.11 and 4.12. Hence, we see that the choice of a network that offers a mapping of the data into a subspace, as offered by the $41-20-41$ network, provides higher performance. Accordingly, we use $41 - 20 - 41$ structure for the performance comparison against the state-of-the-art methods.

Comparison with Unsupervised and Supervised Techniques

We compare the performance of AADRNN with offline learning first with the performance of unsupervised technique based on the SVM-OCC, and then with other supervised state-of-the-art machine learning techniques. Figure 4.14 compares AADRNN with SVM-OCC with respect to Accuracy, TNR and TPR. The results in this figure show that of the AADRNN with offline learning clearly outperforms this state-of-the-art unsupervised one class classifier for detecting both benign and attack traffic, and the performance difference

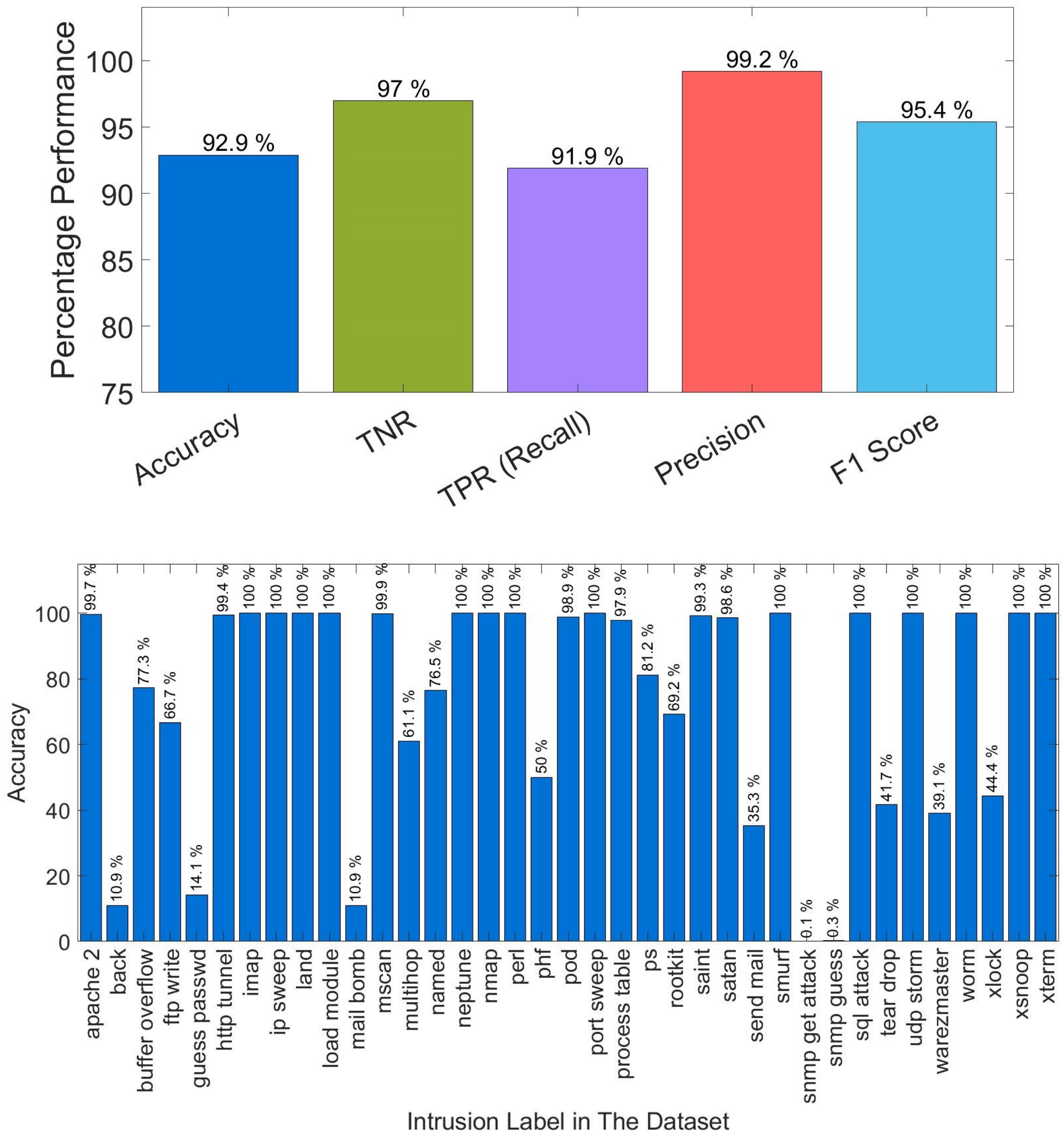

Figure 4.13: Performance of the $41-20-41$ cluster three layer AADRNN with respect to Accuracy, TNR, TPR, Precision and F1 Score metrics for the KDD dataset taken as a whole for all attack types (top) and for each attack type (bottom).

is significant especially for detecting attack traffic.

In addition, in Table 4.3, presents the computation times of AADRNN and SVM-OCC. One may see that AADRNN is two orders of magnitude faster than SVM-OCC. Moreover, since the training of AADRNN requires only benign traffic and takes $1.49\ s$ on average, AADRNN can also be trained online for some applications.

Table 4.4 shows the results on the performance comparison between the $41-20-41$ cluster three layer AADRNN and other well-known machine learning techniques including the unsupervised SVM-OCC and the supervised MLP, LR, KNN, DT, and RF, where the performance of the supervised techniques is taken as reported in a recent publication [210].

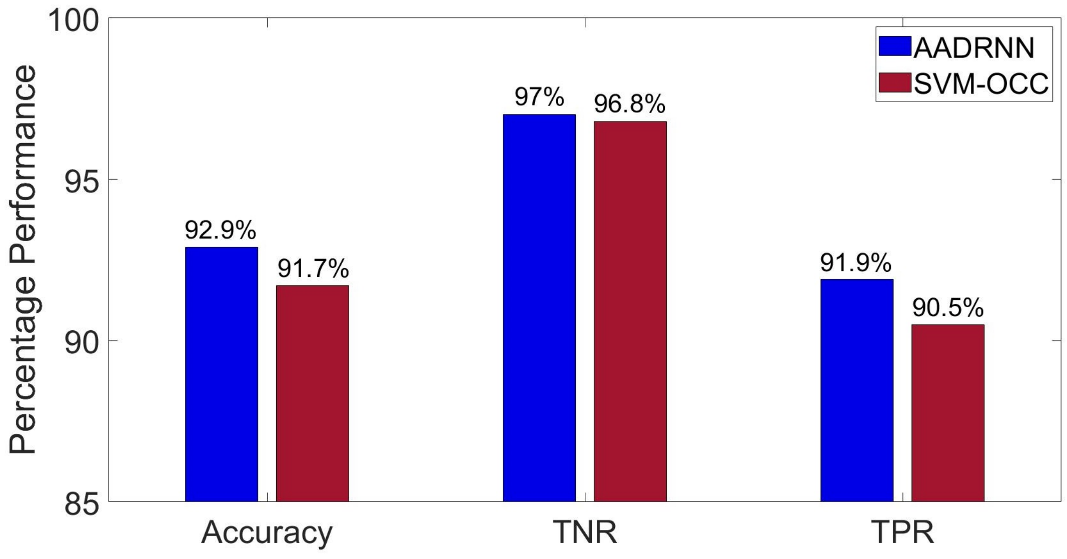

Figure 4.14: Comparison of the $41 - 20 - 41$ cluster three layer AADRNN against the state-of-the-art one class classifier SVM-OCC on the KDD dataset.

Table 4.3: Comparison of the $41-20-41$ three layer AADRNN against a state-of-the-art one class classifier, the SVM-OCC with respect to computation time.

Model	**Total Training Time (seconds)**		**Execution Time per Sample (μsecs)**	
	Mean	**Standard Deviation**	**Mean**	**Standard Deviation**
AADRNN	1.49	0.03	5.13	0.08
SVM-OCC	249.7	9.44	326.3	7.59

These results show that, when evaluated with the KDD dataset, the $41 - 20 - 41$ cluster AADRNN can outperform these other techniques except for the MLP and DT. While the AADRNN achieves almost the same performance as the MLP and DT, it is superior to them in terms of the Recall metric and inferior with regard to Precision. In particular, AADRNN yields slightly lower False Negatives and slightly higher False Positives as compared to MLP and DT. On the other hand, one may recall that AADRNN trained using only normal traffic.

4.4.5 Compromised Device Identification with Sequential Learning

Finally, we present the performance evaluation results of CDIS during an ongoing Botnet attack in an IoT network. Recall that we use data from Kitsune and BotIoT datasets. In order to present the methodology of our experimental study and results more clearly, we first use the Mirai Botnet from the publicly available Kitsune dataset [203, 204]. Then, we present the results for 6 attacks obtained from Kitsune and Bot-IoT datasets. For the performance evaluation of the CDIS, we use the following parameter settings: $r = 1$,

Table 4.4: Performance of the $41 - 20 - 41$ three layer AADRNN compared to several state-of-the-art unsupervised and supervised techniques with the KDD dataset.

Model		**Metrics (%)**			
		Accuracy	**Recall**	**Precision**	**F1 Score**
Unsupervised	**AADRNN**	92.9	91.9	99.2	95.4
	SVM-OCC	91.7	90.5	99.1	94.6
Supervised [210]	**MLP with 4 layers**	93	91.5	99.8	95.5
	LR	81.1	76.9	99.4	86.7
	KNN	92.5	90.9	99.8	95.2
	DT	92.9	91.5	99.7	95.4
	RF	92.7	91.1	99.9	95.3

$\lambda^+ = \lambda^- = 0.005$, and $p = 1/M$, where $M = 6$. We also set the length of a time window $T = 10\ secs$.

Ground Truth:

Recall that Kitsune Mirai Botnet dataset contains the traffic from 107 distinct IP addresses that either sent or receive traffic. For each of these IP addresses, we utilize an instance of the CDIS presented in Section 4.3 to perform infection detection.

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 time t at which it is sent, and the complete representation of packet p is:

$$p \equiv pk(t, s, d) \tag{4.51}$$

where s and d are the source and destination nodes.

The ground truth for the infection level of a device (or IP address) is defined as the ratio of the number of attack packets to the total number of packets that are sent by device or node d in time window k:

$$\phi_k^i = \frac{\sum_{\{p \in \cup_{l=1}^{k} P_l^{i,D}\}} a(p)}{| \cup_{l=1}^{k} P_l^{i,D} |}, \tag{4.52}$$

where $P_k^{s,D}$ is the set of all packets sent by s in time window k:

$$P_k^{s,D} \equiv \sum_{d \in D} P_k^{s,d}\ . \tag{4.53}$$

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

$$v_k^i = \mathbf{1}[\phi_k^i \geqslant \Theta], \tag{4.54}$$

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.

Accordingly, v_k^i is the variable that we use **to test how well our attack detection schemes are working**. It is **not used at all for learning** since we develop a quasi-online sequential learning technique which does not rely on prior offline learning.

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 Kitsune Mirai Botnet dataset [204]. The ordered list of IP addresses in this considered network is as follows: [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).

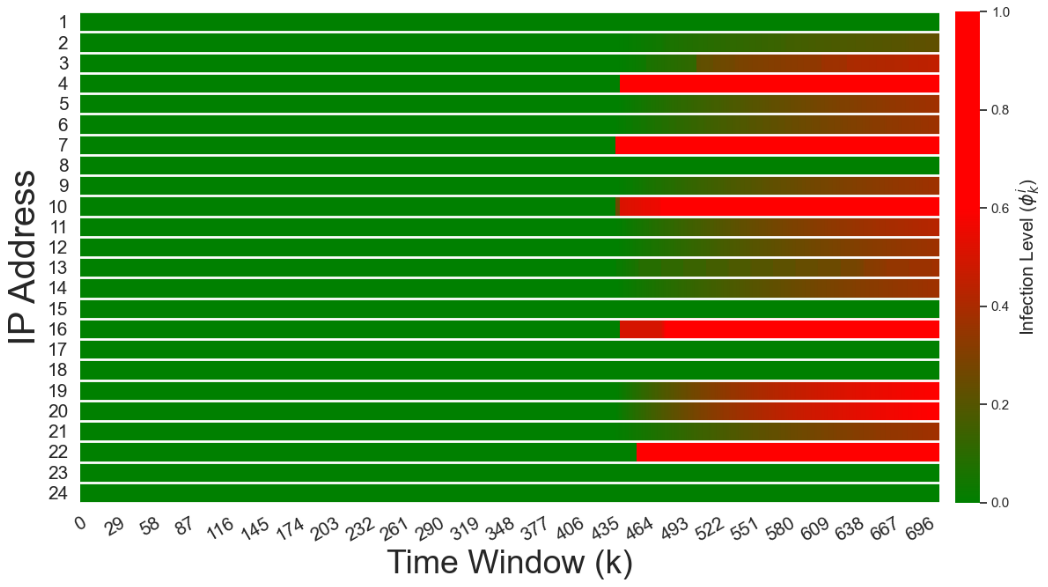

Figure 4.15: Ground truth value of the level of infection (ϕ_k^i) for individual IP addresses over 712 windows

Figure 4.15 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 4th, 7th, 10th, 16th, 19th, 20th and 22nd IP addresses while the infection level remains around 0 for only 1st, 8th, 15th, 17th, 18th, 23rd and 24th IP Addresses.

Based on Figure 4.15, we intuitively consider 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). Thus, we set $\Theta = 0.5$ to calculate the binary ground truth of compromised devices via (4.54). 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 712 windows. In addition, in the binary ground truth for $\Theta = 0.5$, 4th, 7th, 10th, 16th, 19th, 20th, and 22nd IP addresses become compromised with time.

