Anomaly detection in IoT network traffic

ANTOON WOLTERS, University of Twente, The Netherlands

Abstract- With the increasing usage of IoT systems in various sectors, ensuring the reliability and security of them has become a major priority. This paper investigates the effectiveness of certain Deep Learning and Machine Learning algorithms for anomaly detection in IoT network traffic. It aims to evaluate the performance of Machine Learning and Deep Learning algorithms in order to find the most optimal algorithm for a real-time use case. This conclusion is carefully made after analyzing the accuracy and performance on the IoT-23 and IoTID20 datasets for each of the following algorithms: XGBoost, Random Forest, Naïve Bayes, Decision Tree, Convolutional Neural Network, Stochastic Gradient Descent. The findings of this study can provide insight into the effectiveness of algorithms for IoT Security Systems.

Additional Key Words and Phrases: Internet of Things, Anomaly Detection, Network Security, Machine Learning, IoT-23,IoTID20

1 INTRODUCTION

The Internet of Things (IoT) is an ever expanding network of physical devices that connect and exchange data through the internet. This concept has experienced exponential growth in recent years. The estimated amount of IoT devices has been rising throughout the years from around 2 billion in 2006 to a predicted 200 billion in 2020[12]. This exponential growth was caused by a series of innovations in various sectors, from wearables and smart homes to industry 4.0 supply chains [10] and smart cities. IoT has the potential to completely reshape the users current way of living, by connecting every part of their daily lifes.

Obviously the development of this technology has also introduced new security challenges to the equation. As more critical infrastructure like Healthcare service, food supply chains and even firefighting systems [8] get connected to the internet through these IoT systems, it becomes even more crucial to ensure the security of them. Furthermore, the large amounts of user and other classified data generated by these systems must be secured.

An innovative way of making such a network more secure, is by using Artificial Intelligence (AI) methods to determine and detect the different variations of malicious network traffic. The large datasets will provide enough training material to create substantial models.

In this research, the various Machine Learning (ML) and Deep Learning (DL) algorithms will be explored by training them to detect malicious traffic and their effectiveness will be evaluated based on a variety of aspects like model accuracy and model efficiency. The findings of this research can be valuable for the creation of future Intrusion Detection Systems [12].

TScIT 39, July 7, 2023, Enschede, The Netherlands

2 PROBLEM STATEMENT

With the increasing usage of IoT devices throughout recent years, the security of these devices has become an even more important concern. IoT devices are usually poorly protected, which is partially due to the fact that an average IoT device is not able to support complex security schemes because of their low power and computational resource capabilities[2]. This makes it quite simple for malicious actors to compromise these devices. Due to the differing ways a network can be attacked by malicious users, it becomes extremely hard for human actors to analyze and monitor a network properly. Additionally, the sheer amount of network traffic would make this impossible to do for a real-time use case. Therefore, in order to properly identify these malicious users, an algorithm should be applied to properly analyze and detect these security threats.

The aim of this research is to develop ML/DL algorithms, which will be used to detect malicious traffic for IoT devices. The algorithms will be trained and tested on the IoTID20 dataset and an IoT-23 subset. These algorithms will be evaluated on different qualities like efficiency, accuracy and overall performance.

2.1 Research Questions

This leads to the following research question: Which ML/DL algorithm is most effective for detecting anomalies in IoT network traffic? Sub-questions:

- What are the most accurate ML/DL algorithms to detect malicious traffic in IoT network traffic?
- When taking into account the overall performance of the model, which is the best for a real time use case?
- How do the results of the algorithms differ when trained on different datasets?

3 RELATED WORK

Regarding the application of these AI methods, a lot of research has already been conducted. The research can be divided into 2 categories: an application of either Machine Learning or Deep Learning algorithms[1, 4, 7, 9, 11, 13, 14] on existing datasets or a more theoretical paper regarding the overall security [2, 6, 16].

The papers provided insights into the different algorithms and the accuracy and speed of their testing results. For training on the IoT-23 dataset the following algorithms were used: Naive Bayes, Support Vector Machine, Convolutional Neural Network, Decision Tree and CatBoost. In this experiment , the Random Forest obtained a high testing accuracy with an accuracy between 99.5% and 100% [14]. The Catboost algorithm also obtained an incredible accuracy of 99.99% when trained on the IoT-23 dataset[4]. For the IoTID20 dataset a large variety of machine learning algorithms were trained in order to test the classification of subcategories. The experimental results of this research show that the Random Forest, Decision Tree and Ensemble achieved a relatively high accuracy whilst detecting the sub-categories of the dataset[15]. In this case, the Decision Tree algorithm scored an accuracy of 88%, Ensemble 87% and the Random Forest 84%.

