The Internet of Things (IoT) has become an integral requirement to equip common life. According to IDC, the number of IoT devices may increase exponentially up to a trillion in near future. Thus, their cyberspace having inherent vulnerabilities leads to various possible serious cyber-attacks. So, the security of IoT systems becomes the prime concern for its consumers and businesses. Therefore, to enhance the reliability of IoT security systems, a better and real-time approach is required. For this purpose, the creation of a real-time dataset is essential for IoT traffic analysis. In this paper, the experimental testbed has been devised for the generation of a real-time dataset using the IoT botnet traffic in which each of the bots consists of several possible attacks. Besides, an extensive comparative study of the proposed dataset and existing datasets are done using popular Machine Learning (ML) techniques to show its relevance in the real-time scenario.

The rest of this paper is arranged as follows. In Section 2, we have mentioned the contributions of our scheme. In Section 3, we go through some of the existing schemes and related work. Section 4 contains our proposed methodology. Section 5 highlights the description of the selected datasets for comparative analysis. Section 6 contains the results obtained from our proposed method and its comparison with existing datasets. Finally, Section 7 concludes our research work.

### Related Work

As per the advancement of IoT, many real-world smart technologies have recently attracted the attention of different manufacturers to enhance the quality of human existence. Because IoT devices have limited resources and the absence of embedded security mechanisms, attackers may easily target each device and the entire IoT system. As a result, the attacker’s goals shifted from attacking people to attacking the organization. It has necessitated the development of an effective IoT botnet analysis system to discover new, unknown, and groundbreaking botnets in IoT devices.
The major issue in today's IoT botnet analysis is finding invisible and cryptic malicious files or programs. As malware programmers develop various bypassing strategies for information tucking away to avoid detection. Also, with the massive rise in internet users, cyberattacks, including zero-day attacks, have a disastrous influence on people's lives regarding effectiveness and scale. In this section, we have discussed the past research work in two aspects; one is related to IoT botnet dataset generation, and the second is IoT botnet detection framework using machine learning models.

Here, we described the popular dataset generations in the field of IoT botnet. As per the advancement of IoT, a plethora of real-world smart technologies has recently attracted the attention of different manufacturers to enhance the quality of human existence. Because IoT devices have limited resources and also the absence of embedded security mechanisms, attackers may easily target each device as well as the entire IoT system. As a result, the attacker's goals shifted from attacking people to attacking the organization. It has necessitated the development of an effective IoT botnet analysis system to discover new, unknown, and groundbreaking botnets in IoT devices.

Mrutyunjaya Panda11 developed a well-structured dataset that is the UNSW-NB15 dataset. It was utilized to classify cyberattacks in this scheme. This present dataset is based on VM-simulation datasets rather than real-time scenarios. It focused on detecting IoT botnets fast and accurately rather than diving deep into analyzing novel IoT botnets like Mirai. To overcome this, Koroniotis N12 proposed the BoT_IoT dataset, it includes some real and virtual IoT network traffic, as well as several kinds of attacks. A limited number of ML and DL models have been applied to evaluate the results that lack the detection feature of IoT botnet.

Hai-Viet Le13 proposed IoT-BDA framework to combat the smartness of the IoT botnets. The IoT-BDA architecture comprises a variety of honeypots and newly evolved sandboxes, and it is made up of blocks such as BCB and BAB. BCB and BAB are still being improvised to make them more dependable and scalable over various CPU architectures and forthcoming IoT devices. Valerian Rey14 presented a paradigm based on a new technology known as Federated Learning (FL) to overcome this. The present scheme was conducted using the N-BaIoT dataset, which comprises network traffic across various IoT devices that have been tainted by malware. Several adversarial setups with multiple malicious individuals were contacted to test the Federated Learning-based framework's resilience, robustness, and scalability. The model's performance is still being observed, as it is compared with many updated malware versions.

