RESEARCH ARTICLE OPEN ACCESS

Botnet Attack Prevention in Internet of Things (IOT) Devices Using Ai

Sanjeev Kumar, Prof. Shivank Soni
Research Scholar, CSE Oriental Institute of science &technology, Bhopal
Assistant professor, CSE Oriental Institute of science &technology, Bhopal

ABSTRACT

Cybersecurity is seriously threatened by botnet attacks, which need for sophisticated detection systems to successfully reduce threats. In order to increase accuracy of categorisation, this work offers a strong botnet detection system which combines deep learning and machine learning methods. The suggested method classifies network traffic as either normal or botnet-infected using five models: Logistic Regression SVM, Decision Tree, KNN and a Sequential Deep Learning Model. The Software-Defined Networking (SDN) dataset, a comprehensive dataset that captures both benign and malicious network behaviours, is used to train and assess the detection system. StandardScaler is used for feature normalisation in order to maximise classification performance, guaranteeing reliable model training and increased detection accuracy. According to experimental data, the accuracy of Logistic Regression is 77%, SVM is 96.67%, KNN is 98.35%, and Decision Tree is 98.22%. With an astounding 99% accuracy rate, the deep learning-based Sequential MLP model performs better than any other method, reducing false positives and showcasing its excellent detection skills. In order to guarantee scalability and efficiency, the system is built for real-time deployment and uses cloud-based execution on Google Colab. This study offers a highly efficient botnet detection system that improves network security by fusing deep learning with conventional machine learning techniques. According to the findings, deep learning models—in particular, the Sequential model significantly increase detection accuracy, which makes them a practical way to spot botnet activity on networks with a lot of users.

Keywords: **Network traffic analysis, cybersecurity, software-defined networking (SDN), machine learning, botnet detection, and sequential modelling, deep learning.**

I. INTRODUCTION

In the current computerised environment, cyber security becomes a major worry, and one of the biggest dangers to network security is botnet assaults. Numerous online offences, including Distributed Denial of Service, or DDoS, assaults, data breaches, and financial fraud, are caused by botnets, which are networks of infected devices under the direction of malevolent individuals. Because of their adaptive nature and complex evasion strategies, modern botnets are often difficult for traditional security measures, including intrusion detection systems based on rules and signatures, to determine.

For businesses as well as governments that depend on digital infrastructures for communication and trade, as well as for individual users, it is essential to protect networks against botnet assaults. However, since cyber dangers are always changing, identifying botnets is still a difficult process. Botnet architecture are constantly being altered by attackers, making it challenging for traditional detection techniques to stay up to date. Furthermore, human analysis is impossible due to the sheer amount of network data, which raises the need for automated detection methods.

The capacity of deep learning as well as machine learning approaches to spot intricate designs in massive datasets has drawn a great deal of interest in cybersecurity. These algorithms have a high degree of accuracy in classifying hostile activity, analysing network data, and identifying abnormalities. Though they often have trouble with complicated and high-dimensional data, traditional ML models like SVM, KNN, LR as well as DT have shown potential in botnet identification. Deep learning models are ideal for botnet detection because they have shown exceptional ability in recognising complex attack patterns, particularly Recurrent Neural Networks (RNNs) as well as Convolutional Neural Networks (CNNs).

Using a Software-Defined Networking (SDN) dataset, this research focusses on using both ML and DL methods for botnet identification. The main goal is to evaluate how well sophisticated DL models and traditional ML methods detect botnet activity. This study shows the potential using AI-driven security solutions to improve network protection by showcasing the increased accuracy of deep learning algorithms. The study's conclusions are intended to aid in the creation of stronger and more effective cybersecurity frameworks, which will eventually enhance the capacity to identify and stop botnet assaults instantly.

II. LITERATURE REVIEW

A significant cybersecurity risk, botnet assaults need sophisticated detection strategies that go beyond conventional rule-based approaches. Several botnet detection strategies, including as signature-based, behavior-based, as well as anomaly-based methods, are examined in this literature review. It draws attention to how ML along with DL methods are becoming more and more used to detect fraudulent network traffic. Deep learning models like CNNs and LSTMs perform better at identifying intricate attack patterns, even though machine learning techniques like SVM as well as Decision Trees provide respectable accuracy. This analysis highlights the need for AI-driven solutions to improve cybersecurity and real-time threat identification by contrasting current approaches and evaluating their advantages and disadvantages.

(Joshi et al. 2022) [48] investigated a range of botnet detection tactics, including as signature-based, anomaly-based and behavior-based methods. The study highlighted the challenges of high false positive rates and difficulties in detecting new botnets. It also examined Artificial Neural Networks (ANNs) and feature engineering approaches, concluding that fuzzy logic-based feature selection improves detection accuracy and reduces false positives.

