Detection of Zero-day Attacks on IoT

Shay Reardon
Electrical Eng. and Computer Sc.
Florida Atlantic University
Boca Raton, USA
sreardon2018@fau.edu

Murtadha D. Hssayeni
Electrical Eng. and Computer Sc.
Florida Atlantic University
Boca Raton, USA
ORCID 0000-0002-8588-4639

Imadeldin Mahgoub
Electrical Eng. and Computer Sc.
Florida Atlantic University
Boca Raton, USA
mahgoubi@fau.edu

2024 International Conference on Smart Applications, Communications and Networking (SmartNets) | 979-8-3503-8532-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/SMARTNETS61466.2024.10577735

***Abstract*—Zero-day attacks are cybersecurity attacks that seek to exploit an unknown vulnerability in Internet of Things (IoT). This makes zero-day attacks inherently difficult to detect and costly to network administrators. Current methods of detection utilize machine learning methodologies for intrusion detection. However, these methods suffer from low performance in specific zero-day attacks. This study proposes novel features built upon network flow and raw packet data aiming to detect zero-day attacks. Our testing approach utilizes fix traditional machine learning algorithms (Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Logistic Regression (LR), Gaussian Naive Bayes (NB), and Random Forest (RF)) with split-at-scenario cross-validation. We find that our engineered features achieve consistent high detection rates with three models (DT, SVM, and RF), whereas these models fail to detect at least one of the attacks when using raw features. Our results display potential for utilizing the proposed flow-based complex features to detect unknown network attacks with Internet of Battle Things (IoBT) applications.**

***Index Terms*—intrusion detection system, zero-day attacks, IoT, IoBT, machine learning.**

I. Introduction

ZERO-DAY attacks are cybersecurity attacks that seek to exploit a vulnerability that is unknown to network administrators. This vulnerability is known as a zero-day vulnerability. While the vulnerability remains unknown, the developers cannot patch it, and anti-virus software cannot detect attacks exploiting it. But, by the time developers find the vulnerability or attack, the damage to the company or consumer has already been done. Zero-day cost an average of 1.2 million USD per attack [1], [2] and can last an average of 312 days [3]. It is estimated that every 17 days one of these attacks is discovered [4]. Also, there is a 15% increase in these attacks. To monitor for these costly attacks, Intrusion Detection Systems (IDS) are required to detect zero-day attacks and vulnerabilities.

Machine learning algorithms are a promising area for research concerning IDS. Hindy *et al.* implemented multiple traditional machine learning models (such as k-Nearest Neighbor and Random Forest) to detect attacks on unidirectional and bidirectional flows [5]. Their methodology had the advantage of a robust dataset, including MQTT data. Their approach is not evaluated for zero-day attacks. Other studies have shown promising results utilizing machine learning for zero-day attack detection with high accuracy. However, these proposed systems either have a drop in F1 score or have low detection rates for other attacks [1], [6], [7].

In this study, we propose to extract complex engineered features in combination with traditional machine learning algorithms to detect abnormal behaviors in network traffic data and thus detect zero-day attacks. Although deep learning methods recently yield promising results in many domains, they are data-hungry and have low explainability as they do not give details to why traffic data are flagged as attacks. The proposed flow-based system and subsequent extracted features is evaluated using MQTT-IoT-IDS2020 dataset [5]. Split-at-scenario cross-validation is used to check its utility as IDS in IoBT applications.

II. Methodology

A. Segmentation

Traffic data streams are captured as sequences of packets, each distinguished by source and destination IP addresses, irrespective of the utilized ports and protocols. We define a new flow when there is a modification at the IP level, with alterations in port or protocol having no impact on the flow identification. Subsequently, each flow undergoes segmentation into windows consisting of s packets. In this study, flow windows are limited to being no greater than 16 packets in length. This segmentation becomes necessary due to the substantial packet volume associated with the majority of flows, which can impede real-time intrusion detection by consuming significant processing time [8]. The quantity of packets within each window is adaptable during testing to accommodate variations in network resources and detection frequency.

B. Feature Extraction

Two settings for features extraction were implemented and compared which were raw packet features utilized in literature and novel flow-based features.

