IoT Botnet Detection and Classification using Machine Learning Algorithms for Improved Cybersecurity

Pham Van Quan[1][0009-0003-1625-0848], Ngo Van Uc[2][0009-0005-0954-5618], Do Phuc Hao[3] [0000-0003-0645-0021], and Nguyen Nang Hung Van[4][0000-0002-9963-7006]

[1, 2] Dong A University, Da Nang, Vietnam
quan10.work@gmail.com, ngovanuc.1508@gmail.com
[3] The Bonch-Bruevich Saint-Petersburg State University of Telecommunications, Saint-Petersburg, Russian Federation
haodp.sut@gmail.com
[4] Danang University of Science and Technology, Da Nang, Vietnam
nguyenvan@dut.udn.vn

Abstract. The Internet of Things (IoT) is becoming increasingly prevalent, but the rise of connected devices has led to security threats such as botnets. Detecting and classifying botnets is a complex task that requires sophisticated machine learning algorithms. In this study, we explore the effectiveness of several algorithms, including decision tree, kNN (k nearest neighbors), random forest, and extreme gradient boosting, for detecting and classifying IoT botnets. Our results show that all four algorithms are effective at detecting and classifying botnets, with the extreme gradient boosting algorithm achieving the highest accuracy but the execution time of Decision Tree is the shortest. This study highlights the potential of machine learning algorithms in detecting and mitigating security threats in IoT devices.

Keywords: Internet of Things, Supervised learning, Intrucsion detection, IoT Botnet, Cybersecurity.

1 Introduction

The Internet of Things (IoT) is a term coined by Kevin Ashton in 1999, referring to the dramatic increase in the number of connected devices installed on the internet. The European Union Agency for Network and Information Security defines IoT as "a cyber-physical ecosystem of inter-connected sensors and actuators, which enable decision making." IoT has been a part of society for several decades, with various sectors using IoT devices and applications on a small scale. IoT offers the ability to integrate devices, data, and applications employing internet protocols[1]. Industrial IoT (IIoT) is the network of intelligent and highly connected industrial components that are deployed to achieve a high production rate with reduced operational costs through real-time monitoring, efficient management, and controlling of industrial processes, assets, and operational time[2].

However, the convergence of isolated systems and technologies in IoT presents significant security challenges. IoT devices often have vulnerabilities in software and communication protocols, in addition to weak physical security and resource constraints. These security concerns emphasize the need for effective security measures to protect against cyber threats and data breaches in IoT systems[3].

The security of the internet is threatened by various attacks, making it essential to design a system to protect data and users. Intrusion Detection Systems (IDS) have been developed to fulfill this need. Network administrators use IDS to prevent malicious attacks and make it an essential part of security management. IDS can detect and report any intrusion attempts or misuse on the network, block malicious attacks, maintain normal performance during an attack, and perform experienced security analysis. IDS is an effective tool for protecting networks and preventing cyber-attacks, making it a crucial component of modern security systems[4].

Developing IDSs using machine learning is taking a new approach. This research is about nine malware captures of the IoT-23 dataset in both binary and multiclass classification scenarios. The Developed models include Decision Tree, K Nearest Neighbor (KNN), Random Forest, and Extreme Gradient Boost (XGBoost). The above models are all powerful in the classification. The analysis can yield positive results and is suitable for intrusion detection.This paper is organized into multiple sections. Section 2 provides a survey of previous work on machine learning techniques for intrusion detection. Section 3 describes the utilized dataset and models, including the data preprocessing steps and evaluation metrics. Section 4 presents an analysis of the results obtained in each scenario. Finally, section 5 addresses the main conclusions and future research topics.

2 Related work

In recent years, research on IoT intrusion detection has gained attention. As cyber-attacks and detection techniques continue to evolve, new research topics emerge. It is crucial to understand the results and conclusions of previous work to advance research in this field. By examining the findings of previous studies, researchers can build on existing knowledge and develop more effective intrusion detection systems for IoT. Understanding the strengths and limitations of previous research can also help researchers identify gaps in the current knowledge and devise new research questions. In this way, previous research serves as a foundation for future work and can contribute to the development of more robust and effective IoT security measures.

