CISTER
Research Centre in
Real-Time & Embedded
Computing Systems

Conference Paper

One-Class Models for Intrusion Detection at ISP Customer Networks

Nuno Schumacher
Pedro Miguel Santos*
Pedro Souto*
Nuno Martins
Joana Sousa
João Ferreira
Luís Almeida*

*CISTER Research Centre

CISTER-TR-230609

2023/06/16

One-Class Models for Intrusion Detection at ISP Customer Networks

Nuno Schumacher, Pedro Miguel Santos*, Pedro Souto*, Nuno Martins, Joana Sousa, João Ferreira, Luís Almeida*

*CISTER Research Centre

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8321159

E-mail: nunoschumacher@protonmail.com, pss@isep.ipp.pt, pfs@fe.up.pt, nuno.mmartins@parceiros.nos.pt, joana.sousa@parceiros.nos.pt, 1100064@isep.ipp.pt, lda@fe.up.pt

https://www.cister-labs.pt

Abstract

Despite the explosion of IoT deployments at Internet Service Provider (ISP) customer networks, such devices remain vulnerable to cyber-attacks. We present a ML-based anomaly detection system, to be deployed at the Customer Premises Equipment (CPE), that leverages several One-Class Classification algorithms and majority voting to detect anomalous network traffic. We train these models using not only conventional per-flow features but also features extracted from sliding windows of flows. An extensive evaluation, using publicly available datasets shows that our algorithm has a higher detection rate than commonly supervised-learning algorithms, which require the use of labelled datasets. Our evaluation suggests that the detection capabilities of our algorithm are only marginally affected by Packet Acceleration, a technique used by CPEs to improve throughput but that reduces the number of packets (per flow) available to extract features from.

© 2023 CISTER Research Center
www.cister-labs.pt

One-Class Models for Intrusion Detection at ISP Customer Networks

Nuno Schumacher[1][0009−0003−1782−6861], Pedro M. Santos[2,3][0000−0002−7162−0560], Pedro F. Souto[1,3][0000−0002−0822−3423], Nuno Martins[4][0000−0002−4509−2493], Joana Sousa[4][0000−0002−6418−2312], João M. Ferreira[4], and Luís Almeida[1,3][0000−0002−9544−3028]

[1] Universidade do Porto - Faculdade de Engenharia
[2] Instituto Superior de Engenharia do Porto - Instituto Politécnico do Porto
[3] CISTER Research Unit
[4] NOS Inovação, Lisboa, Portugal
nuno.schumacher@fe.up.pt, pss@isep.ipp.pt, pfs@fe.up.pt, {nuno.mmartins,joana.sousa,joao.MFerreira}@nos.pt, lda@fe.up.pt

Abstract. Despite the explosion of IoT deployments at Internet Service Provider (ISP) customer networks, such devices remain vulnerable to cyber-attacks. We present a ML-based anomaly detection system, to be deployed at the Customer Premises Equipment (CPE), that leverages several One-Class Classification algorithms and majority voting to detect anomalous network traffic. We train these models using not only conventional per-flow features but also features extracted from sliding windows of flows. An extensive evaluation, using publicly available datasets shows that our algorithm has a higher detection rate than commonly supervised-learning algorithms, which require the use of labelled datasets. Our evaluation suggests that the detection capabilities of our algorithm are only marginally affected by Packet Acceleration, a technique used by CPEs to improve throughput but that reduces the number of packets (per flow) available to extract features from.

Keywords: IoT · Intrusion Detection System · One-Class Classification.

1 Introduction

By the end of 2020, cybercrime-derived losses approached $1 trillion [3]. A particular type of cyberattacks is Denial-of-Service (DoS), in which a malicious agent floods a target server with a number of requests that exceeds the server's capacity, thus denying access to legitimate users. A Distributed DoS (DDoS) attack generates traffic from a very large number of sources that have been infected previously with malware (*botnet*) that is under remote control from a Command and Control (C&C) server. The distributed nature of this attack makes it hard to counteract, because the traffic pattern generated by the devices may be difficult to distinguish from regular traffic. A well-known malware used in DDoS attacks is Mirai, which targeted domestic IoT devices such as surveillance cameras, DVRs and routers, and was responsible, among others, for a DDoS attack to French webhost and cloud service provider OVH that peaked at 1.1 Tbps [6].