(Abdullayeva and Fargana et al. 2022) [49] analyzed DDoS attack detection methods in cloud environments. They emphasized the importance of examining network traffic patterns and introduced data clustering techniques for effective classification. Their approach achieved high accuracy with minimal false positives, making it suitable for cloud security.

(Owen et al. 2022) [50] provided an extensive review of botnet characteristics, detection techniques, and prevention strategies. The study highlighted advancements in machine learning applications and behavioral analysis, underlining the importance of regulatory measures and technological innovations in cybersecurity.

(Schwengber et al. 2020) [51] proposed an adaptive online botnet detection method addressing concept drift. The study discussed ensemble and window-based strategies to manage statistical variations in network data. Their approach outperformed traditional techniques by maintaining higher detection accuracy and reducing false alarms.

(S. Khanna et al. 2020) [52] examined security challenges in IoT technology, focusing on data transmission vulnerabilities. The study explored AI-based anomaly detection techniques for mitigating threats like Denial of Service (DoS) attacks, offering research directions for improving IoT cybersecurity.

(Rizvi et al. 2020) [53] introduced a multi-layered threat detection model for IoT security. The research covered various IoT vulnerabilities, including botnets, ransomware, and spoofing attacks. It emphasized privacy measures to mitigate risks in medical, household, and industrial IoT applications.

(Ali et al. 2020) [54] proposed an early-stage botnet detection system using feature selection techniques. Their study demonstrated the effectiveness of machine learning classifiers in distinguishing malicious and legitimate messages, achieving 99% accuracy with a low false positive rate.

(Alissa et al. 2020) [55] developed a machine learning-based IoT security framework. The study tested Logistic Regression, Decision Trees, and XGBoost models, with Decision Trees achieving the highest accuracy. Their work emphasized the importance of data preprocessing and standardization in improving classification performance.

(Snoussi and Youssef et al. 2023) [56] developed Variational Autoencoders (VAEs) to identify botnets in IOT networks. By overcoming the drawbacks of traditional ML approaches, their model improved botnet detection in complicated IoT contexts and improved network traffic analysis.

III. PROPOSED METHODOLOGY

The workflow that follows offers a comprehensive, step-by-step methodology used in this study. It covers techniques for preprocessing data, feature engineering, model selection, hyperparameter tuning, training multiple ML models, and performance assessment using crucial parameters like as precision, recall, accuracy, & F1-score.

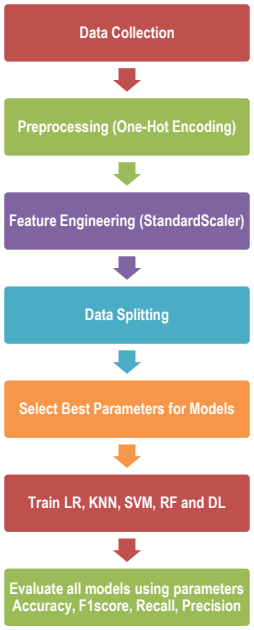

Figure 1 Workflow of proposed methodology

A. LOAD DATASET

The Dataset_SDN is specifically designed for research in Software-Defined Networking or +(SDN), with a primary focus on network security and botnet detection. It comprises network traffic data captured in SDN environments, where a centralized controller manages the flow of data packets. This dataset includes various essential features that facilitate network analysis and intrusion detection. It contains flow-based data, which includes attributes such as packet count, byte count, flow duration, and source-destination IPs. Additionally, it provides traffic labels to differentiate between normal and botnet traffic, making it suitable for classification tasks. The dataset also incorporates time-based features, capturing traffic patterns over time to analyze botnet behavior. Furthermore, it includes protocol information, covering details on TCP, UDP, and ICMP traffic, along with SDN-specific data, which consists of features unique to SDN networks, such as flow rules and controller communication metrics. These characteristics make Dataset_SDN a valuable resource for advancing research in SDN security and botnet detection.