Jun Jeon15 devised a DAIMD model that has been developed to identify both existing IoT Malware and new varieties that are being developed. DAIMD works by first understanding the malware's behavior on data and then acting on it. The DAIMD model falls short in a VM where IoT malware avoids the malware analysis phase, necessitating a hybrid analytic technique. Segun I. Poopla16 has developed a Hybrid analysis model based on Deep Learning (DL) approaches to overcome this. The DL applied on the BoT_IoT dataset is a completely VM-based simulation that lacks the features and parameters compared to Real-Time datasets.

Javed Ashraf17 brought up an IoTBoT-IDS model. The IoTBot-IDS examines the usual behavior of an IoT network data traffic using statistical learning techniques such as Corentropy and BMM. This model's study is still to be used in smart controllers for SDN and by incorporating additional AI, ML, and DL approaches. The analysis is on a statistical approach and must be ML/DL-based to be more effective. To overcome this, Segun I. Poopla18 devised the RNN model for identifying network traffic in unbalanced network traffic in addition to ML or DL approaches. Only one dataset was used for the classification and its comparison with other results. Rather than analyzing IoT botnet variants, the experimental analysis was compared with other ML/DL models.

To address all the problems mentioned above requires an intelligent IDS to detect IoT botnet.

**Proposed Methodology**

The IoT botnet architecture as explained in Fig. 2 demonstrates the complete procedure of testbed setup and attacking procedure of the IoT botnet. The steps followed for its execution are as follows:

- **Build your own botnet**—The attacker will create the IoT botnet using BYOB, which will produce bots, before sending it to the C&C server.

- **C&C server (attacker)**—The IoT botnet will be shared with the linked IoT devices with the help of the local admin.

- **IoT devices (victim)**—The data received from the server will be executed by the IoT devices.
**Dataset analysis.**—The datasets are produced using the Wireshark network analyzer tool and then categorized using the CatBoost and other six different models as specified in phase 4 of the proposed model.

**Build Your Own Botnet**

The entire architecture is built upon IoT botnets, which will be produced with the help of the BYOB tool. The BYOB-generated bots include 13 different features of malware exploitation that will be activated once the bots are installed on IoT devices. The following steps include the workflow of Phase 1.

1. **Step 1:** Using Linux operating system, the attacker will get access to the BYOB web-GUI.
2. **Step 2:** The attacker will create an IoT botnet by accessing the BYOB Payload generating section and choosing the appropriate architecture of the victim’s computer that he intends to target.
3. **Step 3:** Once the IoT botnet has been successfully installed on the victim’s computer, the attacker will generate bots by selecting any one of the 13 specific exploitation modules by visiting the control panel area of the BYOB web-GUI.
4. **Step 4:** Finally, the IoT botnets will be put onto the C&C server in preparation for the attack on the IoT devices.

**C&C Server (Attacker)**

On the C&C server, IoT botnets are loaded. The server must give essential updates or distribute any files with all local or distant users that have access to this server. The sequence of events for phase 2 is outlined in the steps below.

1. **Step 1:** The IoT botnets are loaded on this server without the local administrator being informed, and this operates as a C&C server under the attacker’s control.
2. **Step 2:** Even though the local administrator is unaware of the attack, he just sends the appropriate files and updates to the local server as per its schedule.
3. **Step 3:** The C&C server will then forward (or broadcast) the updates or files associated with the linked IoT botnet to all local IoT devices.
4. **Step 4 (Impacts):** The attacker who is monitoring the server learns about the files or the changes that are sent to victims via local admin. The attacker’s method is depicted in the sequential flow diagram as mentioned in Fig. 3.

As shown in Fig. 3, the attacker starts by spoofing the source address and sending a large number of TCP packets to the victim, making it appear arbitrary or even originating from a legitimate source. Figure 3 shows the involvement of the attacker where (s)he creates a bash script (.bat) and sends it to the victim over FTP (FileZilla). The bash script’s function is to start an auto-download and installation operation in the background as soon as the victim double-clicks the file, without the victim being aware of the attack. The bash script used here is written in the proximity of the attacker by keeping in mind that Window10 installed on Raspberry Pi 3B+, doesn’t treat it as malware.