Selection of the Decision Threshold γ_i:

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 4.16, 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.

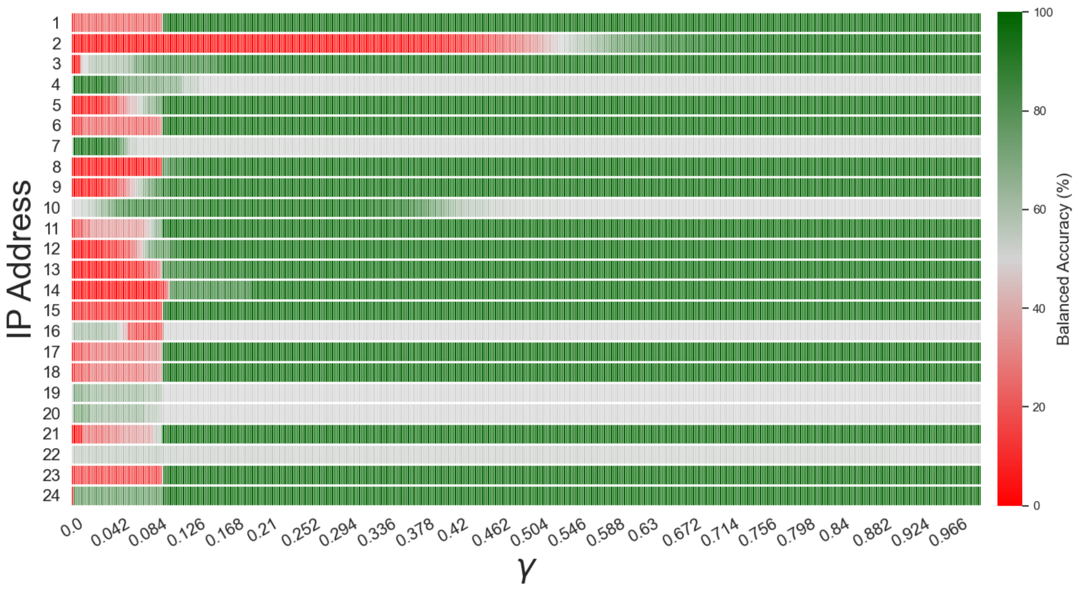

Figure 4.16: Balanced Accuracy during the search for the best value of γ_i in CDIS with sequential training for each IP address $i \in \{1, \ldots, 24\}$

Furthermore, Figure 4.16 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 our 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.

Performance of the CDIS:

Using the best values of γ_i's, we evaluate the performance of the CDIS (presented in Section 4.3) with respect to Balanced Accuracy, Sensitivity, and Specificity. Figure 4.17 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 the CDIS is almost 100%. However there are four IP addresses for which the measured performances are the outliers.

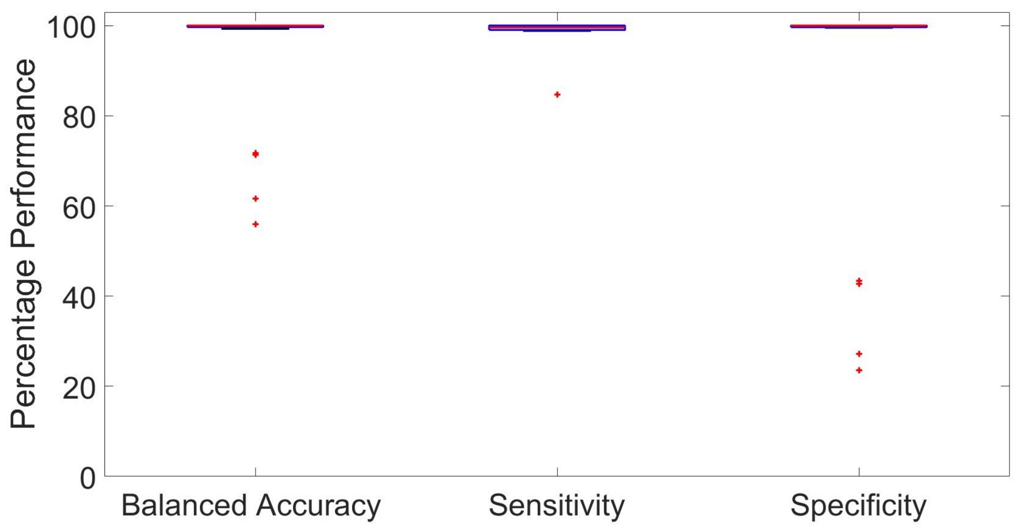

Figure 4.17: Box plot of the Balanced Accuracy, Sensitivity, and Specificity over considered 24 IP addresses

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 4.16.

These results also reveal that our 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 RTMs (Received Traffic Metrics) 1-4 are zero for all time windows.

Performance of the CDIS under Different ML models:

The performance of the 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 4.5. 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 4.5 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 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 4.6 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. Also, recall

Table 4.5: Average percentage performance of the 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

Table 4.6: Training and execution times of the CDIS under different ML models (in milliseconds)

ML Models	Training Time		Execution Time	
	Mean	Standard Deviation	Mean	Standard Deviation
AADRNN	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

that each of the neural network models considered (i.e. AADRNN, MLP, and LSTM) has three hidden layers with six neurons (which is the number of metrics) each. In addition to its three hidden layers, LSTM neural network has also an LSTM layer with six units.

The results in Table 4.6 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.

Performance of the CDIS for a Network with 107 Distinct IP Addresses:

We now evaluate the performance of the CDIS for the extended IoT network scenario where all 107 unique IP addresses provided in the Kitsune dataset [204] are addressed instead of only considering the IPs in “192.168”network. To this end, Figure 4.18 (top) displays the average performance of the CDIS over IP addresses with respect to each of Balanced Accuracy, Sensitivity and Specificity, and Figure 4.18 (bottom) displays the box plot of the performance of the CDIS over IP addresses with respect to the same metrics.

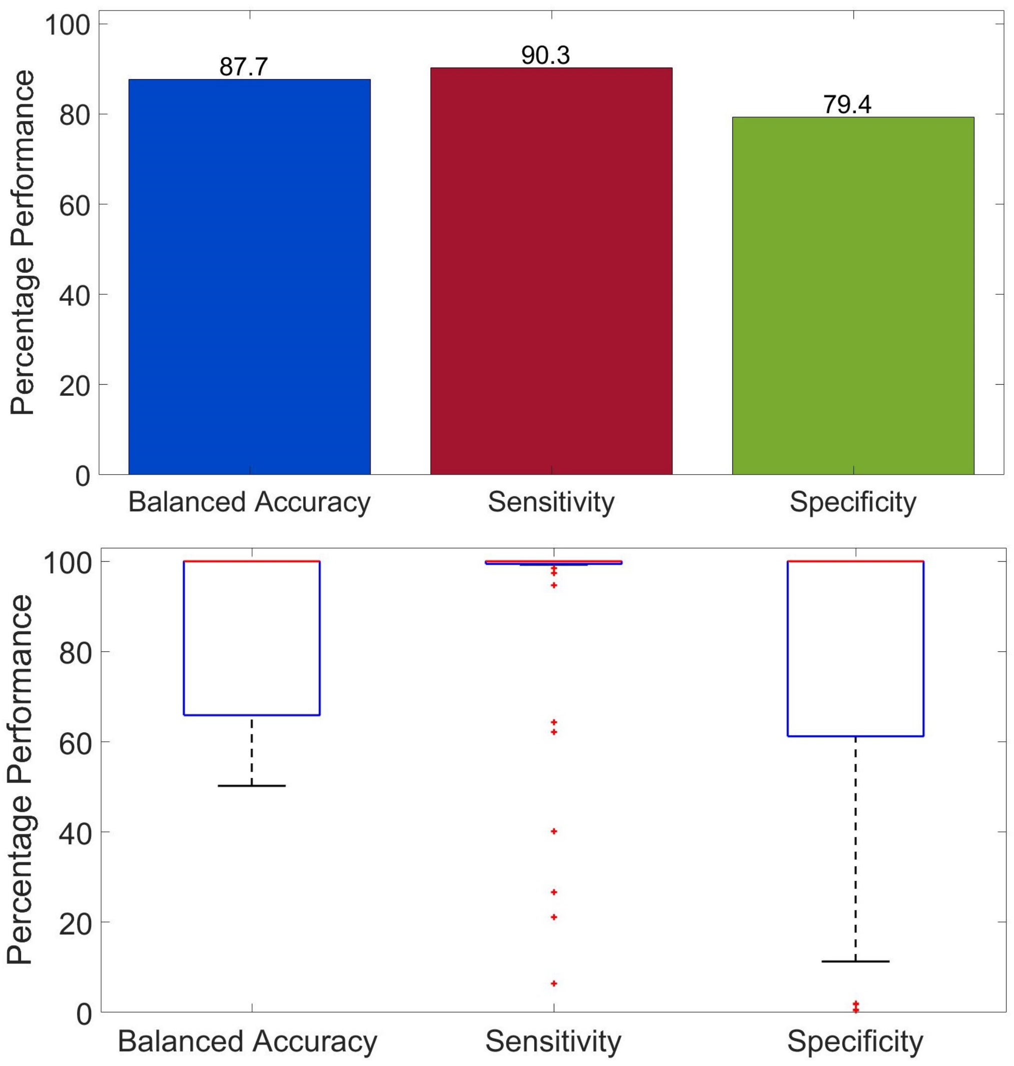

Figure 4.18: Bar graph (top) of the average performance and box plot (bottom) of the performance of the CDIS over all (107) IP addresses in the Kitsune Mirai dataset.

In the top of Figure 4.18 we see that our CDIS 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 4.18 (bottom) reveal that the Balanced Accuracy performance of the CDIS is above 92% for $2/3$ of IP addresses and above 50% for all IP addresses. That is, the Balanced Accuracy performance 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%.

Furthermore, in Figure 4.19, we plot the logarithmic prediction error of the CDIS, defined as

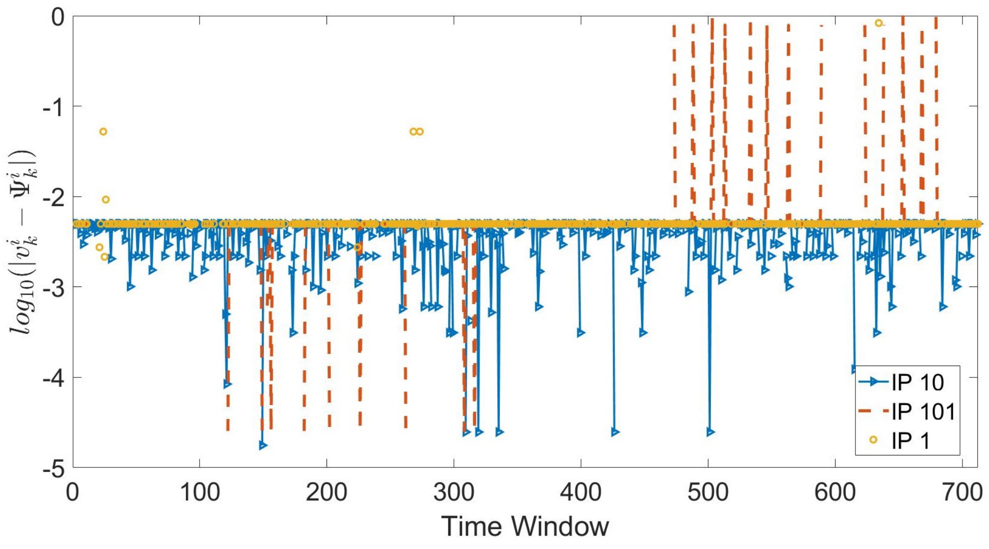

Figure 4.19: In this figure, the logarithmic prediction error $\log_{10}(|v_k^i - \Psi_k^i|),\ 1 \leqslant k \leqslant 712$ of the CDIS is plotted versus the time slot k for three IP addresses: the 1st, 10th, and 101st. Only three IP addresses were chosen to clearly visualize the prediction errors that may be affected by sequential quasi-online training.

$\log_{10}(|v_k^i - \Psi_k^i|),\ 1 \leqslant k \leqslant 712$, versus the slot k for three IP addresses: the 1st, 10th, and 101st. Our purpose is to present the prediction errors of the quasi-online sequential learning clearly. Indeed, the results on the 10th and 101st IP addresses show that the CDIS achieves lower prediction errors for normal non-attack traffic after $k = 100$. On the other hand, the sequential 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 4.15.

Performance of the CDIS on Different Datasets:

Although we mainly focused on identifying the compromised IoT devices during a Mirai Botnet attack, the proposed IDS 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 the CDIS for various types of DDoS attacks provided in two datasets: Kitsune [203,204] and Bot-IoT [206].

Figure 4.20 displays the balanced accuracy results of the performance evaluation of the CDIS for the Kitsune 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 Kitsune. Recall that the Balanced Accuracy is above 92% for $2/3$ of unique IP addresses for Kitsune dataset.

