

A OneM2M Intrusion Detection and Prevention System based on Edge Machine Learning

Nadia Chaabouni

LaBRI, Univ. Bordeaux, Bordeaux INP and CNRS France
Atos Innovation Aquitaine Lab
Pessac, France
chaabouni.nadia14@gmail.com

Akka Zemmari

LaBRI, Univ. Bordeaux, Bordeaux INP and CNRS France
Talence, France
zemmari@u-bordeaux.fr

Mohamed Mosbah

LaBRI, Univ. Bordeaux, Bordeaux INP and CNRS France
Talence, France
mosbah@u-bordeaux.fr

Cyrille Sauvignac

Atos Innovation Aquitaine Lab
Pessac, France
cyrille.sauvignac@atos.net

Abstract—As Internet of Things (IoT) is widely spread and is becoming heterogeneous, a growing number of connected devices are the focus of security threats. Hence, a standardized security strategy seems required. OneM2M [1] is a global standard initiative designed to satisfy the need for a common horizontal platform for the multi-industry M2M/IoT applications. In this paper, we propose an Intrusion Detection and Prevention System (IDPS), for the Service Layer introduced by the oneM2M standard. To our knowledge, it is the first generic IDPS for the oneM2M Service Layer based on Edge Machine Learning (ML). We will detail, in this work, the strategy of the oneM2M-IDPS. Moreover, we investigate the performance of ML algorithms on the oneM2M generated dataset to choose the best ones for our IDPS. Since we are in the context of tiny devices (IoT), we pay attention in our experiments to the features dimension reduction in ML and thus, to the size of trained models.

Index Terms—Internet of Things, IoT, Standardization, OneM2M Standard, Security, Intrusion Detection, IDS, Intrusion Prevention, IPS, IDPS, Machine Learning, ML, Deep Learning, DL

I. INTRODUCTION

Internet of Things (IoT) devices are being increasingly deployed in every field such as in healthcare, transportation, industrial automation, smart homes, etc. Consequently, each industry from each sector relies on proprietary solutions with customized hardwares and softwares for Machine-To-Machine (M2M) systems hence Internet of Things (IoT) applications. To avoid such vertical, mono-industry solutions, eight of the world's leading ICT standards bodies [1] initiate the international partnership project oneM2M in 2012.

OneM2M standard enables the communication between heterogeneous devices and applications by defining a common M2M Service Layer for the multi-industry M2M applications. OneM2M standardizes the IoT ecosystem with respect to the existing worldwide networks and standards.

Therefore, securing the IoT ecosystem can be done through securing the oneM2M standard. We propose in this paper, the first generic Intrusion Detection and Prevention System (IDPS)

for the oneM2M standard. An IDPS [2] is a system which detects and responds immediately to potential threats as soon as they are carried out.

Nowadays, Machine Learning (ML) algorithms are widely used in the intrusion detection systems (IDS) [2] as well as in the Intrusion Prevention Systems (IPS) thanks to their good detection results compared to the traditional systems [2]. In [3], we have proven that with an edge machine learning IDS based on a decision tree algorithm (J48 [4] in Weka tool is an open source Java implementation aiming to generate a pruned or unpruned decision tree), we are able to attend 95.40% of detection rate with 12% of false classifications for a binary classification. We have studied different shallow machine algorithms in [3] (NaiveBayes (NB) [5], MultilayerPerceptron (MLP) [6], J48 [4], RandomForest (RF) [7] and SMO [8]) for binary and multi-class classification. Unfortunately, binary classification in [3] suffers from high false positive rate (FPR) which represents benign inputs predicted as attacks. High FPR values increase the resource consumption as well as the computation time which is inappropriate especially in the context of IoT (tiny devices). Moreover, multi-classification experiments in [3] show low performance of the IDS. Consequently, in this work, we change the IDS strategy proposed in our previous work [3] to be based on three levels of ML intrusion detection in order to improve the attacks detection in IoT systems. Furthermore, we enrich the IDS with a prevention strategy and continuous learning module to design an appropriate oneM2M-IDPS for the IoT context.

The remainder of the paper is organized as follows: Section II presents the state of art of IoT IDPS as well as the existing oneM2M security mechanisms. Moreover, we describe our IDPS strategy and modules in Section III. Last but not least, we experiment machine learning algorithms in Section IV. Finally, we close this paper with a conclusion and details about future works.