J. Hajji et al.[5] applied several unsupervised machine learning algorithms, including k-means, PCA, and autoencoder, to the dataset to detect anomalous network traffic. They found that the k-means and PCA algorithms achieved the best performance in detecting anomalies and the autoencoder algorithm performed well in detecting rare and subtle anomalies in the network traffic.

A. Rahim et al.[6] used statistical analysis to identify the most significant features then applied several machine learning algorithms, including Decision Tree, Random Forest, and Naive Bayes, to classify normal and malicious traffic. The results showed

that the Random Forest algorithm achieved the highest accuracy followed by Decision Tree and Naive Bayes.

F. Alotaibi et al.[7] used a deep learning model based on convolutional neural networks (CNNs) to classify the traffic as normal or malicious. The results showed that the proposed approach achieved high accuracy, precision, recall, and F1-score. The model also outperformed other machine learning algorithms, including decision tree, random forest, and support vector machine.

J. Li et al.[8] used a deep learning model based on long short-term memory (LSTM) and a dense neural network to classify network traffic as normal or malicious. The IDS system achieved an overall accuracy of 99.8%, precision of 99.7%, recall of 99.9%, and F1-score of 99.8%.

Abdallah et al.[9] used a deep learning model based on convolutional neural networks (CNNs) and long short-term memory (LSTM). The model was implemented as a real-time botnet detection system using the Apache NiFi framework. The accuracy of the system for botnets is over 99%.

S. M. Z. Islam et al.[10] developed averaging model and stacking model based on support vector machine, random forest, and gradient boosting algorithms. The system achieved an high accuracy for detecting different types of botnets.

Y. Li et al.[11] developed four individual machine learning models based on decision tree, K-nearest neighbors, logistic regression, and random forest algorithms to classify network traffic as normal or malicious.

A. T. Kiani et al.[12] developed a deep autoencoder neural network to learn the normal behavior of IoT network traffic then used it to detect anomalous behavior in real-time by comparing the input data with the learned normal behavior. The proposed approach achieved high accuracy in detecting various types of IoT network attacks.

S. Rasool et al.[13] experimented with three different transfer learning models, including VGG16, ResNet50, and InceptionV3, which were pre-trained on ImageNet dataset. They fine-tuned the pre-trained models on the IoT-23 dataset for detecting IoT malware. P. H. Do [14] proposes a feature extraction method by dividing the feature sets into different classes, then the author compares the results when performing machine learning algorithms on those attribute classes.

Each of these researches presents different methods for detecting and classifying cyberattacks and all have shown good results. However, some researches have not mentioned the performance of the model as well as the execution time of the model, a few others only present detection or classification.

3 Method

In this section, the dataset will be introduced. Steps to process the data before entering the training and testing model. The first step is data pre-processing. Includes data selection, data visualization, data formatting, and data splitting. The algorithms used for binary and multiclass classification are based on the malware's attack classifier. Finally, the algorithms are compared for accuracy, decision tree, precision, and F1 score.

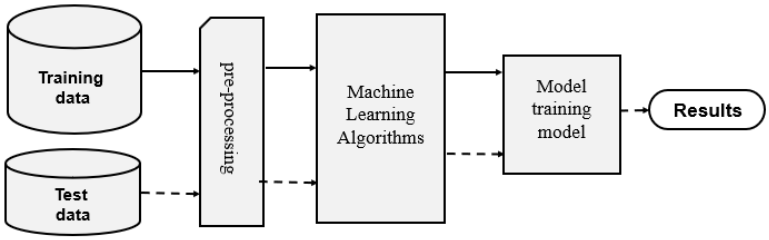

Fig. 1. Proposed model

3.1 Dataset

The dataset used in this project is IoT-23 [15], a dataset created by the Avast AIC laboratory. The dataset contains 20 malware captures from various IoT devices, and 3 captures for benign anomalies. The data set contains a total of 325,307,990 captures, of which 294,449,255 are malicious. The data set registered the following types of attacks:

Table 1: The types of attacks present in the data set

Types of attack	Explaination
Attack	the generic label that is attributed to anomalies that cannot be identified
Benign	generic label for a capture that is not suspicious
C&C	control and command, a type of attack which takes control of the device in order to order it to perform various attacks in the future
C&C- FileDownload	the server that controls the infected device is sending it a file
C&C- Mirai	the attack is performed by the Mirai bot network
C&C- Torii	the attack is performed by the Torii bot network, a more sophisticated version of the Mirai network
DDoS	the infected device is performing a distributed denial of service
C&C- HeartBeat	the server that controls the infected device sends periodic messages the check the status of the infected device, this is captured by looking for small packages being sent periodically from a suspicious source
C&C-HeartBeat Attack	the same as above, but the method is not clear, only the fact that the attack is coming periodically from a suspicious source
C&C-HeartBeat FileDownload	the check-up is done via a small file being sent instead of a data packet
C&CPartOfAHorizontalPortScan	the network is sending data packages in order to gather information for a future attack
Okiru	the attack is performed by the Okiru bot network, a more sophisticated version of the Mirai network
OkiruAttack	the attacker is recognized as the Okiru bot network, but the method of attack is harder to identify

PartOfA HorizontalPortScan	information is gathered from a device for a future attack
PartOfA HorizontalPort Scan-Attack	the same as above, but methods that cannot be identified properly are used

Each of the conn.log.labelled files contain 23 columns of data, whose types are presented in table 1. These columns are:

Table 2: The types of information in the data set

Column	Description	Type
ts	the time when the capture was done, expressed in Unix Time	int
uid	the ID of the capture	str
id_orig.h	the IP address where the attack happened, either IPv4 or IPv6	str
id_orig.p	the port used by the responder	int
id_resp.h	the IP address of the device on which the capture happened	str
id_resp.p	the port used for the response from the device where the capture happened	int
proto	the network protocol used for the data package	str
service	the application protocol	str
duration	the amount of time data was traded between the device and the attacker	float
orig_bytes	the amount of data sent to the device	int
resp_bytes	the amount of data sent by the device int conn_state the state of the connection	str
local_orig	whether the connection originated locally bool local_resp whether the response originated locally	bool
missed_bytes	number of missed bytes in a message	int
history	the history of the state of the connection	str
orig_pkts	number of packets being sent to the device	int
orig_ip_bytes	number of bytes being sent to the device	int
resp_pkts	number of packets being sent from the device	int
resp_ip_bytes	number of bytes being sent from the device	int
tunnel_parents	the id of the connection, if tunnelled	str
label	the type of capture, benign or malicious	str
detailed_label	if the capture is malicious, the type of capture, as described above	str

In this dataset any missing data cells will be highlighted - except for IP addresses marked with double colons "::".

3.2 Preprocessing and Formatting Data

Before the data can be fed into the training model, several steps must be taken to pre-process the data. The data from the sources will be accessed along the path to the destination.

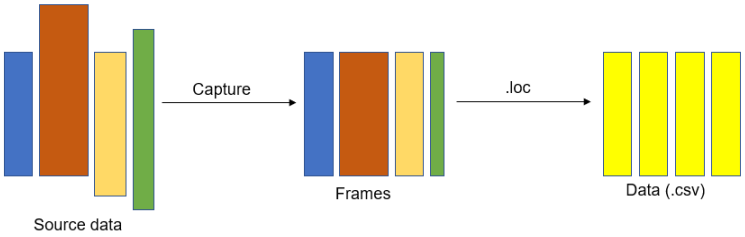

Fig. 2. The steps of pre-processing data

We need to capture and save to the source code. Now each variable will store a destination path. We get the frames, then use the .loc function from the library to get the columns with Labels to process. Now our data can be saved as csv so that it can be easily processed and reused whenever needed. Python has a support library that makes it easy to read and process data in csv format. Below is a schematic representation of data preprocessing. We then encode the labels for easier training and testing.