The results in Figure 4.20 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 attack. 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

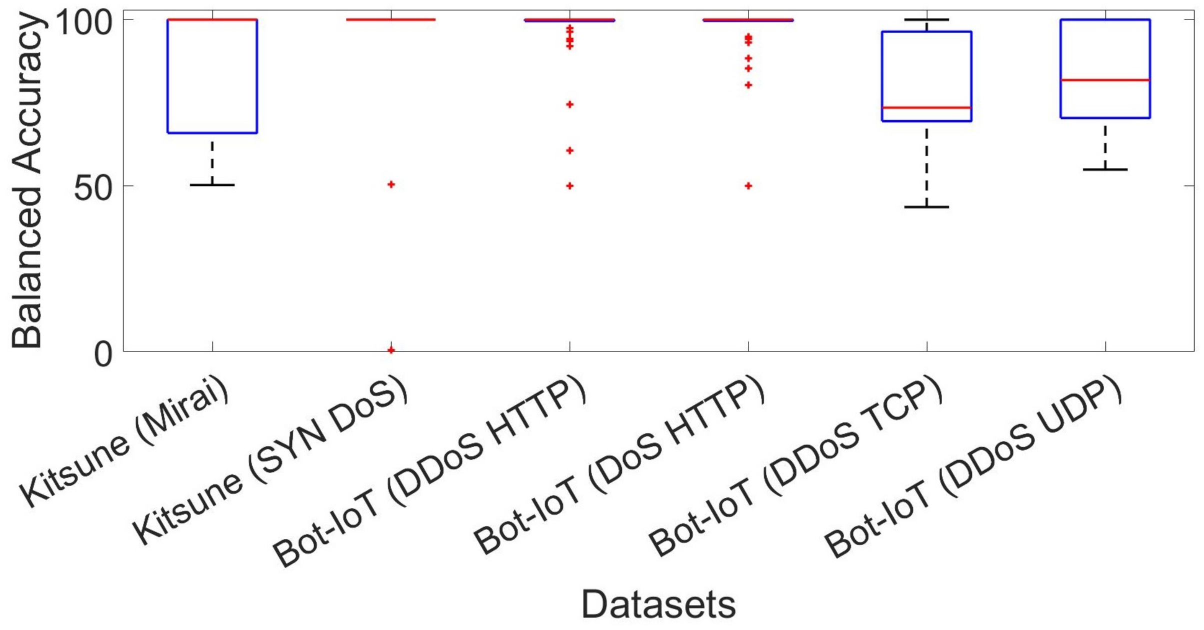

Figure 4.20: Performance of CDIS using the parameters set for Mirai is evaluated for other type of attacks on various datasets

median Balanced Accuracy performance is above 100%. We may also see that the performance of the CDIS slightly lower for DDoS attacks on the network traffic using TCP or UDP protocols. Since the parameters of CDIS 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 metrics.

Further Remarks on the Results for Compromised Device Identification:

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 6 attack data in two different datasets, namely Kitsune and Bot-IoT. The experimental results on compromised device identification show that:

- The CDIS achieves high performance (94% Balanced Accuracy) with low computation time for both execution and sequential quasi-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 metrics and parameter settings of CDIS.

Chapter 5

Fully Online Self-Supervised Intrusion Detection Framework

As the results presented in Chapter 4 as well as the results of earlier research suggest, anomaly-based Intrusion Detection Systems are very promising in detecting zero-day attacks that are based on unknown types of intrusions and often target vulnerable devices and networks. On the other hand, high dependency of anomaly-based IDS on the normal traffic used for parameter optimization causes some significant challenges, such as: 1) The (expected) normal behavior of network traffic may change over time due to both internal and external influences. For example, an IoT device in the considered network may start to generate more data packets over time depending on the variables measured by the device, or new device(s) may be added to the network, resulting in a considerable change in aggregated network traffic. 2) As normal traffic will most likely differ on different networks, in order to achieve acceptable performance, the IDS should be individually adapted for each network on which it will be implemented. To this end, the parameters of the IDS should be optimized (i.e. learned), or the IDS should be restructured for the normal traffic of the particular network. On the other hand, for the practical application of the IDS in end-user networks (such as smart home IoT networks), there is very limited or no offline data collected over this network that can be used to adapt (or train) the IDS.

In order to address these issues, in this chapter, we propose a novel fully online Self-Supervised Intrusion Detection (namely, SSID) framework, which automatically selects normal traffic packets for learning and decides when to update the parameters of the utilized algorithm in order to keep the algorithm up-to-date and the detection accuracy high. The SSID framework can be used with any anomaly detection algorithm that requires parameter optimization, providing fully online (on the fly) self-supervised learning of parameters in parallel with real-time anomaly detection. It also completely eliminates the need for labeled or unlabeled offline data collection, and offline training or parameter optimization. Therefore, the proposed framework contrasts sharply with existing work [211–217] that has implemented self-supervised learning for intrusion detection, which often utilizes offline (small-sized) labeled or unlabeled training data and pseudo-labeling.

5.1 System Design of the Self-Supervised Intrusion Detection Framework

As the main contribution of this thesis, 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 5.1) and the general IDS structure. Next, we explain the learning process performed in our framework.

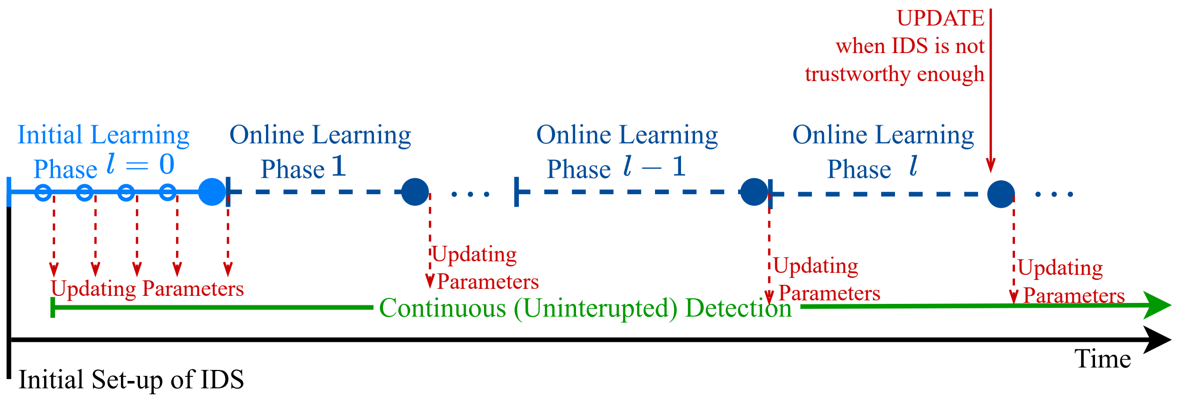

Figure 5.1: Detection and learning processes of IDS within the Fully Online Self-Supervised Intrusion Detection (SSID) framework

5.1.1 Intrusion Detection Process

As shown in Figure 5.1, 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.

General Structure of IDS:

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 5.2 that is taken into account during our analysis.

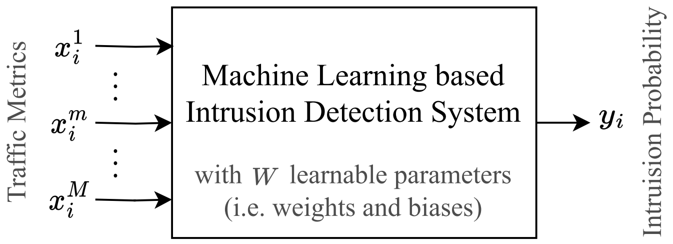
Traffic Metrics
x_i^1
x_i^m
x_i^M
Machine Learning based Intrusion Detection System
with W learnable parameters (i.e. weights and biases)
y_i
Intrusion Probability

Figure 5.2: Structure of IDS

As shown in Figure 5.2, 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 $f : [0,1]^M \to [0,1]$.

Example of an IDS:

We now present an example of the structure of IDS that satisfies our criterion. This example is displayed in Figure 5.3. This IDS is mainly comprised of an ML model and a decision maker component.

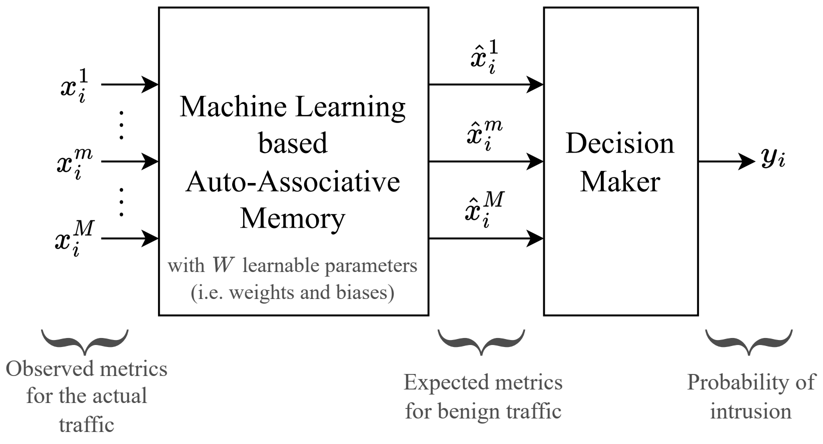
x_i^1
x_i^m
x_i^M
Machine Learning based Auto-Associative Memory
with W learnable parameters (i.e. weights and biases)
$\hat{x}_i^1$
$\hat{x}_i^m$
$\hat{x}_i^M$
Decision Maker
y_i
Observed metrics for the actual traffic
Expected metrics for benign traffic
Probability of intrusion

Figure 5.3: Example for the Structure of IDS

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 this example of IDS structure, 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 $\hat{x}_i = [\hat{x}_i^1, \ldots, \hat{x}_i^m, \ldots, \hat{x}_i^M]$. 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 $f_{aam} : [0, 1]^M \to [0, 1]^M$.

Furthermore, within this IDS, any particular decision maker calculates the probability of attack. In fact, based on the output of ML-based AAM, the decision maker can measure 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. On the other hand, it is possible for the decision maker to learn and update parameters along with the learning process of the ML during the online operation.

5.1.2 Online Self-Supervised Learning Process

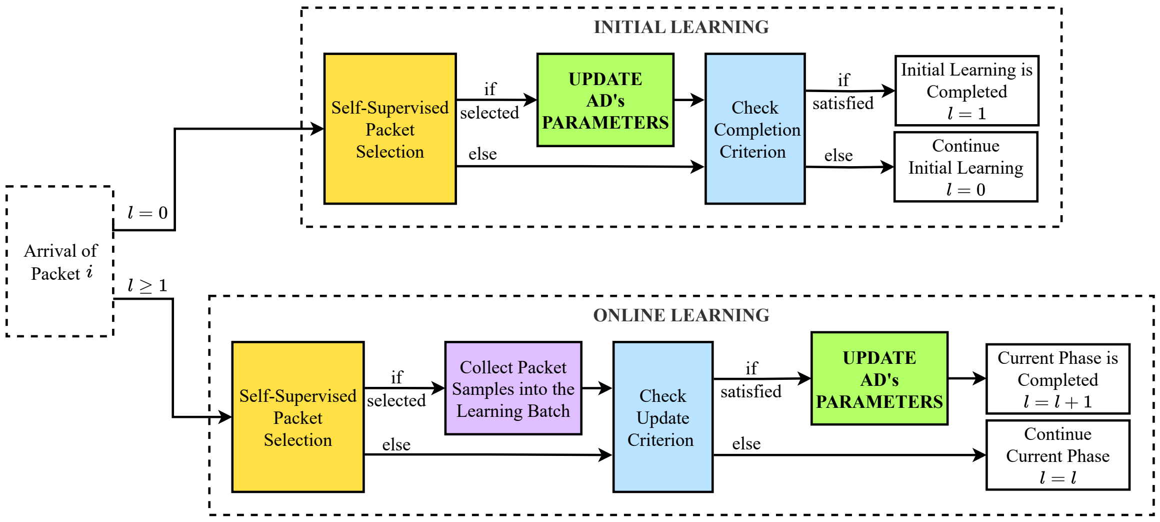
INITIAL LEARNING
Self-Supervised Packet Selection
if selected
UPDATE AD's PARAMETERS
Check Completion Criterion
if satisfied
Initial Learning is Completed $l = 1$
else
else
Continue Initial Learning $l = 0$
Arrival of Packet i
$l = 0$
$l \geq 1$
ONLINE LEARNING
Self-Supervised Packet Selection
if selected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

Figure 5.4: Block diagram of the learning process in SSID framework for the online self-supervised learning of the parameters of IDS

In parallel with attack detection, our SSID framework provides online self-supervised learning of IDS parameters which is shown as the upper process line in Figure 5.1. As seen in that figure and Figure 5.4 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 the network traffic characteristics may vary with time substantially affecting the detection performance of IDS, it is crucial to update the parameters of the IDS online in parallel to its real-time operation. The parameter updates performed by online learning ensure that the detection of the IDS is trustworthy by improving its performance and keeping it up-to-date for the recent traffic characteristics. As one of the main purposes of SSID, online learning also prevents collecting and labeling big data for offline training so it saves time and resources. On the other hand, there may be online available datasets to validate the performance of the IDS (or to train it additionally) on the traffic of the same or other IoT networks, and the IDS can also be pretrained prior to the learning process if any valuable data is available.

In the remainder of this subsection, in order to be able to clearly present the SSID framework, we shall only explain the main functionalities of both initial and online learning phases through the block diagram displayed in Figure 5.4. Then, in Section 5.2, the detailed comprehensive methodology (including all blocks in Figure 5.4) will be presented.

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. That is, we update the parameters of the IDS for each selected packet until a certain criterion on the trustworthiness of the IDS is satisfied.

In detail, as shown in the top block of Figure 5.4, 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 criterion on the trust coefficient for completing of the initial learning phase $l = 0$. This criterion basically measures if SSID's trust in the IDS is greater than a threshold Θ (which refers to the minimum desired trust):

$$\text{if } \Gamma \geqslant \Theta, \text{ complete initial learning and set } l = 1 \tag{5.1}$$

That is, if $\Gamma \geqslant \Theta$, the initial learning phase is completed, and the next packet will be considered for the first phase $l = 1$ of continuous learning.