II. THE STATE OF ART

This section starts by presenting the state of art of IDPS for the IoT ecosystem. Then we detail the security mechanisms proposed in the specifications of the oneM2M standard. The goal of oneM2M is to define, approve and maintain a number of requirements and specifications to standardize the M2M and IoT ecosystem.

A. State of art of IDPS for IoT

To the best of our knowledge, the state of art does not contain IDPS proposals for the IoT systems. However, many IoT IDS have been proposed [2]. They vary according to the analyzed protocols, the deployment strategies, the detection methodologies as well as the targeted threats. The first network IDS for IoT was introduced by Raza et al. [9]. They concentrate on IPv6-connected IoT devices. Their solution is based on signature and anomaly detection techniques and treats the routing attacks with a distributed deployment. Kasinathan et al. [10], [11] propose a centralized signature-based IDS to detect DoS (Denial of Service) attacks at the 6LoWPAN protocol (an adaptation layer for low-power lossy networks). Cervantes et al. [12] tend also to protect the 6LoWPAN networks against the sinkhole attacks with a distributed IDS. Their detection methodology is based on trust and reputation strategy. Hosseinpour et al. [13] present a novel real-time, distributed and lightweight IDS based on Artificial Immune System (AIS) in an effective combination of edge, fog and cloud computing. The IDS detects botnet attacks. Bostani and Sheikhan [14] suggest a real-time hybrid of anomaly-based and specification-based IoT IDS. It enables the detection of sinkhole and selective-forwarding attacks in 6LowPAN networks. As a protocol independent IDS, Midi et al. [15] propose a hybrid IDS (signature and anomaly based) to detect DoS and conventional network attacks. Moreover, Thamilarasu and Chawla [16] develop an Integrated Intrusion-Detection (IID) system based on a deep learning algorithm. Their anomaly based proposal operates independently of IoT protocols and network structure, and requires no prior knowledge of security threats. Many other works present IDS for IoT as discussed in [2] but none of them treat the oneM2M standard.

B. OneM2M Security

Among the specification documents of the oneM2M standard, the oneM2M community focuses on security and privacy aspects in TS-0003 [17]. OneM2M contributions in terms of information security seem to be an interesting step [18] to provide a sustainable development for both M2M and IoT applications while facing the challenges of M2M security and privacy.

The oneM2M security strategy is based on six main categories: a) identification and authentication, b) authorization, c) identity management, d) security association, e) sensitive data handling and f) security administration. To start with, identification (a) is the process of verifying the validity of the identity that asks to authenticate (e.g. if an AE (Application Entity) or CSE (Common Services Entity) fits a certificate).

Then comes the authentication step (a) where the validated identity needs to be associated to a trustworthy credentials (for example in the case of certificate authentication, it is a digital signature that needs to be checked). In oneM2M, it is possible to use a centralized key distribution server hosted by a 3rd party or by M2M Service Provider with a symmetric key access. In the second place, authorization function (b) regulates services and data access authorizations for already authenticated entities. This security mechanism is based on Access Control Policies (ACPs) which are sets of conditions that reflect permitted accesses and role based access control which can be a token based framework (e.g. OAuth). Moreover, identity management (c) guarantees the anonymity of the entities in the oneM2M systems. In fact, oneM2M provides pseudonyms that play the role of temporary identifiers to protect the true identity of the nodes which is considered as sensitive data in the oneM2M architecture [19]. This mechanism is used independently of authentication and authorization functions. Furthermore, security association (d) is about equipping communicating entities with security services such as confidentiality and integrity of the exchanged information thanks to keys provided during the identification an authentication phase (a). Hence, the exchanged contents of resources are encrypted. As an example, at the service layer, exchanged messages between adjacent AE/CSE can be protected with a TLS or DTLS session. Last but not least, the sensitive data handling service (e) is mainly for the security of the application layer. It provides sensitive functions which enable secure storage, cryptographic operations and bootstrapping methods for initial secrets (e.g. GBA). Consequently, an isolated secure environment is guaranteed for data storing and retrieving such as credentials, subscriptions, personal information and for functions including security algorithms. Finally security administration (f) is introduced to manage all of the sensitive resources (data and functions) as well as to configure and extend the security services themselves.