1) Raw Packet Features: The first setting tested the packet features employed in the MQTT-IoT-IDS2020 dataset, as published and reported by Hindy *et al.* Four different attacks and standard network traffic were simulated in a realistic MQTT IoT network, enabling robust evaluation of intrusion detection systems in IoT [5]. The four simulated attacks are as

979-8-3503-8532-8/24/$31.00 ©2024 IEEE

follows: Aggressive Scan (Scan-A), User Datagram Protocol Scan (Scan-sU), Sparta SSH Brute Force (Sparta), and MQTT Brute Force (MQTT-Brute Force).

The dataset comprises packet-level data for packet-based, unidirectional-based, and bidirectional-based features. This study utilizes the packet-based features for testing the first scenario. Thirty total features are extracted for this. Five features consider the source and destination IP address and ports in addition to the protocol used by the packet. Additionally, a feature is used to describe the transport layer protocol. Two features describe the packet length and time to live of the packet. Three packets were utilized for the IP flags and ten were used for the TCP flags [5]. The nine MQTT flags were not used in this study. More details about these features are in the work of Hindy *et al.* [5].

Each of the five '.pcap' files in the dataset representing regular network traffic and the zero-day attacks traffic at the packet level were partitioned into flows and segmented into windows. To obtain one feature vector from each window after segmentation, majority voting was performed for categorical features, and averaging was performed for numerical features.

2) Flow-based Complex Features: The second setting utilized a modified version of the MQTT-IoT-IDS packet-based dataset with complex flow-based features. Thirty-nine features were extracted. Part of these features were based on Lashkari *et al.* work [9], and the other part are novel features developed in this work. Table I describes the complex features of the second setting.

The highest, lowest, and number of unique protocols, sources, and ports are calculated per flow. This would give the algorithm an idea of where the packets in a flow generally originate from. Nine features relate to the time to live and the packet length of the packets in the flow. The mean, standard deviation, max, and min are calculated for each respectively (see Lashkari *et al.* for details on calculation [9]). Packet length can be indicative of flooding attacks due to small packets being used to increase efficiency. The mean of the time to live has also been found to reduce the number of false positives in smaller flows [10]. The total value of each IP and TCP flag is calculated for each packet in the flows.

In this study, we expand on the flags normally used for flow calculation for a more robust view on the attacks. The novel developed features are in bold in Table I. 'multi_dst_flag' is true if the source is interacting with more than one destination (i.e. parallel flows and a single source). 'num_dst' is number of the destinations that the source interact within a predefined period (a day for example). 'fwd_packet_ct' is the log of the order of this forwarded packet in this flow. The reason for using the log is because the count is exploding for some flows, especially flood attacks. 'bwd_packet_ct' is the log of the order of last backward packet in this flow.

TABLE I
COMPLEX FEATURE NAMES, DATA TYPES, AND DESCRIPTIONS.THESE FEATURES WERE CALCULATED FOR EACH 16-PACKET WINDOW. THE NOVEL DEVELOPED FEATURES ARE IN BOLD.

Feature	Data Type	Description
send_time_delta_mean	Decimal	Mean of packet send time delta
send_time_delta_std	Decimal	STD of packet send time delta
send_time_delta_max	Decimal	Maximum send time delta
send_time_delta_min	Decimal	Minimum send time delta
response_time_delta_mean	Decimal	Mean of packet response time delta
response_time_delta_std	Decimal	STD of packet response time delta
response_time_delta_max	Decimal	Maximum response time delta
response_time_delta_min	Decimal	Minimum response time delta
packet_length_total	Integer	Total of packet length
packet_length_mean	Integer	Mean of packet length
packet_length_std	Decimal	STD of packet length
packet_length_max	Integer	Maximum packet length
packet_length_min	Integer	Minimum packet length
ip_protocols_unique	Integer	Number of unique IP protocols
ip_protocols_highest	Integer	Numerically highest IP protocol
ip_protocols_lowest	Integer	Numerically lowest IP protocol
src_port_unique	Integer	Number of unique source ports
src_port_highest	Integer	Numerically highest source ports
src_port_lowest	Integer	Numerically lowest source ports
dst_port_unique	Integer	Number of unique destination ports
dst_port_highest	Integer	Numerically highest destination ports
dst_port_lowest	Integer	Numerically lowest destination ports
ip_flag_df_total	Integer	Total of Don't fragment IP flag
ip_flag_mf_total	Integer	Total of More fragments IP flag
ip_flag_rb_total	Integer	Total of Reserved IP flag
tcp_flag_res_total	Integer	Total of Reserved TCP flag
tcp_flag_ns_total	Integer	Total of Nonce sum TCP flag
tcp_flag_cwr_total	Integer	Total of Cong. Reduced TCP flag
tcp_flag_ecn_total	Integer	Total of ECN Echo TCP flag
tcp_flag_urg_total	Integer	Total of Urgent TCP flag
tcp_flag_ack_total	Integer	Total of Acknowledgement TCP flag
tcp_flag_push_total	Integer	Total of Push TCP flag
tcp_flag_reset_total	Integer	Total of Reset TCP flag
tcp_flag_syn_total	Integer	Total of Synchronization TCP flag
tcp_flag_fin_total	Integer	Total of Finish TCP flag
multi_dst_flag	Boolean	Source IP with multiple distinations
num_dst	Integer	Total number of destinations
fwd_packet_ct	Integer	The order of this forwarded packet
bwd_packet_ct	Integer	The order of last backward packet