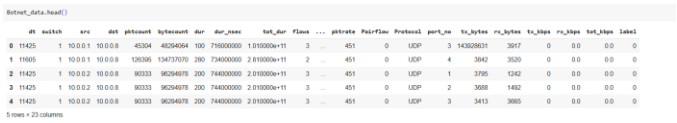

Figure 2 Dataset Sample

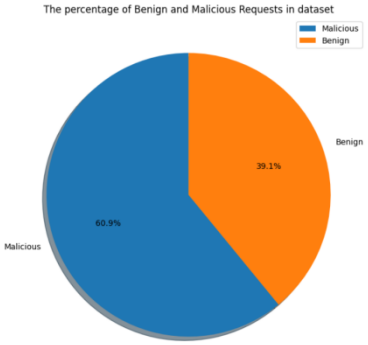

Figure 3 Percentage of Benign and Malicious Requests in dataset

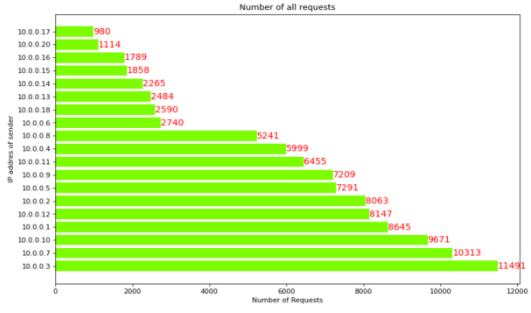

Figure Error! No text of specified style in document. **Number of all requests**

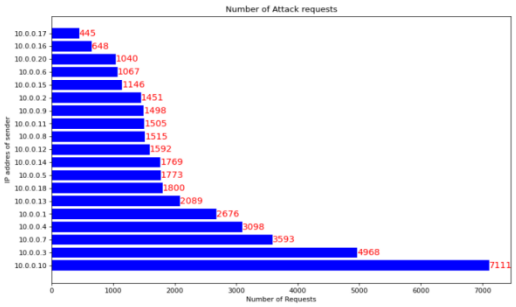

Figure 5 Number of Attack requests

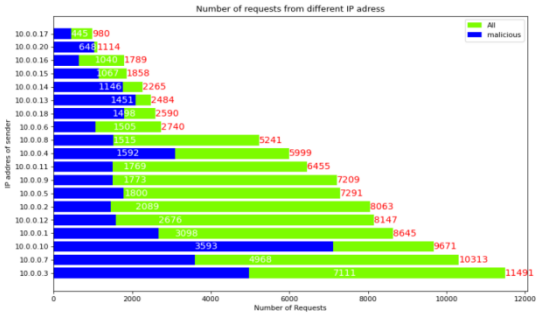

Figure 6 Number of requests from different IP address

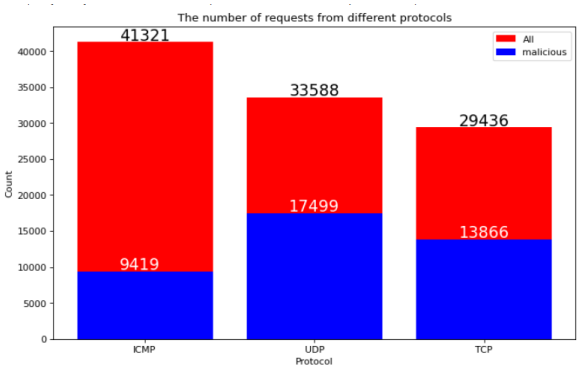

Figure 7 Number of requests from different protocols

B. PREPROCESSING (ONE-HOT ENCODING)

In order to guarantee that categorical characteristics are appropriately transformed into a format that machines can understand, preprocessing is an essential step in getting data ready for ML methods. A technique for converting categorical data into numerical values is called one-hot encoding. Each distinct category is given its own binary column in this procedure, with the existence of a category denoted by a "1" and its absence by a "0." Presenting categorical values like ordinal

numbers may lead to biassed weight assignments, which is why this method is so helpful. Protocol types like TCP, UDP, and ICMP, for instance, are encoded into distinct columns so as to avoid suggesting any hierarchical link. This change improves the model's compatibility and interpretability, enabling it to handle categorical data efficiently and without inadvertently adding biases.

C. FEATURE ENGINEERING (STANDARDSCALER)

Enhancing model efficiency and accuracy requires feature engineering. Numerical characteristics are standardised using the StandardScaler approach to ensure that they all participate equally to the training process. To ensure that every feature has Standardisation is done using a mean of 0 and a standard deviation of 1. by removing each feature's mean and dividing the result by the standard deviation. When managing KNN and other ML methods that rely on distance-based calculations, this step is especially crucial since it keeps certain characteristics from outweighing others because of their size. StandardScaler increases model stability and the efficacy of deep learning algorithms by converting the data into a uniform range, which eventually results in improved predictions and overall performance.

D. DATA SPLITTING

In machine learning, data splitting is a crucial step that guarantees the model is trained efficiently while preserving its capacity to generalise to new data. The dataset is divided into two parts: a testing set and a training set. in order to do this. This division is often carried out using Python's train_test_split() function, whereby 80% of the data is allocated for training and 20% for testing. The model is made possible by this division to assess its performance on a separate test set while learning from the training data. To guarantee an actual distribution of the target labels in the training and evaluation sets, stratified sampling is sometimes used to preserve class balance. By evaluating the method's correctness, resilience & capacity to manage data from the actual world, the test set helps to avoid overfitting and guarantee dependable performance in real-world applications.