Online Learning:

After the initial learning is completed, the parameters of IDS are updated via an online learning phase $l \geqslant 1$ when the trust of SSID in the IDS is unacceptably low. As the lower block in Figure 5.4 shows, the parameters of the 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 $l \geqslant 1$, 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. $|B_l| \geqslant K$), and there is no attack detected by the IDS:

$$\text{if } \Gamma < \Theta \text{ and } |B^l| \geqslant K \text{ and } \frac{1}{I} \sum_{j=i-I+1}^{i} y_j \leqslant \gamma, \text{ update parameters and set } l = l + 1 \tag{5.2}$$

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, i.e. if the average output of the IDS is greater than γ, SSID clears the batch of collected packet samples, B^l. 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. That is,

$$\text{if } \frac{1}{I} \sum_{j=i-I}^{i} y_j > \gamma, \text{ empty } B^l \tag{5.3}$$

5.2 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 5.4. 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 satisfy the assumptions in Section 5.1.

5.2.1 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 use 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

$$p_i^+ \equiv \text{(trust in the IDS) (estimated probability of packet } i \text{ being malicious)} \tag{5.4}$$

$$p_i^+ = \Gamma\, y_i \tag{5.5}$$

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

$$p_i^- \equiv \text{(trust in the IDS)(estimated probability of packet } i \text{ being normal)} \tag{5.6}$$

$$p_i^- = \Gamma\,(1 - y_i) \tag{5.7}$$

Subsequently, since

$$p_i^+ + p_i^- + q_i = 1, \tag{5.8}$$

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

$$\begin{aligned} q_i &= 1 - (p_i^+ + p_i^-) \\ &= 1 - \Gamma. \end{aligned} \tag{5.9}$$

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. This is why an IDS that can learn from only benign traffic is expected to learn faster and perform better under SSID.

5.2.2 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 5.2.3 and Section 5.2.4.

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}:

$$\Gamma = C_{rep} C_{gen} \tag{5.10}$$

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.

5.2.3 Representativeness of Learned Traffic

In order to calculate the representativeness of the packet traffic used during the earlier learning phases, we compare the learned (or memorized) traffic with the total observed traffic through Kullback-Leibler (KL)-Divergence [218]. 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_o^{PL} have exponential distributions with means of $1/\mu_l$ and $1/\mu_o$.

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 IDS. 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 [218], we first calculate $D_{KL}(S_o^{TT}||S_l^{TT})$, which can shortly be denoted by D_{KL}^{TT}, as

$$\begin{aligned} D_{KL}^{TT} &= \int_{-\infty}^{\infty} f(x;\lambda_o) log(\frac{f(x;\lambda_o)}{f(x;\lambda_l)})\, dx \\ &= \mathbb{E}_{f(x;\lambda_o)} \left[log(\frac{f(x;\lambda_o)}{f(x;\lambda_l)}) \right] \\ &= \mathbb{E}_{f(x;\lambda_o)} \left[log(\frac{\lambda_o}{\lambda_l}) - x(\lambda_o - \lambda_l) \right] \end{aligned} \tag{5.11}$$

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

$$D_{KL}^{TT} = log(\frac{\lambda_o}{\lambda_l}) - \frac{(\lambda_o - \lambda_l)}{\lambda_o} \tag{5.12}$$

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

$$\begin{aligned} D_{KL}^{PL} &= \int_{-\infty}^{\infty} f(x;\mu_o) log(\frac{f(x;\mu_o)}{f(x;\mu_l)})\, dx \\ &= \mathbb{E}_{f(x;\mu_o)} \left[log(\frac{f(x;\mu_o)}{f(x;\mu_l)}) \right] \\ &= \mathbb{E}_{f(x;\mu_o)} \left[log(\frac{\mu_o}{\mu_l}) - x(\mu_o - \mu_l) \right] \end{aligned} \tag{5.13}$$

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:

$$D_{KL}^{PL} = log(\frac{\mu_o}{\mu_l}) - \frac{(\mu_o - \mu_l)}{\mu_o} \tag{5.14}$$

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 a 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

$$D_{KL-norm}^{TT} = e^{-D_{KL}^{TT}}, \tag{5.15}$$

$$D_{KL-norm}^{PL} = e^{-D_{KL}^{PL}}. \tag{5.16}$$

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:

$$\begin{aligned} D_{KL-norm}^{PL} &= e^{-\left[log(\frac{\lambda_o}{\lambda_l}) - \frac{(\lambda_o - \lambda_l)}{\lambda_o}\right]} \\ &= \left[\frac{\lambda_l}{\lambda_o} \, e^{-\frac{(\lambda_l - \lambda_o)}{\lambda_o}}\right] \end{aligned} \tag{5.17}$$

Similarly,

$$\begin{aligned} D_{KL-norm}^{PL} &= e^{-\left[log(\frac{\mu_o}{\mu_l}) - \frac{(\mu_o - \mu_l)}{\mu_o}\right]} \\ &= \left[\frac{\mu_l}{\mu_o} \, e^{-\frac{(\mu_l - \mu_o)}{\mu_o}}\right] \end{aligned} \tag{5.18}$$

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

$$C_{rep} = c_1 D_{KL-norm}^{TT} + c_2 D_{KL-norm}^{PL} \tag{5.19}$$

where $c_1 \leqslant 1$ and $c_2 \leqslant 1$ 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:

$$\begin{aligned} C_{rep} &= \frac{1}{2}[D_{KL-norm}^{TT} + D_{KL-norm}^{PL}] \qquad (5.20) \\ &= \frac{1}{2}[e^{-D_{KL}^{TT}} + e^{-D_{KL}^{PL}}] \end{aligned}$$

We can rewrite C_{rep} only in terms of the traffic parameters using (5.17) and (5.18):

$$C_{rep} = \frac{1}{2}\left[\frac{\lambda_l}{\lambda_o} \, e^{-\frac{(\lambda_l - \lambda_o)}{\lambda_o}} + \frac{\mu_l}{\mu_o} \, e^{-\frac{(\mu_l - \mu_o)}{\mu_o}}\right] \tag{5.21}$$

5.2.4 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 [219]:

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

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

$$C_{gen} = c_3\,\Delta + c_4\,\kappa \tag{5.22}$$

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

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 [220].

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:

$$\tilde{\Delta} = \min\Big(\frac{W}{\sum_{k=0}^{l}|B^k|}, 1\Big) \tag{5.23}$$

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 $|B^l| \geqslant 1$ for any learning phase l (in which a learning is performed), $\lim_{l\to\infty}(\tilde{\Delta}) = 0$. In addition, $\tilde{\Delta} = 1$ when $\sum_{k=0}^{l}|B^k| \leqslant W$.

We can then define the adequacy of learning data as

$$\Delta = 1 - \tilde{\Delta} = \Big[1 - \min\Big(\frac{W}{\sum_{k=0}^{l}|B^k|}, 1\Big)\Big] \tag{5.24}$$

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.

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 $0 \leqslant \mathcal{E}(l) \leqslant 1$. 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:

$$\kappa = 1 - \sum_{k=0}^{l} (\frac{1}{2})^{(l-k+1)} \, \mathcal{E}(k) \tag{5.25}$$

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

$$\kappa \leftarrow \frac{1}{2} - \frac{1}{2}\Big[\mathcal{E}(l) - \kappa\Big] \tag{5.26}$$

Generalization Factor:

We now easily calculate the generalization factor C_{gen} combining the "data adequacy" Δ (5.24) and "knowledge" κ (5.25) using the definition of the generalization factor (5.22):

$$C_{gen} = c_3\Big[1 - \min(\frac{W}{\sum_{k=0}^{l} |B^k|}, 1)\Big] + c_4\Big[1 - \sum_{k=0}^{l} (\frac{1}{2})^{(l-k+1)} \, \mathcal{E}(k)\Big] \tag{5.27}$$

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

$$C_{gen} = 1 - \frac{\min(W/\sum_{k=0}^{l} |B^k|, 1) + \sum_{k=0}^{l} (1/2)^{(l-k+1)} \, \mathcal{E}(k)}{2} \tag{5.28}$$

5.3 Results

We evaluate the performance of SSID framework with AADRNN-based IDS for two tasks of malicious traffic detection and compromised device identification which have been presented for both offline and quasi-online learning in Section 4.2 and Section 4.3, respectively.

Accordingly, we first present the performance evaluation results for malicious traffic detection during Mirai Botnet attack. To this end, we use the well-known **Kitsune** dataset [203,204], which contains $764,137$ packet transmissions of both normal and attack traffic cover a consecutive time period of roughly 7137 seconds.

We then present the performance evaluation results for compromised device identification on six different attack data from two datasets, Kitsune [204] and Bot-IoT [206].

In addition, we use the same AADRNN-based IDS presented in Chapter 4 with the following simple decision makers for the malicious traffic detection:

$$y_i = \frac{1}{M} \sum_{m=1}^{M} |x_i^m - \hat{x}_i^m|, \tag{5.29}$$

and for compromised device identification:

$$y_i = \max_{m \in \{1,\ldots,M\}} (|x_i^m - \hat{x}_i^m|). \tag{5.30}$$

During the online learning phase, the parameters of the IDS are updated using the incremental learning algorithm given in Section 4.2.4 for malicious traffic detection and sequential learning algorithm given in Section 4.3.4 for compromised device identification.

Moreover, 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:

$$\mathcal{E}(l) = \frac{1}{|B^l|} \sum_{i=1}^{|B^l|} y_i. \tag{5.31}$$

We also 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.

5.3.1 Performance Evaluation for Malicious Traffic Detection

We first evaluate the performance of SSID for malicious traffic detection during Mirai Botnet attack. Figure 5.5 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}.

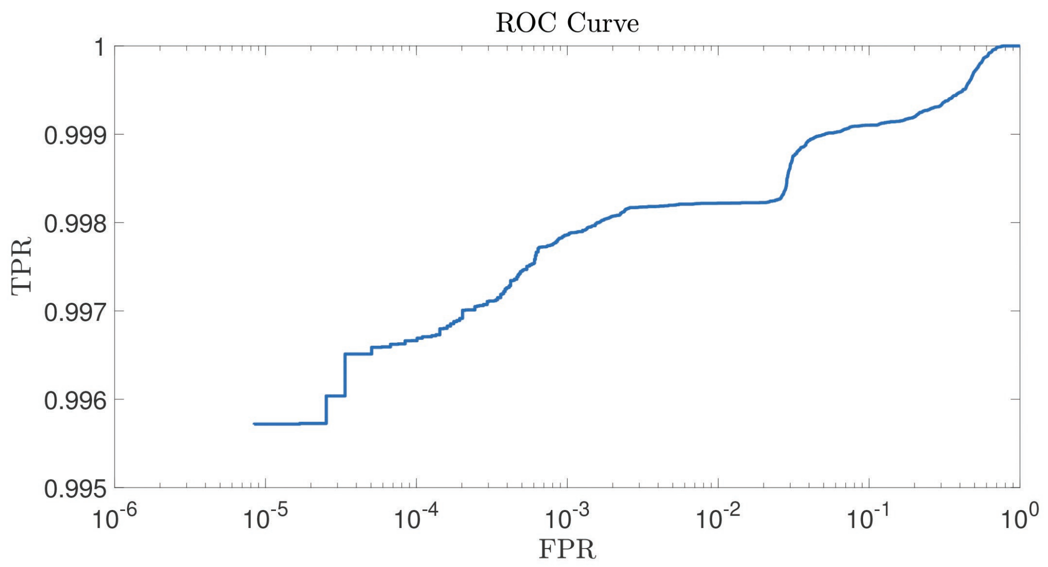

Figure 5.5: ROC curve for the performance of AADRNN-based IDS under the SSID framework for malicious traffic detection

In more detail, in Figure 5.6, 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 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.

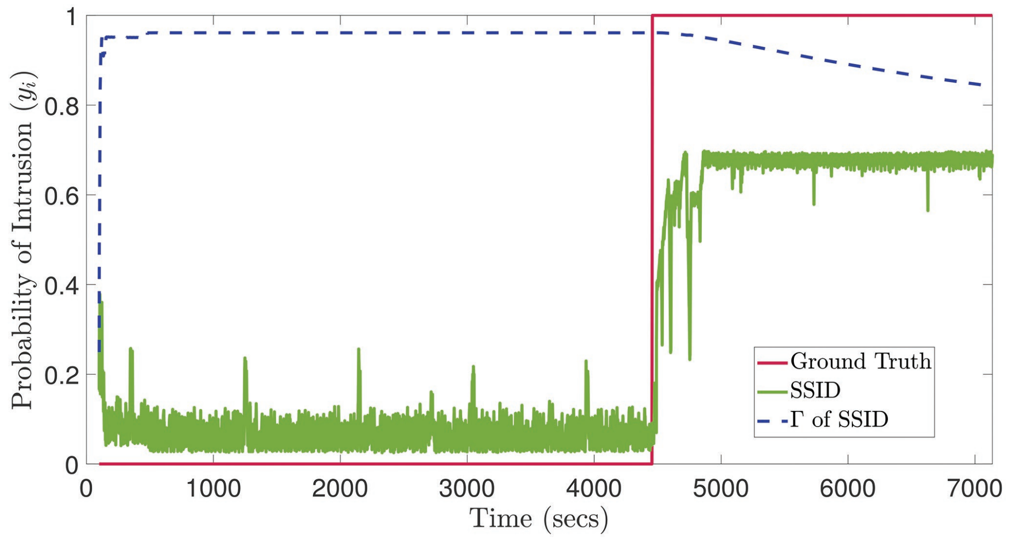

Figure 5.6: Predictions of SSID and the value of trust coefficient Γ with respect to time

Comparison with Quasi-Online and Offline Learning:

We further compare the performance of AADRNN under SSID with the performances of AADRNN with quasi-online and offline learning. Recall (from Section 4.4.2) that all methods with offline learning are trained using approximately $83,000$ benign traffic packets while the AADRNN with incremental (quasi-online) learning trained periodically for the window of 750 packets using AADRNN's own decision where the first 750 packets transmitted are assumed to be normal packets during the cold-start of the network.

