Ultra-Lightweight and Secure Intrusion Detection System for Massive-IoT Networks

Roumaissa Bekkouche, Mawloud Omar, Rami Langar, Bechir Hamdaoui

Ultra-Lightweight and Secure Intrusion Detection System for Massive-IoT Networks

Roumaissa Bekkouche*, Mawloud Omar*, Rami Langar*‡, Bechir Hamdaoui§
* LIGM-CNRS UMR 8049, University Gustave Eiffel, F-77420 Marne-la-Vallée, France
‡ Software and IT Engineering Department, Ecole de Technologie Supérieure (ÉTS), Montréal, QC H3C 1K3, Canada
§ School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA
E-mails: {roumaissa.bekkouche, mawloud.omar, rami.langar}@univ-eiffel.fr; hamdaoui@eecs.oregonstate.edu

***Abstract*—The Internet of Things (IoT) is starting to integrate deeply into our daily lives thanks to the different services it provides. This technology has already made us more closely linked to the external environment through ubiquitous communication devices. However, even though this proximity has numerous benefits, it also has a significant security impact, where the cyber-attack surface has grown dramatically. In this regard, we present, in this paper, our results toward the development of a decision tree-based machine learning model for intrusion detection in Massive-IoT networks. The principal objective of this work is to provide a highly accurate detection model, while preserving resource consumption by developing a real prototype of the intrusion detection system. To this end, we first propose and apply our pre-processing methodology on the well-known Avast IoT-23 dataset, allowing us to reach a high detection rate with 99.99% of accuracy and just 1804KB of the model's size. Then, we propose a new machine learning model based on the decision tree classifier and deploy it in a real environment with malicious attack traffic. Obtained results show that our proposed model allows 88% of real-traffic-based precision rate and up to 90% of specificity.**

***Index Terms*—IoT, machine learning, network security, Avast IoT-23, anomaly detection, intrusion detection, decision tree.**

I. Introduction

The potential increase in mobile traffic, the expansion of communication infrastructures, and the massive use of Internet of Things (IoT) devices have led to crossing the capacities and limits of 4G networks. The latter no longer meets the actual requirements of users. A new generation of wireless mobile communication technology, 5G, is emerging to achieve user satisfaction and best meet user needs. One of the essential 5G's slices is Massive-IoT which offers tremendous benefits for end-users. 5G networks promise to meet the complex requirements of IoT architectures and infrastructures and support the massive number of devices connected simultaneously over the network. A study by Cisco [1] says the number of IoT devices will soon exceed 500 billion in 2030.

The IoT introduces new challenges for network security. Due to the lack of security awareness of devices and end-users, they have become vulnerable and the prime target for malware developers so far to infiltrate and attack a more extensive network. That is why the implementation of security measures is among the essential requirements. Recently, the researchers started looking into more advanced security measures to confront this issue. One of those new measures is the use of machine learning approaches to detect and classify attacks to mitigate them.

The use of machine learning is now attracting a lot of interest among academics. The most effective use of resources is enabled by artificial intelligence-powered security. The devices analyze the network traffic using more straightforward prediction methods, avoiding complex packet filtering approaches using traditional intrusion detection systems. Moreover, the establishment of a security service does not involve the deployment of particular network security hardware. Many datasets of attacks may be found in the literature. Unfortunately, most of them are not dedicated to IoT, and some emerged attacks are missing from the few IoT datasets available. Furthermore, their feature set is rather restricted. In the context of our study, we were interested in a very recent and comprehensible dataset; namely, Avast IoT-23 [6]. The data were gathered from several IoT devices, including 325307990 samples with 23 features. We meticulously pre-processed Avast IoT-23 dataset under which we experimented and implemented a prototype of the decision tree learning model. Our pre-processing approach allowed us to reach a high detection rate with just 1804KB of the model's size, making it strongly suitable for IoT devices. The resulting model has been then implemented in a real environment with malicious attack traffic. Obtained results show that our proposed model allows 88% of real-traffic-based precision rate and up to 90% of specificity.