IoT devices are often not equipped with levels of protection similar to personal devices (e.g., laptops, smart phones), and attackers explore such ill-protected devices for

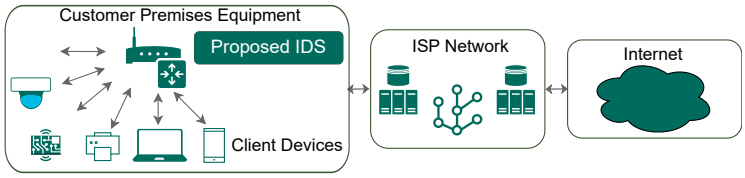

Fig. 1. Context of operation and placement of proposed IDS

infection and attack. With the growing number of IoT devices deployed at their customer premises, Internet Service Providers (ISPs) are interested in protecting their customers networks as well as their own. In this paper we describe Machine Learning (ML) models for an Intrusion Detection System (IDS) to be deployed at the Customer Premises Equipment (CPE), an ISP-provided equipment that enables Internet connectivity and local networking, as shown in Fig. 1. The goal is to monitor the network traffic at the CPE and use ML techniques to detect anomalies. More specifically, we present an anomaly detector that uses several one-class classification (OCC) models and majority voting to classify network traffic as either legitimate or (potentially) malicious. Model training can happen either at the CPE or be offloaded to the cloud. As CPEs have limited computing power, we foresee training to be cloud-based and trained models to be deployed at the CPE for inference; however, we do not address this aspect of the system in this paper.

The deployment of the IDS at the CPE runs into another specific challenge: CPEs typically resort to packet accelerators (PA) to speed up packet processing. As a result, the IDS only has access to the first few packets of each traffic flow, i.e. it must make predictions based only on those packets.

Our contributions can be summarized as follows.

- **Use of features of a sliding window of traffic flows** in combination with traditional traffic flow features.
- **Use of a refined hyperparameter tuning strategy** for training each of the OCC models and of **majority voting** for prediction. This combination leads to a model that has a higher detection rate than models using supervised learning.
- An extensive evaluation of the prediction accuracy with and without **packet acceleration** (PA), indicating that PA does not affect performance significantly.

The remainder of this paper is as follows. Section 2 reviews the relevant state-of-the-art. Section 3 details the datasets identified and used in this work, followed by the developed ML pipeline. A performance evaluation of the system is presented in Section 4. Section 5 draws final remarks.

2 Related Work on ML-based IDS for IoT Networks

In this section, we focus our review on detection of (D)DoS and Botnet infections in IoT networks, due to the growing relevance of such attacks. A broader review of the research on the application of ML to IDS for IoT networks can be found in survey papers like [2] and [4]. Nõmm and Bahşi [17] developed an anomaly-based detection system with emphasis on reducing model dimension. They compared a common model with device-specific models using iForest and SVM and concluded that, for some devices, a common model had significantly lower detection performance, due to the complexity and uniqueness of some devices. Lima Filho et al. [14] developed *Smart Detection*, an ML system for DoS/DDoS detection. The authors used a dataset of their own and three public

datasets to validate their Random Forest based model; results showed that the system was capable of accurately detecting different types of DoS attacks such as TCP/UDP/HTTP floods. Yuan et al. [21] developed *DeepDefense*, a Deep Learning detection system with emphasis on DDoS. By using various types of Recurrent Neural Networks (RNN), such as LSTM, authors achieved a reduction of the error rate from 7.517% to 2.103%, with respect to Random Forest. Other line of research is the detection as early as possible of infection by botnets such as Mirai. Kumar and Lim [13] developed *Early Detection of IoT Malware Network Activity* (EDIMA), a botnet detection system aimed at identifying malware activity during the scanning and infection phases. They used their own dataset to simulate 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. [16] developed an anomaly-based IDS for botnet detection using deep learning techniques. They trained an autoencoder for each IoT device to capture their normal traffic behavior, and evaluated their method by deploying the Mirai and BASHLITE botnets. Every attack was successfully detected with a low false positive rate. McDermott et al. [15] 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 their own dataset with Mirai attack samples; BLSTM-RNN was compared to LSTM-RNN and the results showed high accuracy in the detection of botnets and other related attacks.

Our work focus on one-class classification (OCC) algorithms, as [17], but it considers a larger number of algorithms and uses majority voting to improve the detection rate and reduce the false positive rate. Furthermore, it uses inter-flow (sliding window) features to improve those metrics, and evaluates the effect of packet acceleration, a feature typically used by CPEs to increase their bandwidth. To develop our system, we carried out a careful review of publicly-available datasets and judiciously combined them to obtain a composite dataset that features both personal and IoT traffic with adequate representativeness.

3 Machine Learning Models for Anomaly Detection

We describe the training of the Machine Learning models used in this Intrusion Detection System. The goal of this system is to detect anomalous traffic at or to/from the customer network, i.e., traffic that differs considerably from the typical observed behaviour and that can be presumed to be malicious traffic.