E. SELECTING THE BEST PARAMETERS FOR MODELS

A crucial stage in machine learning model optimisation is hyperparameter modification, which lowers mistakes and increases accuracy. The procedure entails choosing the optimal hyperparameters that govern a model's learning process. A popular method for systematic tuning is GridSearchCV, which tests various hyperparameter arrangements to determine the most effective setup. For instance, the no. of tree is optimised in RF, the kernel types are changed in SVM, & the optimal number of neighbours is chosen in K-Nearest Neighbours (KNN). In order to make sure that the parameters selected maximise accuracy while minimising false positives as well as false negatives, cross-validation is carried out throughout this procedure. The model's efficiency, predictive power, and ability to generalise to new data are all enhanced by adjusting hyperparameters.

F. MACHINE LEARNING AND DEEP LEARNING MODELS

- **Logistic Regression**

By applying a sigmoid function to a linear mixture of input data, the supervised machine learning technique known as logistic regression predicts the likelihood that an event will occur in binary classification tasks. A threshold, usually 0.5, is used to classify the probability of 0 to 1 that the model produces in one of two groups. Newton-cg, lbfgs, liblinear, sag, as well as saga are among the optimisation solvers that are investigated in order to discover the optimal approach for implementing Logistic Regression. The dataset for training (X_train, y_train) is used to train a model for each solver, and accuracy scores are used to assess the method on the testing data set (X_test). The solver utilised for final model training and prediction is the one with the best accuracy. After that, the final model is applied to fresh data and retrained using the best solution. Classification measures like Precision, F1-score & Recall are used in performance analysis to make sure the model successfully distinguishes between classes and operates at its best in practical applications.

- **Support Vector Machine (SVM)**

A SVM is a supervised learning method, finds the best decision boundary, or hyperplane, to maximise the margin between classes in classification problems. By using various kernel functions, SVM can handle both nonlinear and linear classification. To choose the best kernel function, the SVM implementation starts by evaluating many options, such as linear, polynomial (poly), sigmoid and radial basis function (rbf). Each kernel's SVM model is constructed utilising the training dataset (X_train, y_train), & the model's performance on the test dataset (X_test) is evaluated using accuracy scores. For the final model training, the kernel with the best accuracy is chosen. The finished SVM model is then retrained using the selected kernel and then used to make predictions on fresh data. Key classification measures F1-score, accuracy, and recall are among the metrics used to assess the model's effectiveness, guaranteeing that it can correctly classify data and generalise effectively to unknown inputs.

- **K-Nearest Neighbors (KNN)**

K-Nearest Neighbours (KNN) is a well-liked supervised machine learning approach to classification and regression

issues. A data point is categorised using this simple yet effective approach based on the majority of the class of its nearest neighbours. KNN generates predictions by using the feature space's data points' similarities rather than assuming a fixed structure as parametric models do. KNN implementation starts with the definition of hyperparameters such distance metrics (euclidean or Manhattan), weighting techniques (uniform or distance), and the number of neighbours (n_neighbors with values 3, 5, 10). The optimal parameter combination is found by doing an exhaustive search using 3-fold cross-validation using GridSearchCV to optimise these hyperparameters. The KNN models is trained through the chosen values once the ideal hyperparameters have been identified. Accuracy scores and a classification report with key metrics including F1-score, Precision, & Recall are used to evaluate the trained model's efficacy after label prediction for the test dataset (X_test). This ensures that the model properly classifies new data points while maintaining high accuracy and durability.

- **Decision Tree Classifier (DT)**

Strong supervised learning techniques for classification problems are called decision tree classifiers. In order to create a structure that resembles a tree where decision rules generate internal nodes & class labels are allocated to leaf nodes, they recursively partition the dataset depending on feature values. Finding the most informative splits is the algorithm's aim in order to increase classification accuracy while preserving interpretability. Determining hyperparameters such the splitting criteria (gini or entropy), the maximum tree depth (max_depth ranging from 2 to 10), & the highest possible number of leaf nodes (max_leaf_nodes ranging from 2 to 11) is the first step in implementing a decision tree. GridSearchCV uses 5-fold cross-validation and an exhaustive search to optimise these parameters, using parallel processing (n_jobs=-1) to increase efficiency. To balance accuracy and avoid overfitting, the DecisionTreeClassifier is trained through the optimal hyperparameters after they have been determined. The capacity of the training model to generalise to new, unknown data is next evaluated by predicting class labels for the test dataset (X_test). Lastly, a classification report that incorporates important performance measures such as F1-score, accuracy, and recall are used to evaluate the model's correctness and ensure its efficacy across various categorisation categories.