The rest of the paper is organized as follows. In Section II, we review the literature. In Section III, we describe the approach we use in data pre-processing. In Section IV, we describe our intrusion detection system. In Section V, we present the experimental setup and the evaluation methodology, we present also in this section the result achieved and the performance evaluation. Finally, Section VI concludes this work by the perspectives and future research ideas.

II. Literature review

Several works are addressing the conception of machine learning approaches for network security. In this section, we present some related works.

Authors in [10] discussed the idea of using machine learning algorithms to secure IoT networks. They showed the poten-

tial of these approaches by focusing on the deployment of supervised, unsupervised, and reinforcement learning for both host-based and network-based security solutions for the IoT environment. The "host-based" is addressed using the data held by the devices, and the "network-based" is addressed using the metadata of IoT network. They also discussed the strength of machine learning techniques to enable more efficient IoT protection. According to the authors, machine learning algorithms to secure IoT devices are limited by the unique characteristics of IoT and their environments. The challenge is to develop novel, cost-effective, and scalable machine learning techniques to meet the IoT ecosystem's computation power and environmental constraints.

Authors in [9] performed a comparison between different machine learning algorithms applied to Avast IoT-23 with the purpose to secure the IoT network and identify the best model for anomaly detection. They examined the following machine learning algorithms: Random Forest, Naive Bayes, Multi-Layer Perceptron, Artificial Neural Network, Support Vector Machine, and AdaBoost. They provided a comparison between the performance of each algorithm in terms of precision, recall, and f1-score. The obtained results showed that the random forest algorithm had the best performance with 99.5% of accuracy.

Another comparative study of machine learning algorithms is done by Mehmood et al. in [5]. In their work, they employed the dataset KDD99, which is a benchmark for anomaly-based detection technique that contains four attack classes. The authors experimented: Support Vector Machine, Naive Bayes, Decision Tree, and Decision Table. The obtained results show that not a single algorithm has a high detection rate for each class of KDD99 dataset. But the overall accuracy of the decision tree is high among all the other algorithms with a low misclassification rate.

The authors of [3] tested a new solution for intrusion detection problem. In their work, they combined two machine learning algorithms: Decision Tree (DT) and Multilayer Perceptron (MLP) to identify attacks with high accuracy and reliability. To do that, they have used the KDD CUP 99 dataset.Their solution consists of two phases: in the first phase, they have used both MLP and DT on the original dataset to create a new dataset of the predicted values obtained by MLP and DT; in the second phase, they used the new dataset with another MLP network to classify data on attack or normal traffic according to the results of the first phase.To evaluate their hybrid method, they have used False Alarm Rate (FAR), Accuracy (A), and Detection rate (DR).

The work of [8] addresses the problem of features selection for machine learning algorithms in the context of anomaly detection in the Internet of things network. They proposed a new framework that allows to select an effective input feature set and remove the unwanted features from the dataset used by machine learning models. They have used Bot-IoT dataset and four different ML algorithms, namely: C4.5, Naive Bayes, Random forest and SVM to test and validate their solution. Experimental results analysis showed that their solution can achieve higher than 96% results on average.

The authors of [2] proposed a tool to detect cyber-attacks and protect IoT devices that are directly connected to it from various types of malicious traffic, including Port Scanning, HTTP, SSH Brute Force, and SYN Flood attacks. The proposed solution, named Passban, is a platform-independent anomaly-based IDS that can be deployed directly on IoT gateways and learn from the normal behavior of incoming IoT traffic. Passban comprises five main components: Packet Flow Discovery block, Feature Extraction block, Train/Load model block, Action Manager procedure, and Web Management Interface. The Packet Flow Discovery block is used to capture packets and extract network flows. Then, the Feature Extraction block will calculate network flow statistics and build a feature set. The Train/Load model block is responsible for the training of the IDS model using the normal network flow. The learned model is stored on the gateway and used to detect attacks. The Action Manager procedure provides the protection strategies according to the attack classification. And finally, Web Management Interface provides a user interface to manage and explore the IDS parameters.