The theoretical papers gave insights into the current security solutions and in which ways they are limited. They also address the challenges one can encounter when implementing a ML based security system. Currently, IoT devices are limited in their options regarding security, as there is just not enough computational power in the average IoT node to support complex security schemes[2]. So on top of the already present challenges regarding the dataset and algorithm selection in order to tackle a specific problem. There is an additional layer added in the form of a computationally light model. the actual applications of the ML and DL algorithms are useful for an entirely different reason. They gave a rough understanding on how to start with the process of training the models on such a dataset. The overall results of their testing influenced the specific algorithms that were used for training.

4 METHODOLOGY

This section will elaborate upon the methods used in order to conduct this research.

4.1 Dataset Collection

During the research, two different datasets were used for the training of the algorithms, The IoTID20 and the IoT-23 datasets. The 'IoT Intrusion Dataset 2020' acronym as IoTID20 is a dataset generated by I. Ullah et al.[15]. This dataset contains a total of 625,783 records and 86 features, of which three are label features. It is broken down into the catagories that can be seen in Table 1.

Table 1. IoTID20 Categories.

Binary	Category	SubCategory
Benign	-	-
Anomaly	Mirai	Mirai-Ackflooding
		Mirai-HTTP Flooding
		Mirai-UDP Flooding
		Mirai-HostBruteForce
	MITM	MITM ARP Spoofing
	DoS	DoS-Synflooding
	Scan	Scan HostPort
		Scan Port OS

The IoT-23 dataset[5] is a dataset created by the Avast AIC laboratory. this dataset consists of a total of 23 captures, with 20 of them being mainly malicious traffic and 3 of them being benign. The dataset contains a total of 325,307,990 records, with 294,449,255 of them being malicious. This is broken down into the categories that can be seen in Table 2.

4.2 Data Formatting

Since one of the goals was to compare the efficiency of algorithms between the datasets, the decision was made to scale the IoT-23 dataset down to a size comparable to the IoTID20 dataset. Due to the distribution of the IoT-23 dataset, it was not feasible to scale the dataset down while keeping the same ratio's between the different anomalies. As this would lead to certain attacks having only a few entries per training fold. The choice was made to change the ratio's between the anomalies by completely including the anomalies with low representation and taking a subsample of the overrepresented anomalies. The conn.log.labeled files of the IoT-23 dataset were iterated over in order to extract these specific entries to create a distribution as seen in Table 3.

Table 2. IoT-23 Categories.

Binary	Category
Benign	-
Anomaly	Attack
	C&C
	C&C-FileDownload
	C&C-HeartBeat
	C&C-HeartBeat-Attack
	C&C-HeartBeat-FileDownload
	C&C-Mirai
	C&C-PartOfAHorizontalPortScan
	C&C-Torii
	DDoS
	FileDownload
	Okiru
	Okiru-Attack
	PartOfAHorizontalPortScan
	PartOfAHorizontalPortScan-Attack

Table 3. IoT-23 anomaly distribution

Label	Original Dist	New Dist
Benign	30,858,735	120,000
Attack	9,398	9,398
C&C	21,995	21,995
C&C-FileDownload	53	53
C&C-HeartBeat	33,673	33,673
C&C-HeartBeat-Attack	834	834
C&C-HeartBeat-FileDownload	11	11
C&C-Mirai	2	2
C&C-PartOfAHorizontalPortScan	888	888
C&C-Torii	30	30
DDoS	19,538,713	75,000
FileDownload	18	18
Okiru	47,381,241	80,000
Okiru-Attack	13,609,470	60,000
PartOfAHorizontalPortScan	213,852,924	250,000
PartOfAHorizontalPortScan-Attack	5	5
total	325,307,990	652,725

4.3 Data Pre-Processing

After both of the datasets had been acquired, the actual pre-processing of these datasets could begin. The decision was made to train the algorithms based on the overall category of the anomalies. For the IoTID20 dataset this was the category label, for the IoT-23 dataset the subcategories were encoded to ensure that similar attacks would be grouped together to reduce the amount of classes that the algorithm would need to classify between and to increase the amount of entries per encoded category.

