

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO



Anomaly Detection Models for Cloud-Edge Intrusion Detection in Customer Networks

Nuno Schumacher

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ricardo Morla

Co-Supervisor: Pedro Santos

July 29, 2022

Abstract

Home networks are more and more susceptible to cyberattacks that can compromise user operation and data integrity, with the increasing presence of different types of devices, such as Internet of Things (IoT) appliances or personal computers. When vulnerabilities on such devices are discovered, they can be exploited for multiple purposes, such as infection to take part in a botnet for large scale distributed attacks.

Some intrusion detection systems (IDSs) that attempt to detect such attacks rely on predetermined signatures, manually configured based on expected patterns. However, zero-days attacks can be performed before the signatures are employed, or the traffic pattern can be changed to avoid being detected. Other IDSs resort to learning techniques, but require generating or collecting samples of malicious data to train the models for the case of supervised learning. When anomaly detection (AD) systems that don't require malicious examples to be trained are used, their capability of detecting new and unseen benign traffic is often not properly evaluated.

The datasets commonly used for IDS performance evaluation are often outdated or lacking in variety, which can devalue the relevance of some results for today's needs. Furthermore, the different possible types of home traffic (e.g., web browsing, gaming and IoT traffic) are usually not taken into account in evaluation to provide solid results and only some of these are considered instead, which limits the representativity of real networks.

In this work, we make a selection of some appropriate datasets for IDS evaluation and propose a machine learning (ML) architecture with the goal of learning the typical behaviour of the home network. We then evaluate it's capability of accurately classifying previously seen benign traffic, unseen benign traffic from different sources and varied malign traffic.

Acknowledgments

Firstly, a very special thanks to professors Pedro Santos and Ricardo Morla for the continuous support, insightful discussions and relevant feedback that allowed me to successfully carry out this work and learn immensely from the process.

I'm deeply grateful for everything my parents have done for me, specially in this stage of my life. Many, many thanks for my friends and colleges for the encouragement and good times along the way. Lastly, I'm thankful to everyone else that played an important role on my academic and personal life that surely left a lasting mark on me.

This work was partially supported by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020), and by the Portuguese National Innovation Agency (ANI) through the Operational Competitiveness Programme and Internationalization (COMPETE 2020) under the PT2020 Partnership Agreement, through the European Regional Development Fund (ERDF), within project(s) grant nr. 69522, POCI-01-0247-FEDER-069522 (MIRAI).

Nuno Schumacher

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Context	2
1.3	Objectives	3
1.4	Document structure	3
2	Background and Related Work	5
2.1	Background	5
2.1.1	Overview on Cybersecurity and the IoT Threat Model	5
2.1.2	Intrusion Detection Systems as Agents of Cybersecurity	7
2.1.3	Machine Learning for IDS	8
2.2	Related Work on AI/ML-based Cybersecurity	9
2.2.1	General Attack Detection	10
2.2.2	DoS / Botnet Detection	11
2.3	Discussion	12
3	ML-Based System for Anomaly Detection	15
3.1	Workflow of the Anomaly Detection Models	15
3.2	One- vs. Two-Stage Classification Approaches	16
3.3	Supervised Steps for Performance Improvement	17
3.4	Implemented Pipeline for Anomaly Detection	18
3.4.1	Feature Engineering	18
3.4.2	Data Preparation	19
3.4.3	Model Training and Validation	19
3.4.4	Inference	19
3.5	Selected ML Methods and Prediction Merging	20
4	Datasets and Feature Selection for Network Anomaly Detection	21
4.1	Description of the Relevant Datasets	21
4.2	Datasets Preparation	22
4.3	Flow-Based and Temporal-Related Feature Selection	24
4.4	TCP vs. UDP Traffic Comparison and Selection	25
4.5	Packet Accelerator Simulation	27
5	Results and Discussion	29
5.1	Anomaly Detection for Traffic Classification	29
5.2	Anomaly Detection vs. Supervised Learning Comparison	31
5.3	One- vs. Two-Stage Learning Comparison	33

5.4	Packet Acceleration Impact on Models	35
5.5	UDP Traffic Inference	35
6	Discussion and Conclusion	37
6.1	Conclusions	37
6.2	Future Work	37
	References	39

List of Figures

1.1	ML architecture to train and validate the anomaly detectors	2
3.1	ML workflow to train and validate the anomaly detectors	16
3.2	One vs. two stage learning approach	17
3.3	The degree to each we provide information to the models	18
3.4	Developed pipeline for the proposed IDS	18
4.1	Boxplot of the main <i>Tstat</i> generated features for both classes	24
4.2	Boxplot of the newly generated features for both classes	27
4.3	Effect of PA simulation on data distribution	28
5.1	Performance comparison between default and optimized parameters models . . .	30
5.2	Performance comparison between the anomaly detector and supervised models .	31
5.3	Performance evaluation of one-stage and two-stage inference	33
5.4	Performance differences between the majority or selective voting methods	34
5.5	Performance differences when inferring on traffic with or without packet acceleration	35
5.6	Performance evaluation for UDP traffic, between the best performing AD model and the supervised models	36

List of Tables

2.1	Confusion Matrix	8
2.2	Summary of recent related studies	13
4.1	Chosen datasets and types of traces present in them	22
4.2	Benign traces used for training and validating the models	23
4.3	Malign traces used for training and validating the models	23
4.4	Extra features generated from the ones provided by Tstat	25
4.5	<i>Tstat</i> features for TCP flows	26
4.6	<i>Tstat</i> features for UDP "flows"	26
4.7	Differences in protocol distribution on the datasets	28
5.1	Performance evaluation results with default parameters on benign data (T-Training, V-Validation)	30
5.2	Performance evaluation results with default parameters on malignant data	31
5.3	Performance evaluation results with optimized parameters (T-Training, V-Validation)	32
5.4	Performance evaluation results of supervised models, for comparison (T-Training, V-Validation)	32
5.5	Performance evaluation results of the two-stage approach (T-Training, V-Validation)	34
5.6	Performance evaluation results for UDP traffic (one-stage AD) (T-Training, V-Validation)	36

Abreviaturas e Símbolos

AD	Anomaly Detection
C&C, C2	Command and Control
CPE	Customer Premises Equipment
DDoS	Distributed Denial-of-Service
DL	Deep Learning
DoS	Denial-of-Service
EC	Elliptic Curve
IDS	Intrusion Detection System
iForest	Isolation Forest
IoT	Internet of Things
ISP	Internet Service Provider
LOF	Local Outlier Factor
ML	Machine Learning
FN	False Negative
FP	False Positive
FNR	False Negative Rate
FPR	False Positive Rate
KNN	k-Nearest Neighbors
MITM	Man-in-the-Middle
NB	Naive Bayes
OC-SVM	One-Class Support Vector Machines
PA	Packet Accelerator
R2L	Remote-to-Local
RF	Random Forest
SGD	Stochastic Gradient Descent
SVM	Support Vector Machines
TN	True Negative
TP	True Positive
TNR	True Negative Rate
TPR	True Positive Rate
U2R	User-to-Root

Chapter 1

Introduction

This first chapter describes the motivation behind the topic of this work, the industrial context of this dissertation and its goals. It ends with an outline of the structure of this document.

1.1 Motivation

Smart devices are increasingly more present in home networks. However, these devices are often associated with flaws in design and implementation, leading to security vulnerabilities. Such vulnerabilities are being exploited by adversaries to generate a plethora of attacks that can affect the privacy of home users up to the operation of large companies.

One of the most common forms of attacks to smart networks is a botnet attack, with the most iconic case of this being the Mirai attack, where an adversary infects a set of devices in order to later command them to perform distributed denial-of-service (DDoS) attacks. This form of malware generates abnormal traffic that be distinguished and detected with the assistance of machine learning (ML) mechanisms.

Intrusion Detection Systems (IDSs) can employ ML techniques to learn from the typical behaviour of the network or from a given labeled dataset, to then analyse new traffic in the search for anomalous activity. These systems need to be adjusted to the different profiles of network connected devices and accurately and timely detect incoming attacks, while considering factors such as performance constrains, users' privacy and false positive rates.

However, the learning process can be an intensive task to Customer Premises Equipment (CPE), which is the target device of the IDS. This aspect adds more variables to be considered, namely which architecture do adopt, the volume of data to be used for the learning process and the inference strategies for flowing traffic.

The development and deployment of IDSs capable of detecting different classes of attacks through the use of ML techniques while paying attention to the *supra* mentioned concerns is thus of great relevance.

1.2 Context

The goal of this dissertation is the development of an IDS capable of learning the typical network behaviour and accurately detect attacks. It is part of an ongoing project with NOS[1], a Portuguese Internet Service Provider (ISP), who defines objectives and constraints, and provides some datasets for testing purposes.

The ISP infrastructure has different components, the two most relevant for the discussion being the cloud (servers and services established at the ISP network) and the CPE (typically the router and gateway devices of the client). At the cloud level, more computational resources are available, and all the customer traffic that circulates within the Internet is visible. However, internal/local traffic within the customer premises is not visible (e.g., two devices within the same network communicating directly with each other). The CPE has access to all this traffic, but the equipment is more limited in computational power.

Therefore, because of traffic data availability, the IDS should ideally operate at the CPE, performing the inference and acting accordingly, as illustrated in Figure 1.1. The training process can be outsourced to the cloud, provided that the training data is capture the same way the inference data is.

The main constraint associated with CPE inference is the existence of a Packet Acceleration (PA) mechanism. It analyses the first few packets of each flow in order to fetch routing information, and the rest of the packets are "accelerated" and no longer processed. This restricts the available information of each flow to that pertaining only to the first packets. This is taken into account when performing the system validation.

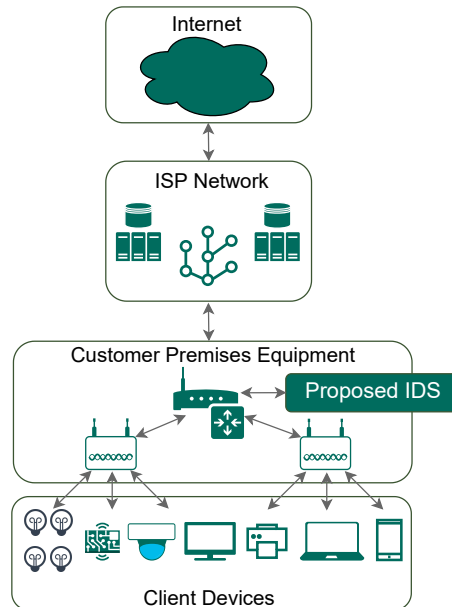


Figure 1.1: ML architecture to train and validate the anomaly detectors

This project and its outputs are framed in the scope of project "MIRAI - Machine Intelligence for smart and sustainable planning and operation of IoT and Edge", with project grant nr. 69522,

POCI-01-0247-FEDER-069522. MIRAI will develop the MIRAI Framework Building Blocks (MFBB) based on AI techniques in order to enable the smart and sustainable planning and operation of IoT and edge computing applications.

1.3 Objectives

The objective of this dissertation is the development of an IDS tailored for smart and home networks, with an emphasis on denial-of-service (DoS) or other forms of attacks that can disrupt the usability of these networks.

To achieve this, the main topics explored in this work and described here are:

- The selection of appropriate datasets of internet of things (IoT) and other normal traffic, as well as malign data to validate the proposed system
- The generation and selection of relevant network features for assisting in the classification task
- The design of the architecture of an ML-based detector that learns the typical behaviour of the network in which it's inserted and the implementation of its respective pipeline
- The exploration of changes to the system that incorporate some malign examples to improve the performance

1.4 Document structure

The structure of this document is divided as follows:

Chapter 2- Background and Related Work: Introduction of essential concepts related to the scope of the thesis (IDS, IoT Security and ML), as well as a summary of recent studies done on the subject of attack detection.

Chapter 3- ML-Based System for Anomaly Detection: Discussion on the architecture and pipeline of the proposed IDS, as well as a discussion regarding different learning approaches that can be considered.

Chapter 4- Datasets and Feature Selection for Network Anomaly Detection: Presentation of the selected datasets of benign and malign traffic, and a description of the feature engineering process.

Chapter 5- Results and Discussion: Presentation of the performance results for different scenarios and discussion of these results.

Chapter 6- Conclusion and Future Work: Conclusions of the work done and how it can be further improved.