**IoT Devices (Victim)**

The server’s updates or its files are received through wi-fi on both of the IoT devices, which are Raspberry Pi 3B+. The victim will get notification of the received data as an update or its file from the server on most priority to install it. The IoT botnet attack will
auto-initialize as soon as it is installed on the victim’s device when s (he) will install the updates or double-click on the files to open it. On the other hand, the attacker is watching the entire process. S(he) will first halt the traffic flood into the victim’s device and then use the BYOB web-GUI’s control panel to select one of the 13 exploitation modules and load the bots appropriately. On the victim’s end, the attacker has complete access to the services provided by IoT devices and hence can be easily exploited. The bot services will then operate in the background on the victim’s devices, resulting in an IoT botnet attack against IoT devices.

Besides, the Wireshark network protocol analyzer tool was utilized to construct our new IoT botnet dataset i.e., RI_BoT, and analyze network traffic.20 We’ll build a .csv dataset using this tool for IoT botnet analysis on IoT devices.

Dataset Analysis

Our new RI_BoT dataset has been created as a result of the aforementioned processes. We will analyze it mainly using the CatBoost classifier model including other six different models namely Logistic Regression, Decision Tree, Support Vector Machine, Neural Network, Gradient Boosting, XGBoost in this phase. The CatBoost model will assist us in determining the results using BYOB with accuracy, precision, recall, and F1 score in IoT devices.

Classification Models and Ensemble Learning

Different classification models and other ensemble learning approaches that we have applied to the datasets have been discussed below.

Classification models.—The classification methods which we have used, are supervised learning techniques that utilize training data for identifying certain classification criteria of newer inferences. It uses ML techniques to figure out how to classify datasets from the concerned domain with the class label. Targets, labels, and categories are also the terms that can be used to define the classes. For example, attack or normal. There are varieties of classical ML and ensemble learning approaches that may be used. Several of them are mentioned below:

Logistic Regression: This technique uses the dataset that is divided into the training ratio of 70 percent and the testing ratio of 30 percent respectively. Since the logistic regression uses a sigmoid function, the default threshold could be 0.5 for the appropriate measure. For the hyperparameter tuning, “fit_intercept” is evaluated false as the maximum iteration is 10. The predicted mean value of Y for all X = 0 is typically referred to as the intercept in logistic regression. In the regression equation, start with a single variable X. An intercept is just Y’s expected mean value at a certain time if X = 0. After running it to 10 iterations, the model achieves an F1—score and an Accuracy of 97.5% and 97.4% respectively.

Decision Tree: This technique has the advantage that we do not need to standardize the data (mean = 0 and variance = 1). As previously, the dataset provided to the scheme has been used in similar divisions. Hyperparameter tuning for this algorithm is also done, such as the maximum depth is set to “4”. When the tree depth is reached at its maximum depth during the creation of the very first decision tree, the maximum tree depth is set to cease for further node splitting. Further, the “min_weight_fraction_leaf” is set to “0.04.” The leaves node of the decision tree should get the same proportion of input samples where the sample weight determines different weights; this is the way out to cope with class imbalance. The splitter is set to random. This procedure is continued across the training process until the homogeneity in the nodes is left. It was the most probable reason that the decision tree works so well. After hyperparameter tuning the model we got an F1—score and an Accuracy of 91% and 91.7% respectively.

Support Vector Machine: Here for the input dataset, SVM with the linear kernel is used. Whenever a data collection contains a significant number of characteristics, it is mostly used. Maximum iteration is set to 10 for the dataset which yields an F1—score and an accuracy of 62.4% and 57.3% respectively.