Table 4. Category encoding

Category	Encoding
Benign	0
C&C	1
C&C-FileDownload	1
FileDownload	1
C&C-Mirai	1
C&C-Torii	1
C&C-HeartBeat	2
C&C-HeartBeat-Attack	2
C&C-HeartBeat-FileDownload	2
DDoS	3
Attack	4
Okiru	5
Okiru-Attack	5
PartOfAHorizontalPortScan	6
PartOfAHorizontalPortScan-Attack	6
C&C-PartOfAHorizontalPortScan	6

4.3.1 *Correlation.*

The application of statistical correlation to the datasets was motivated by the need to investigate and understand the relationship between features within the dataset itself. By performing this correlation and observing the results from the correlation matrices. We are able to remove the features that either have no or a weak correlation to the feature containing the anomaly categories. In this case, a blue color indicates a negative correlation between features and the red color indicates a positive correlation. By performing this statistical correlation and removing certain features that are not relevant to the label, the resulting model can be trained on fewer features which will make it less complicated and this will obviously result in a more efficient model. As seen in Figure 1, the IoT-23 dataset had two features with no correlation to the label at all and multiple features with a weak correlation. In this case, the decision was made to eliminate the following features:

local_orig, local_resp, missed_bytes, uid, resp_ip_bytes and additionally the **tunnel_parents** feature was removed during preprocessing as this was a mostly empty column.

For the IoTID20 dataset a similar correlation matrix was generated but on a larger scale. Due to the matrix having 84 different features it became to large to be readable as a figure for this paper. However, the results of this statistical correlation were also quite clear. The IoTID20 dataset had a total of 10 features which were either weakly correlated or not correlated at all to the label and were thus removed, these were:

Fwd_PSH_Flags, Fwd_URG_Flags, Fwd_Bytsb_Avg, Fwd_Pktsb_-Avg, Fwd_Blk_Rate_Avg , Bwd_Bytsb_Avg, Bwd_Pktsb_Avg, Bwd_Blk_Rate_Avg, Init_Fwd_Win_Byts, Fwd_Seg_Size_Min

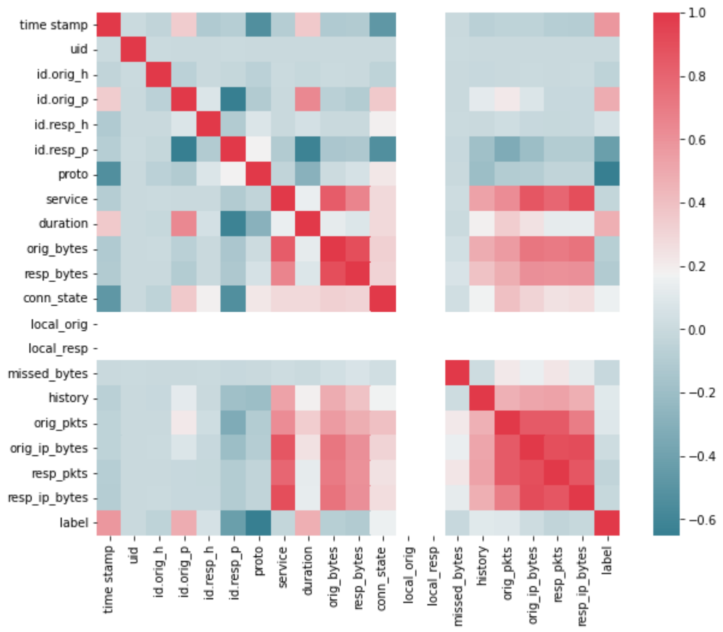

Fig. 1. Correlation Matrix IoT-23

4.4 Algorithm Analysis

After the pre-processing step followed the actual training of the algorithms. This was done using cross validation with a 80-20 train-test split. In this section the various algorithms used in this research will be explained in greater details.

4.4.1 *Decision Tree.*

The Decision Tree is a machine learning algorithm that creates a tree-like model, where each node represents a feature and each branch a decision based on that feature. This algorithm recursively splits the data by selecting the features with the highest information gain. During classification, the data follow the nodes from the root to a leaf in order to assign the right label. The decision tree is a computationally light algorithm but is in cases susceptible to overfitting when the tree becomes to complex.

4.4.2 *XGBoost.*

Extreme Gradient Boosting or XGBoost is a highly effective machine learning algorithm that constructs an ensemble of weak models to improve prediction accuracy. XGBoost uses a tree boosting approach, which focuses on optimizing a loss function through iterative addition of decision trees . XGBoost minimizes the residuals of the loss function by continuously improving performance with each added decision tree. It also allows for parallel and distributed computing [3] which improves the training time and thus enables quicker model exploration. XGBoost can utilize the Graphics Processing Unit (GPU) for faster training and it can leverage multiple Central Processing Unit (CPU) cores to optimize this, making it well-suited for handling large datasets and computationally intensive tasks.