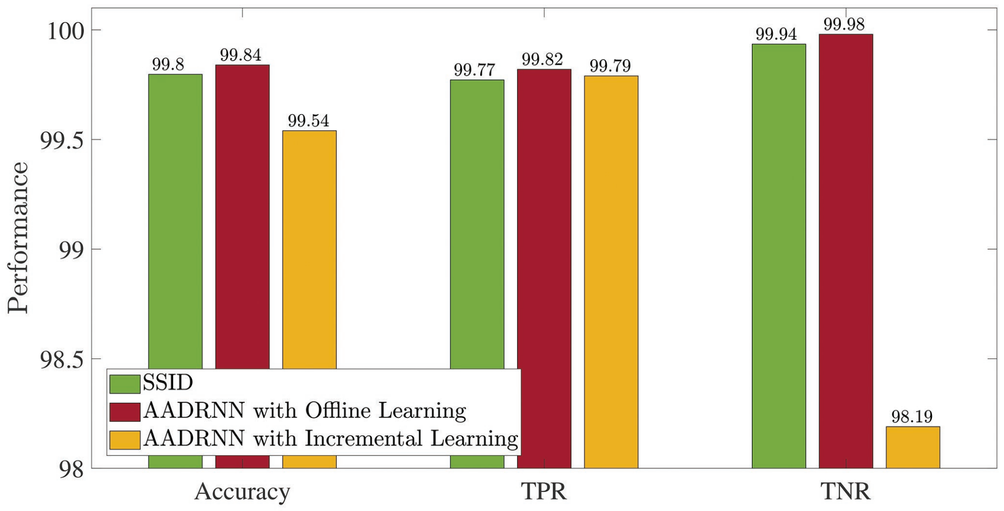

Figure 5.7: Performance comparison between the AADRNN under SSID and the AADRNN with incremental (quasi-online) and offline learning

Figure 5.7 displays the performances of SSID as well as the AADRNN with incremental (quasi-online) and offline learning. The results in this figure first reveal that the AADRNN trained fully online using the SSID framework achieves competitive results with the AADRNN trained offline using approximately $83,000$ packets while the SSID significantly outperforms the AADRNN with incremental learning with respect to all performance metrics. It is worth to note that the SSID learned $4,161$ packets in total along with real-time attack detection.

In contrast with offline and incremental (quasi-online) learning, SSID framework assumes only that the first packet is known to be 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 to train an ML-based attack detector to achieve considerably high performance which is highly competitive against the ML models trained on significantly large dataset.

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 5.8.

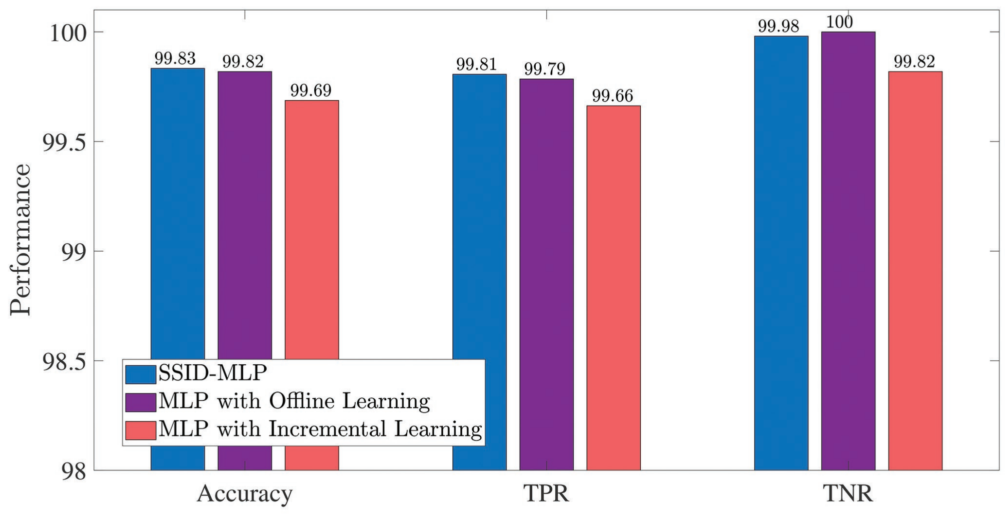

Figure 5.8: Performance comparison between the SSID-MLP and the MLP with incremental (quasi-online) and offline learning

The results in Figure 5.8 show that SSID-MLP achieves slightly higher Accuracy and TPR than MLP with offline learning although MLP with offline learning raises no false alarms (i.e. achieves 100% TNR). Moreover, we see that SSID-MLP significantly outperforms the MLP model that is trained via incremental learning periodically for every 750 packets based on its own output.

Comparison of Different ML Models:

We further compare the performances of AADRNN under SSID (called SSID-AADRNN for clarity) and SSID-MLP with those of the well-known ML models, including KNN and Lasso with offline learning. Figure 5.9 displays the performances of all 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 achieves highly competitive results with the offline trained ML models, while the SSID framework completely eliminates the need for data collection and labeling.

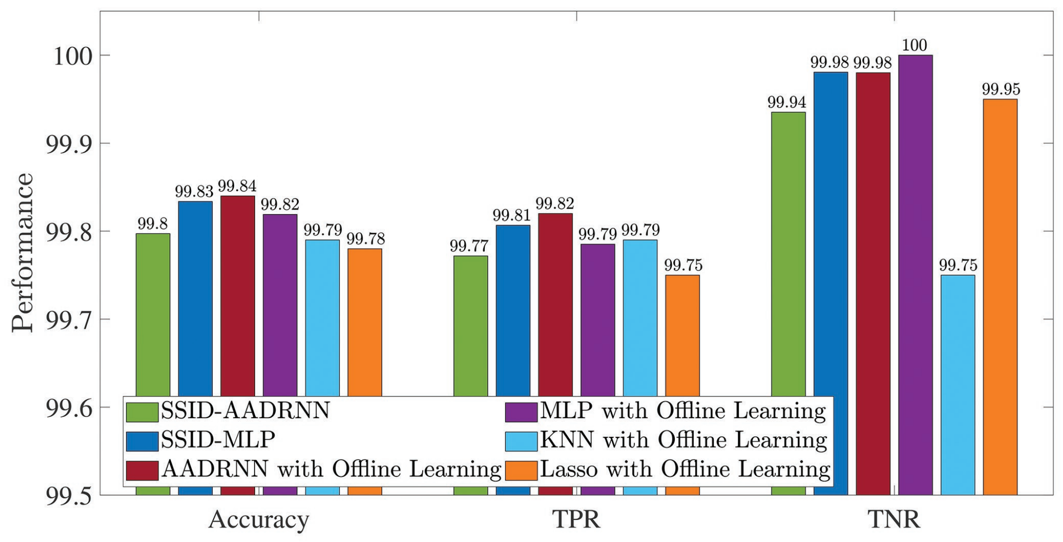

Figure 5.9: Performance comparison between the ML models under the SSID framework and those with offline learning

5.3.2 Performance Evaluation for Compromised Device Identification

We now evaluate the performance of the CDIS presented in Section 4.3 under the SSID framework, in short SSID-CDIS, on two different datasets Kitsune and Bot-IoT. For each dataset, the performance of SSID-CDIS is compared with the original CDIS technique with sequential learning. Recall from Section 4.3 that the compromised device identification is performed for 10 seconds long time window. In order to measure the performance we use the balance accuracy.

Figure 5.10 displays the performance of SSID-CDIS and its comparison with CDIS to identify compromised IP addresses during each of the Mirai Botnet and SYN DoS attacks available in the Kitsune dataset. Results in this figure show that although using the SSID framework provides the same performance as using sequential learning to identify compromised devices during a Mirai Botnet attack, it significantly improves the overall performance of CDIS during a SYN DoS attack. In detail, the box plot on the right of Figure 5.10 shows that the SSID-CDIS achieves 100% median balanced accuracy when there is only one outlier IP address with around 85% accuracy. On the other hand, sequentially trained CDIS has two outlier IP addresses with performances of 50% and 1%, respectively.

Figure 5.11 displays the performance of SSID-CDIS and its comparison with CDIS to identify compromised IP addresses during DDoS and DoS attacks using different communication protocols available in the Bot-IoT dataset. The results in this figure mainly show that the use of the SSID framework enables the decision system CDIS to achieve higher identification performance compared to the use of sequential learning for the majority of attack types. In more detail, starting with the box plot displayed at the far left of this figure, we observe the following results: 1) For DDoS HTTP attack, the overall performance is almost the same for SSID and sequential learning. However, as expected, performance varies slightly for individual

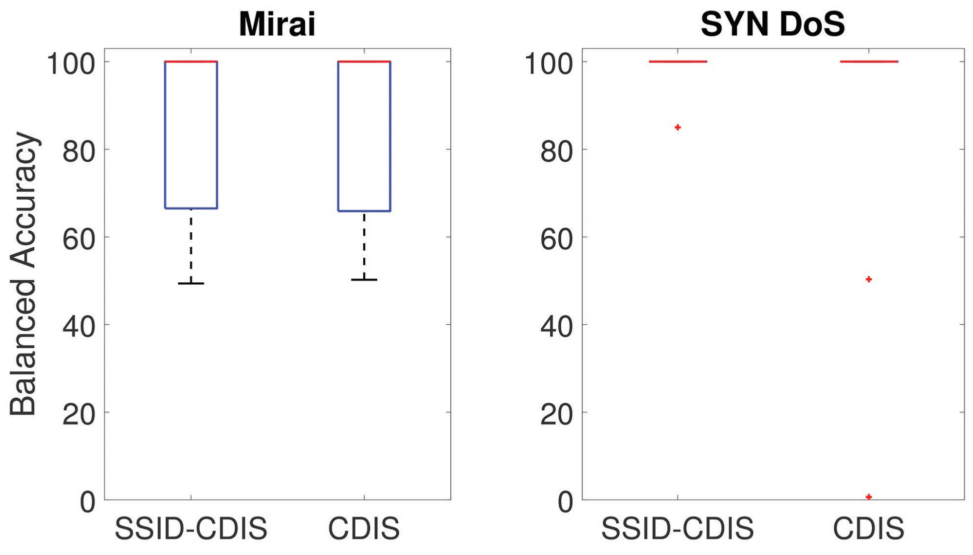

Figure 5.10: Performance comparison of the CDIS trained under the SSID framework with that under sequential quasi-online learning on Kitsune dataset

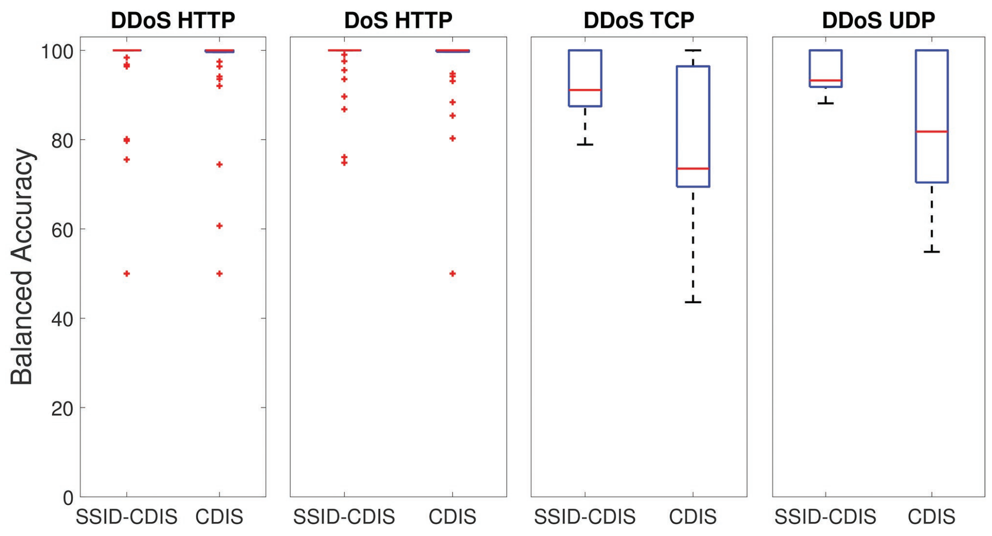

Figure 5.11: Performance comparison of the CDIS trained under the SSID framework with that under sequential quasi-online learning on Bot-IoT dataset

IP addresses. 2) For DoS HTTP attack, using SSID improved the performance by 2% on average with a minimum of 75% balanced accuracy. 3) For DDoS TCP, 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 DDoS TCP attack, SSID provides significant performance improvement to identify compromised devices during DDoS UDP attack. The median accuracy increased by 11%, achieving above 88% balanced accuracy for all IP addresses.