Two particular decisions have influenced our approach. The first one was to train ML models solely with legitimate data, under the assumption that malicious traffic datasets are hard to obtain or of little use, as unknown attacks may operate differently than existing ones. As mentioned in Section 1, this option lead to the use of OCC models. The second one was to use two types of legitimate traffic: *IoT* and *normal* traffic. By IoT traffic, we mean traffic generated by devices that collect data or perform simple home tasks. IoT devices connected to a home network include IP security cameras, smart bulbs and door sensors. The traffic volume generated by these devices is typically low (with exceptions such as camera video streams), the connections are periodic and relatively repetitive [1]. Normal traffic is usually generated directly by humans on their personal devices, such as laptops and smartphones; examples are web browsing, gaming and video conferencing.

The two above decisions informed our selection of publicly available datasets; these are described in Section 3.1. Used features and extraction procedure are described

Table 1. Selected datasets and types of traces.

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

Table 2. Traces Used For Training And Validating The Models

	Dataset	Training	Hyperparameter Tuning	Evaluation	Capture details
Benign	UNSW-IoT #1	2000	2000	1832	Amazon Echo, Netatmo camera, Lixf light, WEMO power switch
	UNSW-IoT #2	-	-	5331	iHome, Samsung camera, Hue bulb, TP Link plug
	IoT-23	-	-	486	Amazon Echo, Hue bulb, Somfy door lock
	NOS - IoT	-	-	10	Smart plug, temperature and humidity sensors
	CTU-Normal #1	2000	2000	27639	Automated capture, top 500 websites: 1-30, 61-120, 241-360
	CTU-Normal #2	-	-	31790	Automated capture, top 500 websites: 31-60, 150-240, 361-500 and manual capture of web traffic
	NOS -Normal	-	-	295	Manual capture of web traffic
Malign	Bot-IoT - TCP DDoS	-	2000	35585	TCP floods generated using hping3
	Bot-IoT - HTTP DDoS	-	-	19180	HTTP request floods using Golden-eye
	Bot-IoT - OS Scan	-	-	68	OS fingerprinting using Nmap and Xprobe2
	Bot-IoT - Service Scan	-	2000	34885	Different port scans using Nmap and Hping3
	Bot-IoT - Keylogging	-	-	1464	Record keystrokes with Logkeys or exploits
	Bot-IoT - Data Theft	-	-	197	Directory exfiltration through system exploits
	IoT-23 - DDoS	-	-	114	DDoS attack part of a botnet
	IoT-23 - Port Scan	-	-	68	Port scans part of a botnet
	IoT-23 - C&C	-	-	4094	C&C communication part of a botnet

in Section 3.2. Lastly, in Section 3.3, we describe the selected OCC ML models, the majority voting procedure developed to aggregate the output of the various models into a single classification, and the hyperparameter tuning procedure carried out at training to improve accuracy.

3.1 Public Datasets Description

The datasets relevant to this work, and that would be captured by the Customer Premises Equipment (CPE), are composed of packet capture traces. Sequences of packets that have similar characteristics (e.g., same source and destination nodes) and occur in temporal proximity can be aggregated in **flows**. In TCP-based transactions, flows are named *connections*, and the start and end of transactions are explicitly identified with dedicated packets. Our OCC models use as features mainly statistical characteristics of flows. After introducing the used datasets, we discuss the process of mapping raw packet captures into flow descriptions.

Used datasets were the **UNSW-IoT** [19], **Bot-IoT** [12, 11, 10, 8, 9, 7], **IoT-23** [5], **CTU-Normal** [20], and the **NOS** data set (produced by portuguese ISP NOS and not publicly disclosed at this moment). 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 1 outlines the types of samples present in each of these datasets.

The **UNSW-IoT** dataset was separated into two subsets with devices of similar types, as shown in Table 2. The first set (**UNSW-IoT #1**) was used for the hyperparameter tuning selection, as described in Section 3.3, and training with different samples from that subset afterwards. The remaining samples are used for evaluation. The second set (**UNSW-IoT #2**) is used solely for inference, to evaluate the model's ability to correctly recognise new traffic of a similar type. We took a similar approach for the normal traffic, with the the first set (**CTU-Normal #1**) being used for training and