4.4.3 *Naïve Bayes.*

Naïve Bayes is a simple classification algorithm based on Bayes' theorem with an assumption of feature independence.

$$P(\mathbf{A}|\mathbf{B}) = \frac{P(\mathbf{B}|\mathbf{A}) * P(\mathbf{A})}{P(\mathbf{B})} \tag{1}$$

This algorithm calculates the posterior probability of each class given the input features. It assumes that the features are conditionally independent of each other given the labels, which allows for efficient computations. The algorithm requires a relatively small amount of training data compared to other algorithms, this makes it a great classifier for situations where there are no strong dependencies between the features.

4.4.4 *Random Forest.*

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. The various decision trees are trained independently on random subsets of the training data. By aggregating the predictions of the individual trees, Random Forest is able to reduce some of the overfitting that might be present when using a singular decision tree.

4.4.5 *Artificial Neural Network (ANN).*

An Artificial Neural Network is a deep learning algorithm consisting of interconnected nodes or neurons in this case. Each neuron takes input values and applies a certain weight and bias to them. It then passes the transformed input through an activation function in order to produce a fitting output. An ANN typically consists of: an input layer, one or more hidden layers and then the output layer. Every neuron in a given layer is connected to next layer, from the input layer down to the output layer. During the training of an ANN, the biases and weights of neurons are updated iteratively in order to minimize the difference between actual outputs and predicted outputs. Overall the ANN is quite a flexible algorithm, however it does need quite a large set of training data in order to perform optimally.

4.4.6 *Stochastic Gradient Descent (SGD).*

The Stochastic Gradient Descent is an optimization algorithm widely used in ML. It updates model parameters iteratively using randomly selected subsets of training samples. By considering only a subset of the training data at each iteration, SGD reduces computational burden and enables faster convergence, making it suitable for larger datasets. The implementation used in research supports the default linear Support Vector Machine (SVM).

4.5 Metrics

In order to properly assess the algorithms used in this research, the following set of metrics were used: Accuracy, Recall, Precision and F1-score. These metrics offer calculations on the amount of True Positives (TP), True Negatives (TN) , False Positives (FP) and False Negatives (FN) in order to evaluate the models overall performance and efficiency. Furthermore, this research considers two approaches regarding these metrics, namely: Weighted averaging and Macro averaging. Weighted averaging takes into account the class distribution within the dataset. It calculates the metric by considering the weighted average of each class based on the number of instances belonging to each class. Macro averaging treats each class equally without considering their distribution or imbalance. It calculates the metric by taking the average of the individual class performances.

4.5.1 *Accuracy.*

Accuracy measures the overall correctness of a model's predictions. It calculates the ratio of correct predictions, being true positives and true negatives in this case, to the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

4.5.2 *Recall.*

Recall measures the ability of the model to find all the positive instances. It calculates the ratio of rue positives to the total number of actual positives.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

4.5.3 *Precision.*

Precision focuses on the correctness of positive predictions. It calculates the ratio of true positives to the total number of positive predictions.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

4.5.4 *F1 score.*

The F1-score is a harmonic mean of precision and recall, this means that it considers both false positives and false negatives. It is particularly helpful in imbalanced datasets, where optimizing for accuracy alone may be misleading.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5}$$

5 RESULTS

5.1 Setup

This experiments were performed on a device running 64-bit windows 10 with the following specifications: An AMD Ryzen 5 1600 6-core CPU, a Nvidia GeForce GTX 1060 6GB GPU and 16 GB of DDR4-3200 RAM. The Pandas and Sci-kit learn libraries were used to pre-process the dataset and train the different algorithms within a Jupyter Notebook environment.

5.2 Analysis

The analysis of the results obtained by the algorithms provided valuable insights into their performance. Table 5, which represents the results for the IoT23 dataset, shows that the algorithms achieved high levels of accuracy, with scores ranging from 0.954 on the SGD classifier to 0.970 with a Naïve Bayes classifier. This indicates that the models were quite successful in accurately classifying the different anomalies within the network traffic. Additionally, the weighted metrics demonstrated the algorithms' ability to detect anomalies while considering class imbalance in the dataset. The classifiers showed competitive performance across these metrics, with precision ranging from 0.954 to 0.984, recall from 0.942 to 0.970, and the F1-score from 0.943 to 0.969, respectively. Moreover, the results of the macro metrics suggested that the algorithms still performed relatively well in detecting the specific anomalies classes, but to a lesser degree than the weighted metrics. This indicated that it had more trouble correctly classifying the less represented anomalies. Here the precision varied from .883 to .942, the recall from .852 to .933 and the F1-score from .846 to .914.