Chapter 6

Conclusions and Future Work

This thesis focused on online learning of ML-based intrusion detection systems, which are especially targeted at detecting zero-day attacks, aiming to perform its learning fully online in parallel with real-time detection, eliminate the need for both labeled and unlabeled offline-collected datasets, and pave the way for adapting the time-variant network traffic behaviour via sequential parameter updates. To this end, this thesis first developed an AADRNN-based IDS with offline unsupervised learning, using only normal "benign" network traffic, with low computation time and considerably high accuracy. Then, this IDS is enhanced with quasi-online learning based on newly developed incremental and sequential learning algorithms for malicious traffic detection and compromised device identification, respectively. Performance evaluation results of offline and quasi-online learning IDS revealed the high – close to offline learning – accuracy of quasi-online learning with a requirement of a small size dataset prior to the real-time operation. That is, although developing fully online learning IDS and achieving competitive performance with offline learning IDS are still needed, the results for quasi-online learning IDS showed that the online learning approach is highly promising for developing an IDS that learns and operates completely in real-time. Subsequently, as the main contribution of this thesis, the novel Self-Supervised Intrusion Detection (SSID) framework has been proposed to enable fully online learning of the parameters of any ML-based IDS, and its performance is evaluated on publicly available datasets revealing that ML models, that are used with the SSID framework using no offline training data, achieve highly competitive performance compared to the offline learning IDS.

In the remainder of this chapter, we first summarize the content of individual chapters in Section 6.1 and present some insights into the future work in Section 6.2.

6.1 Summary

In Chapter 1, we first briefly presented the necessary background on the Internet, networked systems, and the IoT. We then introduced the considered problem, highlighted the main contributions of the present thesis, listed the publications of the author, and outlined the thesis.

In Chapter 2, we reviewed cybersecurity breaches in networked systems, focusing on the most common types of attacks, and examined cybersecurity assurance methods in three categories: intrusion detection systems, authentication and access control mechanisms, and cryptography techniques. Then, we presented a literature review of security issues at each of the perception, network, and application layers of an IoT system.

In Chapter 3, we discussed the use and importance of attack and intrusion detection in IoT networks. Although the majority of existing works focus on detecting malicious traffic, the importance of detecting compromised device identification was also discussed. Subsequently, we presented a comprehensive literature review of the works aimed at detecting: 1) DoS and DDoS Botnet attacks specifically in IoT networks, 2) zero-day (unknown types) attacks, and 3) compromised devices or nodes (i.e. bots).

In Chapter 4, we developed an anomaly-based IDS with offline and quasi-online learning to detect both malicious traffic and compromised IoT devices during Botnet or zero-day attacks. This IDS is comprised of three main functionalities for extracting network traffic metrics, estimating expected metric values for normal "benign" traffic, and making a final decision on whether the analysed metrics indicate an intrusion. Respectively for these functionalities within the proposed IDS, the following modules and algorithms are developed and utilized:

- In order to observe the impact of an intrusion on the network traffic and capture the signatures of an attacker, we proposed original network traffic metrics specifically for each of the tasks malicious traffic detection and compromised device identification. In particular, for malicious traffic detection, three metrics were presented to measure the density of total network traffic while for compromised device identification, six metrics were presented to measure the density of received and transmitted traffic by an individual device.

- We created an auto-associative memory using a DRNN model, called AADRNN, to estimate the metric values expected to be observed during the normal operation of the considered network, i.e. without any intrusion. To this end, the DRNN model is trained using only the normal traffic packets to retrieve the actual values of the metrics from their noise added versions.

- The final decision is made by comparing the expected metric values (i.e. the output of the AADRNN) with the actual metric values. To this end, we developed the novel Statistical Whisker-based Benign Classifier algorithm that detects an intrusion if the actual metrics differ significantly from the expected estimated metrics. The significance of the difference, as well as all parameters of the algorithm, is determined based only on the packet samples used for training.

The performance of the proposed IDS evaluated for malicious traffic detection and compromised device identification during Botnet attacks as well as for detecting zero-day (unknown) attacks. During the performance evaluation, we used publicly available datasets – namely Kitsune, KDD Cup'99, and BotIoT – and compared the performance of the proposed IDS against six well-known ML models. The results revealed that the proposed IDS outperforms the existing methods significantly for both detecting malicious traffic and identifying compromised devices. In addition, quasi-online sequential and incremental learning algorithms have shown high potential for the development of high-performance online learning IDS, which requires low computation time and small data.

In Chapter 5, we proposed the novel Self-Supervised Intrusion Detection (namely SSID) framework designed to train any given IDS – whose parameters are calculated using the network traffic – fully online with no need for human intervention. The SSID framework comprises two successive learning stages initial learning and online learning. Initial learning aims to quickly adapt the IDS parameters for the network where the IDS is newly deployed, while online learning aims to update the parameters whenever an update is required to ensure the high detection accuracy of the IDS. During the real-time operation of the IDS, in parallel to the detection, the SSID framework performs the following main tasks:

- It continually estimates the trustworthiness of intrusion decisions to identify normal and malicious traffic, measuring the ability of the IDS to learn and generalize from data provided by SSID and the extent to which this data can represent current 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 the trust of SSID in those decisions.
- Considering the trustworthiness of the IDS, the selected training packets, and the latest state of network security, the SSID framework determines when to update the IDS parameters.

In this way, the proposed SSID framework eliminates the need for offline data collection, prevents human errors in data labeling, avoids labor costs for model training and data collection through experiments, and – as the most important advantage in terms of performance – enables IDS to easily adapt time varying characteristics of the network traffic.

In this chapter, we also evaluated the performance of the SSID framework for two tasks, malicious traffic detection and compromised device identification, aiming 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 revealed that the ML models trained under the SSID framework requiring no offline dataset achieve considerably high performance compared to the same models with offline and quasi-online (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 the data of 6 different cyberattacks provided by two public datasets Kitsune and Bot-IoT. The results showed that the use of SSID significantly improves the performance of CDIS for the majority of cases.

6.2 Future Work

We now highlight several directions for future work based on the outcomes and foci of this thesis:

- This thesis first presented the development and evaluation of an anomaly-based IDS, which is comprised of Auto-Associative Deep Random Neural Network and Statistical Whisker-based Benign Classifier. The performance evaluation results showed the success of the anomaly-based IDS for various different cases and types of attacks. On the other hand, this thesis did not examine the implementation and performance of the developed IDS in the real IoT test environment. Therefore, future work shall first implement the IDS on a real IoT system and measure its effects on system performance.
- Subsequently, we have proposed the SSID framework targeting to enable fully online self-supervised learning of the IDS parameters. The results revealed that SSID provides fast and successful learning for different ML-based IDS with requiring no human intervention and prior training. Accordingly, we shall evaluate the use of SSID for adapting a pre-trained IDS to use across different unlearned networks as it seems to be a promising approach for fast, self-supervised, and successful adaptation of the IDS parameters for 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. A successful integration of the SSID framework with Federated Learning shall provide secure, distributed and self-supervised online learning for collaborative systems.

- Furthermore, since the proposed self-supervised learning framework is not limited to intrusion detection, future studies may also examine its applications on anomaly detection and other prediction problems in different time-varying systems.

Bibliography

[1] J. Postel, “Internet protocol,” Tech. Rep., 1981.

[2] R. H. Glitho, “Application architectures for machine to machine communications: Research agenda vs. state-of-the art,” in *7th international conference on broadband communications and biomedical applications*. IEEE, 2011, pp. 1–5.

[3] F. Ghavimi and H.-H. Chen, “M2M communications in 3GPP LTE/LTE-A networks: Architectures, service requirements, challenges, and applications,” *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 525–549, 2014.

[4] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[5] K. Xu, Y. Qu, and K. Yang, “A tutorial on the Internet of Things: from a heterogeneous network integration perspective,” *IEEE Network*, vol. 30, no. 2, pp. 102–108, 2016.

[6] O. Bello and S. Zeadally, “Toward efficient smartification of the Internet of Things (IoT) services,” *Future Generation Computer Systems*, vol. 92, pp. 663–673, 2019.

[7] G. Fortino, W. Russo, C. Savaglio, M. Viroli, and M. Zhou, “Modeling opportunistic IoT services in open IoT ecosystems.” in *WOA*, 2017, pp. 90–95.

[8] S. Muralidharan, A. Roy, and N. Saxena, “MDP-IoT: MDP based interest forwarding for heterogeneous traffic in IoT-NDN environment,” *Future Generation Computer Systems*, vol. 79, pp. 892–908, 2018.

[9] F. Terroso-Saenz, A. González-Vidal, A. P. Ramallo-González, and A. F. Skarmeta, “An open IoT platform for the management and analysis of energy data,” *Future Generation Computer Systems*, vol. 92, pp. 1066–1079, 2019.

[10] D. Mendez Mena, I. Papapanagiotou, and B. Yang, “Internet of Things: Survey on security,” *Information Security Journal: A Global Perspective*, vol. 27, no. 3, pp. 162–182, 2018.

[11] K. Zhao and L. Ge, “A survey on the Internet of Things security,” in *2013 Ninth international conference on computational intelligence and security*. IEEE, 2013, pp. 663–667.

[12] K. Lin, M. Chen, J. Deng, M. M. Hassan, and G. Fortino, “Enhanced fingerprinting and trajectory prediction for IoT localization in smart buildings,” *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 3, pp. 1294–1307, 2016.

[13] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social Internet of Things (SIoT)–when social networks meet the Internet of Things: Concept, architecture and network characterization," *Computer networks*, vol. 56, no. 16, pp. 3594–3608, 2012.

[14] A. Hameed and A. Alomary, "Security issues in IoT: A survey," in *2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT).* IEEE, 2019, pp. 1–5.

[15] M. Burhan, R. A. Rehman, B. Khan, and B.-S. Kim, "IoT elements, layered architectures and security issues: A comprehensive survey," *Sensors*, vol. 18, no. 9, 2018.

[16] Intersog, "IoT Security Statistics: 6 Facts [Updated]," Dec 2021, accessed: 2023-03-03. [Online]. Available: https://intersog.com/blog/iot-security-statistics/

[17] 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," *Infocommunications Journal*, vol. 13, no. 2, pp. 32–39, June 2021.

[18] S. Maksuti, M. Zsilak, M. Tauber, and J. Delsing, "Security and autonomic management in system of systems," *Infocommunications Journal*, vol. 13, no. 3, pp. 66–75, September 2021.

[19] "Hp study reveals 70 percent of Internet of Things devices vulnerable to attack." [Online]. Available: https://www.hp.com/us-en/hp-news/press-release.html?id=1744676

[20] 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

[21] NIST, "Information systems security (INFOSEC)," accessed: 2023-03-29. [Online]. Available: https://csrc.nist.gov/glossary/term/information_systems_security

[22] Imperva, "Information Security: The Ultimate Guide," accessed: 2023-03-29. [Online]. Available: https://www.imperva.com/learn/data-security/information-security-infosec

[23] D. Bourgeois and D. T. Bourgeois, "Information systems security," *Information Systems for Business and Beyond*, 2014.

[24] M. Department for Digital, Culture and S. of the United Kingdom, "Cyber Security Breaches Survey 2022," 2022, accessed: 2023-03-08. [Online]. Available: https://www.gov.uk/government/statistics/cyber-security-breaches-survey-2022/cyber-security-breaches-survey-2022

[25] Proofpoint, "2023 State of the Phish," 2023. [Online]. Available: https://www.proofpoint.com/us/resources/threat-reports/state-of-phish

[26] ——, "What Is Phishing?" February 2021, accessed: 2023-03-08. [Online]. Available: https://www.proofpoint.com/us/threat-reference/phishing

[27] U. N. C. S. Center, "Phishing attacks: defending your organisation," February 2018, accessed: 2023-03-08. [Online]. Available: https://www.clickguard.com/blog/recent-botnet-attacks-2022/

[28] Imperva, “Phishing attacks.” [Online]. Available: https://www.imperva.com/learn/application-security/phishing-attack-scam/

[29] Cloudflare, “What is a phishing attack?” accessed: 2023-03-08. [Online]. Available: https://www.cloudflare.com/en-gb/learning/access-management/phishing-attack/

[30] J. R. Tietsort, “17 Types of Cyber Attacks Commonly Used By Hackers,” January 2023, accessed: 2023-03-08. [Online]. Available: https://www.aura.com/learn/types-of-cyber-attacks

[31] C. Griffiths, “The Latest 2023 Ransomware Statistics,” March 2023, accessed: 2023-03-08. [Online]. Available: https://aag-it.com/the-latest-ransomware-statistics

[32] McAfee, “What is Malware? How to Stay Protected from Malware Attacks,” January 2023, accessed: 2023-03-08. [Online]. Available: https://www.mcafee.com/en-us/antivirus/malware.html

[33] C. Cimpanu, “New Silex malware is bricking IoT devices, has scary plans,” June 2019, accessed: 2023-03-08. [Online]. Available: https://www.zdnet.com/article/new-silex-malware-is-bricking-iot-devices-has-scary-plans

[34] D. Read, “New Linux Worm Attacks IoT Devices,” June 2019, accessed: 2023-03-08. [Online]. Available: https://www.darkreading.com/iot/new-linux-worm-attacks-iot-devices

[35] O. Yoachimik, “DDoS attack trends for 2022 Q2,” June 2022, accessed: 2023-03-09. [Online]. Available: https://blog.cloudflare.com/ddos-attack-trends-for-2022-q2/

[36] ExtraHop, “DENIAL OF SERVICE ATTACK: DEFINITION, EXAMPLES, AND PREVENTION,” January 2022, accessed: 2023-03-09. [Online]. Available: https://www.extrahop.com/resources/attacks/dos/

[37] B. B. Gupta and A. Dahiya, *Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures*. CRC press, 2021.