Chapter 2

Background and Related Work

This section provides an introductory explanation on the concerns with home network devices, as well as the current approaches to develop systems to secure these devices. Then, an overview of previous works on this topic are described, and the main limitations of this body of work are outlined.

2.1 Background

2.1.1 Overview on Cybersecurity and the IoT Threat Model

With the increase of the number of devices connected to the Internet in various fields, the security of such devices becomes a critical necessity. Particularly, IoT devices often deal with sensitive information (such as customers' data) or are involved in vital operations (such as medical procedures), so none of these aspects can be jeopardized. As opposed to personal devices, the cybersecurity of IoT devices is often overlooked and taken for granted, while in practice they are subject to similar or tailored attacks.

IoT devices, even though not always valuable themselves to attackers, broaden the attack surface of networks and allow for the expansion of cybercrime activity, which is growing by the day. It was estimated by [2] that by the end of 2020 cybercrime derived losses were approaching \$1 trillion, growing 50% in just two years, however most companies had no plan to prevent and respond to these security incidents.

More and more adversaries are attempting to gain control or gather information from these devices through a variety of attacks that are typically considered passive or active depending on whether or not the adversary directly interferes with the the device and the goal of the adversary in performing such attack.

Passive attacks can come in the form of Eavesdropping, the interception of sensible data over a unsecured network, or traffic analysis, the monitoring of a network connection in order to obtain some knowledge or infer behaviours. Some of these attacks can be avoided with the use of secure and encrypted communication protocols, certificates, and other similar security mechanisms.

Active attacks are the most common and can be the most dangerous form of attacks that affect smart devices. They can be categorized as Probe attacks, Remote-to-Local (R2L), User-to-Root (U2R), and Denial-of-Service (DoS), as explained in [3]. They can also be of other types, such as Man-In-The-Middle (MITM), where the attacker poses as another user in order to attain sensitive information intended for that user, or Bruteforce, where the attacker continuously attempts to discover weak passwords to gain illegal access, among others.

A DoS attack occurs when an adversary floods a target server or host with an exaggerated number of requests and/or exploits certain protocols vulnerabilities, denying access to legitimate users. A Distributed DoS (DDoS) attack generates traffic from multiple sources, improving the scale of the attack and making it tougher to be blocked. These forms of attacks are often the main focus of anomaly detection systems due to their disruptive nature, which affects devices and services in various contexts. A recent report [4] showed that this threat continues to expand, with the number of attacks growing by 241% during 2019. Due to the vulnerabilities of typical IoT devices, they can be performed in two ways. In the first way, the attacker directly attempts to disrupt the service of specific devices by wasting their resources. The second way, the attacker infects numerous devices and then uses them to perform a large scale (DDoS) attack, typically against specific corporations. The set of infected devices which can later on be used to perform attacks is commonly known as a Botnet.

There have been multiple Botnet attacks in the past, with the most famous one arguably being the Mirai Botnet. It can be explained in two stages: first, a brute force technique for cracking passwords was deployed and the malware was injected in hundreds of thousands of devices. Then, once infected, a command and control (C2) server signals these devices to attack the specified target. With the release of the source code, many other variants, improved or not in some way, have appeared and continue to be deployed, making the issue of Botnet attacks very current. In another recent report [5] it was shown that during 2019 there was a 57% increase in the Mirai variants detected and a 71,5% increase in botnet C2 servers, and that botnet attacks alone were responsible for over 97% of all attacks.

All these attacks are facilitated by the fact that smart devices can have multiple associated vulnerabilities, such as:

- Outdated and unsupported Operating Systems
- Unencrypted traffic from/to IoT devices
- Poor network design choices
- Lack of protective software
- Unchanged credentials

In fact, recent reports on this topic that analysed data of networking and security companies show that these vulnerabilities are still largely present. The work in [6] analysed IoT data across the cloud of security company Zscaler (that deals with over 1 billion monthly IoT transactions) and

concluded that over 90% of transactions occur in plain text, only 18% of device communication relies exclusively on SSL for security, and thousands of malware attempts are blocked every month, of which most are botnet attacks. The authors of [7] analysed over a million IoT devices managed by Palo Alto Networks, and found that most devices use unsupported operating systems, 98% of all traffic was unencrypted, 57% of IoT devices are vulnerable to medium- or high-severity attacks, and most attacks are worm-like (self-propagating) and botnet (C2) attacks.

However, traffic encryption by itself doesn't avoid the exploitation of other vulnerabilities. Even when traffic is encrypted, observing IP traffic streams can be done to infer user behavior, posing a great user privacy issue, as shown in [8].

2.1.2 Intrusion Detection Systems as Agents of Cybersecurity

Since the computational power of smart devices is limited, a designated system for monitoring and protecting these devices in the context of a network is a common tool for improving cybersecurity in this context.

This system usually comes in the form of an IDS that monitors the state of the network, identifies possible malicious behaviour and acts to mitigate any damage. As it pertains to network monitoring, these systems can be deployed in one or more devices belonging to the network of interest. In case of a network manager by an ISP, such devices can be the Customer Premises Equipment (CPE; typically the home router/gateway), or a server connected to the ISP core network.

IDS solutions can be classified accordingly to their method of detection, typically one of the following:

- Signature-based, also known as misuse-based, typically more present in legacy systems, that compare the collected traffic to well-known signatures of attacks, which rely on a defined list of Indicators of Compromise (IOCs). The biggest flaw of this approach is the difficulty in recognizing unknown attacks, since their signature is not yet established.
- Anomaly-based, systems that learn from the typical behavior of a network and detect when a traffic pattern is not recognized. This solution calls for machine learning models that can statistically infer if a new traffic pattern is normal or anomalous.

IDS can also be classified accordingly to the target of analysis, being one of the following:

- Host-based (HIDS), which is implemented, as the name implies, on each host, typically operating at a system level, managing resources and events, which then communicate with a central managing module.
- Network-based (NIDS), which alternatively operates at the network level, analysing traffic between hosts and detecting malicious activity accordingly to a set list of rules or model.

2.1.3 Machine Learning for IDS

The nature of a malicious behaviour detection system indicates that this is a binary classification problem, since the intent is to classify each new access as normal or anomalous. In order to solve such classification problems, supervised learning is typically employed. This learning method consists in the use of labelled data to train ML models that are used to accurately classify data. The alternative, unsupervised data, is the use of unlabeled data to detect patterns and association rules in order to extract information from that data.

Machine learning can be used to solve classification or regression problems. Classification is the act of assigning new and unseen data to defined categories, binary or not, whereas regression is used to find relations between variables, dependent and independent, and estimate a value for one of those variables.

Supervised algorithms can be, but not limited to, Random Forest (RF), Decision Trees (DT), Naive Bayes (NB), Support Vector Machines (SVM), k-Nearest Neighbors (KNN) and Linear/Logistic Regression.

Neural networks, also referred to as Artificial Neural Networks (ANN), are a subset of ML that aims at mimicking the behavior of the brain and interactions between neurons for the learning process. A neural network with more than three layers can also be considered a deep neural network, which forms the basis of deep learning (DL). DL makes use of neural network algorithms such as convolutional neural networks (CNNs), recurrent neural network (RNNs), long short-term memory (LSTM) and Autoencoders.

Some algorithms are more prone to being applied in a distributed architecture than others. For example, neural networks aggregate its neurons in layers, therefore the creation of a centralized, generic model could be done in the first layers, and the specialized model for each purpose could be trained during the last layers, resulting in a specialized, quality model that only required a portion of the training on the user end.

When it comes to validating the performance of the trained models, results are usually initially displayed in a confusion matrix. The following confusion matrix enables describing the performance of a supervised algorithm with results from test data with known values.

Actual \ Predicted	Normal	Malicious
	True Negative	False Positive
Normal	True Negative	False Positive
Malicious	False Negative	True Positive

Table 2.1: Confusion Matrix

Considering the positive class (class of interest) as the malicious access, we can define the following:

- **True positive (TP):** correctly predicted a malicious access
- **True negative (TN):** correctly predicted a normal (non-malicious) access

- **False positive (FP)**: incorrectly predicted a normal access as being malicious
- **False negative (FN)**: incorrectly predicted a malicious access as being normal

In order to measure the performance of the employed model(s), common metrics are, but not limited to:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN} \quad F1 = \frac{2 \times (Precision + recall)}{Precision + Recall}$$

Different metrics are used for different purposes and needs, as well as other factors, such as dataset balance. For instance, if the cost of not detecting a malicious access is much greater than the cost of mistakenly classify a normal access as malicious, then a metric that pays attention to the positive class (malicious access) like Recall is preferable.

2.2 Related Work on AI/ML-based Cybersecurity

Since the types of attacks in the past were fewer and patterns of attacks didn't change as often, research used to focus more on signature-based development of IDS. However, with the rise of new technologies associated to the paradigms of IoT, the attack surface becomes larger, leading to many unknown attacks. Since these attacks are not easily matched to known patterns, IDS research has been moving in the direction of anomaly-based detection. Another driver of this interest is the increase of zero-days attacks and fast mutations of existing malware, which can render the signature-based approach useless.

Benkhelifa et al. [9] reviewed the existing solutions of IDS as it pertains to IoT environments. The authors first described most common protocols and standards present in IoT systems, then defined the threat model for such devices. Afterwards, a review of the literature was made, classifying the surveyed IDS by the type of architecture, focused technology and type of detection. The authors proposed a system with external processing in order to mitigate performance issues, with multiple detection methods to cover attacks of different natures, and adapted to deal with multiple types of technologies.

Elrawy et al. [10] also carried on a survey on IoT smart environments and discussed a similar approach. The authors made an introduction to security on these networks and common detection techniques. Then, they classified the IDS by the technique used and made a descriptive statistical analysis of the results, taking into account different performance metrics.

Chaabouni et al. [11] performed a thorough comparison between commonly used tools in many studies to aid in the development of IDS or existing solutions: free datasets, free and open-source (FOSS) network sniffers and open-source NIDS. The authors also compared between NIDS

implementations for IoT environments and similar implementations but based on learning techniques.

2.2.1 General Attack Detection

Many authors decide to validate their approach by testing their models against a wide variety of attacks, that usually fall under one of four categories: DoS, U2R, R2L and Probing attacks [3]. This decision might be motivated by the fact that these are representative classes of attacks that anomaly-based systems should be able to detect, or that these are the common types of attack samples present in public datasets.

Anthi et al. [12] proposed an anomaly-based IDS capable of profiling the present devices, detect incoming wireless attacks and identifying the type of attack. They also compared the results from 9 different classifiers, and concluded that all-around the J48 classifier, a decision tree based algorithm, provided the best results.

Hasan et al. [13] resorted to a public dataset to develop an anomaly detection system that identifies attacks out of 7 different classes. Authors also evaluated the model using several machine learning techniques, concluding that Random Forest (RF) provides the best results for protecting IoT devices from cyberattacks. Doshi et al. [14] also came to the same conclusion, by developing an anomaly detection system focused on DoS attack detection. They used data from commercial available devices and compared 5 different algorithms, of which RF performed the best.

Nguyen et al. [15] developed D²IoT, a self-learning and distributed IDS focused on IoT networks. Through a novel way of using a language analysis technique to detect anomalies, using both a Mirai attack dataset and personal data the system was proven to quickly and accurately detect attacks, with a special emphasis on DoS attacks. This system also implements a federated learning approach: clients locally train models based on their IoT network traffic, then local gateways and the centralized entity share updates in order to improve global performance of the models.

Eskandari et al. [16] implemented Passban, an anomaly-based IDS capable of running on a typical simple IoT gateway or as a separate device. Resorting to an Isolation Forest(iForest) algorithm, it's able to accurately detect common IoT attacks while having a low impact on system resources.

Diro and Chilamkurti [17] presented a deep learning based distributed approach to detect attacks, with an emphasis on the social IoT paradigm. It employs an architecture of edge learning in a federated manner to improve latency and processing requirements. The authors then resort to a public dataset with attack data of the four common categories to compare the performance of the deep model to a shallow model (i.e. common model that requires manual feature engineering), concluding it is a better approach for this scenario.

2.2.2 DoS / Botnet Detection

Since the DoS and Distributed DoS (DDoS) attacks are one of the fastest growing concerns from the perspective of IoT networks, mainly from the point of infecting devices in networks to later attack bigger targets, much of the recent research on IDS has been done under the scope of detecting these particular attacks.