Neural Network: This contains a set of algorithms that seek to find the hidden patterns inside the dataset as used in previous algorithms. The maximum number of iterations used by this algorithm is 50. After that, the model is yielding more validation loss. Another hyperparameter is “Adam” (solver), which is essentially used to solve weight optimization. As Adam is a DL training model that is used instead of stochastic gradient boosting. An “Adam” combines the best aspects of the RMSprop and AdaGrad techniques to provide an optimized solution for noisier problems with limited gradients. The models get an F1—score and an Accuracy of 66.31% and 61.32% respectively.

Ensemble learning.—It’s a generic ML approach that aims to improve prediction results by integrating predictions received from various models. It is mostly used to improve a model’s performance or lessen the chances of picking a bad one randomly. Ensemble learning may also be used to provide a confidence level to the model’s choice, picking optimal (or near ideal) attributes, data acquisition, reinforcement learning, stochastic learning, and error correction. We have used the following ensemble learning approaches for finding the useful results:

Gradient Boosting: This technique employs numerous weak classifiers to create a robust regression and classification model. For hyperparameter tuning in this algorithm, the learning rate is set

Figure 4. Model architecture of the Machine Learning.
to 0.001. This algorithm has a better learning rate by capitalizing on the cost of a semi overall weight matrix. For optimizing the result in terms of global weight the slower learning rate is expected which may be at the cost of the time taken in learning. With the help of this tuning, the model achieves an F1—score of 96.4% and an accuracy of 96.3%.

**XGBoost:** This algorithm dominates comparatively for providing a result on structured or tabular data. It is a high-speed and high-performance version of gradient boosted decision trees. In this algorithm, the learning rate option is used to determine how new trees are weighted. The learning rate is set to 0.01. The model gives an F1—score of 96.4% and an accuracy of 96.3%.

**CatBoost:** This technique generates cutting-edge findings without the need for extensive data training, which is required by the conventional ML approaches. The model is trained on 70% of the total data points. The maximum depth is taken as 3. It is advisable to take the maximum depth in the case of the CatBoost algorithm, in the range of 3 to 10, to get good accuracy and get rid of overfitting situations. The learning rate is taken as 0.001. It is kept small so that it can find local minima more efficiently. The maximum number of iterations is 5, which means the model will iterate 5 times, and whatever the best result will come, that will consider as output. Another hyperparameter of CatBoost is taken as default. With the help of this tuning, the model gets an F1 – score and an accuracy of 96.4% and 96.2% respectively.

**Dataset Description**

**UNSW_NB15:** UNSW_NB15 dataset’s unprocessed network traffic was created in UNSW Canberra’s Cyber Range Laboratory using the IXIA PerfectStorm technology for generating the mixture of genuine normal and simulated modern attacking activities. Using the tcpdump software, 100 Gigabytes of raw form of traffic were recorded that is Pcap files. The UNSW_NB15 dataset contains 9 distinct forms of attacks including worms, analysis, shellcode, fuzzers, exploits, generic, DoS, reconnaissance, and backdoors. The Bro-IDS and Argus tools are used to create a total number of 49 features associated with class labels, and 12 methods are created. The UNSW_NB15 features.csv dataset describes these below features.