III. Dataset Description and Pre-processing

The dataset used for this study is Avast IoT-23 [6]. It is a new dataset created as part of the Avast AIC laboratory with the funding of Avast Software. Avast IoT-23 is a labeled dataset with malicious and benign traffic captured using IoT devices. It consists of 23 captures (called scenarios) executed in IoT devices with 20 captures for malicious traffic and 3 captures for benign traffic. The full IoT-23 dataset contains the original capture files and the log files of Zeek network analyzer. It contains labeled data and other files containing different information about each capture.

A. Data Visualization and Encoding

The database consists of 23 features, where 21 of them represent the flow characteristics. In Table I, we give the description of each feature. Indeed, the flaws of certain protocols make the network vulnerable to attacks. However, we notice that the protocol feature is of nominal type. Due to the importance of this feature, we re-encoded it as an integer: 0 for UDP, 1 for TCP, and 2 for ICMP.

To provide a more detailed information to network malware researchers and analysts, this dataset contains labels to describe the relation between flows related to (possible) malicious activities. These labels were created in the stratosphere laboratory considering the malware captures analysis [6]. In Table II, we give the description of the labels used for malicious flows detection.

B. Data Pre-processing

Data pre-processing is an important phase in machine learning that enhances the quality of data and the learning process. In this step, we clean, format, and organize the data

Feature	Description
ts	Time when the capture was done.
uid	Capture's ID.
id.orig_h	Sender's IP address.
id.orig_p	Sender's port.
id.resp_h	Receiver's IP address.
id.resp_p	Receiver's port.
proto	Used protocol (TCP, UDP or ICMP).
service	Application protocol.
duration	Packet exchange duration.
orig_bytes	Amount of sent data.
resp_bytes	Amount of received data.
conn_state	Connection state (specific variable used by ZEEK).
local_orig	True if the sender is within the local network.
local_resp	True if the receiver is within the local network.
missed_bytes	Number of missed bytes in the capture.
history	History of the connection state.
orig_pkts	Number of sent packets to the device.
orig_ip_bytes	Number of sent bytes to the device.
resp_pkts	Number of sent packets from the device.
resp_ip_bytes	Number of sent bytes from the device.
tunnel_parents	Connection's ID (if tunnelled).
label	Attack or benign traffic.
detailed-label	Attack type.

TABLE I
FEATURES DESCRIPTION

Label	Description
benign	No suspicious or malicious activity.
Attack	Suspicious or malicious activity.
C&C	Device connected to a Command and Control server.
HeartBeat	Sent packets are used to keep a track on the infected host by the C&C server.
Mirai	Connections have the characteristics of Mirai botnet.
Torii	Connections have the characteristics of Torii botnet.
PartOfA-Horizontal-PortScan	Indicates that the connections are used to perform a horizontal port scan to gather information and carry-out further attacks.
Okiru	Connections have the characteristics of Okiru botnet.
DDoS	Distributed Denial of Service attack is being executed by the infected device.
FileDownload	A file is being downloaded into the infected device.

TABLE II
LABELS DESCRIPTION

to get the most pertinent characteristics that improve the model ability in terms of attack detection. To do so, we performed the following steps:

- *Discarding additional and nominal attributes:* we were interested in the attack category. Thus, we discarded the label attribute since it represents redundant information regarding the detailed-label attribute. We also discarded uid, $history$, $id.orig\_h$ and $id.resp\_h$, which are of nominal type.
- *Discarding specific attributes:* we discarded the attribute ts, which is a specific feature related to Wireshark capture. Moreover, $conn\_state$ is a specific variable of Zeek (network analysis framework) that represents the connection state between two devices. The targeted model has to be generic and applicable for any network configuration. Therefore, we discarded $conn\_state$.
- *Discarding missing values:* in this step, we discarded $tunnel\_parents$, $local\_orig$, $local\_resp$, $service$ and $missed\_bytes$, where more than 80% of their values are missed, and hence, do not contribute to the classification.
- *Discarding correlated features:* we identified the highly correlated features that degrade the detection capability of the target model by discarding them. We used a correlation coefficient of 0.9. The two features that are highly correlated are $orig\_ip\_bytes$ and $resp\_ip\_bytes$.

