Machine Learning Methods for Anomaly Detection in IoT Networks, with Illustrations

Vassia Bonandrini[1], Jean-François Bercher[2](✉), and Nawel Zangar[2]

[1] ESIEE Paris, Marne-la-Vallée, France
vassia.bonandrini@edu.esiee.fr
[2] LIGM, UMR 8049, École Des Ponts, UPEM, ESIEE Paris, CNRS, UPE, Marne-la-Vallée, France
{jf.bercher,nawel.zangar}@esiee.fr

Abstract. IoT devices have been the target of 100 million attacks in the first half of 2019 [1]. According to [2], there will be more than 64 billion Internet of Things (IoT) devices by 2025. It is thus crucial to secure IoT networks and devices, which include significant devices like medical kit or autonomous car. The problem is complicated by the wide range of possible attacks and their evolution, by the limited computing resources and storage resources available on devices. We begin by introducing the context and a survey of Intrusion Detection System (IDS) for IoT networks with a state of the art. So as to test and compare solutions, we consider available public datasets and select the CIDDS-001 Dataset. We implement and test several machine learning algorithms and show that it is relatively easy to obtain reproducible results [20] at the state-of-the-art. Finally, we discuss embedding such algorithms in the IoT context and point-out the possible interest of very simple rules.

Keywords: Internet of Things · IoT · IDS · NIDS · Intrusion detection system · Rules · CIDDS-001

1 Introduction

The problem of securing electronic devices is as old as computers exist, but with time computers have gained more and more resources, so IDS in these devices became more efficient. Now, a lot of different small devices without the power of modern computers are connected to a network and are the target of many attacks. Moreover, every IoT system is different and has specific worries depending on the type of the attack (DDoS, Blackhole, Sybil Attack…) they want to be protected from. Wireless Sensor Networks (WSN) for instance has unique characteristics such as limited power supply, low transmission bandwidth, small memory size, and data storage [3]. It is thus crucial to develop and deploy new IDS.

Section 2 presents a brief review of the literature and Sect. 3 presents the problem of selecting or simulating a dataset to test IDS. Section 4 presents the implementation, tests, and results of several machine learning algorithms for outlier detection for CIDDS-001 dataset. Section 5 presents some simple decision rules that can be

S. Boumerdassi et al. (Eds.): MLN 2019, LNCS 12081, pp. 287–295, 2020.
https://doi.org/10.1007/978-3-030-45778-5_19

introduced in an IoT network to work like an IDS. Section 6 presents a summary of the conclusions and future work.

2 Related Work

Doshi et al. [4] simulate IoT networks with raspberry Pi and virtual machines. They collect network data from their system and test this data with 5 algorithms: K-nearest neighbors, Support Vector Machine (SVM) with linear kernel, decision tree, random forest, and neural network. The specificity of their work is to use stateful features and they get up to 30% better performance compared to without these features.

Hussain et al. [5] list for each problem many surveys that use machine learning techniques (Table 4 on the document). For anomaly and intrusion detection:

- K-means clustering and Decision Tree [6]
- Artificial Neural Network ANN [7]
- Novelty and Outlier Detection [8]
- Decision Tree [9]
- Naive Bayes [9, 10]

Butun et al. [3] classify the IDS methodology of IDS in 3 categories:

1 Anomaly based detection:
We create an activity profile for each member of the network and a certain amount of deviation is reported as an anomaly. This method is adequate to detect never known attacks but we need to update the profiles periodically because the network behavior can change rapidly.
2 Misuse based detection
A signature (profile) of the previously known attack is used and is used as a reference to flag the next attacks. The disadvantage of this method is that it cannot detect new type of attacks, but the false positive rate is very low.
3 Specification based detection
That’s a mix of the previous ones, “a set of specifications and constraints that describe the correct operation of a program or protocol is defined.” [3] But it takes a lot of time to develop special rules to get a low false-positive rate.

Some surveys have unclear results, sometimes there is no result. There are also very few simulations and implementations in real systems.