| Table I. Classification results of various IoT botnet. |
| MOD | Datasets | Accuracy | Precision | Recall | F1 |
|------|---------|---------|-----------|-------|----|
| Logistic Regression | UNSW_NB15 | 69.635 | 0.756 | 0.667 | 0.708 |
| | BoT_IoT | 78.689 | 0.742 | 0.879 | 0.804 |
| | RI_BoT | 97.444 | 0.955 | 0.996 | 0.975 |
| Decision Tree | UNSW_NB15 | 99.820 | 0.998 | 0.998 | 0.998 |
| | BoT_IoT | 96.104 | 0.943 | 0.981 | 0.961 |
| | RI_BoT | 91.733 | 0.998 | 0.837 | 0.910 |
| Support Vector Machine | UNSW_NB15 | 58.597 | 0.607 | 0.719 | 0.658 |
| | BoT_IoT | 54.300 | 0.522 | 0.999 | 0.686 |
| | RI_BoT | 57.359 | 0.994 | 0.531 | 0.624 |
| Neural Network | UNSW_NB15 | 54.717 | 0.594 | 0.674 | 0.521 |
| | BoT_IoT | 97.477 | 0.989 | 0.960 | 0.974 |
| | RI_BoT | 61.322 | 0.752 | 0.553 | 0.663 |
| Gradient Boosting | UNSW_NB15 | 99.888 | 0.998 | 0.999 | 0.999 |
| | BoT_IoT | 92.477 | 0.890 | 0.969 | 0.927 |
| | RI_BoT | 96.342 | 0.935 | 0.996 | 0.964 |
| Xgboost | UNSW_NB15 | 99.893 | 0.998 | 0.999 | 0.999 |
| | BoT_IoT | 92.477 | 0.891 | 0.969 | 0.929 |
| | RI_BoT | 96.342 | 0.935 | 0.996 | 0.964 |
| Catboost | UNSW_NB15 | 99.983 | 0.998 | 0.994 | 0.999 |
| | BoT_IoT | 96.645 | 0.985 | 0.947 | 0.965 |
| | RI_BoT | 96.290 | 0.934 | 0.996 | 0.964 |

**Figure 5.** Confusion Matrix for RI_BoT dataset.

The Training.csv and Testing.csv are two partitions from this dataset that were specified as training and testing sets. The overall size of the data in the dataset is 2 million and 540,044 records. The training part and testing part consists of 175,341 records as well as 82,332 records respectively and that includes various forms, including normal and attack. The dataset consists of GT.csv which is the ground-truth table, while the EVENTS.csv is the collection that includes complete events file. The UNSW_NB15 simulated dataset is 100Gigabytes in size. This dataset is composed of PCAP files including a mix of typical and invasive traffic as well as other significant characteristics: Analysis, Worms, Generic, Fuzzers, Shellcode, Exploits, DoS, Reconnaissance, and Backdoors are among the IoT botnet attack types covered by the UNSW_NB15 dataset.

**BoT_IoT:** In UNSW Canberra’s Cyber Range Lab, the BoT_IoT dataset is generated by simulating a practical real-time environment. In this environment, there is a mixture of normal and attack network traffic. This dataset’s source files come in
several formats that include raw pcap files and the .csv files. Furthermore, these files are divided into attack types and subtypes to make the tagging process easier. With almost 72,000,000 records, the recorded pcap files are approximately 69.3 Gigabytes in size. In CSV format, the captured flow traffic is 16.7 Gigabytes in size. Data Exfiltration, Service Scan, OS, DoS, Keylogging, and DDoS attacks are all included in the dataset. DoS and DDoS attacks are being further categorized on the basic protocols used. The BoT_IoT dataset has been extracted by 5% of the entire dataset by using MySQL queries to enable handling the dataset easily. The 5% dataset is made up of four files with a combined size of 1.07 Gigabytes and around 3 million entries.

**Results and Discussion**

In this section, the performance metrics that are used to analyze the performance of three different datasets i.e., RI_BoT dataset, Bot_IoT dataset, and UNSW_NB15 dataset are described. The below Fig. 4 is the model architecture of the ML that has been applied on our datasets. We have measured the performance based upon the parameters like Precision, Accuracy, Recall, Confusion Matrix, and F1 score. These results have further been classified using the above-mentioned classification models and further used to compare the efficiency of the three datasets. We present such findings to determine the best suitable among three datasets for the best performance and to present the confusion matrix for the best classification models performance.