Table 3. *Tstat* (top) and sliding window (bottom) features

Feature	Unit	Description
packets	-	total number of packets observed form the client/server
RST sent	0/1	0 = no RST segment has been sent by the client/server
ACK sent	-	number of segments with the ACK field set to 1
PURE ACK sent	-	number of segments with ACK field set to 1 and no data
unique bytes	bytes	number of bytes sent in the payload
data pkts	-	number of segments with payload
data bytes	bytes	number of bytes transmitted in the payload, including retransmissions
rexmit pkts	-	number of retransmitted segments
rexmit bytes	bytes	number of retransmitted bytes
out seq pkts	-	number of segments observed out of sequence
SYN count	-	number of SYN segments observed (including rtx)
FIN count	-	number of FIN segments observed (including rtx)
Completion time	ms	Flow duration since first packet to last packet
C first payload	ms	Client first segment with payload since the first flow segment
S first payload	ms	Server first segment with payload since the first flow segment
C last payload	ms	Client last segment with payload since the first flow segment
S last payload	ms	Server last segment with payload since the first flow segment
C first ack	ms	Client first ACK segment (without SYN) since the first flow segment
S first ack	ms	Server first ACK segment (without SYN) since the first flow segment
sw_fRate_IPSrc	flows/s	Rate of flows with same origin IP
sw_bRate_IPSrc	bytes/s	Rate of bytes with same origin IP
sw_pRate_IPSrc	pkts/s	Rate of packets with same origin IP
sw_fRate_IpPortDst	flows/s	Rate of flows with same destination IP and Port
sw_bRate_IpPortDst	bytes/s	Rate of bytes with same destination IP and Port
sw_pRate_IpPortDst	pkts/s	Rate of packets with same destination IP and Port
sw_isComplete_percentage	-	Percentage of flows that are complete

hyperparameter tuning and the second (**CTU-Normal #2**) for evaluation. The rest of the benign samples from the remaining datasets (**IoT-23** and **NOS**) was used solely for evaluating the models.

Selected datasets typically either do not have absolute *Ground Truth* labels, i.e., a label indicating whether a given flow is legitimate or malicious, or use rule-based external tools to attribute labels, such as *Zeek* (https://zeek.org/) or *Argus* (https://openargus.org/), which is not appropriate for inference on unseen anomalous traffic patterns since their rules are too specific. In our work, we resort in great measure to the *tstat* (http://tstat.polito.it/) tool (discussed in the next section) to extract flows from raw packet captures. The issue arises to map one tool's flows (e.g., *tstat*) into another tool's flows (*Zeek* or *Argus*). For that, we developed a tool that allows us to map these labels of existing datasets into *tstat* generated logs, thus allowing us to use these datasets to train the models and still have the large feature generation of *tstat*.

3.2 Feature Extraction & Description

The raw packet captures are processed to extract flow and inter-flow characteristics that serve as features in the model training and evaluation procedures.

Flow-based features: The publicly available packet traces were parsed with *tstat*, a tool that identifies flows from packet traces. Features that have no statistical relevance are removed, such as identifiers (e.g., IP addresses), timestamps and highly-contextual characteristics (e.g., whether address are anonimized/internal or not). Table 3 presents the flow descriptions produced by *tstat* that we use to train our models. The feature values were normalized before applied to the models via Min-Max normalization

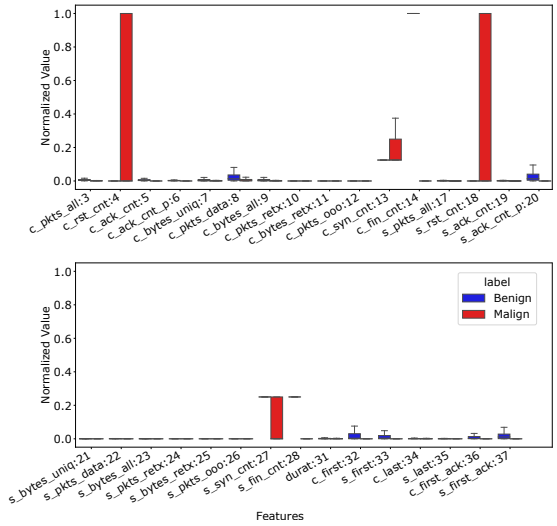

Fig. 2. Boxplot of *tstat*-generated features for both classes

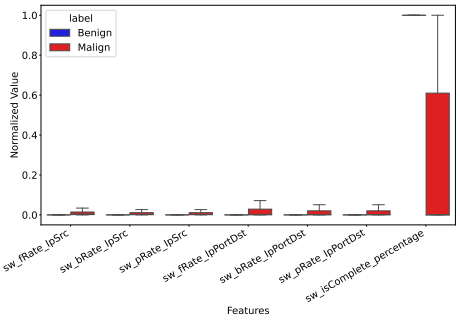

Fig. 3. Boxplot of the window-based features per traffic class.