Nömm and Bahşi [18] developed an anomaly-based detection system with an emphasis on dimensional reduction, which lowers the complexity of the models. They also compare a common model to device-specific models and evaluate these differences, while applying the iForest and SVM algorithms, and observed that for some devices, a common model had considerably lower detection performance, which can be explained by the complexity and uniqueness inherent to some devices.

Lima Filho et al. [19] developed Smart Detection, a machine learning detection system focused on DoS/DDoS detection. The authors used both a dataset of their own and three other public datasets to validate their Random Forest based model, and the results showed that the system is capable of accurately detecting different types of DoS attacks, such as TCP/UDP/HTTP floods.

Research has also been made in the perspective of detecting a botnet infection as early as possible, such as the infamous Mirai botnet or newer variations.

Kumar and Lim [20] developed EDIMA (Early Detection of IoT Malware Network Activity), a botnet detection system aimed at identifying malware activity during the scanning and infecting phases. An own dataset was used with scripted activity simulating the characteristics of the original Mirai code during the initial phases and this data was fed to models based on Random Forest, k-NN and Naive Bayes, obtaining good results on the scoring metrics.

Meidan et al. [21] also developed an anomaly-based IDS for botnet detection, this time using deep learning techniques. For each IoT device, an autoencoder was trained in order to capture its normal traffic behaviour. To evaluate the detection method, Mirai and BASHLITE botnets were deployed, and every attack performed was successfully detected, with a low to null false positive rate.

McDermott et al. [22] also deployed a Deep Learning approach to detect botnet activity, based on Bidirectional Long Short Term Memory based Recurrent Neural Networks (BLSTM-RNN). The authors produced an own dataset with Mirai attack samples, and the data was pre-processed with the use of Word Embedding. BLSTM was compared to an LSTM-RNN approach, and the results showed high accuracy for botnet and other related attacks detection.

Yuan et al. [23] developed DeepDefense, another Deep Learning detection system with an emphasis on DDoS. By the use of various types of Recurrent Neural Networks (RNN), such as LSTM, the authors achieved a reduction of the error rate from 7.517% to 2.103%, comparing to conventional ML models, specifically Random Forest.

2.3 Discussion

A summary of the work presented can be found in table 2.2. One of the common challenges is to create a model that is capable of encompassing the behaviour of all smart devices present in the network, meaning having a high detection rate while not raising too many false positives, which can affect the usability in a real scenario.

The data used to validate these models is often outdated, i.e., doesn't represent traffic generated by modern devices in home networks, or doesn't account for all the expected types of traffic in these environments.

The attacks simulated for generating malign datasets aimed at validating IDSs for home networks are usually basic, representing the simple components of known attacks such as a botnet infection. Current attacks are often more complex, making use of newly found vulnerabilities and detection avoidance mechanisms, for instance.

Another difficulty comes in the implementation of a system that accounts for the limited resources of the CPE. This is often not taken into account, as models take long periods of time to be trained, during which the network operations can be impaired, which is not acceptable. When distributed approaches take this into consideration they're usually implemented in a simple manner that is not easily scalable to larger networks.

Reference	Technique(s)	Data Used	Threat Focus	Distributed Architecture
Anthi et al. [12]	Various Weka classifiers, J48 best results	Own	Scanning, DoS, MITM and iot-toolkit	-
Hasan et al. [13]	LR, SVM, DT, RF and ANN	DS2OS	DoS, Probing, Malicious Control and Operation, Scan, Spying and Wrong Setup	-
Doshi et al. [14]	k-NN, Linear Kernel SVM (LSVM), DT, RF, ANN	Own	DoS	-
Nguyen et al. [15]	Gated Recurrent Unit (GRU)	Own	DoS	Federated Learning
Eskandari et al. [16]	iForest	Own	Port Scanning, HTTP/SSH Brute Force and SYN Flood	-
Diro and Chilamkurti [17]	Deep Learning	NSL-KDD	Probe, R2L, U2R and DoS	Distributed Learning
Nömm and Bahşı [18]	iForest and SVM	Own	Mirai and BASH-LITE traffic	-
Lima Filho et al. [19]	RF	Own and CIC-DoS, CICIDS2017, and CSE-CIC-IDS2018 Datasets	DoS	-
Kumar and Lim [20]	RF, k-NN and Gaussian Naive Bayes	Own	DoS / Botnet infection	-
Meidan et al. [21]	Deep Learning (Autoencoders)	Own	DoS / Botnet infection	-
McDermott et al. [22]	Deep Learning (BLSTM-RNN)	Own	DoS / Botnet infection and similar attacks	-
Yuan et al. [23]	Deep Learning (CNN, RNN, LSTM and GRU)	UNB ISCX Intrusion Detection Evaluation 2012	DDoS	-

Table 2.2: Summary of recent related studies

Chapter 3

ML-Based System for Anomaly Detection

This chapter serves as a presentation and discussion of the proposed ML system. In section 3.1 we describe the workflow of the ML-based Anomaly Detection (AD) methods and define the difference between IoT and normal traffic. Section 3.2 shows a further possible adjustment to the learning process to account for different traffic types. Additionally, in Section 3.3 we discuss the inclusion of traffic from both classes to improve performance. In Section 3.5 we list the chosen algorithms and how they'll take part of the system.

3.1 Workflow of the Anomaly Detection Models

Attack patterns affecting devices in home environments can be quite varied, so collecting samples representative of every one of these attacks for training models is impractical. Also, the statistical attributes of these samples depend on the traffic conditions of the network in which it was captured. Furthermore, the generation of these samples could impose security issues and require additional data transfer and storage in devices that are already limited in resources.

A solution for these constraints is to train models with only benign data, so it learns the typical traffic behaviour of the network. Any traffic that statistically differs considerably from the typical behaviour is considered anomalous, and thus presumed to be, with high likelihood, malicious traffic. This can allow for a model that does not require malign data at the training to still be capable of detecting it.

The model should also take into account the different possible types of benign traffic expected, with the main two types being the IoT and normal traffic. They should be well represented throughout the learning process if present in the network, as to ensure correct representativity of those types in the given network.

By IoT traffic, we refer to traffic generated by devices that are often part of an automation system, that collect data and inform the consumer or perform simple home tasks, for instance. Some examples of these devices are IP security cameras, smart bulbs and door sensors. The

traffic volume generated by these devices is typically low (with exceptions, such as camera video streaming), the connections are periodic and relatively repetitive.

Normal traffic is usually generated directly by the user on his personal devices, such as laptops and smartphones. Examples of this traffic include web browsing, gaming and video conferencing. Normal traffic is different than IoT since it's typically more bandwidth intensive with less repetitive flows. The protocols used in communication are typically more up-to-date, and encrypted traffic is more common.

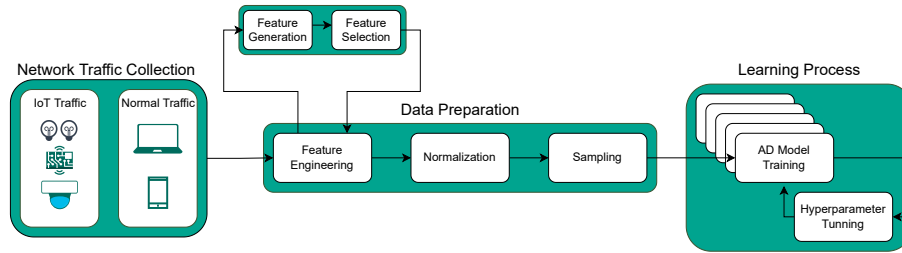


Figure 3.1: ML workflow to train and validate the anomaly detectors

Figure 3.1 shows the considered workflow for training models for network anomaly detection. Firstly, network traffic has to be collected. Different network analysis tools can be used for the purpose, but in our case the data is provided mainly by publicly available datasets, described in Chapter 4. Afterwards, the data goes through a process of feature creation and manipulation to facilitate the learning process. The data has to be processed in different forms, and is then fed to the learning portion of the system, which may or may not contain an optimization process.

3.2 One- vs. Two-Stage Classification Approaches

As mentioned in the previous section, traffic generating devices connected to customer networks can be for the most part divided in two categories: IoT devices and normal devices.

The differences in nature of these types of traffic might justify the use of individual models for each. Since the ML approach of this work is outlier detection, fitting an outlier detector for each type of traffic might provide a level of specificity that a single generic model doesn't. Figure 3.2 shows the difference between the two approaches at the inference level.

The output of an anomaly detector can be an outlier, if the sample differs considerably from the training data, or inlier, otherwise. In the case of the one-stage detector, outliers are classified as malign and inliers as benign, so we refer to this classifier as a benign traffic classifier.

For the two-stage detector, the traffic is run through a model trained with IoT traffic, and if classified as an outlier, it is then run through a model trained with normal traffic. If the sample is an outlier to both models, it is then classified as malicious. Naturally, it adds some overhead to both the training and inference processes, which might be the cost to improve the inference performance.

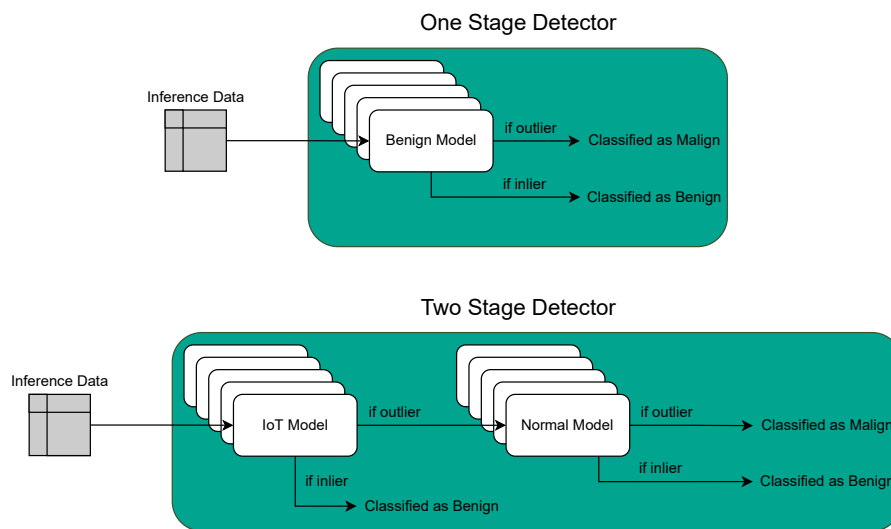


Figure 3.2: One vs. two stage learning approach

3.3 Supervised Steps for Performance Improvement

Not all ML models are trained with the same type or amount of data. Some methods require labeled datasets in the learning process with samples from each target class, which are classified as supervised models [24]. Unsupervised models don't need any prior information of classes in order to provide statistical information about the data, as is the case of clustering methods [25].

The degree to which ML models use information about the data in classification or knowledge extraction tasks can fall within a spectrum, with the typical extremes being unsupervised and supervised learning methods.

However, there are many other methods that fall somewhere along this spectrum, where taxonomy is not always even well defined. We identify methods that learn from the samples of only one class as anomaly or outlier detectors, since they're neither unsupervised or supervised methods. Anomaly detectors require previous knowledge of the data (it assumes all/most samples come from one class) in order to classify new samples as inliers (within the training class) or outliers (falling outside of that class), but don't require any knowledge about the other class.

Still, there are certain processes along the pipeline that can be designed to improve its performance, but require some knowledge of the outlier class. This processes from a conceptual perspective will shift the degree of information to the supervised end, since we are now using labeled data from all classes.

In our case, we opt to perform hyperparameter tuning (where samples from both classes are tested to determine a certain choice of parameters' performance and find the optimal set) or feature selection (where statistical analysis determines the features that are more relevant for the classification task). While the results of these actions can be carried over to new scenarios without any re-processing, they still increase the data demands in an initial phase.