- **Multi-Layer Perceptron (MLP) for Classification**

An artificial neural network called a Multi-Layer Perceptron (MLP) is made for tasks involving regression and classification. An input layer, one or more hidden layers, and an output layer are among the many layers of neurones that make up this structure. In order to interpret inputs and discover intricate patterns, each neurone uses an activation function. Backpropagation is used in MLP training, where weights are adjusted to reduce mistakes and increase model accuracy.

Architecture of MLP Model

1. **Input Layer**
   - Accepts input features (input_dim = X_train.shape[1]).
2. **Hidden Layers**

   1st Hidden Layer:
   - There are 128 neurones that have ReLU activity.
   - The dropout value is 0.15 to avoid overfitting.

   2nd Hidden Layer:
   - 64 neurones that are activated by ReLU.
   - To further regularise, dropout (0.15).

   3rd Hidden Layer:
   - ReLU-activated 32 neurones for deep feature learning.
3. **Output Layer**
   a. Number of neurons = **number of target classes (nb_classes)**.
   b. **Softmax activation** for multi-class classification.
4. **Compilation**
   a. **Loss Function:** Categorical Crossentropy (for multi-class classification).
   b. **Optimizer:** RMSprop (adapts learning rate dynamically).
   c. **Metric:** Accuracy (used to evaluate model performance).
5. **Training Strategy**
   a. **Epochs:** 20 (number of times the dataset is processed).
   b. **Batch Size:** 16 (samples that have been processed prior to weight updates).
   c. **Validation Split:** Validation is done using 10% of the training data.

d. **Verbose Level:** 2 (detailed training logs).

6. Prediction and Evaluation

a. Predict class probabilities using model.predict(X_test).

b. Convert predictions into class labels using np.argmax(y_pred, axis=1).

c. Generate a **Confusion Matrix** to compare actual and predicted labels.

d. Use classification_report() to display F1-score, precision, & recall.

IV. RESULT

The advantage of deep learning in cybersecurity is shown by a comparison of machine learning and deep learning models for botnet detection.Among traditional models, Logistic Regression has the lowest accuracy (77%), while Decision Trees, KNN, and SVM achieve 98.35%, 96.67%, and 98.22%, respectively. However, our Decision Tree model, trained with different parameters, outperforms existing models with 92% accuracy. The proposed deep learning model, MLP, further enhances detection, achieving 99% accuracy with minimal false positives. StandardScaler improves feature normalization, optimizing training and classification. Cloud-based execution on Google Colab ensures scalability for real-time deployment. Integrating deep learning with traditional methods strengthens botnet detection, paving the way for advanced AI-driven cybersecurity solutions.

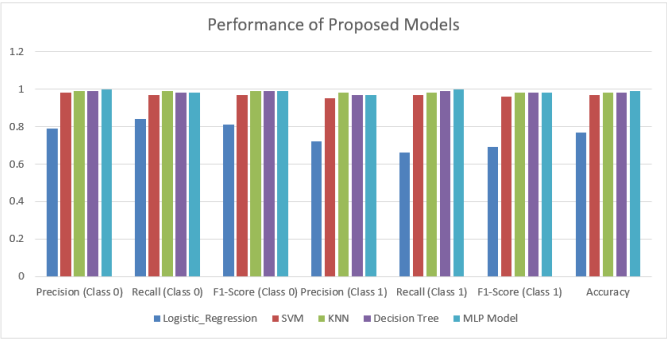

Figure 8 Performance of Proposed Models

Table 1 Result of Proposed Models

Models	Accuracy	TP	TN	FP	FN
Logistic Regression	76.64%	15866	8008	3120	4158
SVM	96.67%	18350	11766	636	400
KNN	98.35%	18737	11901	249	265
Decision Tree	98.22%	18588	12011	398	155
MLP	99%	18581	12121	405	45

Table 1 Compare Our Work With Existing Work

Models	Accuracy
Proposed Decision Tree	0.98%
Existing Decision Tree [68]	0.9221 %

V. CONCLUSION

This research shows that deep learning outperforms conventional machine learning models in botnet identification. While traditional models with good accuracy include SVM, KNN, Decision Trees, and Logistic Regression, they are still prone to misclassifications. Our optimized Decision Tree model, trained with different parameters, performs better than existing approaches, achieving 92% accuracy. However, our proposed deep learning model, MLP, surpasses all with an impressive 99% accuracy, minimizing false positives. The use of StandardScaler for feature normalization and cloud-based execution on Google Colab enhances scalability and efficiency, making real-time deployment feasible. By integrating deep learning with traditional methods, this research paves the way for robust, AI-driven cybersecurity solutions. Future work will focus on automated countermeasures, real-time threat mitigation, and adaptive learning to strengthen network security against evolving cyber threats.

REFERENCE

[1]. "Meaning of 'bot' in the English Dictionary." [Online]. Available: http://dictionary.cambridge.org/dictionary/english/bot. [Accessed: 15-Aug-2017].