(rescaling the training data to [0,1]). Fig. 2 shows a boxplot of the numerical normalized features for the benign and malign sets for hyperparameter tuning (see Table 2). It shows that some features have different distributions for each class, and therefore should be used to improve the classifiers' performance. Note that the rest of the data (hyperparameter tuning and evaluation data) is rescaled using the same reference values as the training set.

Sliding Window features: While informative, observing specific flows does not directly give network context (e.g., network throughput) or device context (e.g., device throughput), and it does not provide much information at a packet level either. We generated extra features (from *tstat*'s) using a sliding window of 100 flows (a value selected empirically), that are also shown in Table 3 (bottom section).

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 (refered to as client by Tstat, thus specifying the origin device) and by destination (refered to server by *tstat*) IP address and port (thus specifying the destination device/service). Lastly, the percentage of flow

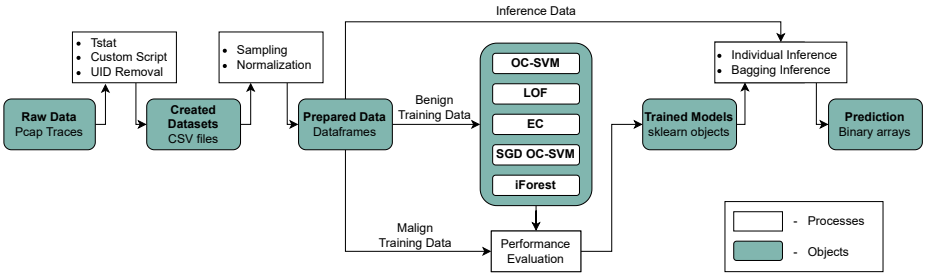

Fig. 4. Developed prediction pipeline for the proposed IDS

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).

Fig. 3 shows the normalized distribution of these derived features per traffic class. We observe larger values for the malign class, as expected for flood attacks, thus indicating potential usefulness to the classifiers. 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.

3.3 Selected ML Methods, Decision and Tuning

As refered earlier, we focus on the use of one-class classification (OCC) algorithms. OCC models receive only benign/legitimate (or negative class) data and attempt to fit boundaries around these data points. The OCC models used in this work are:

- **One-Class Support Vector Machine** (OC-SVM)
- **Stochastic Gradient Descent One-Class Support Vector Machine** (SGD OC-SVM)
- **Local Outlier Factor** (LOF)
- **Elliptic Curve** (EC)
- **Isolation Forest** (iForest)

Model implementations were provided by the *scikit-learn* library [18]. At the end, a single classification output must be provided. Once the models are trained and a classification is requested, we implement a **majority vote strategy**, in which the class with the highest number of votes is output. Fig. 4 describes the complete prediction ML pipeline, including the feature extraction stage described earlier.

During the training stage, each model has a set of user-defined parameters (i.e., hyperparameters) that can be changed as part of a validation process. Optimizing these based on performance for only the benign class (i.e., true negative rate) might lead to large or underfit boundaries that maximize performance for the benign class, but then incorrectly classify malicious samples as also 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 of the model with the chosen parameters. 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 for training new models without the need for validation and, consequently, any malign samples.

Table 4. Performance of optimized, supervised and PA models

	Dataset	Tuning		Supervised				PA
		Default	Tuned	SVM	RF	NB	KNN	
Benign	UNSW-IoT #1 (T,V)	0.9333	**0.9890**	1.0000	0.9996	0.9976	0.9999	0.9239
	UNSW-IoT #2	0.9650	**0.9923**	1.0000	0.9998	0.9999	0.9999	0.9939
	IoT-23	0.5483	**0.9055**	1.0000	1.0000	1.0000	1.0000	0.8704
	NOS - IoT	1.0000	**1.0000**	1.0000	1.0000	0.9836	0.9979	1.0000
	CTU-Normal #1 (T,V)	0.8392	**0.9801**	1.0000	0.9995	0.9964	0.9998	0.9893
	CTU-Normal #2	0.8378	**0.9822**	1.0000	0.9997	0.9957	0.9999	0.9929
	NOS - Normal	0.0946	**0.4358**	1.0000	1.0000	0.9223	1.0000	0.4257
Malign	Bot-IoT - TCP DDoS (T,V)	1.0000	**1.0000**	0.9931	1.0000	1.0000	1.0000	1.0000
	Bot-IoT - HTTP DDoS	1.0000	**0.8612**	0.3163	0.9767	0.4203	0.3905	0.9174
	Bot-IoT - OS Scan	0.9980	**0.9696**	0.6419	0.9956	0.9668	0.9594	0.9693
	Bot-IoT - Service Scan (T,V)	0.9986	**0.9926**	0.8118	0.9994	0.9890	0.9946	0.9914
	Bot-IoT - Keylogging	1.0000	**0.9590**	0.0000	0.9980	0.9495	0.8198	0.9235
	Bot-IoT - Data Theft	0.9898	**0.7959**	0.0102	0.9796	0.7602	0.6633	0.7449
	IoT-23 - Port Scan	1.0000	**1.0000**	0.0000	0.0142	1.0000	1.0000	N/A
	IoT-23 - DDoS	1.0000	**1.0000**	0.0000	0.9913	0.9826	0.9913	N/A
	IoT-23 - C&C	0.9995	**0.9900**	0.0000	0.0139	1.0000	1.0000	N/A