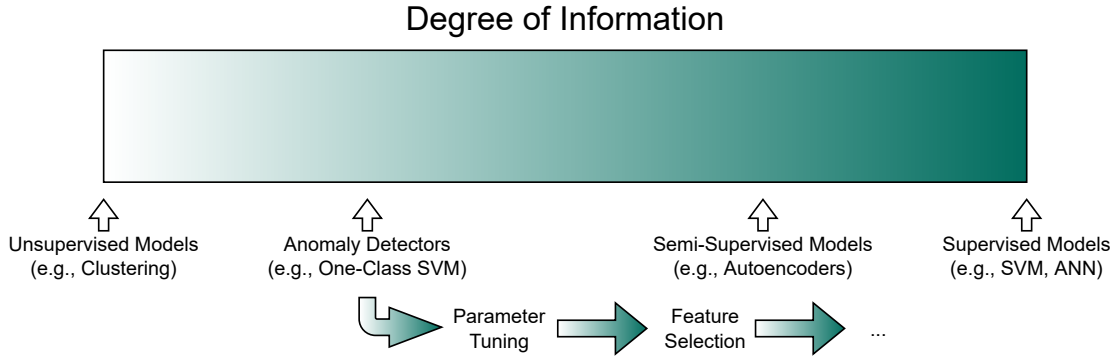


Figure 3.3: The degree to each we provide information to the models

Our proposed architecture begins as a simple set anomaly detection methods, but then adds some of these steps to study their impact on the performance of the models for this context.

3.4 Implemented Pipeline for Anomaly Detection

This section describe how various components of the system were implemented, which are illustrated in Figure 3.4

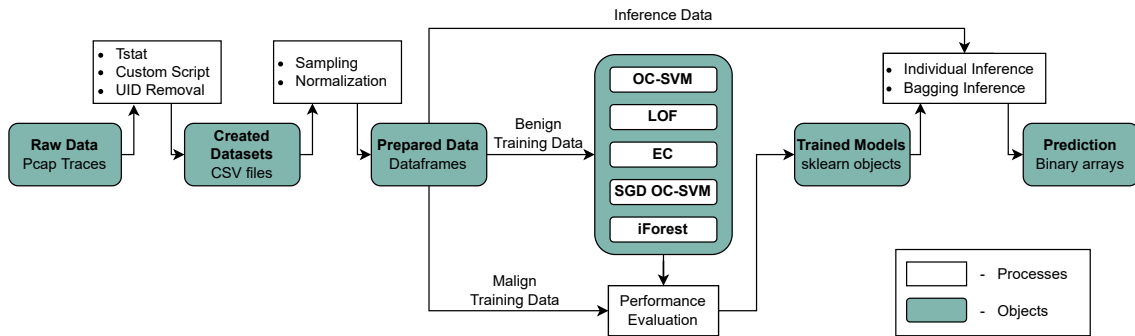


Figure 3.4: Developed pipeline for the proposed IDS

3.4.1 Feature Engineering

As previously mentioned, *Tstat* was chosen as the main tool for generating network flow features. It either takes packets from a live capture or a file trace, the latter being more relevant for our case. We executed *Tstat* for all the selected traces, which outputs the information regarding the networks in a comma-separated values (CSV) format.

After the *Tstat* feature generation, the data goes through a process of creation of additional features, all of which described in section 4.3.

Afterwards, features that have no statistical relevance to models are removed, such as identifiers (e.g., IP addresses), timestamps and highly-contextual characteristics (e.g., if addresses are anonymized and internal or not).

An additional process of feature selection can be performed based on both classes. For that purpose, we can utilize the validation portion of the data to assess feature correlation to the target variable, and choose the most correlated features. This process adds an extra layer of supervision to the system, which can be then carried over to new scenarios.

3.4.2 Data Preparation

In order to fulfill the requirements of some of the used models, the data has to be normalized before applied to the models. A traditional method of Min-Max normalization is performed, rescaling the training data to $[0, 1]$. The rest of the data (validation and testing data) is rescaled using the same reference values as the training set.

To take into account the limited computation power of the CPE, the models are trained with a small set of data. For that, each training and each validation set is sampled to $n = 2000$ samples, in order to take into account the limited computational resources of the target device of this system. Testing data is not sampled, as most of the computational demands of the ML approach derive from the training process, and by doing so we preserve the representativity of the results for all the data.

3.4.3 Model Training and Validation

With the prepared benign data, the multiple anomaly detection models are trained. These models receive only benign (or negative class) data and attempt to fit boundaries around these data points. These models are described in detail in section 3.5.

Each model has a set of user-defined parameters (i.e., hyperparameters) that can be changed as part of a validation process. However, trying to select the best hyperparameters based on a scoring metric (e.g., accuracy) for just the benign data might lead to large or underfit boundaries that correctly classify benign samples, which maximizes performance, but then incorrectly classify malicious samples as benign leading to high false negative rates.

To compensate for this factor, we used samples of both classes (benign and malign) when calculating the accuracy for the validation data. The models are still trained with only benign data, but this validation informed by the two classes allow for the selection of a good set of hyperparameters that can be later on used on new models without the need for any malign samples.

3.4.4 Inference

After training the models, we then get a set of detectors that are prepared to infer incoming data. These models are independent from one another, so a decision method has to be decided upon.

Test data samples are parsed through these models, and the predictions from each are combined into a global prediction. The method to perform this combination can be a regular majority vote, where in this case of 5 (odd number) models the decision is the class with the highest number of votes. This decision results in a classifier that can then be evaluated similarly to the others, by accuracy or false/true negative/positive rates.

3.5 Selected ML Methods and Prediction Merging

As explained earlier, the emphasis is on models that do not require any positive class samples at the training stage. They are commonly referred to as anomaly detection (AD), outlier detection, novelty detection or one-class models. They are also considered to be a special case of unsupervised learning, since no labels are provided, yet assumes prior knowledge of it. So, their main purpose is to classify new samples as belonging to the training class or not. Some of the most frequently used, that are also those that are currently implemented, are the following:

- **One-Class Support Vector Machines (OC-SVM):** Support Vector Machines is a supervised ML algorithm commonly used for classification in high-dimensional feature spaces. It defines a boundary that maximizes distance to the closest data points of each class.

OC-SVM is an adaptation of this model for novelty detection where instead of a hyperplane, it fits the smallest possible hypersphere to the data. Points that fall outside of this space are considered outliers.

OC-SVM with Stochastic Gradient Descent (SGD OC-SVM) solves the linear OC-SVM using SGD, which has the advantage of efficiency due to the linear scale with the data.

- **Local Outlier Factor (LOF):** Based on the local neighborhood, the model attempts to identify outliers in the data. It does so by comparing an estimated local density to that of the k closest neighbors and deciding upon that estimation.
- **Elliptic Curve (EC):** This model assumes a normally distributed data, and similarly to the others, is used for identifying outliers. It fits an ellipse to the gaussian distributed data and considers points that fall outside of it anomalous.
- **Isolation Forest (iForest):** This algorithm builds an ensemble of trees that separate the data points by recursively generating partitions for the attributes until they're isolated. An outlier should take less iterations until isolated compared to a regular point, therefore a decision threshold is used to separate this two classes.

The output of these AD models can be then combined in order to achieve a possibly better overall model, which is commonly referred to as Bagging. After independently training the models and at the inference level, the predictions of each model can be combined by a given method, such as the majority vote, in order to obtain a more precise estimate. If certain models perform better on some datasets than others, this method can account for that variability and provide a more reliable prediction.

All the model implementations are provided by the scikit-learn library [26][27], a scientific open source, complete and convenient toolbox of anything machine learning related. It provides many implementations of models for supervised and unsupervised learning, as well as anything in between and special cases of each.

Chapter 4

Datasets and Feature Selection for Network Anomaly Detection

The previous section outlined the proposed architecture for achieving anomaly detection in the context of home networks, as well as the implementation details. Here, we discuss the selected datasets to assist in the learning process of the detectors and to assess their ability in classifying benign and malign traffic. In section 4.2 we detail which portions of data are used for each stage of the whole process, and the features created for the data are explained in section 4.3. Section 4.4 gives a remark on the difference between TCP and UDP traffic distributions on the datasets and its impact on the goals of this work. Finally, section 4.5 provides insight on the approach taken to simulate a packet acceleration technology, relevant to CPE inference.

4.1 Description of the Relevant Datasets

Common datasets used for evaluating IDSs, such as KDD CUP 99 and NSL-KDD, are generally not representative of the traffic generated by modern devices in home networks. IoT samples are scarce when present and normal samples often do not use modern protocols or match what a typical user might perform on the Internet. This has been highlighted in chapter 2.

Furthermore, these datasets often come in the form of preprocessed network flows with pre-determined features, which limits the ability to understand the dataset in detail at the packet level, manipulate it and generate specific features. When packet captures are provided, benign and malign traffic is typically not separated, which is a task done by a network analysis tool. This may limit the interoperability of different tools for a given dataset, requiring extra efforts to match the output of each tool for labeling purposes.

With that in mind, the datasets used for this detection system were the **UNSW-IoT**[28], **Bot-IoT** [29, 30, 31, 32, 33, 34], **IoT-23**[35] and **CTU-Normal**[36], as well as some datasets provided by **NOS**[1]. The combination of these provides sets of IoT and normal traffic samples from different scenarios, as well as samples from different types of attacks that commonly affect the devices in question. Table 4.1 outlines the types of samples present in each of these datasets.

Datasets	IoT Traces	Normal Traces	Attack Traces
UNSW-IoT	✓	X	✓
Bot-IoT	✓	X	✓
IoT-23	✓	X	✓
CTU-Normal	X	✓	X
NOS provided	✓	✓	X

Table 4.1: Chosen datasets and types of traces present in them

UNSW-IoT provides captures of both benign traffic generated by real devices as well as malicious traffic of attacks launched against those devices. A list of some devices of this dataset, as well as an indication of whether that data was used for training or testing the models, can be seen on table 4.2. Since these captures include traces with only benign data of many devices, easily separated by device by their IP addresses, they're particularly useful for generating the main benign IoT datasets for training and testing the models.

Bot-IoT consists of a large collection of both benign and malicious traffic. The dataset includes traces of multiple types of DoS/DDoS attacks, scans and data theft, designed in an attempt to simulate the expected traffic of conventional botnets. The raw data in the form of pcap files is conveniently separated by attack type, which facilitates the ML testing process for the malicious data.

IoT-23 is a dataset that combines benign IoT captures from three devices (Amazon Echo, Philips Hue and Somfy door lock) and malicious botnet captures that simulate the expected network traffic generated by devices infected with different variants of botnets. This dataset, while not diverse in benign data, provides a good amount of malicious data captured in a different environment to test the ML models.

CTU-Normal includes various captures of normal traffic, predominantly in the form of web browsing. These traces include mostly HTTPS traffic of access and interaction with the top 500 websites [37] in an automated fashion, as well as manual access to some of these, which adds variety to the captures.

NOS also provided a few datasets as mentioned in section 1.2. These include captures of IoT and normal traffic under different conditions which are used to further validate the models.

4.2 Datasets Preparation

Since the selected anomaly detection models require only benign data to be trained, the first step of the process is selecting the appropriate datasets for each relevant instance of the learning process.

The **UNSW-IoT** dataset was separated in two testbeds that contain devices of similar types, as seen in table 4.2. The first set (**UNSW-IoT #1**) was used for selecting the best hyperparameters and features for a portion of the data, as described in 3.1, and trained with a different portion

Dataset	Training	Validation	Testing	Capture details
UNSW-IoT #1	✓	✓	✓	Amazon Echo, Netatmo camera, Lixf light, WEMO power switch
UNSW-IoT #2	X	X	✓	iHome, Samsung camera, Hue bulb, TP Link plug
IoT-23	X	X	✓	Amazon Echo, Hue bulb, Somfy door lock
NOS - IoT	X	X	✓	Smart plug, temperature and humidity sensors
CTU-Normal #1	✓	✓	✓	Automated capture, top 500 websites: 1-30, 61-120, 241-360
CTU-Normal #2	X	X	✓	Automated capture, top 500 websites: 31-60, 150-240, 361-500
				Manual capture of web traffic
NOS -Normal	X	X	✓	Manual capture of web traffic

Table 4.2: Benign traces used for training and validating the models

afterwards. Finally, unseen data from that set is also used for inference. The second set (**UNSW-IoT #2**) is used for inference, as to test the possibility of the model to recognise new traffic of similar type correctly, captured under similar network condition.

We took a similar approach for the normal traffic, with the the first set (**CTU-Normal #1**) being used for training and validation and the second (**CTU-Normal #2**) for testing the model performance for new benign traffic.

The rest of the benign samples for the other datasets (**IoT-23** and **NOS**) was used solely for testing the models.

As for the malign data chosen, as the models are not trained with this class, they're mainly used for testing the models and tune some parameters of the detectors, as shown in Table 4.3.

Samples from the **BoT-IoT** TCP DDoS and service scan captures were chosen for the hyperparameter tuning process. The rest of the samples of performed attacks are used for assessing the detectors performance and testing if the tuning process carries over to other forms of attacks.