Indeed oneM2M provides a large panel of security mechanisms to protect the service layer itself as well as the communication between the oneM2M architecture layers. However, none of the specified techniques protect the IoT systems once the malicious user has gained access to the system and bypassed the first-line security measures. Therefore, a second security line is required to detect and prevent intrusions from affecting the IoT systems. In [3], we proposed a distributed oneM2M-IDS architecture based on edge ML (one-layer of ML detection). We experimented with different shallow ML algorithms and ended up with the decision tree algorithm (J48) that achieves the best performances. It attends 95.40% of detection rate with 12% of false classifications for the binary classification and respectively 77.07% and 25.02% for the multi-classification.

III. THE PROPOSED SOLUTION

Due to the limitation of the security techniques presented in Section II, in terms of detection and prevention of the IoT systems threats, we decide to propose an IDPS for the

oneM2M service layer based on machine learning. In this work, we change the IDS strategy proposed in our previous work [3] to be based on three levels of ML intrusion detection. Furthermore, we enrich the IDS with a prevention strategy and continuous learning module to design an appropriate oneM2M-IDPS for the IoT context.

The oneM2M-IDPS has the same distributed architecture as the oneM2M-IDS proposed in [3]. The IDPS is integrated in powerful IoT devices (Application Service Nodes (ASN) and not Application Dedicated Nodes (ADN)), gateways and servers. Technically, it is included in each (ASN), Middle Node (MN) and Infrastructure Node (IN) having a CSE layer. Each IDPS instance analyzes its own exchanged flows which leads to an edge lightweight processing suitable for the IoT systems that have challenges related to resource constraints.

Regarding the detection techniques, we use ML algorithms [3] for: i) their ability to have better detection for zero-days attacks; ii) their effectiveness in detecting the threats variations; and iii) their efficiency in terms of high detection rate and low false alarms. This oneM2M-IDPS can be added to the basic architecture of the oneM2M standard or can be added as a plugin to existing systems based on the oneM2M standard.

A. OneM2M-IDPS Strategy

An IDPS is mainly composed of two phases: i) the detection phase and ii) the response phase. An IDPS has the ability of the attack controlling in real time. In this paper, we propose an intelligent oneM2M-IDPS with the following objectives:

- the detection of IoT threats whether they are security attacks or abnormal behaviors that could lead to the IoT system bug or shutdown. This intelligent detection should not only identify oneM2M threats already seen, but also be able to learn unknown threats all the time.
- the prevention against the detected threats through immediate and appropriate actions. It needs to be an active, real time prevention to assure the oneM2M services availability.

The oneM2M-IDPS strategy is composed of four main modules as shown in Figure 1; i) the data acquisition and features extraction module where the oneM2M messages are prepared to be injected to the IDS module, ii) the IDS module to detect the threats, iii) the IPS to prevent the threats from damaging the system and iv) the continuous learning and human annotation module to guarantee the update of the IDPS against the new threats. We will detail each module in the rest of this section.

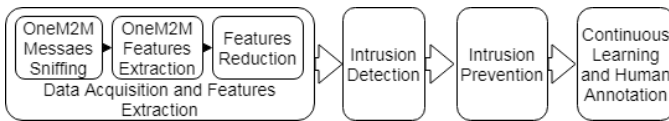


Fig. 1: OneM2M IDPS Strategy

B. Data Acquisition and Features Extraction

This module captures the oneM2M messages to be analyzed, processed and enriched. It sniffs the oneM2M flow composed of a pair of Request and Response messages as mentioned in the oneM2M specifications [20]. Each flow is composed of mandatory and optional parameters depending on the requested operation. After having captured this flow, we extract and generate the needed features for the machine learning detection. These features constitute the *GFlow*, a new abstraction for the oneM2M messages introduced in [3] based on the mandatory parameters. A *GFlow* encompasses multiple oneM2M flows on the basis of a key composed of the request originator, the request receiver, the type of the requested operation and the status of the response. For each NB exchanged flows, a set of *GFlows* is generated. A *GFlow* is composed of 26 features as presented in [3].