Considering the number of samples is limited in one of the chosen datasets, and that models size can increase drastically with number of training samples, each training and hyperparameter tuning set is sampled to **n=2000 samples**, while the remaining samples are used for evaluation.

4 Results and Discussion

We evaluate the models for different scenarios. The main metric used is **accuracy**; 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. Note that the results that combine different datasets are a simple average of the performance for each dataset, i.e. it doesn't take into account the size of theses datasets.

4.1 Baseline & Tuned Classification Performance

We assess the ability of the ML models to correctly classify unseen traffic samples of the devices present in a network, as well as from other devices similar to those seen during training. The models were trained with traffic from **UNSW-IoT #1** and **CTU-Normal #1**. The most relevant test datasets of benign traffic are the **UNSW-IoT #2** and **CTU-Normal #2**, since these constitute (different) traffic generated in the same networks. This subset of data is hereby refered to as **UNSW+CTU** for simplicity. The other benign datasets serve the purpose of assessing correct benign classification of captures in different conditions.

Table 4 (*Default* column) shows the accuracy when evaluating the ML models for each benign test dataset without any parameter tuning. Fig. 5 (left side) summarizes the accuracy of the majority voting of the models with the untuned hyperparameters. The aggregated decision has a 89.4% accuracy, i.e., TNR, when inferring on **UNSW+CTU**, but the average (non-weighted) of the model accuracy per benign dataset is 74.6%. For the malign datasets, the average of the per dataset accuracy, i.e. TPR, is 99.8%.

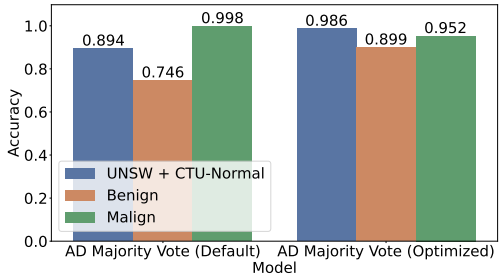

Fig. 5. Comparison between models with untuned (default) and tuned (optimized) parameters

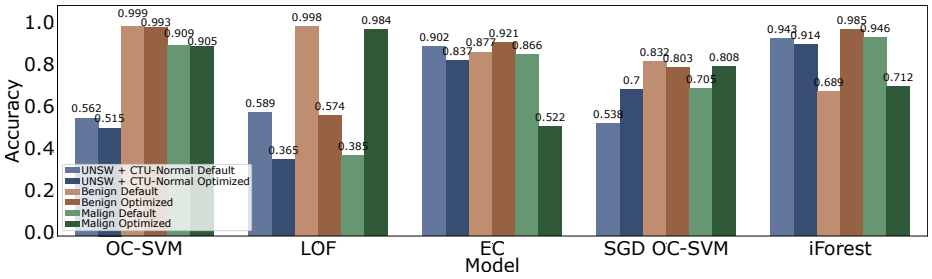

Fig. 6. Performance of individual models with default and tuned parameters.

We perform the hyperparameter tuning process (Section 3.3) to decrease the false positive rate of the models. This procedure uses samples from **UNSW-IoT #1** and **CTU-Normal #1**, and also samples from **Bot-IoT TCP DDoS** and **Bot-IoT Service Scan** for the positive class. Afterwards, we evaluate the tuned models using the datasets previously used for evaluating the models with default hyperparameters. As earlier, Table 4 (*Tuned* column) shows the classification performance per dataset aggregated across all tuned models, and Fig. 5 (right side) summarizes the accuracy of the majority voting of the models with tuned hyperparameters. The majority vote of the tuned models achieves a performance of 98.6% for the **UNSW+CTU** subset, and an overall TNR of 89.9% and TPR of 95.2%.