[2]. "Meaning of 'crossword puzzle' in the English Dictionary." [Online]. Available: https://dictionary.cambridge.org/dictionary/english/crossword-puzzle. [Accessed: 15- Aug-2017].

[3]. Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov, "Spamming botnets: Signatures and characteristics," Computer Communication Review, vol. 38, no. 4, pp. 171–182, Aug. 2008.
[4]. J. Fu, P. Lin, and S. Lee, "Detecting spamming activities in a campus network using incremental learning," Journal of Network and Computer Applications, vol. 43, pp. 56– 65, 2014.
[5]. A. West and I. Lee, "Towards the effective temporal association mining of spam blacklists," in ACM International Conference Proceeding Series, 2011, no. September, pp. 73–82.
[6]. P. A. R. Kumar and S. Selvakumar, "Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems," Computer Communications, vol. 36, no. 3, pp. 303–319, 2013.
[7]. S. McGregory, "Preparing for the next {DDoS} attack," Network Security, vol. 2013, no. 5, pp. 5–6, 2013.
[8]. G. Pellegrino, C. Rossow, F. J. Ryba, T. C. Schmidt, and M. Wählisch, "Cashing Out the Great Cannon? On Browser-Based DDoS Attacks and Economics," 9th USENIX Workshop on Offensive Technologies (WOOT 15), no. 1, pp. 1–36, 2015.
[9]. B. Alshehry and W. Allen, "Proactive approach for the prevention of DDoS attacks in cloud computing environments," in Studies in Computational Intelligence, vol. 695, 2017, pp. 119–133.
[10]. H. Haddadi, "Fighting Online Click-Fraud Using Bluff Ads," SIGCOMM Comput. Commun. Rev., vol. 40, no. 2, pp. 21–25, Apr. 2010.
[11]. H. Shahriar and V. K. Devendran, "Classification of Clickjacking Attacks and Detection Techniques," Information Security Journal: A Global Perspective, vol. 23, no. 4–6, pp. 137–147, 2014.
[12]. S. A. Alrwais, A. Gerber, C. W. Dunn, O. Spatscheck, M. Gupta, and E. Osterweil, "Dissecting ghost clicks," in Proceedings of the 28th Annual Computer Security Applications Conference on - ACSAC '12, 2012, p. 21.
[13]. C. M. R. Haider, A. Iqbal, A. H. Rahman, and M. S. Rahman, "An ensemble learning based approach for impression fraud detection in mobile advertising," Journal of Network and Computer Applications, vol. 112, pp. 126–141, Jun. 2018.
[14]. Song, LP, Jin, Z & Sun, GQ 2011, 'Modeling and analyzing of botnet interactions', Physica A: Statistical Mechanics and its Applications, vol. 390, no. 2, pp. 347-358.
[15]. Egele, M, Scholte, T, Kirda, E & Kruegel, C 2012, 'A survey on automated dynamic malware-analysis techniques and tools', ACM Computing Surveys (CSUR), vol. 44, no. 2, pp. 1-6.
[16]. . Zeidanloo, HR & Manaf, AA 2009, 'Botnet command and control mechanisms', Proceedings of the second IEEE International Conference on Computer and Electrical Engineering, pp. 564-568.
[17]. Marupally, PR & Paruchuri, V 2010, 'Comparative analysis and evaluation of botnet command and control models', Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 82-89.
[18]. Wang, B, Li, Z, Li, D, Liu, F & Chen, H 2010, 'Modeling connections behavior for web-based bots detection', Proceedings of second IEEE International Conference on e-Business and Information System Security (EBISS), pp. 1-4.
[19]. S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," Computer Networks, vol. 57, no. 2, pp. 378–403, Feb. 2013
[20]. "mIRC: Internet Relay Chat Client." [Online]. Available: http://www.mirc.com/. [Accessed: 01-Jan-2016]
[21]. C. Gañán, O. Cetin, and M. van Eeten, "An Empirical Analysis of ZeuS C&C Lifetime," Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15, pp. 97–108, 2015.
[22]. "EVGENIY MIKHAILOVICH BOGACHEV - FBI." [Online]. Available: https://www.fbi.gov/wanted/cyber/evgeniy-mikhailovich-bogachev. [Accessed: 01-Jan2016].
[23]. C. Gañán, O. Cetin, and M. van Eeten, "An Empirical Analysis of ZeuS C&C Lifetime," in Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15, 2015, pp. 97–108.
[24]. H. Binsalleeh et al., "On the analysis of the Zeus botnet crimeware toolkit," in PST 2010: 2010 8th International Conference on Privacy, Security and Trust, 2010, pp. 31–38
[25]. "Major ISPs targeted in Internet of Things botnet attacks," Network Security, vol. 2016, no. 12. pp. 1–2, 2016.
[26]. C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," Computer, vol. 50, no. 7, pp. 80–84, 2017.
[27]. M. Khosroshahy, M. K. Mehmet Ali, and D. Qiu, "The SIC botnet lifecycle model: A step beyond traditional epidemiological models," Computer Networks, vol. 57, no. 2, pp. 404–421, 2013.
[28]. R. A. Rodriguez-Gomez, G. Macia-Fernandez, and P. Garcia-Teodoro, "Survey and taxonomy of botnet research through life-cycle," ACM Computing Surveys, vol. 45, no. 4, pp. 1–33, Aug. 2013.
[29]. McAfee Labs, "McAfee Labs Threats Report: December 2018," Computer Fraud & Security,