3 Dataset

Selecting a dataset to design and evaluate NIDS ML-based algorithms is not immediate and may be a full part challenge.

One of the most used datasets is the KDD cup99 set, but it still presents defaults, as emphasized by Tavallaee et al. [11]:

– a lot of redundant measures
– some parts of the train set were used as test sets in some studies
– set is too long forcing to take only part of the set.

Ring et al. [12] did an exhaustive list of network-based detection data set and compared them. One of the recent and not too heavy dataset is the CIDDS-001 (Coburg Intrusion Detection Data Set) [13] which was described as follows:

"*The CIDDS-001 data set was captured within an emulated small business environment in 2017, contains four weeks of unidirectional flow-based network traffic, and comes along with a detailed technical report with additional information. As special feature, the data set encompasses an external server which was attacked in the internet. In contrast to honeypots, this server was also regularly used by the clients from the emulated environment. The CIDDS-001 data set is publicly available and contains SSH brute force, DoS and port scan attacks as well as several attacks captured from the wild*" [12].

The dataset contains 14 features as follow (Table 1).

Table 1. Features within the CIDDS-001 data set, from [13]

Id	Attribute name	Attribute description
1	Src IP	Source IP Address
2	Src Port	Source Port
3	Dest IP	Destination IP Address
4	Dest Port	Destination Port
5	Proto	Transport Protocol (e.g. ICMP, TCP, or UDP)
6	Date first seen	Start time flow first seen
7	Duration	Duration of the flow
8	Bytes	Number of transmitted bytes
9	Packets	Number of transmitted packets
10	Flags	OR concatenation of all TCP Flags
11	Class	Class label (Normal, Attacker, Victim)
12	AttackType	Type of Attack (PortScan, DoS, Bruteforce, PingScan)
13	AttackID	Unique Attack id
14	AttackDescription	Additional information about the set attack parameters

In our experiment, we used the "Class" attribute as the target for classification, removed the AttackType, AttackID and AttackDescription features which are obviously correlated with the "attacker" class. Furthermore, since IPs were anonymized, they do not convey information so we also removed them. We also use "Date first seen" as the x-axis. Finally, we transformed Flags, Class and Proto, which are categorical features, into "dummy variables" by one-hot encoding.

In the CIDDS-01 dataset, we used the internal-week1 subset of observations, as it contains 42 of the 92 attacks on the entire dataset.

Anomalies are labeled as victim or attacker. However, this file has more than 8 million rows and less than 20% are anomalies. Hence we face a case of imbalanced classes. In such a case, the more represented class can have a “masking effect” on the others; this has been studied in [14] for this dataset. Given the high number of instances available, rebalancing the classes can be simply done by subsampling the majority class (otherwise, one can also oversample the minority classes by creating new, synthetic, instances). In our case, we decided to (a) shuffle the data, (b) keep half of the data for a final evaluation (c) subsample the other half to keep about 180000 instances per class.

4 Experiments and Results

The experiments were carried out using Google Colaboratory with 32 GB of ram and the Tensor Processing Unit acceleration material.

The metrics that were used to evaluate the performance of the algorithm include the classification accuracy, precision, recall and F1-score. These metrics are expressed by the equations below,

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

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

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

$$\text{F1-score} = \frac{2.Precision.Recall}{Precision + Recall}$$

where, TP, TN, FP and FN stand for true positives, true negatives, false positives and false negatives, respectively.

We shuffle the set and we take 33% of the set as the test set and 66% as the train set. Then, we classify the traffic with 4 algorithms: K-Nearest Neighbors (KNN), Decision Tree (DT), Random Forest (RF) and Neural Network (NN). We used the python sklearn package to do our test.

For the KNN, we use only 1 neighbour with a uniform weight function. We get a global accuracy of 99.27%; other metrics are reported Table 2.

Table 2. Results with the K Nearest Neighbour algorithm

KNN	Precision	Recall	F1-score
Attacker	0.9633	0.9972	0.9799
Normal	0.9996	0.9916	0.9956
Victim	0.9573	0.9988	0.9976

For the Decision Tree, we used the default parameters, which is the Gini criterion for measuring the quality of a split and maximum expansion of nodes. We already obtained a global accuracy of 99.89%; other performance metrics are given Table 3.