3.3 Analysis Method

Decision Tree

Decision trees are a type of supervised learning model that can solve both classification and regression problems. The structure of a decision tree involves nodes that represent variables, and branches that represent the relationship between these variables. Each leaf node represents the predicted value of the target variable, given the values of the variables along the path from the root node to that leaf node. The calculation and update of node values use two formulas:

The entropy of probability distribution $\boldsymbol{p} = (p_1, p_2, p_3, \dots, p_n)$ satisfied $\sum_{i=1}^{n} p_1 = 1$

$$H(p) = -\sum_{i=1}^{n} p_i \times \log p_i \qquad (1)$$

The information gain is a node test formula that yields the amount of that node information that remains after switching to $\boldsymbol{k}$ child node.

$$Gain(p) = H(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} \times H(i)\right) \quad (2)$$

Random Forest

Random Forest (RF) is a supervised classification algorithm that averages the results of a number of decision trees. The number of decision trees increases the accuracy of the algorithm. The advantage of this algorithm is that the accuracy increases over time. Compared to other similar algorithms it is computationally lighter when compared to other algorithms.

K-Nearest Neighbor (KNN)

KNN is one of the supervised learning algorithms widely used in data mining and machine learning. The idea of this algorithm is that it doesn't learn anything from the learning dataset (so KNN is classified as lazy learning), all computations are performed when it needs to predict the labels of new data. A new data object's label can be predicted from the labels by getting the mean of ***k*** labels nearest.

$$label\ of\ new\ data = \frac{\sum_{i=1}^{k} label(i)}{k} \qquad (3)$$

Extreme Gradient Boosting (XGBoost)

XGBoost (XGB) is the most advanced algorithm for solving supervised math problems with a fairly high level of accuracy besides Deep learning model. XGBoost takes as input a tabular tuple of any data size and format including classifications. XGboost used in this project has a much faster training speed than other algorithms.

3.4 Performance Evaluation

To evaluate the algorithmic models used above, we use a number of methods to measure the accuracy of the results from which to draw general conclusions for each model used.

Some of the concepts are presented below:

- TP is the actual number of positives that have been determine exactly
- TN is the actual number of negatives that have been determine exactly
- FP is the actual number of positives that have been identified as negative
- FN is the actual number of negations that have been obtained identified as positive

Precision

The precision is a metric that evaluates the model by calculating the fraction of correctly identified positives. Its formula is:

$$precison = \frac{TP}{TP + FP} \qquad (4)$$

Accuracy

The accuracy is a metric that evaluates the model by calculating the fraction of correct predictions over the total number of predictions. Its formula is:

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (5)$$

Recall Score
The recall score is a metric that evaluates the model by calculating the fraction of actual positives that were correctly identified. Its formula is:

$$recall = \frac{TP}{TP+FN} \quad (6)$$

F1-Score for Binary class
The F1-score is a calculated average that harmonizes accuracy and recall score. It is considered an extension of the precision measure. Because it takes into account both false positives and negatives. Its formula is:

$$F1 - score = 2 \times \frac{precision \times recall}{precision + recall} \quad (7)$$

F1-Score for Multi-class
For multi-class classifiers, there are two options: micro-averaging (taking into account the frequency of each class) and macro-averaging (all classes are counted equally). Macro-averaging is more suitable for data sets that are relatively uniform in size, while micro-averaging is more suitable for data sets has a large disproportion between the sizes of the classes[16]. The formulas for them are:

$$F1 - micro = \sum_{i=1}^{n}[f1(i) \times \frac{supportscore(i)}{setsize}] \quad (8)$$

$$F1 - macro = \sum_{i=1}^{n}[F1(i) \times \frac{1}{n}] \quad (9)$$

Where n is the number of classes and setssize is the total number of predictions made. For the results of this paper, the formula with micro-average is used because there is a clear disproportion in the size of the classes from the data set.