Features for ML algorithms should be chosen carefully and accurately to avoid inaccuracy and unnecessary time consumption especially in the context of IoT where devices are limited in resources. Therefore, for a better resource and time consumption as well as a lighter feature extraction phase, irrelevant features must be removed from the set. Such a process is known as features reduction or dimensionality reduction [2]. One of the used techniques for the features reduction is the entropy computation [21]. This technique is processed by the decision trees algorithms. Tree based models calculate feature importance to keep the best performing features as close to the root of the tree. Regarding deep learning algorithms, the feature reduction is automatically performed.

C. Intrusion Detection

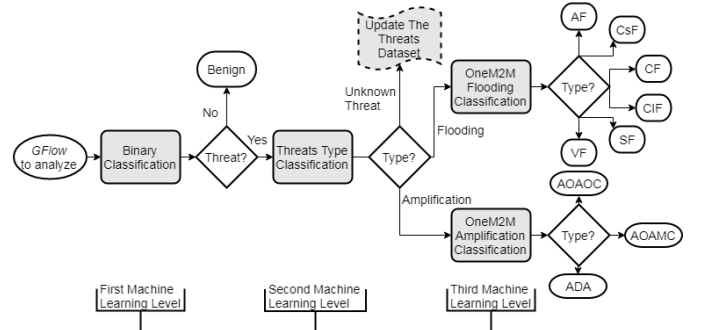


Fig. 2: Flow Chart Detection

The data acquisition and the features extraction phase has generated a *GFlow* which is the input for the detection phase. To guarantee an accurate and light detection, we suggest a three level of machine learning detection as shown in Figure 2. First, the *GFlow* is analyzed in a binary classification. It is either a benign input or a threat. If it is a benign *GFlow*, no action is taken. However, if it is a threat to the IoT system, the *GFlow* needs to be analyzed more deeply to identify the exact type of the threat. This step is at the second level of detection which is based on a semi-supervised learning. This entry can be classified as one of the main classes of oneM2M attacks presented in [3] (oneM2M flooding, oneM2M amplification

and oneM2M protocol exploit). Otherwise, this *GFlow* is considered as a new threat that will be analyzed and added to the threats database in the next step of the IDPS flow. To conclude, if the *GFlow* is classified as oneM2M flooding or amplification, further classification needs to be carried out to identify the exact type of the attack. Such a three-layer detection allows the analysis to be stopped as soon as possible, especially for benign inputs, which represents an interesting strategy for a better resource consumption in the context of IoT.

To have an ML detection, a training phase is needed before embedding the ML models in the IoT devices. The first and the third ML levels are based on supervised ML algorithms which means that the training of the model is based on a labeled dataset. We use the previously generated oneM2M dataset described and tested in [3]. It was generated using the Atos Standardized IoT Platform (ASIP) initiated in May 2017 by Atos Innovation Aquitaine Lab. The dataset is composed of 223273 of *GFlow* inputs (165253 lines of attacks and 58020 of benign *GFlows*). We summarize the number of *GFlow* generated for each type of attack. About the flooding sub-types, the dataset is composed of 22258 of AF (Ae Flooding), 18867 of CsF (Containers Flooding), 13489 of CF (Container Flooding), 9569 of CIF (ContentInstance Flooding), 14922 of SF (Subscription Flooding) and 25356 of VF (Various Flooding). Regarding the amplification sub-types, we have 24449 of AOAOC (Amplify One Ae One Container), 17865 of AOAMC (Amplify One Ae Multiple Containers) and 18478 of ADA (Amplify Discovery Ae). For the first level of training, we use the benign *GFlow* and we rename the labels of all sub-attacks as "threat" to have a binary classification. However, for the third level of training, we use only the sub-types of each attack for each model training. In other words, we use only the sub-types of flooding for the oneM2M flooding training and the amplification sub-types for the oneM2M amplification training.

Regarding the second level of machine learning which is about the threat types classification, we decide to use a semi-supervised learning. Such machine learning algorithms enables the detection of known and unknown threats which is one of the main goals of our IDPS. To do that, five steps are needed:

- 1) remove all the dataset labels
- 2) apply an unsupervised machine learning algorithm to get 3 clusters
- 3) re-associate the labels for the known threats
- 4) assign the labels to the known clusters: flooding and amplification
- 5) hence, the third cluster corresponds to unknown threats which will be furthermore examined in the next module.