Finally, the malign samples from the **IoT-23** dataset are used for testing the models. These are captured under different conditions than the previous ones, so we can assess whether attacks samples have carryover from one network to another regarding the impact on the models.

Dataset	Training	Validation	Testing	Capture details
Bot-IoT - TCP DDoS	X	✓	✓	TCP floods generated using hping3
Bot-IoT - HTTP DDoS	X	X	✓	HTTP request floods using Golden-eye
Bot-IoT - OS Scan	X	X	✓	OS fingerprinting using Nmap and Xprobe2
Bot-IoT - Service Scan	X	✓	✓	Different port scans using Nmap and Hping3
Bot-IoT - Keylogging	X	X	✓	Record keystrokes with Logkeys or exploits
Bot-IoT - Data Theft	X	X	✓	Directory exfiltration through system exploits
IoT-23 - DDoS	X	X	✓	DDoS attack part of a botnet
IoT-23 - Port Scan	X	X	✓	Port scans part of a botnet
IoT-23 - C&C	X	X	✓	C&C communication part of a botnet

Table 4.3: Malign traces used for training and validating the models

4.3 Flow-Based and Temporal-Related Feature Selection

Many available network datasets already include a file with features generated for each network flow. Commonly used tools for this purpose are, for instance, *Zeek* (used in **IoT-23** and **CTU-Normal**) and *Argus* (used in **UNSW-IoT** and **Bot-IoT**). These tools generate features such as IP addresses and ports, timestamp, protocol, bytes and packets. While some of these features can be important for the classification problem, they're also limited and not protocol specific.

And, while some datasets have the original traces separated by category of traffic, others don't. In that case, either the traffic is separated by device using a tool such as *Wireshark* or by a tool that matches *Zeek* flows to *Tstat* flows. We developed a script for that purpose that maps each *Tstat* flow to a *Zeek* to take advantage of the latter's label. This tool encodes uniquely identifying features (IP addresses, ports and timestamps) on both logs, iteratively searches for the matching flows and attributes the *Zeek* label to the *Tstat* flow.

Therefore, for the purpose of feature generation from the raw data (in packets), *Tstat* provides more information specific to each flow. Apart from the same basic features, it also includes protocol-specific ones whenever possible, as seen in tables 4.5.

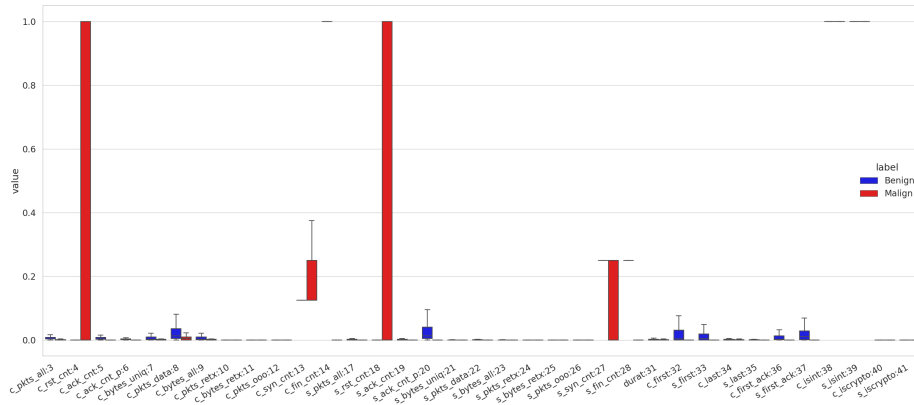


Figure 4.1: Boxplot of the main *Tstat* generated features for both classes

Figure 4.1 shows a boxplot of the numerical normalized features for benign and malign validation sets (see Tables 4.2 and 4.3). Analysing that plot, we can notice that some features have different distributions for each class, which can improve the classifiers' performance.

The features generated by *Tstat* are important to extract information of each network flow. However, observing specific flows does not directly give much temporal context (e.g., network throughput) or device context (e.g., device throughput), and it doesn't provide much information at a packet level either.

Because of that, we generated extra features based on the original *Tstat* ones for a sliding windows of 100 connections (arbitrary and adjustable value that seems appropriate for this context), which are present in Table 4.4. These features provide information about the state of the network otherwise not present.

Feature	Details
sw_fRate_IPSrc	Rate of flows with same origin IP
sw_bRate_IPSrc	Rate of bytes with same origin IP
sw_pRate_IPSrc	Rate of packets with same origin IP
sw_fRate_IpPortDst	Rate of flows with same destination IP and Port
sw_bRate_IpPortDst	Rate of bytes with same destination IP and Port
sw_pRate_IpPortDst	Rate of packets with same destination IP and Port
sw_isComplete_percentage	Percentage of flows that are complete

Table 4.4: Extra features generated from the ones provided by Tstat

The rate of flows/bytes/packets are calculated by the number of flows/bytes/packets that occurred in that window, divided by the elapsed time of that window, respectively. These rates are calculated by original IP address (thus specifying the origin device) and by destination IP address and port (thus specifying the destination device/service). Lastly, the percentage of flow that are classified as complete is calculated. Complete flows are properly initiated (by a three-way handshake) and closed (by a FIN or RST flag, or after a certain timeout).

Figure 4.2 shows the normalized distribution of these newly created features. Simply put, these rate features have larger values for the malign class, as expected for flood attacks, meaning they can be helpful for the detectors. Additionally, the percentage of complete flows feature is nearly maximum for the benign class, indicating most benign flows are correctly terminated, but low for the malign class, which is expected as many attacks never get to properly initiate and terminate the connections.

More features could be created in a similar fashion as these. However, it's important to take into account network condition factors that can lead to an unexpected result on the distribution. For instance, it's tempting to create features such as the average flow/byte/packet rate across the window (e.g., mean #packets/duration), but in the event of an ongoing denial of service attack, it is expected for the server response time to be much greater than usual. This increases the duration of the flow, and decreases estimated rate, which is counter-intuitive, since the real traffic rate during a flood attack is likely higher than the typical traffic rate.

4.4 TCP vs. UDP Traffic Comparison and Selection

Tstat generates features on a connection-basis. Naturally, this method works best for TCP traffic, since it's a connection-oriented protocol, contrary to UDP. For the latter, *Tstat* attempts to identify flows based on socket pairs and defined timeouts. Also, the features generated for this type of traffic are more limited, as seen on table 4.6. Effectively, we only use completion times, numbers of packets and bytes for inferring UDP traffic.

The available datasets for malign traffic are also very biased towards TCP traffic. This limits the variation of malign samples present for testing. Because of that, the only relevant tests to be done for UDP are regarding the models ability to distinguish between known traffic and UDP flood

C2S	S2C	Short description	Unit	Long description
1	15	Client/Server IP addr	-	IP addresses of the client/server
2	16	Client/Server TCP port	-	TCP port addresses for the client/server
3	17	packets	-	total number of packets observed from the client/server
4	18	RST sent	0/1	0 = no RST segment has been sent by the client/server
5	19	ACK sent	-	number of segments with the ACK field set to 1
6	20	PURE ACK sent	-	number of segments with ACK field set to 1 and no data
7	21	unique bytes	bytes	number of bytes sent in the payload
8	22	data pkts	-	number of segments with payload
9	23	data bytes	bytes	number of bytes transmitted in the payload, including retransmissions
10	24	rexmit pkts	-	number of retransmitted segments
11	25	rexmit bytes	bytes	number of retransmitted bytes
12	26	out seq pkts	-	number of segments observed out of sequence
13	27	SYN count	-	number of SYN segments observed (including rtx)
14	28	FIN count	-	number of FIN segments observed (including rtx)
	29	First time abs	ms	Flow first packet absolute time (epoch)
	30	Last time abs	ms	Flow last segment absolute time (epoch)
	31	Completion time	ms	Flow duration since first packet to last packet
	32	C first payload	ms	Client first segment with payload since the first flow segment
	33	S first payload	ms	Server first segment with payload since the first flow segment
	34	C last payload	ms	Client last segment with payload since the first flow segment
	35	S last payload	ms	Server last segment with payload since the first flow segment
	36	C first ack	ms	Client first ACK segment (without SYN) since the first flow segment
	37	S first ack	ms	Server first ACK segment (without SYN) since the first flow segment
	38	C Internal	0/1	1 = client has internal IP, 0 = client has external IP
	39	S Internal	0/1	1 = server has internal IP, 0 = server has external IP
	40	C anonymized	0/1	1 = client IP is CryptoPAN anonymized
	41	S anonymized	0/1	1 = server IP is CryptoPAN anonymized
	42	Connection type	-	Bitmap stating the connection type as identified by TCPL7 inspection engine
	43	P2P type	-	Type of P2P protocol, as identified by the IPP2P engine
	44	HTTP type	-	For HTTP flows, the identified Web2.0 content

Table 4.5: *Tstat* features for TCP flows

C2S	S2C	Short description	Unit	Long description
1	10	Client/Server IP addr	-	IP addresses of client/server
2	11	Client/Server UDP port	-	UDP port addresses of client/server
3	12	First time	ms	client/server first packet in absolute time (epoch)
4	13	Completion time	s	Time between the first and the last packet from the 'client'
5	14	Data bytes	bytes	Number of bytes transmitted in the payload
6	15	Packets	-	Total number of packets observed from the client/server
7	16	Internal	0/1	1 = IP address is internal
8	17	Anonymized	0/1	1 = IP address is CryptoPAN anonymized
9	18	UDP Type	-	Protocol type
	19	FQDN	-	Fully Qualified Domain Name recovered using DNHunter

Table 4.6: *Tstat* features for UDP "flows"

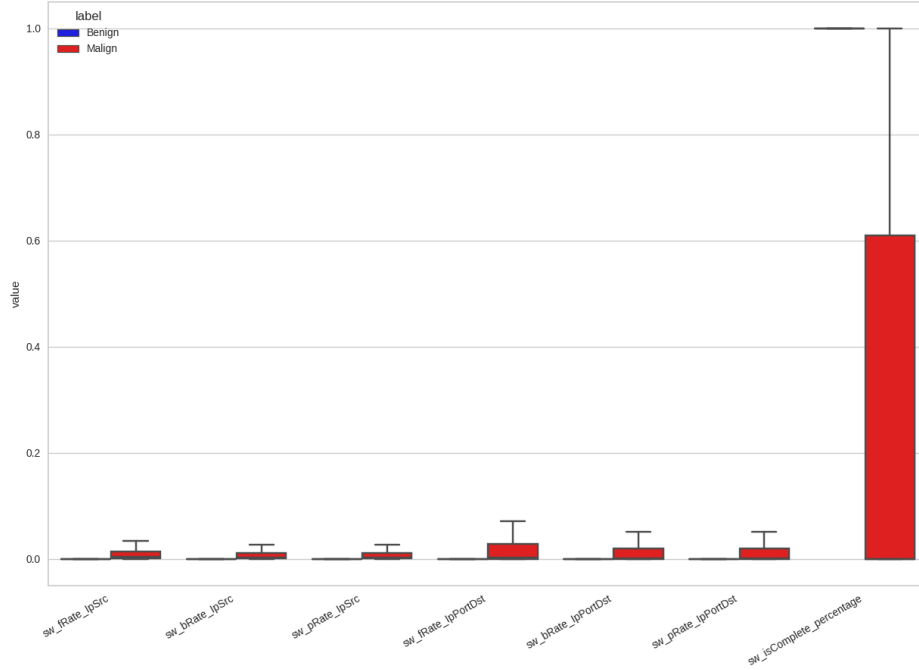


Figure 4.2: Boxplot of the newly generated features for both classes

attacks. This compensates the very limited set of features used, which under normal conditions would most likely not be large or diverse enough.

Furthermore, Domain Name System (DNS) requests are a special case of UDP traffic. They (usually) serve the sole purpose of resolving domain names into IP addresses, so they're (again, usually) inherently benign requests. However, they can be the first step of many attacks, particularly when the target lies behind a defined domain name. Therefore, these requests alone can be a case study of IDSs development, falling outside of just flow-based anomaly detection. But because of the specifics mentioned before, UDP traffic is not the primary focus of this work.

4.5 Packet Accelerator Simulation

As explained in Section 1.2, an imposed constraint of the CPE is the existence of a packet acceleration technique.