IV. The Proposed Intrusion Detection Technique

The proposed approach is based on the decision tree classifier. It is a supervised algorithm based on a set of decision rules inferred from the data structure. Decision trees can analyze data and identify important features in the network that indicate malicious activities. We have chosen the decision tree as the classification algorithm for the many advantages of our use case. It is an efficient classifier with a low cost compared to the others regarding computation time and resource consumption.

A. Learning Phase

The outcome of this phase is the attack prediction model. Our goal is to create a model that accurately predicts the value of a target variable based on several input variables. To do so, as a first step, the dataset is randomly split into two parts: 80% for the training phase and the remaining 20% of the data is used to test the trained model.

B. Attacks Prevention Mechanism

After training the model and validating it based on performance metrics that we will present in Section V-B, we implement it as a new intrusion detection system in a real environment. Our proposed attack prevention mechanism, described in Algorithm 1, goes through four steps, as follows:

1) The program sniffs the network and recovers the incoming/outgoing packets to/from the machine. This process is executed periodically by storing continuously the temporary PCAP (Packet Capture, file format used by Wireshark to save captured packets) file.

Algorithm 1 Attacks prevention mechanism

/*Sniffing the network*/
$Sessions \leftarrow getSessionsInfo()$
/*Getting sessions information*/
for each s in Sessions **do**
$protocol \leftarrow s.protocol$
$source\_address \leftarrow s.source\_address$
$destination\_address \leftarrow s.destination\_address$
$source\_port \leftarrow s.source\_port$
$destination\_port \leftarrow s.destination\_port$
end for
/*Features extraction*/
for each packet in Sessions.packets **do**
if packet.source_address = s.source_address **and** packet.destination_address = s.destination_address **and** packet.source_port = s.source_port **and** packet.destination_port = s.destination_port **then**
$nb\_packets\_sent \leftarrow nb\_packets\_sent + 1$
$amount\_data\_sent \leftarrow amount\_data\_sent + packet.length$
else if packet.source_address = s.destination_address **and** packet.destination_address = s.source_address **and** packet.source_port = s.destination_port **and** packet.destination_port = s.source_port **then**
$nb\_packets\_received \leftarrow nb\_packets\_received + 1$
$amount\_data\_received \leftarrow amount\_data\_received + packet.length$
end if
$connexion\_duration \leftarrow connexion\_duration + packet.duration$
end for
/*Model loading and prediction*/
$model \leftarrow loadModel()$
$model.predict(extractedFeatures)$

2) We retrieve the connection sessions information from the temporary file: the source IP address, destination IP address, source port, destination port, and communication protocol.
3) For each connection, we calculate the number of sent/received packets, the amount of sent/received data as well as the connection duration.
4) Once all the features are extracted, we pre-process every flow by removing the IP addresses and encoding the communication protocol.
5) Finally, the last step consists of loading the trained model to classify and perform prediction.

C. Blacklist Mechanism

To further improve the prediction time and the quality of the results, we set up a blacklist mechanism, which consists of storing information about detected attacks to be able, for future connections, to block connections from the same IP address directly without going through our prediction model, as illustrated in Figure 1.