**Performance Parameters**

Performance Metrics refers to the results of the data models to evaluate the performance of datasets. Based upon the standard parameters like Precision, Recall, Accuracy, Confusion Matrix, F1
The simplest intuitive performance metric is accuracy, which is just the number of expected observations divided by the total number of observed observations.

\[
\text{Accuracy} = \frac{\text{Number of Predicted Observations}}{\text{Total Number of Observation}}
\]

**Precision**: The ratio of precisely anticipated true observations divided by the total number of true positives observations, with a high False Positive score is known as Precision.

\[
\text{Precision} = \frac{\text{Number of Predicted Observations}}{\text{True Positive} + \text{False Positive}}
\]

**Recall**: The ratio of precisely anticipated true observations divided by the total number of true observations, with a high False Negative score is known as Recall.

\[
\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}
\]

**F1 Score**: The F1 Score is the result of averaging Precision and Recall. As a result, this score considers both FP and FN values.

\[
\text{F1 Score} = \frac{2 \times (\text{Recall} \times \text{Precision})}{\text{Recall} + \text{Precision}}
\]

**Confusion Matrix**: The confusion matrix isn’t a criterion for evaluating a model, but it can help you understand the insights of the predictions. The accurate and wrong (i.e. T (True) or F (False)) predictions for every class in the confusion matrix, extend beyond classification results. It is a 2-D matrix in the context of the binary classification problem and a 3-D matrix when there are 3 independent classes, etc.

**ROC (Receiver Operating Characteristic Curve)**: It’s a statistics graph for the evaluation of binary classification tasks. It is the probabilistic slope that compares TPR Rate versus FPR Rate at various threshold levels, successfully isolating the noisy data as per below Fig. 4. It shows how a classifier’s diagnostic performance changes as the classification threshold changes.

### Evaluation

This section highlights the different Classification models and Ensemble Learning technique’s results on several performance criteria that are Precision, Accuracy, F1 Score, and Recall. We evaluated and compared the efficiency of the best-performing dataset which is the BoT_IoT dataset and UNSW_NB15 with our new RI_BoT dataset. Overall performance metrics of three IoT botnet datasets are summarized in Table I. On three separate datasets that are RI_BoT, UNSW_NB15, and BoT_IoT, we achieved an accuracy of 96.290, 99.983, and 96.645 respectively by applying the above-mentioned ML models with the CatBoost Model, which is our major focus. By using performance metrics, we discovered that the CatBoost model produces the best results.

### Confusion Matrix and ROC Curve

This section represents the confusion matrix for the CatBoost Model and the combined ROC Curve as we have applied on the three datasets.

RI_BoT Dataset classification results
- Confusion Matrix
- ROC Curve
- UNSW_NB15 Dataset
- Confusion Matrix
- ROC Curve
- BoT_IoT Dataset
- Confusion Matrix
- ROC Curve

The above Figs. 4 and 5 represent the confusion matrix and ROC curve for the RI_BoT dataset. Further, Figs. 6 and 7 represent the confusion matrix and ROC curve for the UNSW_NB15 dataset. Furthermore, Figs. 8 and 9 represent the confusion matrix and ROC curve for the BoT_IoT dataset. In classification model the Decision Tree performs well with UNSW_NB15 and BoT_IoT whereas in Logistic Regression RI_BoT has better results than that of Decision Tree. From Table I it has been concluded that the CatBoost classifier model performs better on our newly generated RI_BoT dataset and on the best performing datasets like UNSW_NB15 and BoT_IoT dataset. Figures 6, 8, 10 represents the ROC curve, which is used to compare the results, obtained several ML models applied. Finally, by viewing the results based on ROC curve and the aspect of real-time based generated datasets the RI_BoT has provided best results with CatBoost model, which has the best results in Accuracy and Recall.

### Conclusion and Future Work

This paper presents a novel scheme that produces a non-existent real-time dataset, named RI_BoT, which consists of the detailed descriptions of real-time traffic of sensors and actuators implemented using the Raspberry pi 3B+ model. The dataset was created on a realistic testbed and also has been tested including pre-defined datasets like the UNSW_NB15 and the BoT_IoT using the above-mentioned ML models. A detailed comparative analysis for the above-mentioned datasets is also provided in the result section where RI_BoT achieves the same percentage of accuracy as BoT_IoT by applying the CatBoost model. As this work is based upon the real-time environment so it could be a benchmark for further investigation of IoT-Botnets in terms of vulnerabilities or potential threats associated with them. The proposed real-time dataset (RI_BoT) could be further investigated using hybrid techniques for IoT botnet traffic analysis.