In order to assess the impact of this feature on the learning process, we simulated this operation on the datasets and re-evaluated the models. For this simulation, we experimented with two different tools (*scapy* and *Tstat* with a run-time module) and chose *Tstat* for this operation due to processing efficiency. By specifying the number of packets to analyse on each flow, we can effectively trim the length of the connections similarly to the PA.

Since we have access to datasets with and without this feature enabled, provided by NOS, we can compare the real effect of this simulation and how it compares to the real PA. Setting the packet

Dataset	TCP (%)	UDP(%)
UNSW #1	46.6	53.4
UNSW #2	86.7	13.3
NOS IoT	76.5	23.5
CTU-Normal #1	29.1	70.9
CTU-Normal #2	29.3	70.7
NOS Normal	88.4	11.6
Bot-IoT- TCP DDoS	100	0
Bot-IoT- UDP DDoS	0	100
Bot-IoT- HTTP DDoS	100	0
Bot-IoT- Service Scan	96.7	3.3
Bot-IoT- OS Scan	99.4	0.6
Bot-IoT- Data Theft	99	1
Bot-IoT- Keylogging	99.9	0.1
IoT-23 – Botnet	100	0

Table 4.7: Differences in protocol distribution on the datasets

limit per flow to 5, the distribution changes as seen in figure 4.3. While the result is not equal to the real PA process, it provides a similar trimming effect, which is evident when comparing to the original distribution.

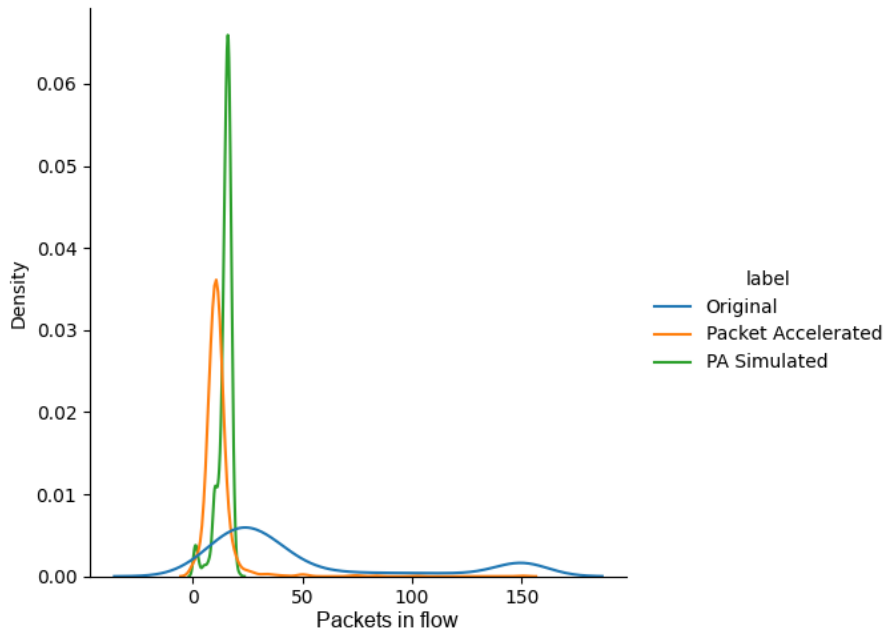


Figure 4.3: Effect of PA simulation on data distribution

Chapter 5

Results and Discussion

In this chapter, we test the models for different scenarios in order to validate the proposed system. To measure its performance for each dataset, the main metric used is **accuracy**, previously defined in Section 2.1.3. Since the datasets are separated by class, this directly translates to the True Negative Rate (**TNR**) and True Positive Rate (**TPR**) when prediction for benign or malign traffic, respectively. Then, the False Positive Rate (**FPR**) and False Negative Rate (**FNR**) are easily deducted from the those.

5.1 Anomaly Detection for Traffic Classification

We assess the ability of the ML models that learn from the behaviour of the network to correctly classify unseen traffic samples of the devices present in that network. Furthermore, we also test for the ability to identify new devices that are added to the network, which are similar in nature to previously existing ones. For this purpose, and as mentioned in section 4.2, the models were trained with traffic from **UNSW-IoT #1** and **CTU-Normal #1**. Therefore, the most relevant test datasets of benign traffic are the **UNSW-IoT #2** and **CTU-Normal #2**, since these constitute new traffic generated in the same networks. This subset of data is hereby referred to as **UNSW+CTU** for simplicity purposes. The other benign datasets serve the purpose of assessing correct benign classification of captures within different conditions.

Table 5.1 shows the accuracy when evaluating the ML models for each benign test dataset without any parameter tuning. The aggregated decision has a 89.4% accuracy when inferring on **UNSW+CTU**, and an overall 74.6% accuracy on benign traffic, i.e., TNR. Table 5.2 shows the accuracy for the malign test datasets, on which the combined models have an accuracy of 99.8%, i.e., TPR.

We then perform the parameter tuning process as described in Section 3.4.3 mainly in order to decrease the false positive rate of the models. This process also utilizes samples from **UNSW-IoT #1** and **CTU-Normal #1** for parameter tuning, but also some samples from **Bot-IoT TCP DDoS** and **Service Scan** for the positive class. Afterwards, the same tests are performed to assess the same scenarios. The comparison in performances between the scenarios is presented in Figure 5.1.

Type	Dataset	Accuracy					Majority Vote
		OC-SVM	LOF	EC	SGD OC-SVM	iForest	
Benign	UNSW-IoT #1 (T)	0.6090	0.7865	0.9616	0.2365	0.9663	0.9333
	UNSW-IoT #2	0.8542	0.0948	0.9917	0.1386	0.9922	0.9650
	IoT-23	0.3552	0.0616	0.8049	0.4025	0.9610	0.5483
	NOS -IoT	1.0000	0.0000	1.0000	0.1818	1.0000	1.0000
	CTU-Normal #1 (T)	0.4010	0.7470	0.8293	0.8862	0.9024	0.8392
	CTU-Normal #2	0.3846	0.7288	0.8262	0.8896	0.9125	0.8378
	NOS – Normal	0.0034	0.0743	0.4426	0.0642	0.6622	0.0946

Table 5.1: Performance evaluation results with default parameters on benign data (T-Training, V-Validation)

The majority vote of the optimized models achieves a performance of 98.6% for the **UNSW+CTU** subset, and an overall TNR of 89.9% and TPR of 95.2%. The full results for the optimized models, for both classes, can be found in Table 5.3.

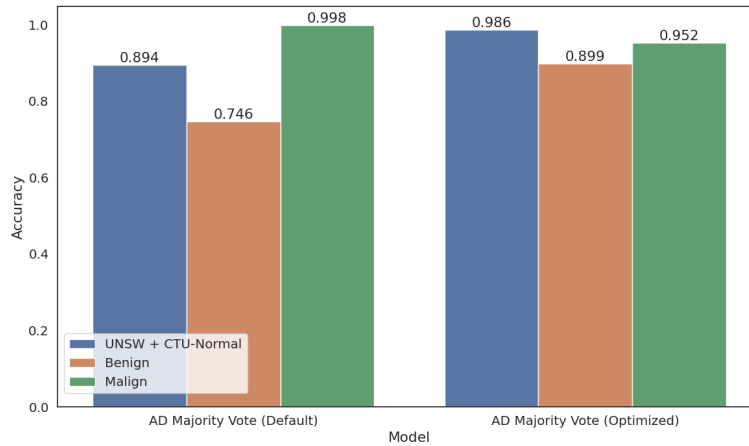


Figure 5.1: Performance comparison between default and optimized parameters models

Discussion

Comparing the two solutions, for the **UNSW+CTU** subset we increase the accuracy from 89.4% to 98.6%. This means that the optimized models are better at recognizing traffic captured within same network conditions as the training data, which is to be expected, given the optimization process. The optimized models generally perform better than the default ones for benign traffic from all sources, resulting in a increased TNR from 74.6% to 89.9%. This comes at the cost of lower positive class performance, with the TPR lowering from 99.8% to 95.2%. Overall, this trade-off alone is advantageous, since the FNR is still under a reasonable 5%, but the FPR, which is important for this context in order not to raise too many false alerts to the customer, is much lower, particularly for the **UNSW+CTU**.

Type	Dataset	Accuracy					Majority Vote
		OC-SVM	LOF	EC	SGD OC-SVM	iForest	
Malign	Bot-IoT - TCP DDoS	1.0000	1.0000	1.0000	0.3729	1.0000	1.0000
	Bot-IoT - HTTP DDoS	1.0000	1.0000	0.9616	0.6830	0.8103	1.0000
	Bot-IoT - OS Scan	0.9926	0.9953	0.9544	0.6711	0.9156	0.9980
	Bot-IoT - Service Scan	0.9993	0.9994	0.9957	0.8191	0.9072	0.9986
	Bot-IoT - Keylogging	1.0000	1.0000	0.7038	1.0000	0.7276	1.0000
	Bot-IoT - Data Theft	1.0000	0.9898	0.6939	0.9439	0.0816	0.9898
	IoT-23 - Port Scan	1.0000	1.0000	1.0000	1.0000	0.9881	1.0000
	IoT-23 - DDoS	1.0000	1.0000	1.0000	1.0000	0.0174	1.0000
	IoT-23 - C&C	1.0000	0.9995	0.5880	0.9995	0.7563	0.9995

Table 5.2: Performance evaluation results with default parameters on malign data

There's also a clear difference in performance between the **UNSW+CTU** subset and the other benign datasets. The other ones, being captured in different conditions, by different people and for different purposes, appear to be statistically different enough from the training data so that the models misclassify a reasonable portion of that data incorrectly, particularly the NOS provided normal dataset, on which the models perform poorly. This indicates that this group of models as is benefits from a tailored training process to the customer traffic.

5.2 Anomaly Detection vs. Supervised Learning Comparison

In order to assess how the proposed one-class models compare to traditional supervised methods, we trained four models of similar nature for comparison, namely traditional SVM, Random Forests (RF), Naive Bayes (NB) and k-Nearest Neighbors (KNN), using their respective scikit-learn implementations. These are trained with the same data used to train and validate the anomaly detectors (**UNSW-IoT #1** and **CTU-Normal #1** for benign data, **Bot-IoT TCP DDoS** and **Service Scan** for malign data). They are also optimized (i.e., have their parameters tuned) and then tested on the same datasets.

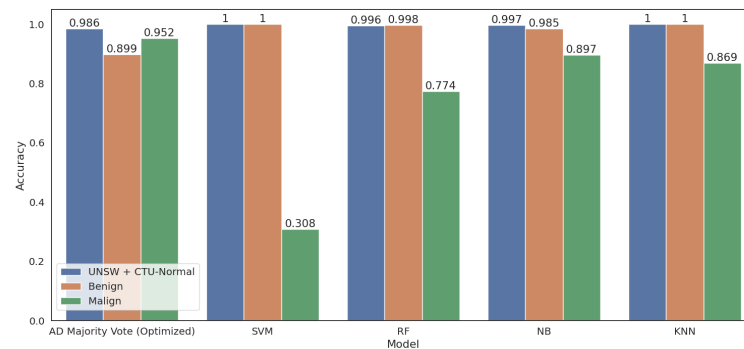


Figure 5.2: Performance comparison between the anomaly detector and supervised models

In Figure 5.2 we can see how the trained supervised models compare to the propose anomaly detection approach. Overall, the models present a higher TNR (most over 99%) but lower TPR (the

Type	Dataset	Model					Majority Vote
		OC-SVM	LOF	EC	SGD OC-SVM	iForest	
Benign	UNSW-IoT #1 (T,V)	0.9914	0.7444	0.8701	0.4751	0.9929	0.9890
	UNSW-IoT #2	0.9991	0.1226	0.9919	0.9481	0.9926	0.9923
	IoT-23	0.9281	0.1170	0.8131	0.6078	0.9733	0.9055
	NOS -IoT	1.0000	0.0909	1.0000	1.0000	1.0000	1.0000
	CTU-Normal #1 (T,V)	0.9913	0.7423	0.9074	0.8918	0.9740	0.9801
	CTU-Normal #2	0.9920	0.6884	0.9129	0.8959	0.9785	0.9822
	NOS – Normal	0.4628	0.1892	0.5642	0.1182	0.7095	0.4358
Malign	Bot-IoT - TCP DDoS (V)	1.0000	1.0000	1.0000	0.3156	1.0000	1.0000
	Bot-IoT - HTTP DDoS	0.4426	0.9851	0.8691	0.6690	0.4986	0.8612
	Bot-IoT - OS Scan	0.9699	0.9938	0.8817	0.6032	0.9520	0.9696
	Bot-IoT - Service Scan (V)	0.9890	0.9989	0.9087	0.7453	0.9859	0.9926
	Bot-IoT – Keylogging	0.9584	0.9795	0.0096	1.0000	0.7549	0.9590
	Bot-IoT - Data Theft	0.7959	0.8980	0.0204	0.9388	0.4388	0.7959
	IoT-23 – Port Scan	1.0000	1.0000	0.0002	1.0000	0.9881	1.0000
	IoT-23 – DDoS	1.0000	1.0000	0.9913	1.0000	0.1304	1.0000
	IoT-23 – C&C	0.9895	0.9993	0.0227	0.9995	0.6637	0.9900

Table 5.3: Performance evaluation results with optimized parameters (T-Training, V-Validation)

highest being under 90%) than the AD counterpart. The full performance results for the chosen supervised models in detail can be seen in Table 5.4.