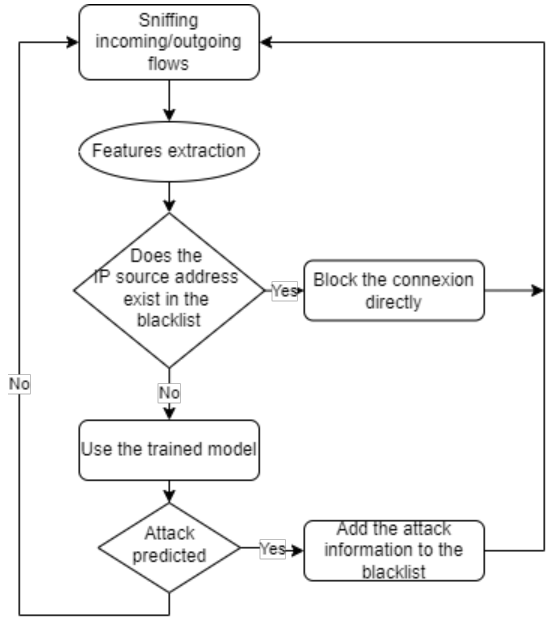

Fig. 1. Blacklist mechanism

V. Experimental Setup and Evaluation Results

As stated earlier, we used the Avast IoT-23 dataset [6] in our experimentation. This dataset was split into two parts: 80% for training and 20% for testing. Once the model is trained, we evaluate the trained model following two steps: first, we use the test dataset to validate our model based on several performance metrics (see Section V-B), then we evaluate the validated model with real-time traffic.

The experiments were conducted on a machine running under Windows 10 Professional 64bits, with processor Intel(R) Core(TM) $i5$-10310U CPU @ 1.70GHz 2.21GHz, 16GO of Memory, and Intel(R) UHD Graphics for the graphic card. The algorithms were implemented using Sklearn of python. Pycharm and Scapy libraries were used in the prototype implementation.

In what follows, we start by describing our test environment used to validate our model. Then, we present the obtained performance results.

A. Test Environment

To test our model with real-time traffic, including benign and malicious traffic, we deployed a network of virtual machines under VirtualBox to emulate IoT devices. The network is composed of 4 machines. We have implemented a prototype of the model on a machine running under Ubuntu 20.04.3 (64 bits) and set 3 machines to generate traffic on the network, as illustrated in Figure 2.

1) Model deployment: We deployed the anomaly detection system based on the trained model on the Ubuntu machine to test and visualize the model's performances under real-time conditions, with real-time traffic.

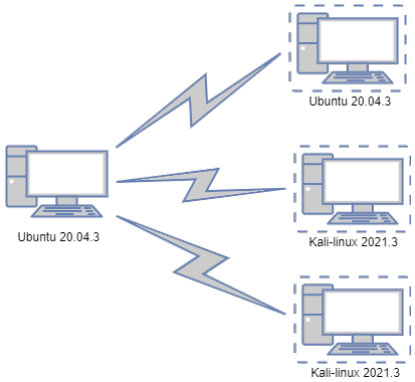

Fig. 2. Test environment

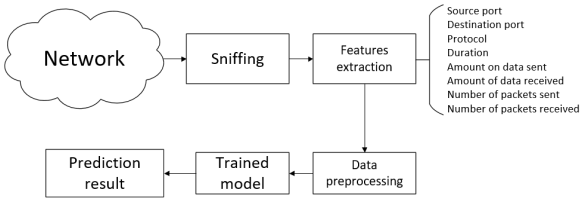

Fig. 3. Overview of the developed prototype

As illustrated in Figure 3, we retrieve the incoming/outgoing flows from which we extract the features, perform the preprocessing and predict the attack existence, following our proposed model presented in Section IV.

2) Traffic attack generation: To generate real network traffic, Mausezahn [4], and HPing [7] tools were deployed on the 3 attackers' machines. Both tools are traffic generators, which allow sending nearly every possible and impossible packet. We used several parameters to generate various attack scenarios based on flooding (DDoS, PortScan, etc.). The attack is started from the 3 attackers and the results are recorded on the target machine implementing the intrusion detection system. An example of using these tools to simulate attacks are shown in Figure 4.

```
machine@machine-VirtualBox:~$ sudo mz enp0s8 -A rand -B 192.168.0.3 -c 0 -t tcp
 "dp=1-1023, flags=syn"
Mausezahn will send frames infinitly...
```

Fig. 4. Example of attack simulation

B. Performance Results

Several performance metrics are used to evaluate the trained model: recall, specificity, precision, and accuracy. We define in the following each metric:

- *Recall:* or true positive rate (TPR), also known as sensitivity. It is computed as follows:

$$Recall = \frac{TP}{TP + FN},$$

where TP represents the number of actual positives that were correctly identified, and FN is the number of actual positives that were identified as negatives.