4 Results and Discussion

4.1 Binary Classification

The process of training and testing the model gives us different results. Overall, the difference between these results is not too much. The results of the accuracy and training time of each model are presented in the table below:

Table 3: Evaluation of binary classification

Evaluation	Decission Tree	Random Forest	KNN	XGB

Accuracy	0.999	0.895	0.996	0.998
Precison	1.0	0.880	1.0	1.0
Recall	1.0	0.860	0.990	1.0
F1	1.0	0.870	1.0	1.0
Time (s)	5.9	40.2	58.2	99.4

In the binary classification table, Decision Tree gives the highest results and the shortest execution time. Decision Tree works well on data with high error due to the "yes" or "no" mechanism. On the contrary, KNN is an algorithm that completely uses data properties, the difference in data between classes leading to the mean value causes high errors. XGB is a powerful algorithm so it gives good results but the execution time is quite long making it difficult to apply in real.

4.2 Multi-classificaion

The results of these models with multilayer attack are similar to the results presented above. The results of the Random Forest model are lower than that of other models. And the time to train models for multithreaded attack is also longer. Below is a table of results and training time for each model:

Table 4: Evaluation of multi-classification

Evaluation	Decission tree	Random forest	KNN	XGB
Accuracy	0.999	0.782	0.998	0.999
Precision	0.999	0.596	0.997	1.0
Recall	0.997	0.461	0.987	0.998
F1	0.996	0.498	0.992	0.999
Time (s)	11.5	206.1	85.2	956.2

4.3 Models evaluation

Performance evaluation is a method to measure whether the usage model fits the original problem or not. The following methods used are accuracy, precission, recall, F1-score.

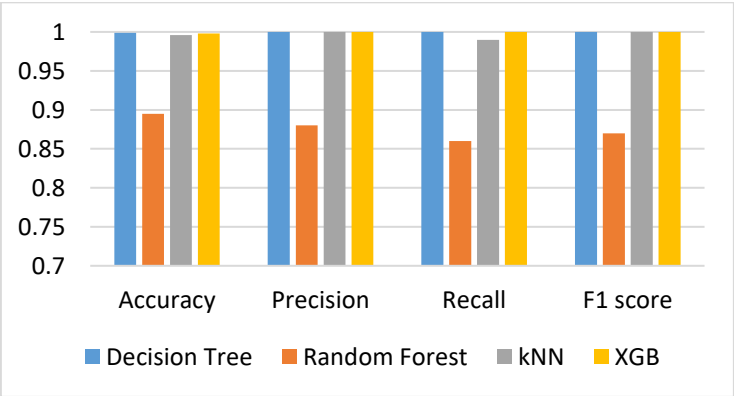

Fig. 3. Accuracy, F1-score, Precision and Recall of some algorithms for binary classification.

In figure 3, the accuracy, precision, Recall and F1-score values of the algorithms are Decision tree, Random forest, kNN and XGB respectively for the binary classification problem. The results show that Decision tree and XGB algorithms give very good and better results than the rest of the algorithms. However, the time used to train the model, the Decision tree algorithm achieves better performance.

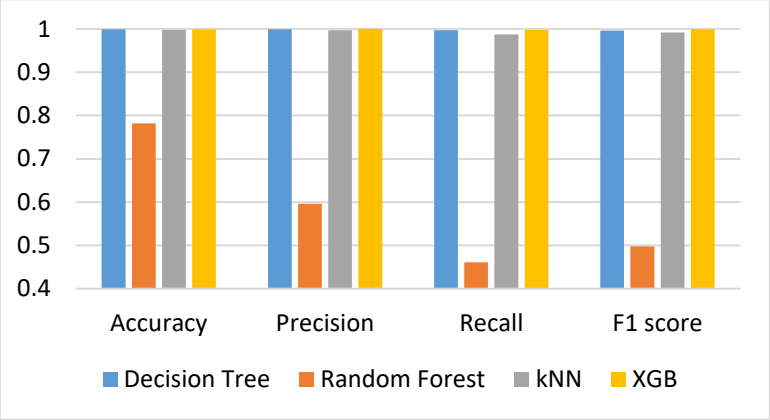

Fig. 4. Accuracy, F1-score, Precision and Recall of some algorithms for multi-classification.