[38] Netcraft, “95% of HTTPS servers vulnerable to trivial MITM attacks,” March 2016, accessed: 2023-03-10. [Online]. Available: https://news.netcraft.com/archives/2016/03/17/95-of-https-servers-vulnerable-to-trivial-mitm-attacks.html

[39] Z. Cekerevac, Z. Dvorak, L. Prigoda, and P. Cekerevac, “Internet of Things and the man-in-the-middle attacks–security and economic risks,” *MEST Journal*, vol. 5, no. 2, pp. 15–25, 2017.

[40] A. Raj, “Unauthorized access the biggest cause of data breaches,” July 2022, accessed: 2023-03-10. [Online]. Available: https://techwireasia.com/2022/07/unauthorized-access-the-biggest-cause-of-data-breaches/

[41] H. W. Glaspie and W. Karwowski, “Human factors in information security culture: A literature review,” in *Advances in Human Factors in Cybersecurity: Proceedings of the AHFE 2017 International Conference on Human Factors in Cybersecurity, July 17- 21, 2017, The Westin Bonaventure Hotel, Los Angeles, California, USA 8*. Springer, 2018, pp. 269–280.

[42] L. Alzahrani and K. P. Seth, “The impact of organizational practices on the information security management performance,” *Information*, vol. 12, no. 10, 2021.

[43] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung, “Intrusion detection system: A comprehensive review,” *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[44] J. Augusto-Gonzalez, A. Collen, S. Evangelatos, M. Anagnostopoulos, G. Spathoulas, K. M. Giannoutakis, K. Votis, D. Tzovaras, B. Genge, E. Gelenbe *et al.*, “From internet of threats to Internet of Things: A cyber security architecture for smart homes,” in *2019 IEEE 24th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 2019, pp. 1–6.

[45] H. Bi and E. Gelenbe, “Emergency management systems and algorithms: a comprehensive survey,” *arXiv preprint arXiv:1907.04136*, 2019.

[46] R. G. Bace, P. Mell *et al.*, “Intrusion detection systems,” 2001.

[47] K. Scarfone, P. Mell *et al.*, “Guide to intrusion detection and prevention systems (idps),” *NIST special publication*, vol. 800, no. 2007, p. 94, 2007.

[48] E. Gelenbe, P. Fröhlich, M. Nowak, S. Papadopoulos, A. Protogerou, A. Drosou, and D. Tzovaras, “Iot network attack detection and mitigation,” in *2020 9th Mediterranean Conference on Embedded Computing (MECO)*. IEEE, 2020, pp. 1–6.

[49] M. Nasereddin, M. Nakip, and E. Gelenbe, “Measurement based evaluation and mitigation of flood attacks on a lan test-bed,” *arXiv preprint arXiv:2305.10565*, 2023.

[50] Z. M. Yusop and J. Abawajy, “Analysis of insiders attack mitigation strategies,” *Procedia-Social and Behavioral Sciences*, vol. 129, pp. 581–591, 2014.

[51] E. Gelenbe, J. Domanska, P. Fröhlich, M. P. Nowak, and S. Nowak, “Self-aware networks that optimize security, qos, and energy,” *Proceedings of the IEEE*, vol. 108, no. 7, pp. 1150–1167, 2020.

[52] J. Rosenberg, “Chapter e6 - embedded security,” in *Rugged Embedded Systems*, A. Vega, P. Bose, and A. Buyuktosunoglu, Eds. Boston: Morgan Kaufmann, 2017, pp. e1–e74.

[53] D. Bolzoni and S. Etalle, “Approaches in anomaly-based network intrusion detection systems,” *Intrusion Detection Systems*, vol. 38, pp. 1–15, 2008.

[54] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, “Survey of intrusion detection systems: techniques, datasets and challenges,” *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[55] C. Modi, D. Patel, B. Borisaniya, H. Patel, A. Patel, and M. Rajarajan, “A survey of intrusion detection techniques in cloud,” *Journal of network and computer applications*, vol. 36, no. 1, pp. 42–57, 2013.

[56] N. F. Syed, Z. Baig, A. Ibrahim, and C. Valli, “Denial of service attack detection through machine learning for the IoT,” *Journal of Information and Telecommunication*, vol. 4, no. 4, pp. 482–503, 2020.

[57] E. Gelenbe, O. H. Abdelrahman, and G. Gorbil, "Detection and mitigation of signaling storms in mobile networks," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, 2016, pp. 1–5.

[58] Z. Tan, A. Jamdagni, X. He, P. Nanda, R. P. Liu, and J. Hu, "Detection of denial-of-service attacks based on computer vision techniques," *IEEE Transactions on Computers*, vol. 64, no. 9, pp. 2519–2533, 2015.

[59] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE communications surveys & tutorials*, vol. 16, no. 1, pp. 266–282, 2013.

[60] Z. Yang, X. Liu, T. Li, D. Wu, J. Wang, Y. Zhao, and H. Han, "A systematic literature review of methods and datasets for anomaly-based network intrusion detection," *Computers & Security*, p. 102675, 2022.

[61] G. Qu, S. Hariri, and M. Yousif, "Multivariate statistical analysis for network attacks detection," in *The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005.*, 2005, pp. 9–.

[62] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A system for denial-of-service attack detection based on multivariate correlation analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, pp. 447–456, 2014.

[63] H. Hindy, R. Atkinson, C. Tachtatzis, J.-N. Colin, E. Bayne, and X. Bellekens, "Utilising deep learning techniques for effective zero-day attack detection," *Electronics*, vol. 9, no. 10, p. 1684, 2020.

[64] X. Wang, Z. Yan, R. Zhang, and P. Zhang, "Attacks and defenses in user authentication systems: A survey," *Journal of Network and Computer Applications*, vol. 188, p. 103080, 2021.

[65] S. Boonkrong, *Authentication and Access Control: Practical Cryptography Methods and Tools*. Springer, 2021.

[66] T. Nandy, M. Y. I. B. Idris, R. Md Noor, L. Mat Kiah, L. S. Lun, N. B. Annuar Juma'at, I. Ahmedy, N. Abdul Ghani, and S. Bhattacharyya, "Review on security of Internet of Things authentication mechanism," *IEEE Access*, vol. 7, pp. 151 054–151 089, 2019.

[67] A. Conklin, G. Dietrich, and D. Walz, "Password-based authentication: a system perspective," in *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, 2004, pp. 10 pp.–.

[68] B. Lampson, M. Abadi, M. Burrows, and E. Wobber, "Authentication in distributed systems: Theory and practice," *ACM Transactions on Computer Systems (TOCS)*, vol. 10, no. 4, pp. 265–310, 1992.

[69] K. M. Renuka, S. Kumari, D. Zhao, and L. Li, "Design of a secure password-based authentication scheme for M2M networks in IoT enabled cyber-physical systems," *IEEE Access*, vol. 7, pp. 51 014–51 027, 2019.

[70] C. Meshram, R. W. Ibrahim, L. Deng, S. W. Shende, S. G. Meshram, and S. K. Barve, “A robust smart card and remote user password-based authentication protocol using extended chaotic maps under smart cities environment,” *Soft Computing*, vol. 25, no. 15, pp. 10 037–10 051, 2021.

[71] O. Lucia, B. Isong, N. Gasela, and A. M. Abu-Mahfouz, “Device authentication schemes in IoT: A review,” in *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 2019, pp. 1–6.

[72] S. Agrawal and P. Ahlawat, “A survey on the authentication techniques in Internet of Things,” in *2020 IEEE International Students’ Conference on Electrical,Electronics and Computer Science (SCEECS)*, 2020, pp. 1–5.

[73] M. Naveed Aman, S. Taneja, B. Sikdar, K. C. Chua, and M. Alioto, “Token-based security for the Internet of Things with dynamic energy-quality tradeoff,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2843–2859, 2019.

[74] M. Dammak, O. R. M. Boudia, M. A. Messous, S. M. Senouci, and C. Gransart, “Token-based lightweight authentication to secure IoT networks,” in *2019 16th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, 2019, pp. 1–4.

[75] R. Amin, N. Kumar, G. Biswas, R. Iqbal, and V. Chang, “A light weight authentication protocol for iot-enabled devices in distributed cloud computing environment,” *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[76] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen, and Y. Koucheryavy, “Multi-factor authentication: A survey,” *Cryptography*, vol. 2, no. 1, p. 1, 2018.

[77] J. Flynn, “17 essential multi-factor authentication (MFA) statistics [2023],” February 2023, accessed: 2023-03-11. [Online]. Available: https://www.zippia.com/advice/mfa-statistics/

[78] H. N. Noura, R. Melki, and A. Chehab, “Secure and lightweight mutual multi-factor authentication for IoT communication systems,” in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–7.

[79] A. Y. F. Alsahlani and A. Popa, “LMAAS-IoT: Lightweight multi-factor authentication and authorization scheme for real-time data access in IoT cloud-based environment,” *Journal of Network and Computer Applications*, vol. 192, p. 103177, 2021.

[80] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2020.

[81] G. C. Kessler, “An overview of cryptography (updated version 24 january 2019),” 2019.

[82] A. Menezes and D. Stebila, “Challenges in cryptography,” *IEEE Security & Privacy*, vol. 19, no. 2, pp. 70–73, 2021.

[83] W. Diffie and M. E. Hellman, “Multiuser cryptographic techniques,” in *Proceedings of the June 7-10, 1976, national computer conference and exposition*, 1976, pp. 109–112.

[84] S. Benzarti, B. Triki, and O. Korbaa, "A survey on attacks in Internet of Things based networks," in *2017 International conference on engineering & MIS (ICEMIS)*. IEEE, 2017, pp. 1–7.

[85] R. Duan, X. Chen, and T. Xing, "A QoS architecture for IoT," in *2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing*, 2011, pp. 717–720.

[86] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of Internet of Things (IoT)," *International Journal of Computer Applications*, vol. 111, no. 7, pp. 1–6, 2015.

[87] I. Mashal, O. Alsaryrah, T.-Y. Chung, C.-Z. Yang, W.-H. Kuo, and D. P. Agrawal, "Choices for interaction with things on internet and underlying issues," *Ad Hoc Networks*, vol. 28, pp. 68–90, 2015.

[88] M. Yun and B. Yuxin, "Research on the architecture and key technology of Internet of Things (IoT) applied on smart grid," in *2010 International Conference on Advances in Energy Engineering*. IEEE, 2010, pp. 69–72.

[89] O. Said and M. Masud, "Towards Internet of Things: Survey and future vision," *International Journal of Computer Networks*, vol. 5, no. 1, pp. 1–17, 2013.

[90] B. K. Mohanta, D. Jena, S. Ramasubbareddy, M. Daneshmand, and A. H. Gandomi, "Addressing security and privacy issues of IoT using blockchain technology," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 881–888, 2020.

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

[92] A. Kamble and S. Bhutad, "Survey on Internet of Things (IoT) security issues & solutions," in *2018 2nd International Conference on Inventive Systems and Control (ICISC)*. IEEE, 2018, pp. 307–312.

[93] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of Things (IoT) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015, pp. 336–341.

[94] M. Gharooni, M. Zamani, M. Mansourizadeh, and S. Abdullah, "A confidential rfid model to prevent unauthorized access," in *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*. IEEE, 2011, pp. 1–5.

[95] U. Mujahid, M. Najam-ul Islam, and M. A. Shami, "Rcia: A new ultralightweight rfid authentication protocol using recursive hash," *International Journal of Distributed Sensor Networks*, vol. 11, no. 1, p. 642180, 2015.

[96] H. Ding, J. Han, Y. Zhang, F. Xiao, W. Xi, G. Wang, and Z. Jiang, "Preventing unauthorized access on passive tags," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1115–1123.

[97] H. Ding, J. Han, C. Zhao, G. WANG, W. Xi, Z. Jiang, and J. Zhao, "Arbitrator2. 0: Preventing unauthorized access on passive tags," *IEEE Transactions on Mobile Computing*, 2020.

[98] X. Luo, L. Yin, C. Li, C. Wang, F. Fang, C. Zhu, and Z. Tian, "A lightweight privacy-preserving communication protocol for heterogeneous IoT environment," *IEEE Access*, vol. 8, pp. 67 192–67 204, 2020.

[99] M. S. Wara and Q. Yu, "New replay attacks on zigbee devices for Internet-of-Things (IoT) applications," in *2020 IEEE International Conference on Embedded Software and Systems (ICESS)*, 2020, pp. 1–6.

[100] H. Kim, "Protection against packet fragmentation attacks at 6lowpan adaptation layer," in *2008 International Conference on Convergence and Hybrid Information Technology*, 2008, pp. 796–801.

[101] K. M. Malik, A. Javed, H. Malik, and A. Irtaza, "A light-weight replay detection framework for voice controlled IoT devices," *IEEE Journal of Selected Topics in Signal Processing*, vol. 14, no. 5, pp. 982–996, 2020.

[102] H. T. Reda, A. Anwar, and A. Mahmood, "Comprehensive survey and taxonomies of false data injection attacks in smart grids: attack models, targets, and impacts," *Renewable and Sustainable Energy Reviews*, vol. 163, p. 112423, 2022.

[103] Y. Li, X. Wei, Y. Li, Z. Dong, and M. Shahidehpour, "Detection of false data injection attacks in smart grid: A secure federated deep learning approach," *IEEE Transactions on Smart Grid*, vol. 13, no. 6, pp. 4862–4872, 2022.