Table 5. Classifier Results IoT-23

Metrics		**DT**	**XGB**	**NB**	**RF**	**ANN**	**SGD**
Accuracy		.967	.968	.970	.963	.955	.954
Weighted	Recall	.967	.968	.970	.959	.942	.954
	Precision	.982	.984	.973	.982	.974	.954
	F1-score	.968	.969	.966	.959	.943	.954
Macro	Recall	.929	.933	.887	.915	.910	.852
	Precision	.928	.937	.919	.942	.935	.883
	F1-score	.908	.914	.879	.902	.885	.846

Moving on to Table 6, which represents the results for the IoTID20 dataset, a greater variation in performance was observed between the different algorithms. Regarding this dataset, the accuracy scores ranged from 0.723 with Naïve Bayes to a 1.0 accuracy with the XGBoost algorithm. For the IoTID20 dataset the same metrics were used to give insight into the overall performance. For the weighted metrics most of the classifiers had an incredible performance, with precision ranging from .855 to 1.0 , recall from .723 to 1.0 and F1-score from .766 to 1.0. The macro metrics gave similar results with all the classifiers except for SGD and Naïve Bayes giving exceptional scores. For them, the precision ranged from .732 to 1.0, the recall from .713 to 1.0 and the F1-score from .736 to 1.0.

Table 6. Classifier Results IoTID20

Metrics		**DT**	**XGB**	**NB**	**RF**	**ANN**	**SGD**
Accuracy		.999	1.0	.723	.999	.999	.861
Weighted	Recall	.999	1.0	.723	.999	.999	.860
	Precision	.999	1.0	.897	.999	.999	.855
	F1-score	.999	1.0	.766	.999	.999	.848
Macro	Recall	.999	1.0	.862	.998	.999	.713
	Precision	.999	1.0	.732	.999	.999	.829
	F1-score	.999	1.0	.737	.999	.999	.736

5.3 Accuracy

Within the context of the IoT-23 and the IoTID20 dataset, it becomes quite clear that the algorithms that were used within this experiment are able to obtain a high accuracy given the training data. For the IoT-23 dataset Naïve Bayes was the most accurate algorithm. It is highly likely that this is due to the fact that the data within the IoT-23 subdata set that was created is independent. Here XGBoost and the Decision Tree were very close in accuracy. For the IoTID20 dataset the resulting accuracy's looked entirely different, here XGBoost had a perfect accuracy of 100% across all folds whilst in comparison Naïve Bayes performed quite poorly on this dataset. Overall, when taking into account the performance on both datasets, XGBoost is the most accurate algorithm out of the ones used.

5.4 Performance

When considering the overall performance of the models for a real-time use case, it is important to prioritize not only accuracy but also the ability to minimize False Positives and False Negatives in anomaly detection. The amount of anomalies that bypass the model and the amount of benign data that gets wrongly classified as an anomaly should be as little as possible. Considering this importance the algorithm that has the highest F1-score, and thus the best distribution between Precision and Recall, will serve as the most optimal model for a real time setting. Due to the highly differing entries between the classes, the decision was made to value Precision and Recall per respective class and thus the macro F1-score was used as main metric. Regarding the two datasets used, the XGBoost algorithm obtained the highest macro F1-score on both, namely .914 for IoT-23 and 1.0 for IoTID20.

5.5 Differences

In this case, the results of the algorithms clearly differ when trained on different datasets. Comparing the performance of the algorithms on the IoT23 and IoTID20 datasets, it is evident that the performance can vary significantly. The accuracy, weighted metrics, and macro metrics demonstrated varying levels of effectiveness when applied to different datasets. For example, The Naïve bayes algorithm which performed quite well on the IoT-23 dataset had underwhelming results on the IoTID20 dataset. This suggests that the performance of ML/DL algorithms for anomaly detection in IoT network traffic is influenced heavily by the variety of anomalies within the dataset, the distribution of the entries within the dataset and finally the features used while training the model. Therefore, it is important to carefully consider the dataset which is used for training and the algorithms performance regarding the relevant anomalies.