In figure 4, the accuracy, precision, Recall and F1-score values of the algorithms are Decision tree, Random forest, kNN and XGB respectively for the multiclass problem. The results show that the Decision tree algorithm gives better results than the remaining algorithms.

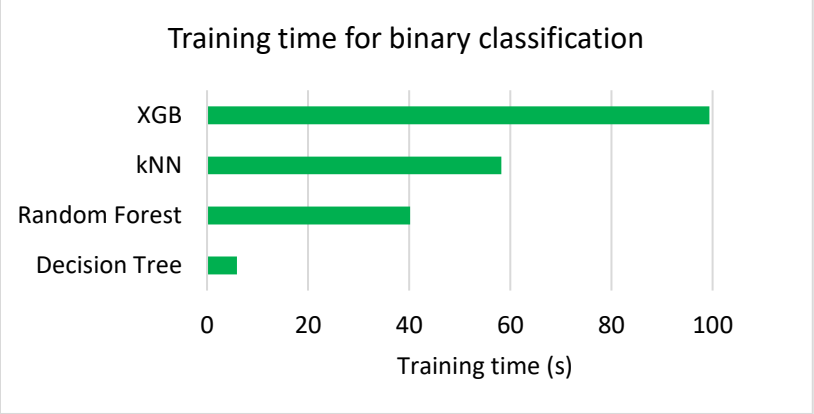

Fig. 5. Training time for binary classification

The figure 5 shows the training time of the algorithms for the binary classification problem. The results show that the training time of the XGB algorithm is very long compared to the rest of the algorithms. And the training time of the decision tree algorithm is the best.

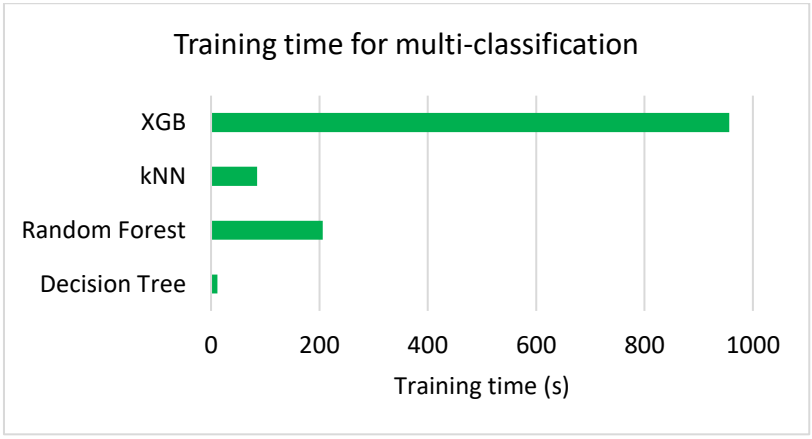

Fig. 6. Training time for multi-classification

Figure 6 shows the training time of the algorithms for the multi-classification problem. The results show that the training time of the XGB algorithm is very long compared to the rest of the algorithms and nearly 100 times that of the decision tree algorithm. And the training time of the decision tree algorithm is the best.

5 Conclusions

The development of machine learning models for IoT intrusion detection is relatively new. However, the results of the study were significant. In this study, decision tree, random forest, k-nearest neighbor, and extreme gradient boosting are some of the popular machine learning algorithms that have been used on the Iot-23 dataset to evaluate the performance of nine malware captures bot. The results of these studies have generally shown that decision tree, random forest, k-nearest neighbor, and extreme gradient boosting can all be effective for IoT intrusion detection on the IoT-23 dataset. However, the performance of these algorithms can vary depending on the specific features and parameters used, as well as the type of IoT devices in the dataset.