Table 3. Results with the Decision Tree algorithm

DT	Precision	Recall	F1-score
Attacker	0.9956	0.9982	0.9969
Normal	0.9998	0.9990	0.9994
Victim	0.9946	0.9996	0.9970

For instance, we get something as shown in Fig. 1 for the beginning of the tree:

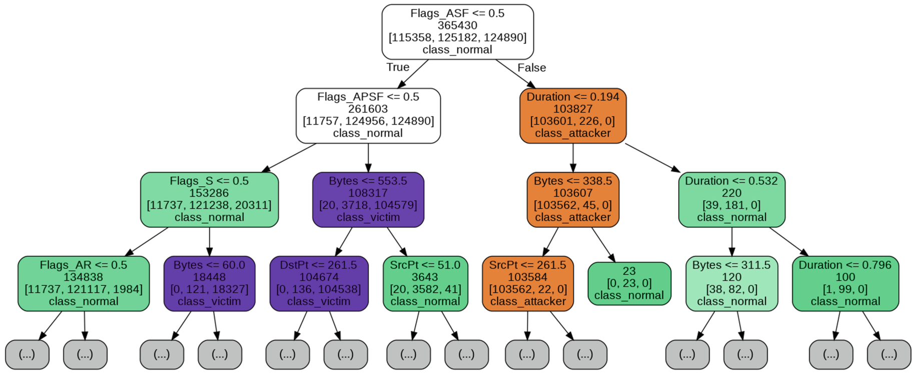

Fig. 1. First nodes of the decision tree

The decision tree has a depth of 31, a total of 563 nodes and 282 leaves, thus 281 tests. From this tree, it is possible to deduce, and even generate automatically, a classification script (see the source code [20] for an example). Running this code on an instance will predict the classification with 99.88% accuracy.

For the RF, we selected the best parameters using a grid search strategy, which consists of computing the performance, by cross validation, on a grid of possible parameters values, and then selecting the best estimator. We used global accuracy as the performance metric. In particular, we used 800 trees with a max depth of 20. We get a global accuracy of 99.95%, with the performances reported in Table 4.

Table 4. Results with the Random Forest algorithm

RF	Precision	Recall	F1-score
Attacker	0.9982	0.9981	0.9982
Normal	0.9997	0.9996	0.9997
Victim	0.9981	0.9993	0.9997

An interesting outcome of the random forest is that we can extract which are the most important features in computing the classification. The features are shown by importance on Fig. 2.