Type	Dataset	Model			
		SVM	RF	NB	KNN
Benign	UNSW-IoT #1 (T,V)	1.0000	0.9996	0.9976	0.9999
	UNSW-IoT #2	1.0000	0.9998	0.9999	0.9999
	IoT-23	1.0000	1.0000	1.0000	1.0000
	NOS -IoT	1.0000	1.0000	0.9836	0.9979
	CTU-Normal #1 (T,V)	1.0000	0.9995	0.9964	0.9998
	CTU-Normal #2	1.0000	0.9997	0.9957	0.9999
	NOS – Normal	1.0000	1.0000	0.9223	1.0000
Malign	Bot-IoT - TCP DDoS (T,V)	0.9931	1.0000	1.0000	1.0000
	Bot-IoT – HTTP DDoS	0.3163	0.9767	0.4203	0.3905
	Bot-IoT - OS Scan	0.6419	0.9956	0.9668	0.9594
	Bot-IoT - Service Scan (T,V)	0.8118	0.9994	0.9890	0.9946
	Bot-IoT – Keylogging	0.0000	0.9980	0.9495	0.8198
	Bot-IoT - Data Theft	0.0102	0.9796	0.7602	0.6633
	IoT-23 – Port Scan	0.0000	0.0142	1.0000	1.0000
	IoT-23 – DDoS	0.0000	0.9913	0.9826	0.9913
	IoT-23 – C&C	0.0000	0.0139	1.0000	1.0000

Table 5.4: Performance evaluation results of supervised models, for comparison (T-Training, V-Validation)

Discussion

By looking at the results, it is immediately evident that the supervised models are more capable at inferring benign traffic correctly consistently than the AD, particularly for traffic the AD struggled with, such as the normal dataset provided by NOS. However, they generally perform worse for malign traffic. Therefore, opting for a supervised solution would likely lead to less false alerts

delivered to the user, but would likely let more malicious connections go through unnoticed. This trade-off has to be studied for each use-case, as the acceptable false/true negative/positive rates need to be defined in order to make an informed decision.

Moreover, and as mentioned in Section 3.1, the generation or selection of the malign class samples might not be a straight forward process. So, choosing this solution should be done if aligned with the system requirements and this data collection is not an issue.

5.3 One- vs. Two-Stage Learning Comparison

Taking into account the two main types of traffic collected, we now include one instance of the anomaly detectors for each one of these two types, forming the two-stage detector. The training and optimization processes are similar, but done with the **UNSW-IoT** and **CTU-Normal** data for the IoT and normal models, respectively. After performing the same parameter tuning process as before, the main results achieved are displayed in Figure 5.3.

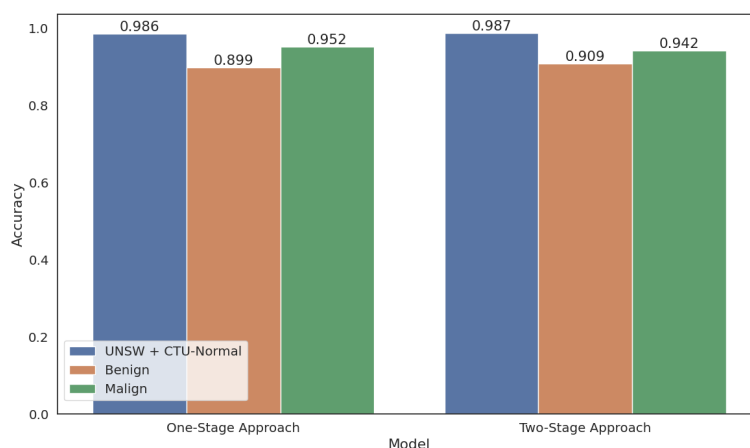


Figure 5.3: Performance evaluation of one-stage and two-stage inference

With this solution, and for the majority vote, we achieve an accuracy of 98.7% for the **UNSW-CTU** subset, and an overall TNR of 90.9% and a TPR of 94.2%. This represents an absolute increase of the TNR by 1% and decrease of the TPR by 1% as well. More details of the results can be found in Table 5.5.

Selective Vote vs. Majority Vote

Knowing by analysing the results the strengths and weaknesses of each outlier detector model, we can perform a more selective selection method for the prediction. In this case and for the two-stage model, LOF achieves a low FNR, therefore we can value its negative prediction more than the others, and SGD OC-SVM a low FPR, so we can value its positive prediction more than the others. Having done that, we improve the performance of the **UNSW-CTU** subset to 99.2%. This

Type	Dataset	Accuracy					Majority Vote
		OC-SVM	LOF	EC	SGD OC-SVM	iForest	
Benign	UNSW-IoT #1 (T,V)	0.8974	0.7991	0.8900	1.0000	0.9966	0.9838
	UNSW-IoT #2	0.9811	0.1337	0.9915	1.0000	0.9979	0.9973
	IoT-23	0.8665	0.0472	0.7762	1.0000	0.9815	0.9220
	NOS -IoT	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000
	CTU-Normal #1 (T,V)	0.9846	0.5061	0.9135	0.9996	0.9807	0.9832
	CTU-Normal #2	0.9879	0.4372	0.9212	0.9999	0.9851	0.9840
	NOS – Normal	0.3649	0.0912	0.4662	0.9324	0.8885	0.4955
Malign	Bot-IoT - TCP DDoS (V)	1.0000	1.0000	1.0000	0.0000	0.9963	1.0000
	Bot-IoT – HTTP DdoS	0.9771	0.9996	0.7228	0.0000	0.6033	0.8226
	Bot-IoT - OS Scan	0.9791	0.9960	0.9008	0.2800	0.9121	0.9689
	Bot-IoT - Service Scan (V)	0.9980	0.9994	0.9052	0.0923	0.9027	0.9899
	Bot-IoT – Keylogging	1.0000	1.0000	0.0109	0.9488	0.2464	0.9543
	Bot-IoT - Data Theft	0.9082	0.9541	0.0204	0.7908	0.0204	0.8010
	IoT-23 – Port Scan	1.0000	1.0000	0.0001	1.0000	0.0828	1.0000
	IoT-23 – DdoS	1.0000	1.0000	0.8783	0.9826	0.1304	1.0000
	IoT-23 – C&C	0.9963	0.9993	0.1579	0.9277	0.2906	0.9451

Table 5.5: Performance evaluation results of the two-stage approach (T-Training, V-Validation)

selective voting has no relevant impact on the one-stage approach, so for that only the majority vote is considered.

However, it should be considered that this approach adds an extra layer of supervision, and we can't claim with certainty that it remains valid for different datasets than the ones used here. This results just validate the hypothesis that at least for some data, the majority vote is not always the best prediction aggregation method, as is the case for these datasets for the two-stage detector.

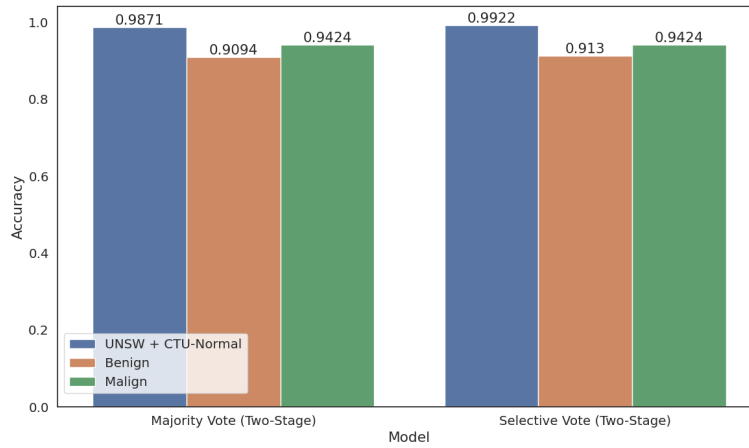


Figure 5.4: Performance differences between the majority or selective voting methods

Discussion

The results show a small difference in performance between the two approaches, with a slightly higher accuracy for benign data and slightly lower accuracy for malign data. Similarly to the previous section, this compromise is something that should align with defined requirements in order to take a thought-out decision.

Performance constraints might also be considered. This two-stage approach requires twice as many models to be trained, and twice as many predictions to be produced. This adds an overhead to the process that might not justifiable.

5.4 Packet Acceleration Impact on Models

The same training and testing processes as before were performed again, this time with data whose flows have been trimmed as described in 4.5. The impact of this change was assessed for both one- and two-stage approaches, and the main results are presented in Figure 5.5.

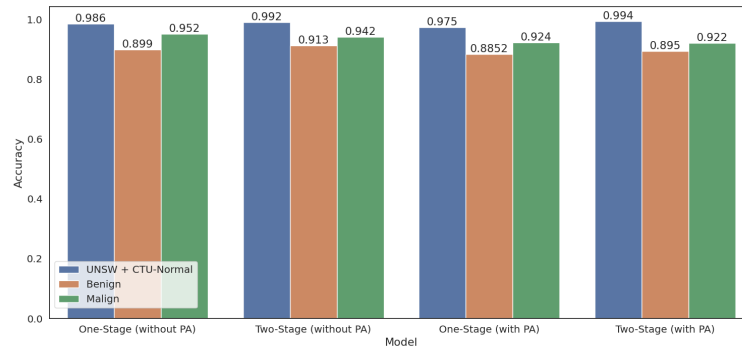


Figure 5.5: Performance differences when inferring on traffic with or without packet acceleration

Overall, this feature doesn't seem to severely impact the models' classification ability. TNR and TPR are slightly lower, and UNSW-CTU accuracy is similar. This tells us that this constraint imposed by the CPE should not lead to a need to make drastic changes in the architecture or implementation.

5.5 UDP Traffic Inference

Finally, the models were tested for UDP traffic. As mentioned before, mostly due to a lack of representative datasets of malign traffic, this protocol not as emphasized as TPC. However, we still tested the selected models ability to detect this type of traffic. The models were retrained and optimized with data from the same datasets, similarly as before.

Figure 5.6 shows the results for the best performing anomaly detector model, OC-SVM, and the selected supervised models. The AD model achieves a TNR of 99.8% and a TPR of 100%.

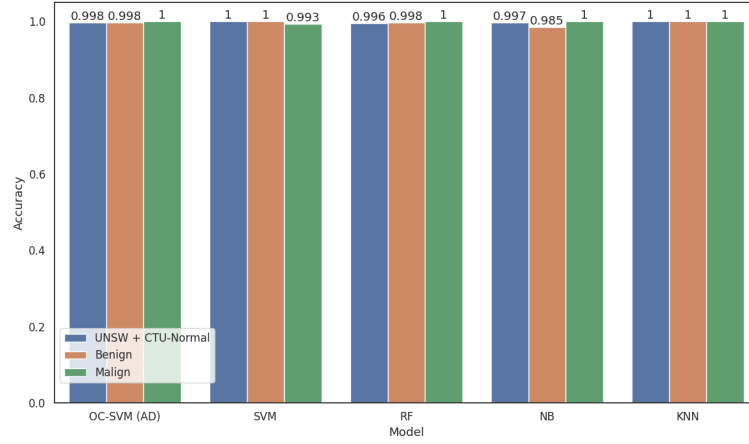


Figure 5.6: Performance evaluation for UDP traffic, between the best performing AD model and the supervised models

The best performing supervised model achieves a rounded accuracy of 100% on all datasets. Table 5.6 shows the full results for the one-stage anomaly detector for all the data.