Comparing default and tuned, hyperparameter tuning increases the TNR for the **UNSW+CTU** subset from 89.4% to 98.6%. In general tuned models outperform untuned ones for benign datasets, resulting in a increased TNR from 74.6% to 89.9%. This comes at the cost of the TPR, that decreases 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 is lower (particularly for the **UNSW+CTU** dataset), which is important to avoid raising too many false alerts to the ISP.

Finally, Fig. 6 presents the performance of individual models with default and tuned parameters aggregated per class. It is worth noting that some models clearly under-perform with the training data **UNSW+CTU** dataset (e.g., OC-SVM, LOF), but provide reasonable accuracy when classifying the other benign datasets. This may indicate statistical differences between the training data and the other benign datasets. Nevertheless, we observe that the aggregated classification accuracy, obtained through majority vote of the classifications of all models, improves over the accuracy of the individual models.

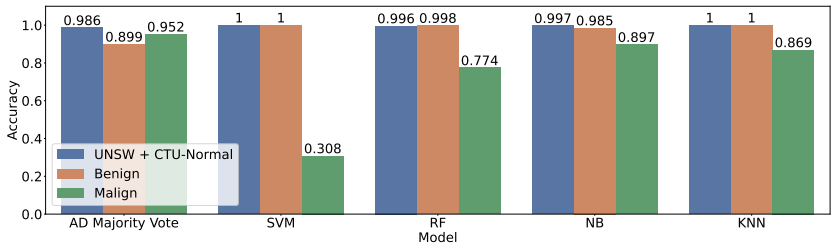

Fig. 7. Performance comparison between the anomaly detector and supervised models

4.2 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 using traditional SVM, Random Forests (RF), Naive Bayes (NB) and k-Nearest Neighbors (KNN), more specifically their respective *scikit-learn* implementations. We trained these models using the datasets used for training and for hyperparameter tuning of the one-class models. For hyperparameter tunning, we used disjoint subsets of the datasets used for training.

Fig. 7 summarizes the accuracy of these supervised models for the evaluation datasets previously used. Overall, the models present a higher TNR (most over 99%) but lower TPR (the highest being under 90%) than the AD counterpart. The performance results for the supervised models are shown in Table 4 (*Supervised* columns).

The supervised models are better at inferring legitimate traffic correctly than the tuned version of the OCC models, including datasets for which OCC performed poorly (e.g., *NOS-Normal* dataset). Interestingly enough, they generally perform worse for malign traffic. Therefore, opting for a supervised solution would likely lead to less false alerts delivered to the ISP, but allow more malicious connections to go through unnoticed. This trade-off has to be assessed for each use-case, as the acceptable rates of each kind may vary, e.g., in accordance to Service Level Agreements.

4.3 Impact of Packet Acceleration (PA) on Model Accuracy

As described in Section 1, CPEs can be equipped with packet accelerator to improve network throughput. This means that the classifiers deployed at the CPE may only have access to the first few packets of each flow for the inference process (the remaining packets of the flow are forwarded directly to the outgress port).

To simulate the effect of PA on the available datasets, we trimmed the flows according to what a PA would do. To do so, **NOS** (a portuguese ISP) provided us datasets containing samples of flows with the PA deactivated (untrimmed flows) and activated (trimmed flows) captured on an actual CPE. We computed the probability mass functions (PMF) of these two sets of data, and configured *tstat* to trim flows in all datasets in a way that the resulting PMF mimics, to the extent possible, the PMF of the **NOS** trimmed dataset. datasets. This procedure was done by instructing *tstat* to trim flows to 5 packets – the value that we found, by visual inspection, to provide the best results. While the PMF of the new trimmed datasets did not replicate exactly the PMF of the **NOS** trimmed data, we consider a sufficiently similar result was obtained.

We trained and tuned the hyperparameters of the various one-class models using the same datasets as before, but now using the trimmed versions. Fig. 8 summarizes

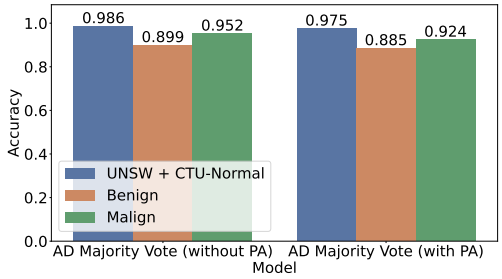

Fig. 8. Performance with or without PA.

the accuracy of these models for the evaluation datasets previously used, but with the respective flows trimmed. Table 4 (*PA* column) shows the complete results. Overall, TNR is about 1.5% lower whereas TPR is about 3.3% lower. This suggests that PA does not to significantly affect the models classification ability, even when they are built with trimmed flows.