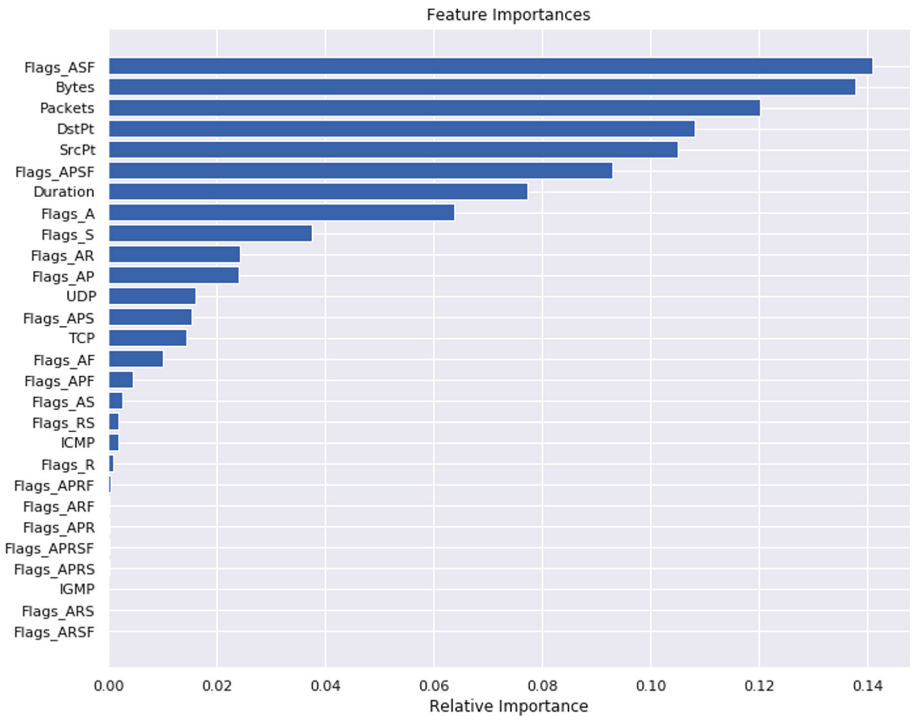

Fig. 2. Relative importance of features in the CIDDS-01 dataset.

For the Neural Network, we used the multi-layer perceptron classifier. This model optimizes the log-loss function using LBFGS or stochastic gradient descent. We used 100 neurons in the hidden layer with the rectified linear unit function for the hidden layer activation function. The maximum iteration was set to 200. We finally obtained a global accuracy of 99.25%; performances metrics for the different classes are shown Table 5.

Table 5. Results with the Neural Network algorithm

NN	Precision	Recall	F1-score
Attacker	0.9929	0.9938	0.9933
Normal	0.9906	0.9962	0.9934
Victim	0.9914	0.9883	0.9899

For the CIDDS-01 dataset and the ML-based algorithms, we obtained very high accuracies. As mentioned in [12], for this particular dataset, balancing the classes or not has a very tight influence on the accuracy, which is already so high. Anyway, by a careful selection of the hyperparameters we recover results similar to the best RF-WHICD in [14] (where only two classes normal/attacker were considered) (Table 6).

Table 6. Comparison with related work

Survey	Approach	Accuracy (%)
Verma and Ranga [15]	2NN	99.60
Verma and Ranga [16]	DT	99.90
Tama and Rhee [17]	DNN-10-FCV	99.90
Idhammad et al. [18]	Entropy + RF	99.54
Abdulhammed et al. [14]	RF-WHICD	99.99
Proposed KNN	KNN-1	99.27
Proposed DT	DT	99.89
Proposed RF	RF	99.95
Proposed NN	NN	99.25

5 Rules

The most accurate algorithm is the RF with 99.95% accuracy. However, it may be difficult to embed on an IoT and it lacks interpretability. The problem of extracting simple rules from a forest of decision trees has been considered in the machine learning community. The goal was to find a trade-off between the modelization power of random forests and some simple rules interpretable as in a (small) decision tree. The Skope rules Python library [19] enables us to extract such rules from a random forest. In our experiments, we took all the instances to train the model and extract the rules. For the victim class, the skope rules are:

- Bytes > 100
- Duration <= 0.03749999962747097
- Flags == ASF

And for the attacker class, the identified rules are:

- Dst Pt <= 261
- Duration <= 0.032500000670552254
- Flags == APSF

With these very simple rules, we already get a global accuracy of 86.88%. Furthermore, by a simple inspection of data, we discovered that adding the instances flagged with AR (class victim) or S (class attacker) TCP flags enables us to improve the accuracy to 98.45%, with only 0.35% are miss classifications.

6 Conclusion

We have considered the problem of detecting anomalies or intrusions in IoT networks. We have first presented the context and reviewed approaches in the literature. We have focused on machine learning-based methods that can learn directly from data and find what are the important features, without resorting to specialized models of the network or specialized signatures. We have then selected a dataset of network activity with several attacks, which is regularly used to develop NIDS and as a benchmark of proposals. Using standard open-source libraries, we have implemented and evaluated several ML-based algorithms, with performances that are at the state-of-the-art. The sources are available and results easily reproducible [20].