D. Intrusion Prevention

To have the best prevention against an identified threat, human annotation is needed. For the already known threats such as flooding and amplification, the prevention actions are already defined and associated to the attacks when implementing the IDPS. Once a *GFlow* is identified as an unknown threat

further examination is needed. If the *GFlow* does not exist already in the threats database, then the dataset will be updated with this flow, an alert will be launched to capture the attention of the IoT human operator and an annotation procedure of the *GFlow* as well as the definition of the associated prevention action need to be performed. The flow chart of the prevention procedure is detailed in Figure 3.

To guarantee efficient continuous learning, human annotation is needed. To help human intervention, more information are provided for more precision. The *GFlow* is enriched, on the one hand, by the source and destination addresses as well as the timestamp of the incident. On the other hand, data related to the device resource consumption are displayed. For example in gateways, we report the disk, the memory, the network, the processes, the processor and the usb ports consumption / use.

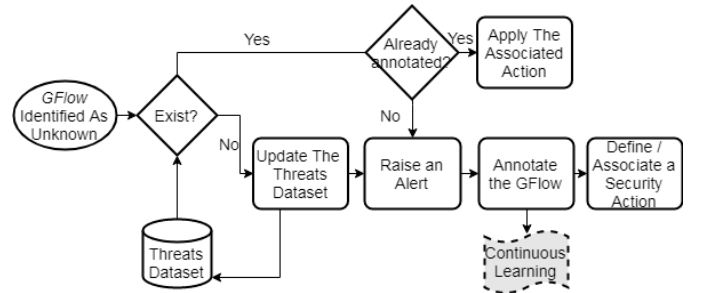


Fig. 3: Flow Chart Prevention

E. Continuous Learning and Human Annotation

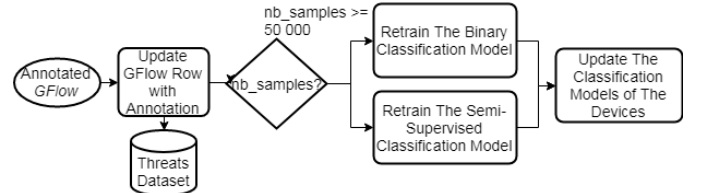


Fig. 4: Flow Chart Continuous Learning

To have a continuous learning after the human annotation, the machine learning models of the first and second level in the Subsection III-C need to be updated to take into consideration the new already-seen threats. To retrain the ML models, enough data needs to be available. To have a balanced dataset (Figure 5) for the training phase, we decide to have 50 000 samples of the new threat before retraining the models as mentioned in Figure 4. This step is performed on a distant server then the new models will be injected in the IoT devices. Since the retraining needs an important amount of samples, it will be performed occasionally when enough data are available which is adequate in the IoT context. The size of the models to be injected will be studied in Section IV.

IV. EXPERIMENTS

To experiment our IDPS strategy, we concentrate, in this paper, on the first and third levels of machine learning detection. We did not experiment the second level of machine

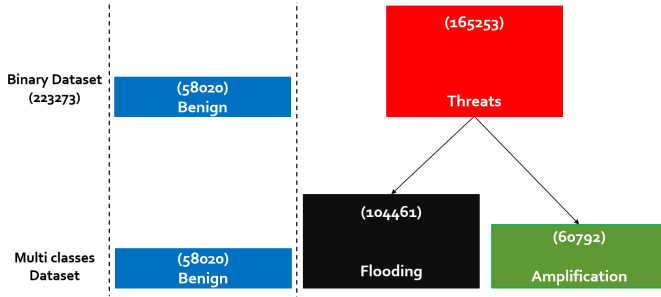


Fig. 5: OneM2M dataset

learning detection especially since it requires a large number of samples of an unknown threat. The evaluation of five of the shallow machine learning algorithms with the oneM2M dataset in [3] shows that J48 ML algorithm has achieved the best performances. In [3], we consider binary and multi-class classification. Consequently, we decide to experiment in this paper, a deep learning (DL) algorithm and to compare its results to the J48 ones. We will concentrate on the results of the binary classification (first ML level in Figure 2) and the third level of ML detection in Figure 2.

We start this Section with the presentation of the test environment as well as the deep learning hyper-parameters and the relevant machine learning metrics in the context of IoT systems security.