2019. [Online]. Available: www.mcafee.com/us/mcafee-labs.aspx. [Accessed: 04-Apr-2019].

[30]. "The Spamhaus Project." [Online]. Available: https://www.spamhaus.org/statistics/botnet-cc/. [Accessed: 04-Apr-2019]

[31]. S. S. C. Silva, R. M. P. Silva, R. C. G. Pinto, and R. M. Salles, "Botnets: A survey," Computer Networks, vol. 57, no. 2, pp. 378–403, Feb. 2013.

[32]. A. Karim, R. Bin Salleh, M. Shiraz, S. A. A. Shah, I. Awan, and N. B. Anuar, "Botnet detection techniques: review, future trends, and issues," Journal of Zhejiang University: Science C, vol. 15, no. 11, pp. 943–983, Nov. 2014.

[33]. P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 2, pp. 113–127, 2010.

[34]. De Neira, Anderson Bergamini, Alex Medeiros Araujo & Michele Nogueira 2020, 16th International Conference on Mobility, Sensing and Networking (MSN), 17th to 19th December, "Early botnet detection for the internet and the internet of things by autonomous machine learning", pp. 516-523, https://doi.org/ 10.1109/ MSN 50589.2020.00087

[35]. Lingam, G, Rout, RR, Somayajulu, DV & Das, SK 2020, In Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, 5th October, "Social botnet community detection: a novel approach based on behavioral similarity in twitter network using deep learning", Taipei, Taiwan, https://doi.org/10.1145/3320269.3384770

[36]. P. Mishra, V. Varadharajan, U. Tupakula, and E. S. Pilli, "A detailed investigation and analysis of using machine learning techniques for intrusion detection," IEEE communications surveys & tutorials, vol. 21, no. 1, pp. 686–728, 2018.

[37]. H. Li, K. Ota, and M. Dong, "Learning IoT in edge: Deep learning for the Internet of Things with edge computing," IEEE network, vol. 32, no. 1, pp. 96–101, 2018.

[38]. Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on Random Forest and Support Vector Machine," in 2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC), vol. 1, pp. 635–638, IEEE, 2017.

[39]. O. Y. Al-Jarrah, Y. Al-Hammdi, P. D. Yoo, S. Muhaidat, and M. Al-Qutayri, "Semi-supervised multi-layered clustering model for intrusion detection," Digital Communications and Networks, vol. 4, no. 4, pp. 277–286, 2018.

[40]. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., "Human-level control through deep reinforcement learning," nature, vol. 518, no. 7540, pp. 529–533, 2015.

[41]. P. Soucy and G. W. Mineau, "A simple KNN algorithm for text categorization," in Proceedings 2001 IEEE international conference on data mining, pp. 647–648, IEEE, 2001.

[42]. K. Goeschel, "Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive Bayes for off-line analysis," in SoutheastCon 2016, pp. 1–6, IEEE, 2016.

[43]. Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," in 2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), pp. 127–132, IEEE, 2017.

[44]. Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2432–2455, 2017.

[45]. L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in 2013 IEEE international conference on acoustics, speech and signal processing, pp. 8599–8603, IEEE, 2013.

[46]. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," Advances in neural information processing systems, vol. 27, 2014.

[47]. Z. Xueqing, Z. Zhansong, and Z. Chaomo, "Bi-LSTM deep neural network reservoir classification model based on the innovative input of logging curve response sequences," IEEE Access, vol. 9, pp. 19902–19915, 2021.