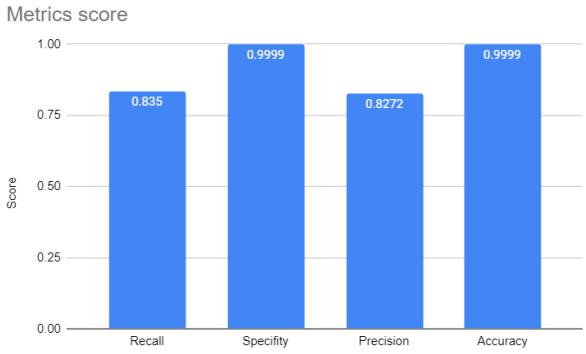

Fig. 5. The average metrics score of the trained model using the test dataset

- *Specificity:* or true negative rate (TNR), which is computed as follows:

$$Specificity = \frac{TN}{TN + FP},$$

where TN represents the number of actual negatives that were correctly identified, and FP is the number of actual negatives that were identified as positives.

- *Precision:* or positive predictive values (PPV), which is computed as follows:

$$Precision = \frac{TP}{TP + FP}.$$

- *Accuracy:* is the ratio of correctly predicted observations to the total observations. It is computed as follows:

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

The obtained results are illustrated in Figure 5. As can be seen in this figure, our model performs a high classification rate for both benign traffic and attacks with an error rate of 0.01%. Moreover, it shows promise in terms of other evaluation metrics. High recall (83.5%) refers to the ability of the model to detect benign traffic correctly. On the other hand, high specificity (99.99%) refers to the ability of the model to detect the attacks correctly. Finally, the high precision (82.72%) means that only few positives are detected as attacks.

Table III, summarized in Figure 6, shows the performance details of these metrics. As it is a multi-class classification problem, we have computed for each class the TP, TN, FP and FN rates. The reported values are the average of the obtained results regarding all classes.

C. Results on Real-Time Traffic Conditions

Under the test environment, we tested our model, which is deployed on the first machine of benign traffic and generated malicious traffic. The obtained results demonstrate the ability of our model to distinguish efficiently between benign and malicious traffic with 88% of precision rate and up to 90% of specificity, as mentioned on Table IV.

One of the strong points of the system is its ability to detect attacks from the first attacking flows, and with the use of

Metric	PartOf AHoriz ontal Port Scan	Okiru	DDoS	Benign	Attack	C&C	C&C: Heart-Beat	C&C: PartO-fA-Hor-izon-tal-PortScan	C&C: Heart-Beat Attack	File Down-load	C&C: Heart-Beat File Down-load	C&C: Torii	C&C: File Down-load
TP	1833	2326	9	4309	132	1	0	2	3.12e6	4	2.73e6	1.12e7	8.44e5
TN	1.79e7	1.79e7	1.79e7	1.79e7	1.79e7	1.79e7	1.79e7	1.79e7	1.48e7	1.79e7	1.52e7	6.71e6	1.71e7
FP	53	8	1	1	30	0	3	2	12	3	5	78	24
FN	41	9	2	1	41	1	173	0	12	1	0	11	28
Recall	0.978	0.996	0.818	0.999	0.763	0.5	0	1	0.999	0.8	1	0.999	0.999
Specificity	0.999	0.999	0.999	0.999	0.999	1	0.999	0.999	0.999	0.999	0.999	0.999	0.999
Precision	0.971	0.996	0.9	0.999	0.814	1	0	0.5	0.999	0.571	0.999	0.999	0.999
Accuracy	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999	0.999