A. Test Environment

The ML classifications were made offline, on a machine with Intel(R) Xeon(R) CPU E3-1225 v3 @ 3.20GHz and 16Go of RAM. We use Weka tool [22] for the J48 tests and the Python Deep Learning library Keras [23] for the deep learning algorithm implementation and tests.

B. Deep Learning Hyper-Parameters

To implement a deep learning algorithm with Keras Library [24], we need to precise different hyper-parameters to get better results. The ones that we adjust in our case to have good detection are:

- hyper-parameters for the layers:
 - the units hyper-parameter which is an integer that represents the dimensionality of the output space,
 - the activation functions [25] are an important feature in the artificial neural networks. They decide the relevance of the neuron information with a non linear transformation,
 - the kernel initializers [26] define the way to set the initial random weights of the layer,
- hyper-parameters for the learning process [23]:
 - the loss function which represents the objective function that the model will try to minimize by changing the parameters (weights) of the model,
 - the optimizer in machine learning is a function that describes how to adjust the parameters of the model for the loss minimization.

C. Machine Learning Metrics for the IoT context

In machine learning, there are numerous metrics that could be used to compare the performance of the different algorithms. All the metrics are calculated from the confusion matrix which is composed of:

- TP: True Positive which represents the correct classification of the oneM2M *GFlows* attacks as attacks.
- TN: True Negative which represents the correct classification of the benign oneM2M *GFlows* as benign.
- FP: False Positive which represents incorrect classification where the benign oneM2M *GFlow* were classified as attacks. High FP value increases the computation time. However it is less harmful than the high FN value.
- FN: False Negative which represents incorrect classification where oneM2M *GFlows* attacks are predicted as benign flows. High FN is a serious problem since it means that the attacker succeed to pass through the IDS.

In our study, we decide to concentrate on the following metrics:

- Recall or detection rate which reflects the samples that should have been identified as attacks [$Recall = TP / (TP + FN) = 1 - (FN / (TP + FN))$]
- Accuracy which estimates the ratio of the correctly predicted flows to the entire dataset[$Accuracy = (TP + TN) / (TP + TN + FP + FN)$]
- Precision which estimates the ratio of the correctly predicted attacks to the number of all predicted attack records [$Precision = TP / (TP + FP)$]
- False Postive Rate (FPR) which represents the false alarms: the ratio of the benign flows predicted as attacks to all the actual benign inputs [$FPR = FP / (TN + FP)$]
- Model Size which is the size of the generated model that needs to be injected in the IoT devices for the ML classification. The smaller the models are the better is since we are in the context of tiny devices (IoT)
- CPU Training Time which represents the time needed to build the ML model. This metric is important for the continuous learning module of the Section III-E

D. Edge Machine Learning for IoT

We experiment in this paper a deep neural networks algorithm to improve the classification results in [3]. The model that gives the better results will be injected in the IoT devices to ensure autonomous, decentralized and faster threat detection since edge computing allows processing closer to where the problem is located rather than sending all data over long distances to data centres or clouds. In this subsection, we analyze the binary classification as well as the multi-classification of each attack type.

1) *Binary Classification:* In [3], the best performance for the binary classification was achieved by the J48 algorithm which is a decision tree algorithm in Weka tool. J48 detects attacks with 95.40%, 87.81%, 88.9% and 34% of recall, accuracy, precision and FPR, respectively. It takes 27 490 ms of CPU training time to generate a model of 301Ko.

To find good results with deep learning networks, we tried to vary the number of layers as well as the different hyper-parameters introduced in Section IV-B. The best results among the different combinations was achieved by a network of three fully connected neural network layers. The input layer was initialized by normally distributed weights (*random_normal* initializer) and has a Rectified Linear Unit [27] (*ReLu*) activation function. The second layer is composed of eight units with a *random_normal* as a kernel initializer and the *sigmoid* [27] function for the activation. Regarding the output layer, it is a two units layer since we are in the context of binary classification. *Softmax* was the used activation function to guarantee a probability between 0 and 1 for each classification. The used loss function was the *binary_crossentropy* and the optimizer was the *Stochastic Gradient Descent (SGD)* optimizer.

The best results in terms of detection rate (97.41%) and false positive rate (13.39%) are achieved by the DL algorithm with a model of 18 ko (Table I). It is true that J48 accomplish better accuracy and precision results as well as lower CPU training time, but in terms of IoT security it is more important to have better detection rate and FPR than accuracy and precision. Consequently, we decide to use the generated DL model for the first level of intrusion detection of our strategy.