Overall, the use of machine learning algorithms for IoT intrusion detection on the IoT-23 dataset shows promise for improving the security of IoT networks. However, more research is needed to further evaluate the performance of these algorithms and to develop more accurate and effective intrusion detection systems for IoT networks.

References

1. Williams, P., Dutta, I.K., Daoud,. H, & Bayoumi, M.: A survey on security in internet of things with a focus on the impact of emerging technologies. Elsevier. (2022).
2. Khan, W.Z., Rehman, M.H., Zangoti, H.M., Afzal, M.K., Armi, N., & Salah, K.: Industrial internet of things: Recent advances, enabling technologies and open challenges. Elsevier. (2020).

3. Vitorino, J., Andrade, R., Praça, I., Sousa, O., & Maia, E.: A Comparative Analysis of Machine Learning Techniques for IoT Intrusion Detection. Foundations and Practice of Security (pp.191-207).Springer.(2022).
4. Haq, N.F., Onik, A.R., Hridoy, M.A.K, Rafni, M., Shah, F.M., & Farid, D.M.: Application of Machine Learning Approaches inIntrusion Detection System: A Survey. Article Published in International Journal of Advanced Research in Artificial Intelligence(IJARAI), Volume 4 Issue 3. (2015).
5. Hajji, J., Khalily, M., Moustafa, N., & Nelms, T. IoT-23: A Dataset for IoT Network Traffic Analysis. Springer. (2019).
6. Rahim, A., Razzaque, M.A., Hasan, R., & Hossain, M.F. Effective IoT Network Security through Feature Selection and Machine Learning Techniques. IEEE. (2020).
7. Alotaibi, F., Al-Qaness, M.A., Abunadi, A., & Alghazzawi, M.A. A Deep Learning Approach for Intrusion Detection in IoT Networks using the IoT-23 Dataset. IEEE. (2020).
8. Li, J., Hu, C., Yang, K., Zhang, X., & Lu, J. An IoT-23 based IoT Intrusion Detection System using Deep Learning. IEEE. (2020).
9. Abdallah, A., Khalil, I., Al-Emadi, N., Almohaimeed, A., & Kim, H. Real-Time IoT Botnet Detection Using Deep Learning on IoT-23 Dataset. IEEE. (2020)
10. Islam, S.M.Z, Bhuiyan, M.Z.H, & Hasan, R. Fusion of Machine Learning Models for Intrusion Detection in IoT Networks using the IoT-23 Dataset. IEEE. (2020)
11. Li, Y., Qiu, L., Chen, Y., & Chen, Y. Ensemble-based Intrusion Detection System for IoT Networks using the IoT-23 Dataset. IEEE. (2020)
12. Kiani, A.T., Abbas, R.A., Abbasi, A.Z., & Khan, M.K. Deep Learning-based Anomaly Detection for IoT Networks using the IoT-23 Dataset. IEEE. (2020)
13. Rasool, S., Saeed, S., Farooq, F., & Madani, A. A Comparative Study of Transfer Learning Approaches for IoT Malware Detection Using IoT-23 Dataset. IEEE. (2021).
14. P. H. Do, T. D. Dinh, D. T. Le, V. D. Pham, L. Myrova and R. Kirichek, "An Efficient Feature Extraction Method for Attack Classification in IoT Networks," 2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, Czech Republic, 2021, pp. 194-199, doi: 10.1109/ICUMT54235.2021.9631726.
15. Garcia, S., Parmisano, A., & Erquiaga, M.J. IoT-23: A labeled dataset with malicious and benign IoT network traffic. Stratosphere IPS. (2020).
16. Stoian, N.A. Machine Learning for Anomaly Detection in IoT Networks : Malware analysis on the IoT-23 data set. EEMCS: Electrical Engineering, Mathematics and Computer Science. (2020)