[104] Y. Han, H. Feng, K. Li, and Q. Zhao, "False data injection attacks detection with modified temporal multi-graph convolutional network in smart grids," *Computers & Security*, vol. 124, p. 103016, 2023.

[105] S. Tan, P. Xie, J. M. Guerrero, and J. C. Vasquez, "False data injection cyber-attacks detection for multiple dc microgrid clusters," *Applied Energy*, vol. 310, p. 118425, 2022.

[106] W.-T. Lin, G. Chen, and Y. Huang, "Incentive edge-based federated learning for false data injection attack detection on power grid state estimation: A novel mechanism design approach," *Applied Energy*, vol. 314, p. 118828, 2022.

[107] M. Abomhara and G. M. Køien, "Security and privacy in the Internet of Things: Current status and open issues," in *2014 international conference on privacy and security in mobile systems (PRISMS)*. IEEE, 2014, pp. 1–8.

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

[109] CISA, "Understanding Denial-of-Service attacks," accessed: 2023-04-08. [Online]. Available: https://www.cisa.gov/news-events/news/understanding-denial-service-attacks

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

[111] M. De Donno, N. Dragoni, A. Giaretta, and A. Spognardi, "Ddos-capable IoT malwares: Comparative analysis and mirai investigation," *Security and Communication Networks*, vol. 2018, pp. 1–30, 2018.

[112] P. Kumari and A. K. Jain, "A comprehensive study of DDoS attacks over IoT network and their countermeasures," *Computers & Security*, p. 103096, 2023.

[113] E. Bertino and N. Islam, "Botnets and Internet of Things security," *Computer*, vol. 50, no. 2, pp. 76–79, 2017.

[114] 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 *Proceedings of the 26th USENIX Security Symposium*, 2017.

[115] A. Kumar and T. J. Lim, "Early detection of Mirai-like IoT Bots in large-scale networks through sub-sampled packet traffic analysis," 2019.

[116] H. Sinanović and S. Mrdovic, "Analysis of mirai malicious software," in *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. IEEE, 2017, pp. 1–5.

[117] J. Margolis, T. T. Oh, S. Jadhav, Y. H. Kim, and J. N. Kim, "An in-depth analysis of the mirai botnet," in *2017 International Conference on Software Security and Assurance (ICSSA)*. IEEE, 2017, pp. 6–12.

[118] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommunication systems*, vol. 73, no. 1, pp. 3–25, 2020.

[119] S. Mergendahl, D. Sisodia, J. Li, and H. Cam, "Source-end DDoS defense in IoT environments," in *Proceedings of the 2017 workshop on Internet of Things security and privacy*, 2017, pp. 63–64.

[120] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in iot: a survey," *The Journal of Supercomputing*, vol. 76, pp. 5320–5363, 2020.

[121] N. Ravi and S. M. Shalinie, "Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3559–3570, 2020.

[122] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, "Flowguard: An intelligent edge defense mechanism against IoT DDoS attacks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, 2020.

[123] B. Gupta, P. Chaudhary, X. Chang, and N. Nedjah, "Smart defense against distributed denial of service attack in IoT networks using supervised learning classifiers," *Computers & Electrical Engineering*, vol. 98, p. 107726, 2022.

[124] C. Li, Z. Qin, E. Novak, and Q. Li, "Securing sdn infrastructure of iot–fog networks from mitm attacks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1156–1164, 2017.

[125] J. J. Kang, K. Fahd, S. Venkatraman, R. Trujillo-Rasua, and P. Haskell-Dowland, "Hybrid routing for man-in-the-middle (mitm) attack detection in IoT networks," in *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2019, pp. 1–6.

[126] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[127] A. Lahmadi, A. Duque, N. Heraief, and J. Francq, "Mitm attack detection in ble networks using reconstruction and classification machine learning techniques," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2020, pp. 149–164.

[128] "What is phishing? examples and phishing quiz," Jul 2021, accessed: 2023-03-08. [Online]. Available: https://www.cisco.com/c/en/us/products/security/email-security/what-is-phishing.html

[129] M. N. Banu and S. M. Banu, "A comprehensive study of phishing attacks," *International Journal of Computer Science and Information Technologies*, vol. 4, no. 6, pp. 783–786, 2013.

[130] A. Sadiq, M. Anwar, R. A. Butt, F. Masud, M. K. Shahzad, S. Naseem, and M. Younas, "A review of phishing attacks and countermeasures for Internet of Things-based smart business applications in industry 4.0," *Human behavior and emerging technologies*, vol. 3, no. 5, pp. 854–864, 2021.

[131] D. Liu and J.-H. Lee, "Cnn based malicious website detection by invalidating multiple web spams," *IEEE Access*, vol. 8, pp. 97 258–97 266, 2020.

[132] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE Access*, vol. 7, pp. 15 196–15 209, 2019.

[133] E. Gandotra and D. Gupta, "Improving spoofed website detection using machine learning," *Cybernetics and Systems*, vol. 52, no. 2, pp. 169–190, 2021.

[134] L. Bustio-Martínez, M. A. Álvarez-Carmona, V. Herrera-Semenets, C. Feregrino-Uribe, and R. Cumplido, "A lightweight data representation for phishing urls detection in IoT environments," *Information Sciences*, vol. 603, pp. 42–59, 2022.

[135] S. B. Gopal, C. Poongodi, D. Nanthiya, T. Kirubakaran, D. Logeshwar, and B. K. Saravanan, "Autoencoder based architecture for mitigating phishing url attack in the Internet of Things (iot) using deep neural networks," in *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, 2022, pp. 427–431.

[136] A. R. Dargad and S. G. Sutar, "Enhanced data leakage detection in IoT network backup with cloud using paillier homomorphic cryptosystem," *Asian Journal For Convergence In Technology (AJCT) ISSN-2350-1146*, vol. 3, 2017.

[137] Z.-K. Zhang, M. C. Y. Cho, C.-W. Wang, C.-W. Hsu, C.-K. Chen, and S. Shieh, "IoT security: Ongoing challenges and research opportunities," in *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, 2014, pp. 230–234.

[138] X. Yu, J. Qiu, X. Yang, Y. Cong, and L. Du, "An graph-based adaptive method for fast detection of transformed data leakage in IoT via WSN," *IEEE Access*, vol. 7, pp. 137 111–137 121, 2019.

[139] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial iot," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4177–4186, 2020.

[140] D. Wei and X. Qiu, "Status-based detection of malicious code in Internet of Things (IoT) devices," in *2018 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2018, pp. 1–7.

[141] J. Jeon, J. H. Park, and Y.-S. Jeong, "Dynamic analysis for IoT malware detection with convolution neural network model," *IEEE Access*, vol. 8, pp. 96 899–96 911, 2020.

[142] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-g. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3187–3196, 2018.

[143] A. Mohaisen, O. Alrawi, and M. Mohaisen, "Amal: high-fidelity, behavior-based automated malware analysis and classification," *computers & security*, vol. 52, pp. 251–266, 2015.

[144] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *computers & security*, vol. 77, pp. 578–594, 2018.

[145] 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 *International ISCIS Security Workshop*. Springer, Cham, 2018, pp. 79–89.

[146] S. Evmorfos, G. Vlachodimitropoulos, N. Bakalos, and E. Gelenbe, "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.

[147] S. Rathore and J. H. Park, "Semi-supervised learning based distributed attack detection framework for IoT," *Applied Soft Computing*, vol. 72, pp. 79–89, 2018.

[148] A. Iqbal, S. Aftab, I. Ullah, M. A. Saeed, and A. Husen, "A classification framework to detect DoS attacks," *International Journal of Computer Network & Information Security*, vol. 11, no. 9, 2019.

[149] A. I. Al-issa, M. Al-Akhras, M. S. ALsahli, and M. Alawairdhi, "Using machine learning to detect DoS attacks in wireless sensor networks," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, 2019, pp. 107–112.

[150] S. Wankhede and D. Kshirsagar, "DoS attack detection using machine learning and neural network," in *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. IEEE, 2018, pp. 1–5.

[151] SAM, "Security IoT Landscape," p. 1–17, 2021, accessed: 2023-03-03. [Online]. Available: https://securingsam.com/2021-iot-security-landscape/

[152] C. McCart, "15+ shocking botnet statistics and facts for 2023," Oct 2022, accessed: 2023-03-03. [Online]. Available: https://www.comparitech.com/blog/information-security/botnet-statistics/

[153] S. M. Team, "Spamhaus Botnet Threat Update: Q4-2021," Jan 2022, accessed: 2023-03-03. [Online]. Available: https://www.spamhaus.org/news/article/817/spamhaus-botnet-threat-update-q4-2021

[154] 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.

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

[156] 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.

[157] 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.

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

[159] 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.

[160] 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.

[161] 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.

[162] 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.

[163] 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.

[164] 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.

[165] 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.

[166] 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.

[167] 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.

[168] M. Moradi and M. Zulkernine, "A neural network based system for intrusion detection and classification of attacks," in *Proceedings of the IEEE international conference on advances in intelligent systems-theory and applications*. IEEE Lux-embourg-Kirchberg, Luxembourg, 2004, pp. 15–18.

[169] M. Catillo, M. Rak, and U. Villano, "2l-zed-ids: A two-level anomaly detector for multiple attack classes," in *Web, Artificial Intelligence and Network Applications*, L. Barolli, F. Amato, F. Moscato, T. Enokido, and M. Takizawa, Eds. Cham: Springer International Publishing, 2020, pp. 687–696.

[170] Z. E. Huma, S. Latif, J. Ahmad, Z. Idrees, A. Ibrar, Z. Zou, F. Alqahtani, and F. Baothman, "A hybrid deep random neural network for cyberattack detection in the Industrial Internet of Things," *IEEE Access*, vol. 9, pp. 55 595–55 605, 2021.

[171] S. Latif, Z. Zou, Z. Idrees, and J. Ahmad, "A novel attack detection scheme for the Industrial Internet of Things using a lightweight random neural network," *IEEE Access*, vol. 8, pp. 89 337–89 350, 2020.

[172] I. H. Sarker, "Cyberlearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks," *Internet of Things*, vol. 14, p. 100393, 2021.

[173] B. Subba, S. Biswas, and S. Karmakar, "A neural network based system for intrusion detection and attack classification," in *2016 Twenty Second National Conference on Communication (NCC)*, 2016, pp. 1–6.

[174] Y. Zhang, P. Li, and X. Wang, "Intrusion detection for IoT based on improved genetic algorithm and deep belief network," *IEEE Access*, vol. 7, pp. 31 711–31 722, 2019.

[175] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[176] Y. N. Kunang, S. Nurmaini, D. Stiawan, and B. Y. Suprapto, "Attack classification of an intrusion detection system using deep learning and hyperparameter optimization," *Journal of Information Security and Applications*, vol. 58, p. 102804, 2021.

[177] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3369–3388, 2018.

[178] P. Laskov, P. Düssel, C. Schäfer, and K. Rieck, "Learning intrusion detection: supervised or unsupervised?" in *Image Analysis and Processing–ICIAP 2005: 13th International Conference, Cagliari, Italy, September 6-8, 2005. Proceedings 13*. Springer, 2005, pp. 50–57.

[179] M. Z. Alom and T. M. Taha, "Network intrusion detection for cyber security using unsupervised deep learning approaches," in *2017 IEEE National Aerospace and Electronics Conference (NAECON)*, 2017, pp. 63–69.

[180] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *The Journal of Supercomputing*, vol. 75, pp. 5597–5621, 2019.

[181] D. Goodin, "100,000-strong Botnet built on router 0-day could strike at any time," December 2017, accessed: 2023-03-22. [Online]. Available: https://arstechnica.com/information-technology/2017/12/100000-strong-botnet-built-on-router-0-day-could-strike-at-any-time/

[182] Imperva, "2022 Imperva Bad Bot Report," p. 1–37, 2022, accessed: 2023-03-03. [Online]. Available: https://www.imperva.com/resources/resource-library/reports/bad-bot-report/

[183] J. Biggs, "Hackers release source code for a powerful DDoS app called Mirai," October 2018, accessed: 2023-03-22. [Online]. Available: https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/

[184] R. Hackett, "Why a hacker dumped code behind colossal website-trampling botnet," October 2016, accessed: 2023-03-22. [Online]. Available: https://finance.yahoo.com/news/why-hacker-dumped-code-behind-145847907.html

[185] N. Statt, "How an army of vulnerable gadgets took down the web today," October 2016, accessed: 2023-03-22. [Online]. Available: https://www.theverge.com/2016/10/21/13362354/dyn-dns-ddos-attack-cause-outage-status-explained

[186] T. ClickGUARD, "The Most Recent Botnet Attacks: The 2022 Edition," June 2022, accessed: 2023-03-03. [Online]. Available: https://www.clickguard.com/blog/recent-botnet-attacks-2022/

[187] 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.

[188] 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.

[189] 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.

[190] 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.

[191] 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.

[192] 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.

[193] 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.

[194] 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.

[195] 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.

[196] 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.

[197] 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.

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

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

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

[201] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[202] N.-y. Liang, G.-b. Huang, P. Saratchandran, and N. Sundararajan, “A fast and accurate online sequential learning algorithm for feedforward networks,” *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[203] 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.

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

[205] “KDD Cup 1999 Data.” [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[206] 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.

[207] 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," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[208] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, no. 347-352, pp. 240–242, 1895.

[209] R. G. Lomax, *Statistical concepts: A second course.* Lawrence Erlbaum Associates Publishers, 2007.

[210] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

[211] 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.

[212] 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.

[213] 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.

[214] 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.

[215] 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.

[216] 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.

[217] 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.

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

[219] 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.

[220] 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.