TABLE I: Comparison of binary classification results

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
J48	95.40	87.81	88.90	34.00	301	27 490
DL	97.41	86.91	86.61	13.39	18	596 463

2) *Multi-Classification*: Before studying the multi-classification for each type of attack, we tried to improve the results that we get in [3] with a DL algorithm for the global multi-classification (benign, the sub-types of flooding and the sub-types of amplification).

TABLE II: Comparison of multi-classification results

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
J48	77.07	74.98	74.80	2.31	3 677	60 850
DL	70.18	69.16	73.97	3.16	33.50	339 544

As shown in Table II, the DL algorithm does not improve the results. Hence, the J48 gives always the best intrusion detection results. In the two following paragraphs, we will concentrate on the multi-classification for the flooding and the amplification attacks for the third level detection of the Section III-C.

3) *Flooding Classification*: For this experiment, we use only the flooding *GFlows* with the sub-types labels.

TABLE III: Comparison of flooding-classification results

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
J48	93.80	92.32	92.95	1.53	290	9 280
DL	86.62	82.60	83.82	3.52	32	152 435

As reported by the Table III, the J48 algorithm achieves the best results of 93.80%, 92.32%, 92.95%, 1.53% and 9 280ms of detection rate, accuracy, precision, FPR and CPU training time. Even though, the DL algorithm is the one which generates the smaller model, 290 Ko (the size of the J48 model)

still always acceptable for the IoT context. Hence, we decide to adopt the J48 algorithm for the flooding classification. The feature reduction phase for this classification eliminates 6 features from the oneM2M dataset features detailed in [3] which are: *isSameFromTo*, *isToRemote*, *fromResourceType*, *counterSameFromOperationResponseType*, *counterSameToResponseType* and *counterSameOperationResponseCategory*.

4) *Amplification Classification*: Regarding the amplification classification (Table IV), the J48 algorithm achieves the best results it terms of recall (62.77%), accuracy (65.04%), precision (62.83%) and training time (12 500ms). However, the DL gets the best FPR and model size. Since the difference of the FPR between the two algorithms is less than 0.3% and that the model size of J48 (953 Ko) remains acceptable, we choose to deploy the J48 for the amplification classification.

To have lighter feature extraction phase, J48 eliminates eight features from the oneM2M dataset features (detailed in [3]) for the amplification classification which are *isSameFromTo*, *isFromRemote*, *isToRemote*, *fromResourceType*, *counterSameFromOperationResponseType*, *counterSameFromOperationResponseCategory*, *counterSameToResponseType* and *counterSameOperationResponseCategory*. Therefore, we remove these features for the amplification classification.

TABLE IV: Comparison of amplification-classification results

ML algorithm	Recall (%)	Accuracy (%)	Precision (%)	False Positive Rate (%)	Model Size (Ko)	CPU Training Time (ms)
J48	62.77	65.04	62.83	17.07	953	12 500
DL	61.28	63.43	60.81	16.80	29	126 504

V. CONCLUSION AND FUTURE WORK

Since IoT devices are widely deployed and threatened, we propose in this paper, the first generic intrusion detection and prevention system related to the oneM2M standard based on edge machine learning. OneM2M is a global standard initiative designed to satisfy the need for a common horizontal platform for the multi-industry M2M/IoT applications. The introduced IDPS guarantees the security of the IoT devices with three levels of intrusion detection, a prevention module against the detected attacks and a continuous learning of the new unknown threats. It enables not only the detection of the security attacks but also of the abnormal behaviors of the IoT systems with an edge machine learning classification. We have experimented in this work the deep learning algorithm to improve the detection results. In the binary classification (first detection level), it achieves the best metrics. However, J48 accomplishes better results in the third layer of multi-classification with a reduction of six features for the flooding attacks and eight features for the amplification attack. Since we concentrate only on the first and the third detection level in this work, we plan to concentrate on the second level of machine learning detection in future works. We intend the deploy the whole IDPS in the future with online detections.