[48]. Joshi, Chirag, Ranjeet Kumar Ranjan & Vishal Bharti 2022, "A Fuzzy Logic based feature engineering approach for Botnet detection using ANN", Journal of King Saud University-Computer and Information Sciences, vol. 34, no. 9, pp. 6872-6882

[49]. Abdullayeva & Fargana 2022, "Distributed denial of service attack detection in Egovernment cloud via data clustering", In 2022 International Open Access

Multidisciplinary Journal - Elsevier, vol. 15, https://doi.org/10.1016/j.array. 2022.100229

[50]. Owen, Harry, Javad Zarrin & Shahrzad M Pour 2022, "A survey on botnets, issues, threats, methods, detection and prevention", Journal of Cybersecurity and Privacy, vol. 2, no. 1, pp. 74-88

[51]. Schwengber, Bruno Henrique, Andressa Vergütz, Nelson G Prates & Michele Nogueira 2020, IEEE, In GLOBECOM 2020-2020 IEEE Global Communications Conference, 7 th to 11th December, "A method aware of concept drift for online botnet detection", Taipei Taiwan pp. 1-6, https://doi.org/10.1109/ GLOBECOM 42002.2020.9347990

[52]. S. Khanam, I. B. Ahmedy, M. Y. Idna Idris, M. H. Jaward, and A. Q. Bin Md Sabri, "A Survey of Security Challenges, Attacks Taxonomy and Advanced Countermeasures in the Internet of Things," IEEE Access, vol. 8, pp. 219709–219743, 2020.

[53]. S. Rizvi, R. Orr, A. Cox, P. Ashokkumar, and M. R. Rizvi, "Identifying the attack surface for IoT network," Internet of Things, vol. 9, p. 100162, 2020

[54]. M. Waqas, K. Kumar, A. A. Laghari, U. Saeed, M. M. Rind, A. A. Shaikh, F. Hussain, A. Rai, and A. Q. Qazi, "Botnet attack detection in Internet of Things devices over cloud environment via machine learning," Concurrency and Computation: Practice and Experience, vol. 34, no. 4, p. e6662, 2022.

[55]. K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, S. Sakib, et al., "Botnet attack detection in IoT using machine learning," Computational Intelligence and Neuroscience, vol. 2022, 2022.

[56]. Snoussi, R & Youssef, H 2023, "VAE-Based Latent Representations Learning for Botnet Detection in IoT Networks", Journal of Network and Systems Management, vol. 31, no. 1, p. 4

[57]. Y. Mahmoodi, S. Reiter, A. Viehl, O. Bringmann, and W. Rosenstiel, "Attack surface modeling and assessment for penetration testing of IoT system designs," in 2018 21st Euromicro Conference on Digital System Design (DSD), pp. 177–181, IEEE, 2018.

[58]. M. Asadi, "Detecting IoT botnets based on the combination of cooperative game theory with deep and machine learning approaches," Journal of Ambient Intelligence and Humanized Computing, vol. 13, no. 12, pp. 5547– 5561, 2022.

[59]. M. Waqas, K. Kumar, A. A. Laghari, U. Saeed, M. M. Rind, A. A. Shaikh, F. Hussain, A. Rai, and A. Q. Qazi, "Botnet attack detection in Internet of Things devices over cloud environment via machine learning," Concurrency and Computation: Practice and Experience, vol. 34, no. 4, p. e6662, 2022.

[60]. K. Alissa, T. Alyas, K. Zafar, Q. Abbas, N. Tabassum, S. Sakib, et al., "Botnet attack detection in IoT using machine learning," Computational Intelligence and Neuroscience, vol. 2022, 2022.

[61]. H. Alazzam, A. Alsmady, and A. A. Shorman, "Supervised detection of IoT botnet attacks," in Proceedings of the second international conference on data science, E-Learning and information systems, pp. 1–6, 2019.

[62]. V. H. Bezerra, V. G. T. da Costa, S. Barbon Junior, R. S. Miani, and B. B. Zarpelao, "IoTDS: A one-class classification approach to detect botnets in ˜ Internet of Things devices," Sensors, vol. 19, no. 14, p. 3188, 2019.

[63]. Shao, Zhou, Sha Yuan & Yongli Wang 2021, "Adaptive online learning for IoT botnet detection", Information Sciences, vol. 574, no. 4, pp. 84-95

[64]. Moubayed, Abdallah, MohammadNoor Injadat & Abdallah Shami 2020, IEEE, 32nd International Conference on Microelectronics (ICM),14th to 16th December, "Optimized random forest model for botnet detection based on DNS queries", Aquba, Jordan, https://doi.org/10.1109/ICM50269.2020.9331819

[65]. Catak, FO & Mustacoglu, AF 2019, "Distributed denial of service attack detection using autoencoder and deep neural networks", Journal of Intelligent & Fuzzy Systems, vol. 37, no. 3, pp. 3969-3979

[66]. Cooke, E., Jahanian, F., & McPherson, D. (2005). The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets.

[67]. McLaughlin, L. (2004). Bot software spreads, causes new worries. ,(6), 1.

[68]. Moorthy, R. S. S., & Nathiya, N. (2023). Botnet detection using artificial intelligence. Procedia Computer Science, 218, 1405-1413.