6 CONCLUSION

In conclusion, the findings suggest that XGBoost is the best classifier for anomaly detection in the context of IoT-23 subset and the IoTID20 dataset. It obtained both the highest average scores regarding accuracy and F1-score, however these metrics are as we have deduced highly dependent on characteristics of the dataset and the objectives of the intrusion detection system. So further research and experimentation would be necessary in order to make more conclusive statements regarding IoT anomaly detection as a whole.

REFERENCES

[1] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul. 2022. Generative Deep Learning to Detect Cyberattacks for the IoT-23 Dataset. *IEEE Access* 10 (2022), 6430–6441. https://doi.org/10.1109/ACCESS.2021.3140015 Conference Name: IEEE Access.

[2] Mohamed Abomhara and Geir M Køien. 2015. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. *Journal of Cyber Security and Mobility* (2015), 65–88.

[3] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[4] Maryam Douiba, Said Benkirane, Azidine Guezzaz, and Mourade Azrour. 2023. An improved anomaly detection model for IoT security using decision tree and gradient boosting. *The Journal of Supercomputing* 79, 3 (2023), 3392–3411.

[5] Sebastian Garcia, Agustin Parmisano, and Maria Jose Erquiaga. 2020. IoT-23: A labeled dataset with malicious and benign IoT network traffic. https://doi.org/10.5281/zenodo.4743746

[6] Fatima Hussain, Rasheed Hussain, Syed Ali Hassan, and Ekram Hossain. 2020. Machine Learning in IoT Security: Current Solutions and Future Challenges. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 1686–1721. https://doi.org/10.1109/COMST.2020.2986444 Conference Name: IEEE Communications Surveys & Tutorials.

[7] Nahida Islam, Fahiba Farhin, Ishrat Sultana, M Shamim Kaiser, Md Sazzadur Rahman, Mufti Mahmud, AS Hosen, and Gi Hwan Cho. 2021. Towards machine learning based intrusion detection in IoT networks. *Comput. Mater. Contin* 69, 2 (2021), 1801–1821.

[8] Sohail Jabbar, Farhan Ullah, Shehzad Khalid, Murad Khan, and Kijun Han. 2017. Semantic Interoperability in Heterogeneous IoT Infrastructure for Healthcare. *Wireless Communications and Mobile Computing* 2017 (March 2017), e9731806. https://doi.org/10.1155/2017/9731806 Publisher: Hindawi.

[9] Falaq Jeelani, Dhajvir Singh Rai, Ankit Maithani, and Shubhi Gupta. 2022. The Detection of IoT Botnet using Machine Learning on IoT-23 Dataset. In *2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM)*, Vol. 2. 634–639. https://doi.org/10.1109/ICIPTM54933.2022.9754187

[10] E. Manavalan and K. Jayakrishna. 2019. A review of Internet of Things (IoT) embedded sustainable supply chain for industry 4.0 requirements. *Computers & Industrial Engineering* 127 (Jan. 2019), 925–953. https://doi.org/10.1016/j.cie.2018.11.030

[11] Pascal Maniriho, Ephrem Niyigaba, Zephanie Bizimana, Valens Twiringiyimana, Leki Jovial Mahoro, and Tohari Ahmad. 2020. Anomaly-based Intrusion Detection Approach for IoT Networks Using Machine Learning. In *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*. 303–308. https://doi.org/10.1109/CENIM51130.2020.9297958

[12] Asadullah Momand, Sana Ullah Jan, and Naeem Ramzan. 2023. A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy. *Journal of Sensors* 2023 (Feb. 2023), e6048087. https://doi.org/10.1155/2023/6048087 Publisher: Hindawi.

[13] Monika Roopak, Gui Yun Tian, and Jonathon Chambers. 2019. Deep Learning Models for Cyber Security in IoT Networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*. 0452–0457. https://doi.org/10.1109/CCWC.2019.8666588

[14] Nicolas-Alin Stoian. 2020. *Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set.* B.S. thesis. University of Twente.

[15] Imtiaz Ullah and Qusay H Mahmoud. 2020. A scheme for generating a dataset for anomalous activity detection in iot networks. In *Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33*. Springer, 508–520.

[16] Liang Xiao, Xiaoyue Wan, Xiaozhen Lu, Yanyong Zhang, and Di Wu. 2018. IoT Security Techniques Based on Machine Learning: How Do IoT Devices Use AI to Enhance Security? *IEEE Signal Processing Magazine* 35, 5 (Sept. 2018), 41–49. https://doi.org/10.1109/MSP.2018.2825478 Conference Name: IEEE Signal Processing Magazine.