REFERENCES

- [1] OneM2M, "oneM2m - Home (<http://www.onem2m.org/>)," Apr. 2019.
- [2] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network Intrusion Detection for IoT Security Based on Learning Techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.
- [3] N. Chaabouni, M. Mosbah, A. Zemmari, and C. Sauvignac, "An Intrusion Detection System for the OneM2m Service Layer based on Edge Machine Learning," *18th International Conference on Ad Hoc Networks and Wireless, AdHoc-Now 2019 Luxembourg*, pp. 508–523, Oct. 2019.
- [4] S. L. Salzberg, "C4.5: Programs for Machine Learning by J. Ross Quinlan. Morgan Kaufmann Publishers, Inc., 1993," *Machine Learning*, vol. 16, no. 3, pp. 235–240, Sep. 1994.
- [5] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 338–345.
- [6] M. W. Gardner and S. R. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627–2636, Aug. 1998.
- [7] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.
- [8] Z. Zeng, H. B. Yu, H. R. Xu, Y. Q. Xie, and J. Gao, "Fast training Support Vector Machines using parallel sequential minimal optimization," vol. 1, Nov. 2008, pp. 997–1001.
- [9] S. Raza, L. Wallgren, and T. Voigt, "SVELTE: Real-time intrusion detection in the Internet of Things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2661–2674, Nov. 2013.
- [10] P. Kasinathan, C. Pastrone, M. A. Spirito, and M. Vinkovits, "Denial-of-Service detection in 6lowpan based internet of things," in *IEEE 9th International Conference on Wireless and Mobile Computing, Networking and Communications*, 2013, pp. pp. 600–607.
- [11] P. Kasinathan, G. Costamagna, H. Khaleel, C. Pastrone, and M. A. Spirito, "DEMO: An IDS Framework for Internet of Things Empowered by 6lowpan," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, Nov. 2013, pp. 1337–1340.
- [12] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos, "Detection of sinkhole attacks for supporting secure routing on 6lowpan for Internet of Things," in *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, May 2015, pp. 606–611.
- [13] F. Hosseinpour, P. Vahdani Amoli, J. Plosila, T. Hämäläinen, and H. Tenhunen, "An Intrusion Detection System for Fog Computing and IoT based Logistic Systems using a Smart Data Approach," *International Journal of Digital Content Technology and its Applications*, vol. 10, pp. 34–46, Dec. 2016.
- [14] H. Bostani and M. Sheikhan, "Hybrid of anomaly-based and specification-based IDS for Internet of Things using unsupervised OPF based on MapReduce approach," *Computer Communications*, vol. 98, no. Supplement C, pp. 52–71, Jan. 2017.
- [15] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalis A System for Knowledge Driven Adaptable Intrusion Detection for the Internet of Things," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 656–666.
- [16] G. Thamilarasu and S. Chawla, "Towards Deep-Learning-Driven Intrusion Detection for the Internet of Things," *Sensors* 2019, Apr. 2019. [Online]. Available: <https://www.mdpi.com/1424-8220/19/9/1977>
- [17] OneM2M, "TS-0003-V2.12.1 Security Solutions," p. 427, Mar. 2018.
- [18] G. Tuna, D. G. Kogias, V. C. Gungor, C. Gezer, E. Taşkın, and E. Ayday, "A survey on information security threats and solutions for Machine to Machine (M2m) communications," *Journal of Parallel and Distributed Computing*, vol. 109, pp. 142–154, Nov. 2017.
- [19] OneM2M, "TS-0001-V2.18.1 Functional Architecture," p. 427, Mar. 2018.
- [20] OneM2M, "TS-0004-V2.7.1 Service Layer Core Protocol Specification," p. 427, Aug. 2016.
- [21] Q. R. Wang and C. Y. Suen, "Analysis and Design of a Decision Tree Based on Entropy Reduction and Its Application to Large Character Set Recognition," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 4, pp. 406–417, 1984.
- [22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009.
- [23] Keras, "Guide to the Sequential model - Keras Documentation (<https://keras.io/getting-started/sequential-model-guide/>)," May 2019.
- [24] Keras, "Core Layers - Keras Documentation (<https://keras.io/layers/core/>)," May 2019.
- [25] P. Dayan and L. F. Abbott, "Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems," 2001.
- [26] Keras, "Initializers - Keras Documentation (<https://keras.io/initializers/>)," May 2019.
- [27] Keras, "Activations - Keras Documentation (<https://keras.io/activations/>)," May 2019.