TABLE III
RESULTS OF THE TEST DATASET

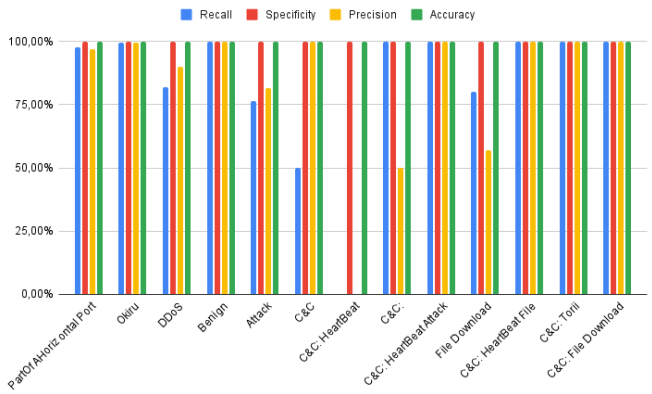

Fig. 6. Metrics score for each class of the test dataset

the blacklist mechanism, the system directly blocks the next attacking flows and exploit its past experience to predict future attacks. The prediction time varies between 1.35 sec to 6 sec with a model size of 1804KB, which is lightweight enough to be supported by IoT environment.

Metric	**Score**
Precision rate	88%
Specificity	90%
Prediction time	1.35sec to 6sec

TABLE IV
RESULTS ON REAL-TIME TRAFFIC

VI. CONCLUSION AND PERSPECTIVES

In this paper, we proposed and deployed our decision tree machine learning model toward the design of a resource-aware intrusion detection system for IoT. The Avast IoT-23 served as the basis for our research under which we trained and deployed our model. Obtained results demonstrate the efficiency of our proposed approach. In particular, the model's size is 1804KB, which fits the imposed constraints and makes it strongly suitable for IoT. The developed model is tested and proved its effectiveness even under real-time conditions and with traffic containing benign flows and malicious flows. The obtained results are promising and motivate further research to refine the model by improving its detection rate and optimizing its prediction time.

The most important next step is the design of the security-as-a-service process. The service should be dynamically migrated to vulnerable IoT devices based on their requirements in terms of protection and hardware capability. The security process will be integrated with a trust-based orchestration paradigm to be deployed in 5G for Massive-IoT slices. The final phase is to design a distributed experience exchange protocol to make the devices smarter at detecting new attacks.

ACKNOWLEDGEMENT

This work was partially supported by the ANR 5G-INSIGHT project (Grant no. ANR-20-CE25-0015).

REFERENCES

[1] CISCO. Internet of things. https://www.cisco.com/c/en/us/products/collateral/se/internet-of-things/at-a-glance-c45-731471.pdf, 2016.
[2] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli. Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices. *IEEE Internet of Things Journal*, 7(8):6882–6897, 2020.
[3] J. Esmaily, R. Moradinezhad, and J. Ghasemi. Intrusion detection system based on multi-layer perceptron neural networks and decision tree. In *2015 7th Conference on Information and Knowledge Technology (IKT)*, pages 1–5. IEEE, 2015.
[4] H. Haas and D. Borkmann. mausezahn(8) — linux manual page. https://man7.org/linux/man-pages/man8/mausezahn.8.html, 2013.
[5] T. Mehmood and H. B. M. Rais. Machine learning algorithms in context of intrusion detection. In *3rd International Conference on Computer and Information Sciences (ICCOINS)*, pages 369–373. IEEE, 2016.
[6] A. Parmisano, S. Garcia, and M. Erquiaga. A labeled dataset with malicious and benign iot network traffic. *Stratosphere Laboratory: Praha, Czech Republic*, 2020.
[7] S. Sanfilippo. hping3(8) - linux man page. https://linux.die.net/man/8/hping3.
[8] M. Shafiq, Z. Tian, A. K. Bashir, X. Du, and M. Guizani. Corrauc: a malicious bot-iot traffic detection method in iot network using machine-learning techniques. *IEEE Internet of Things Journal*, 8(5):3242–3254, 2020.
[9] N.-A. Stoian. Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set. B.S. thesis, University of Twente, 2020.
[10] S. Zeadally and M. Tsikerdekis. Securing internet of things (iot) with machine learning. *International Journal of Communication Systems*, 33(1):e4169, 2020.