5 Conclusion

We report the development of ML-based anomaly detection (AD) techniques to be integrated in an Intrusion Detection System (IDS). The system is able to learn the typical traffic behaviour in the customer network and flag any anomalous traffic. We trained a range of AD methods with different underlying operating natures and combine their outputs into a single verdict using majority voting. To this end, we developed an ML pipeline that includes, at the training phase, an hyperparameter tuning stage that uses examples of malicious traffic for refinement. Lastly, we list the publicly-sourced datasets that used for training and evaluating the models, and that encompass the different types of traffic expected in home networks. Performance evaluated showed correct classification rates of around 90% for legitimate traffic; this value increases to around 99% when considering traffic generated within same network conditions as the training data. It also detects different types of attacks with an average accuracy of around 95%.

Acknowledgements. 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). The authors thank João Tagaio for creating the **NOS** dataset.

References

1. Apthorpe, N., Reisman, D., Feamster, N.: A Smart Home is No Castle: Privacy Vulnerabilities of Encrypted IoT Traffic. arXiv:1705.06805 [cs] (May 2017), http://arxiv.org/abs/1705.06805, arXiv: 1705.06805
2. Benkhelifa, E., Welsh, T., Hamouda, W.: 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 (Oct 2018)

3. Center for Strategic and International Studies (CSIS): The Hidden Costs of Cybercrime. Tech. rep. (Dec 2020), https://www.csis.org/analysis/hidden-costs-cybercrime
4. Elrawy, M.F., Awad, A.I., Hamed, H.F.A.: Intrusion detection systems for IoT-based smart environments: a survey. Journal of Cloud Computing **7**(1), 21 (Dec 2018)
5. Garcia, S., Parmisano, A., Erquiaga, M.J.: 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). https://doi.org/10.5281/zenodo.4743746
6. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017). https://doi.org/10.1109/MC.2017.201
7. Koroniotis, N.: Designing an effective network forensic framework for the investigation of botnets in the Internet of Things. Ph.D. thesis, UNSW Sydney (2020)
8. Koroniotis, N., Moustafa, N.: Enhancing network forensics with particle swarm and deep learning: The particle deep framework. arXiv preprint arXiv:2005.00722 (2020)
9. Koroniotis, N., Moustafa, N., Schiliro, F., Gauravaram, P., Janicke, H.: A holistic review of cybersecurity and reliability perspectives in smart airports. IEEE Access **8**, 209802–209834 (2020). https://doi.org/10.1109/ACCESS.2020.3036728
10. Koroniotis, N., Moustafa, N., Sitnikova, E.: 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)
11. Koroniotis, N., Moustafa, N., Sitnikova, E., Slay, J.: Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques. In: International Conference on Mobile Networks and Management. pp. 30–44. Springer (2017)
12. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: 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)
13. Kumar, A., Lim, T.J.: EDIMA: Early Detection of IoT Malware Network Activity Using Machine Learning Techniques. In: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). pp. 289–294 (Apr 2019). https://doi.org/10.1109/WF-IoT.2019.8767194
14. Lima Filho, F.S.d., Silveira, F.A.F., de Medeiros Brito Junior, A., Vargas-Solar, G., Silveira, L.F.: Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. Security and Communication Networks **2019**, 1–15 (Oct 2019)
15. McDermott, C.D., Majdani, F., Petrovski, A.V.: Botnet Detection in the Internet of Things using Deep Learning Approaches. In: 2018 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (Jul 2018). https://doi.org/10.1109/IJCNN.2018.8489489
16. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y.: N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. IEEE Pervasive Computing **17**(3), 12–22 (Jul 2018)
17. Nõmm, S., Bahşi, H.: Unsupervised Anomaly Based Botnet Detection in IoT Networks. In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). pp. 1048–1053 (Dec 2018)
18. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
19. Sivanathan, A., Gharakheili, H.H., Loi, F., Radford, A., Wijenayake, C., Vishwanath, A., Sivaraman, V.: Classifying iot devices in smart environments using network traffic characteristics. IEEE Transactions on Mobile Computing **18**(8), 1745–1759 (2018)
20. Strasak, Garcia, S., Parmisano, A., Erquiaga, M.J.: Detecting malware even when it is encrypted - machine learning for network https analysis. Stratosphere Lab., Praha, Czech Republic, Tech. Rep (2017)
21. Yuan, X., Li, C., Li, X.: DeepDefense: Identifying DDoS Attack via Deep Learning. In: 2017 IEEE International Conference on Smart Computing. pp. 1–8 (May 2017)