Discussion

As mentioned before, the positive class data is severely underrepresented, with almost all samples coming from simple UDP flood attacks. Nonetheless, we show that the ML methods here employed provide good results for the selected datasets, with few false positives and the full ability to detect simple DDoS attacks. The supervised model KNN performs even better, with even less false positives, and maintaining the ability to detect these UDP DDoS samples. With a lack of more attack data to make an informed decision, it's difficult to derive their performance on more varied scenarios.

Type	Dataset	Accuracy					
		OC-SVM	LOF	EC	SGD OC-SVM	iForest	Majority Vote
Benign	UNSW-IoT #1	0.9970	0.7484	0.8525	1.0000	0.9928	0.9945
	UNSW-IoT #2 (T,V)	0.9956	0.4078	0.4975	1.0000	0.9961	0.9951
	IoT-23	0.9976	0.3529	0.8671	1.0000	0.9988	0.9988
	NOS -IoT	1.0000	0.3333	0.0000	1.0000	0.6667	0.6667
	CTU-Normal #1 (T,V)	0.9993	0.7617	0.9394	1.0000	0.9871	0.9985
	CTU-Normal #2	0.9990	0.7485	0.9709	1.0000	0.9798	0.9974
	NOS – Normal	0.9981	0.5526	0.0827	1.0000	0.9944	0.9944
Malign	Bot-IoT - UDP DDoS (V)	1.0000	1.0000	1.0000	0.0000	1.0000	1.0000

Table 5.6: Performance evaluation results for UDP traffic (one-stage AD) (T-Training, V-Validation)

Chapter 6

Discussion and Conclusion

Having presented the proposed system, the data used to validate that system, and the performance results for that data, we now draw conclusions regarding the work performed and the points that can be explored past this point.

6.1 Conclusions

The main goal of this dissertation was the development of an IDS capable of learning from the typical behaviour of the network and detect any ongoing attacks.

We first presented the workflow for achieving such task and different learning approaches that can be taken for such case. We then described the ML pipeline and the specific models that were implemented. Afterwards, we discussed the relevant datasets for training and evaluating the ML models, taking into account the different types of traffic expected in home networks.

The results then showed the system is capable of correctly classify around 90% of benign traffic, and this value is increased to around 99% if considering traffic generated within same network conditions as the training data. It is also capable of detecting different types of attacks with an average accuracy of around 95%.

Lastly, we explored variations of the learning process (such as supervised learning and multi-stage inference) and how they compare to the core solution. Each of this variations achieves slightly different results, and ultimately the chosen implementation should be in agreement with any defined requirements.

6.2 Future Work

Many different components of this work could be further explored and expanded, as a means to improve its classification performance, scalability to a larger application or simply as research questions. Some of these things could be, but not limited to:

- **Data and Feature Extraction:** In this work, features are selected on a network flow-basis. As mentioned in section [4.3](#), this limits the IDS perspective on the network to only this

dimension. More features collected at different levels, such as the packet level, device level or general network level would improve upon this existing feature space.

- **Feature and Parameter Spaces Carryover:** It should be assessed if the hyperparameter tuning process result (the resulting parameter space) and possible feature selection result (the resulting feature space) can be carried over to new environments and preserve its performance, thus eliminating the need for any malign sample usage. This requires better datasets with realistic IoT and normal traffic, with multiple scenarios, preferably with different network conditions as well.
- **Architectural Changes:** We implemented the training, validation and testing processes of the ML system all within the same device. However, in order to scale this solution to, say, an ISP level, the architecture should be adapted to the available resources. The more resource-consuming tasks can be implemented at the cloud level, whereas the simpler inference process can be done at the CPE, for instance. This might require changes to the model selection to account for this new architecture.

References

- [1] NOS. About us. Available at: <https://www.nos.pt/institucional/EN/nos/about-us/Pages/about-us.aspx> (Accessed: 28 July 2022).
- [2] Center for Strategic and International Studies (CSIS). The Hidden Costs of Cybercrime. Technical report, December 2020. URL: <https://www.csis.org/analysis/hidden-costs-cybercrime>.
- [3] Mahbod Tavallaei, Ebrahim Bagheri, Wei Lu, and Ali A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pages 1–6, July 2009. ISSN: 2329-6275. doi: [10.1109/CISDA.2009.5356528](https://doi.org/10.1109/CISDA.2009.5356528).
- [4] ENISA. ENISA Threat Landscape 2020 - Distributed denial of service. Report/Study, October 2020. URL: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-distributed-denial-of-service>.
- [5] ENISA. ENISA Threat Landscape 2020 - Botnet. Report/Study, October 2020. URL: <https://www.enisa.europa.eu/publications/enisa-threat-landscape-2020-botnet>.
- [6] Zscaler Cloud Security. IoT in the Enterprise, An analysis of traffic and threats. Technical report, May 2019. URL: <https://www.zscaler.com/blogs/research/iot-traffic-enterprise-rising-so-are-threats>.
- [7] Palo Alto Networks. 2020 Unit 42 IoT Threat Report. Technical report, March 2020. URL: <https://unit42.paloaltonetworks.com/iot-threat-report-2020/>.
- [8] Noah Apherpe, Dillon Reisman, and Nick Feamster. A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. *arXiv:1705.06805 [cs]*, May 2017. arXiv: 1705.06805. URL: <http://arxiv.org/abs/1705.06805>.
- [9] Elhadj Benkhelifa, Thomas Welsh, and Walaa Hamouda. A Critical Review of Practices and Challenges in Intrusion Detection Systems for IoT: Toward Universal and Resilient Systems. *IEEE Communications Surveys & Tutorials*, 20(4):3496–3509, October 2018. URL: <https://doi.org/10.1109/COMST.2018.2844742>, doi: [10.1109/COMST.2018.2844742](https://doi.org/10.1109/COMST.2018.2844742).
- [10] Mohamed Faisal Elrawy, Ali Ismail Awad, and Hesham F. A. Hamed. Intrusion detection systems for IoT-based smart environments: a survey. *Journal of Cloud Computing*, 7(1):21, December 2018. URL: <https://doi.org/10.1186/s13677-018-0123-6>, doi: [10.1186/s13677-018-0123-6](https://doi.org/10.1186/s13677-018-0123-6).

- [11] Nadia Chaabouni, Mohamed Mosbah, Akka Zemmari, Cyrille Sauvignac, and Parvez Faruki. Network Intrusion Detection for IoT Security Based on Learning Techniques. *IEEE Communications Surveys Tutorials*, 21(3):2671–2701, 2019. Conference Name: IEEE Communications Surveys Tutorials. doi:10.1109/COMST.2019.2896380.
- [12] Eirini Anthi, Lowri Williams, Małgorzata Słowińska, George Theodorakopoulos, and Pete Burnap. A Supervised Intrusion Detection System for Smart Home IoT Devices. *IEEE Internet of Things Journal*, 6(5):9042–9053, October 2019. Conference Name: IEEE Internet of Things Journal. doi:10.1109/JIOT.2019.2926365.
- [13] Mahmudul Hasan, Md. Milon Islam, Md Ishrak Islam Zarif, and M. M. A. Hashem. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet of Things*, 7:100059, September 2019. URL: <https://www.sciencedirect.com/science/article/pii/S2542660519300241>, doi:10.1016/j.iot.2019.100059.
- [14] Rohan Doshi, Noah Apthorpe, and Nick Feamster. Machine Learning DDoS Detection for Consumer Internet of Things Devices. *2018 IEEE Security and Privacy Workshops (SPW)*, pages 29–35, May 2018. arXiv: 1804.04159. URL: <http://arxiv.org/abs/1804.04159>, doi:10.1109/SPW.2018.00013.
- [15] Thien Duc Nguyen, Samuel Marchal, Markus Miettinen, Hossein Fereidooni, N. Asokan, and Ahmad-Reza Sadeghi. D²IoT: A Federated Self-learning Anomaly Detection System for IoT. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767, July 2019. ISSN: 2575-8411. doi:10.1109/ICDCS.2019.00080.
- [16] Mojtaba Eskandari, Zaffar Haider Janjua, Massimo Vecchio, and Fabio Antonelli. Passban IDS: An Intelligent Anomaly-Based Intrusion Detection System for IoT Edge Devices. *IEEE Internet of Things Journal*, 7(8):6882–6897, August 2020. Conference Name: IEEE Internet of Things Journal. doi:10.1109/JIOT.2020.2970501.
- [17] Abebe Abeshu Diro and Naveen Chilamkurti. Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, 82:761–768, May 2018. URL: <https://www.sciencedirect.com/science/article/pii/S0167739X17308488>, doi:10.1016/j.future.2017.08.043.
- [18] Sven Nömm and Hayretin Bahşi. Unsupervised Anomaly Based Botnet Detection in IoT Networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1048–1053, December 2018. doi:10.1109/ICMLA.2018.00171.
- [19] Francisco Sales de Lima Filho, Frederico A. F. Silveira, Agostinho de Medeiros Brito Junior, Genoveva Vargas-Solar, and Luiz F. Silveira. Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. *Security and Communication Networks*, 2019:1–15, October 2019. URL: <https://www.hindawi.com/journals/scn/2019/1574749/>, doi:10.1155/2019/1574749.
- [20] Ayush Kumar and Teng Joon Lim. EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques. In *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, pages 289–294, April 2019. doi:10.1109/WF-IoT.2019.8767194.

- [21] Yair Meidan, Michael Bohadana, Yael Mathov, Yisroel Mirsky, Asaf Shabtai, Dominik Breitenbacher, and Yuval Elovici. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, July 2018. Conference Name: IEEE Pervasive Computing. doi:[10.1109/MPRV.2018.03367731](https://doi.org/10.1109/MPRV.2018.03367731).
- [22] Christopher D. McDermott, Farzan Majdani, and Andrei V. Petrovski. Botnet Detection in the Internet of Things using Deep Learning Approaches. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, July 2018. ISSN: 2161-4407. doi:[10.1109/IJCNN.2018.8489489](https://doi.org/10.1109/IJCNN.2018.8489489).
- [23] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. DeepDefense: Identifying DDoS Attack via Deep Learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8, May 2017. doi:[10.1109/SMARTCOMP.2017.7946998](https://doi.org/10.1109/SMARTCOMP.2017.7946998).
- [24] IBM. Supervised learning. Available at: <https://www.ibm.com/cloud/learn/supervised-learning> (Accessed: 29 June 2022).
- [25] IBM. Unsupervised learning. Available at: <https://www.ibm.com/cloud/learn/unsupervised-learning> (Accessed: 29 June 2022).
- [26] 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*, 12:2825–2830, 2011.
- [27] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.
- [28] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, 2018.
- [29] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Benjamin Turnbull. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.
- [30] Nickolaos Koroniotis, Nour Moustafa, Elena Sitnikova, and Jill Slay. Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In *International Conference on Mobile Networks and Management*, pages 30–44. Springer, 2017.
- [31] Nickolaos Koroniotis, Nour Moustafa, and Elena Sitnikova. A new network forensic framework based on deep learning for internet of things networks: A particle deep framework. *Future Generation Computer Systems*, 110:91–106, 2020.
- [32] Nickolaos Koroniotis and Nour Moustafa. Enhancing network forensics with particle swarm and deep learning: The particle deep framework. *arXiv preprint arXiv:2005.00722*, 2020.

- [33] Nickolaos Koroniotis, Nour Moustafa, Francesco Schiliro, Praveen Gauravaram, and Helge Janicke. A holistic review of cybersecurity and reliability perspectives in smart airports. *IEEE Access*, 8:209802–209834, 2020.
- [34] Nickolaos Koroniotis. *Designing an effective network forensic framework for the investigation of botnets in the Internet of Things*. PhD thesis, UNSW Sydney, 2020.
- [35] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. Iot-23: A labeled dataset with malicious and benign iot network traffic (version 1.0.0) [data set]. *Stratosphere Lab., Praha, Czech Republic, Tech. Rep*, 2020. doi:<http://doi.org/10.5281/zenodo.4743746>.
- [36] Sebastian Strasak, Garcia, Agustin Parmisano, and Maria Jose Erquiaga. Detecting malware even when it is encrypted - machine learning for network https analysis. *Stratosphere Lab., Praha, Czech Republic, Tech. Rep*, 2017.
- [37] Moz. The moz top 500 websites. Available at: <https://moz.com/top500> (Accessed: 23 January 2022).