**ORCID**

Umang Garg @ https://orcid.org/0000-0002-1815-5794
References

1. Statista Research Department, (2022), Internet of Things - number of connected devices worldwide 2015-2025 (accessed: Jan 5, 2022), https://statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

2. V. Kuzin, M. Shmelev, and Y. Kuskov, (2018), Securelist kaspersky lab (accessed: Jan 12, 2022), https://securelist.com/new-trends-in-the-world-of-iot-threats/87991/.

3. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, “DDoS in the IoT: mirai and other botnets.” Computer, Institute of Electrical and Electronics Engineers (IEEE), 50, 80 (2017).

4. E. Bertino and N. Islam, “Cyber-attacks on internet of things security.” Computer, Institute of Electrical and Electronics Engineers (IEEE), 50, 76 (2017).

5. M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, “A survey of machine and deep learning methods for internet of things (IoT) security.” IEEE Communications Surveys & Tutorials, Institute of Electrical and Electronics Engineers (IEEE), 22, 1646 (2020).

6. Kerbs, (2016), KrebsOnSecurity in-depth security news and investigation, https://krebsoscope.com/2016/10/source-code-for-iot-botnet-mirai-released/.

7. D. Paul, (2021), Digit, (accessed: Feb 4, 2022), https://digit.fyi/iot-security.

8. S. Popoola, B. Adebisi, M. Hammoudeh, H. Gacanin, and G. Gui, “Stacked recurrent neural network for botnet detection in new smart homes.” Computers & Electrical Engineering, 92, 107039 (2021).

9. Malwaredlc, (2021), “BYOB”, Malwaredlc. (accessed: March 12, 2022), https://github.com/malwaredlc/byob.

10. R. Technology, (2022), Wireshark (accessed: March 12, 2022), https://wireshark.org/download.html.

11. N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set).” Military Communications and Information Systems Conference (MICS). IEEE, Nov. 2015.

12. N. Moustafa and J. Slay, “The evaluation of network anomaly detection systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set.” Information Security Journal: A Global Perspective, Informa UK Limited, 25, 18 (2016).

13. N. Moustafa, J. Slay, and G. Creech, “Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks.” IEEE Transactions on Big Data, Institute of Electrical and Electronics Engineers (IEEE), 5, 481 (2019).

14. N. Moustafa, G. Creech, and J. Slay, “Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models.” Data Analytics and Decision Support for Cybersecurity (Springer, Cham) 127 (2017).

15. M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, “Netflow datasets for machine learning-based network intrusion detection systems.” Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (Springer International Publishing) 117 (2021).

16. N. Moustafa, N. Moustafa, and E. Sitnikova, “A new network forensic framework based on deep learning for internet of things networks: a particle deep framework.” Future Gener. Comput. Syst., 110, 91 (2020).

17. N. Koroniotis and N. Moustafa, “Enhancing network forensics with particle swarm and deep learning: the particle deep framework.” 7th International Conference on Artificial Intelligence and Applications. Aircc Publishing Corporation, 28 (2020).

18. N. Koroniotis, N. Moustafa, F. Schifino, P. Gauravaram, and H. Janicke, “A holistic review of cybersecurity and reliability perspectives in smart airports.” IEEE Access, Institute of Electrical and Electronics Engineers (IEEE), 8, 209802 (2020).

19. N. Koroniotis, N. Moustafa, and E. Sitnikova, “Forensics and deep learning mechanisms for botnets in internet of things: a survey of challenges and solutions.” IEEE Access, Institute of Electrical and Electronics Engineers (IEEE), 7, 61764 (2019).

20. N. Koroniotis, N. Moustafa, E. Sitnikova, and J. Slay, “Towards developing network forensic mechanism for botnet activities in the iot based on machine learning techniques.” Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (Springer, Cham) 30 (2018).