Using and implementing such solutions in an IoT network requires to consider the possible computational overhead. For a router or network supervisor, we need a network traffic module to capture the incoming network, and the classifiers, once trained by a decision tree or a random forest, can probably be implemented. For the IoT devices themselves, where consumption and computational costs can be more severely constrained, it is possible to use a decision tree classifier (at most 20 comparison tests and a 840 lines program in Python) or even to use the very simple rules (3 tests) derived from a random forest, with a 98.5% accuracy and a low false-positive rate. Testing these ideas on real devices and real data is the objective of future efforts.

References

1. IoT under fire: Kaspersky detects more than 100 million attacks on smart devices in H1 (2019). https://www.kaspersky.com/about/press-releases/2019_iot-under-fire-kaspersky-detects-more-than-100-million-attacks-on-smart-devices-in-h1-2019. Accessed 29 Oct 2019
2. Newman, P.: IoT Report: How Internet of Things technology growth is reaching mainstream companies and consumers. Business Insider, 28 January 2019
3. Butun, I., Morgera, S.D., Sankar, R.: A survey of intrusion detection systems in wireless sensor networks. IEEE Commun. Surv. Tutor. **1**(16), 266–282 (2014)
4. Doshi, R., Apthorpe, N., Feamster, N.: Machine learning DDoS detection for consumer Internet of Things devices. In: 2018 IEEE Security and Privacy Workshops (SPW), pp. 29–35 (2018)
5. Hussain, F., Hussain, R., Hassan, S.A., Hossain, E.: Machine Learning in IoT Security: Current Solutions and Future Challenges. arXiv [cs.CR], 14 March 2019
6. Shukla, P.: ML-IDS: a machine learning approach to detect wormhole attacks in Internet of Things. In: 2017 Intelligent Systems Conference (IntelliSys), pp. 234–240 (2017)
7. Cañedo, J., Skjellum, A.: Using machine learning to secure IoT systems. In: 2016 14th Annual Conference on Privacy, Security and Trust (PST), pp. 219–222 (2016)
8. Nesa, N., Ghosh, T., Banerjee, I.: Non-parametric sequence-based learning approach for outlier detection in IoT. Fut. Gener. Comput. Syst. **82**, 412–421 (2018)
9. Viegas, E., Santin, A., Oliveira, L., França, A., Jasinski, R., Pedroni, V.: A reliable and energy-efficient classifier combination scheme for intrusion detection in embedded systems. Comput. Secur. **78**, 16–32 (2018)

10. Pajouh, H.H., Javidan, R., Khayami, R., Ali, D., Choo, K.-K.R.: A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Trans. Emerg. Top. Comput. **2**(7), 314–323 (2019)
11. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009)
12. Ring, M., Wunderlich, S., Scheuring, D., Landes, D., Hotho, A.: A survey of network-based intrusion detection data sets. Comput. Secur. **86**, 147–167 (2019)
13. Ring, M., Wunderlich, S., Gruedl, D., Landes, D., Hotho, A.: Technical Report CIDDS-001 data set (2017)
14. Abdulhammed, R., Faezipour, M., Abuzneid, A., AbuMallouh, A.: Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. IEEE Sensors Lett. **1**(3), 1–4 (2019)
15. Verma, A., Ranga, V.: On evaluation of network intrusion detection systems: Statistical analysis of CIDDS-001 dataset using machine learning techniques. Pertanika J. Sci. Technol. **3**(26), 1307–1322 (2018)
16. Verma, A., Ranga, V.: Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. Proc. Comput. Sci. **125**, 709–716 (2018)
17. Tama, B.A., Rhee, K.-H.: Attack classification analysis of IoT network via deep learning approach. Res. Briefs Inf. Commun. Technol. Evol. (ReBICTE) **3**, 1–9 (2017)
18. Idhammad, M., Afdel, K., Belouch, M.: Detection system of HTTP DDoS attacks in a cloud environment based on information theoretic entropy and random forest. Secur. Commun. Netw. **2018** (2018). Article ID 1263123
19. https://github.com/scikit-learn-contrib/skope-rules
20. Bonandrini, V., et al.: https://github.com/VasgoTheTotoroo/IDS_IoT