C. Models

In this section, we give a brief overview of the traditional machine learning algorithms utilized for testing IDS capabilities.

A Decision Tree algorithm (DT) was one of the algorithms implemented to detect zero-day attacks within our two scenarios. It recursively partition the input space into subsets, assigning labels or predicting values based on a series of decision rules represented by a tree-like structure [11], [12]. This hierarchical structure allows the algorithm to make sequential decisions at each node, leading to the assignment of labels or prediction of values based on the learned rules. Decision Trees are particularly well-suited for their interpretability, providing a transparent representation of the decision-making process. It is proposed by Rai *et al.* as a signature-based IDS system [13].

Support Vector Machine (SVM) creates a hyperplane between two classes in an N-dimensional space [14]. A linear SVM finds this dimension in a 2D space and, therefore, is linearly separable. The hyperplane is strategically positioned to maximize the margin between the two classes, where the margin represents the distance between the hyperplane and the nearest data points of each class. By maximizing this margin, linear SVMs aim to enhance their generalization ability, allowing them to perform well on unseen data. The linear separability assumption makes linear SVMs particularly efficient for binary classification tasks where the decision

boundary can be represented as a straight line in a two-dimensional feature space. It is proposed by Jha *et al.* with feature weight as an IDS [15]

k-Nearest Neighbor (KNN), a model implemented for the detection of zero-day attacks in our system, operates by assigning labels to new, unlabeled data points based on the labels of their k-nearest neighbors. In this approach [16], the algorithm measures the distance between the new data point and its k-nearest neighbors in the feature space. The labels of these neighbors are then aggregated to determine the label for the new data point. KNN is a non-parametric and lazy learning algorithm, meaning that it does not make assumptions about the underlying data distribution and defers computation until prediction time. The choice of the parameter k influences the algorithm's sensitivity to local variations in the data.

Logistic regression is a regression model utilized commonly for binary classification. This model enables the separation of non-linear data utilizing Equation 1 [17]. The horizontal shift is represented by b_0 and the slope of the curve is represented as b_1.

$$p = \frac{1}{1 + e^{-(b_0 + b_1 x)}} \tag{1}$$

Gaussian Naive Bayes is a classification model based on probability and Bayes' theory [18]. The probability of being a part of a label is done for each feature independently in this model, with the highest probability leading to a chosen label. Specifically, it assumes that features are conditionally independent given the label. For new data points, the model computes the probabilities for each label based on the observed values of each feature. The chosen label corresponds to the one with the highest calculated probability, reflecting the most likely classification. Gaussian Naive Bayes is particularly well-suited for scenarios where features are continuous and follow a Gaussian (normal) distribution.

Gaussian Naive Bayes, a classification model incorporated into our detection system for zero-day attacks, is grounded in probability theory and Bayes' theorem, as outlined by Meerja et al. [18]. This model operates by independently calculating the probability of each feature contributing to a specific label. Specifically, it assumes that features are conditionally independent given the label, despite the "naive" assumption that might not hold in all real-world scenarios. For new data points, the model computes the probabilities for each label based on the observed values of each feature. The chosen label corresponds to the one with the highest calculated probability, reflecting the most likely classification. Gaussian Naive Bayes is particularly well-suited for scenarios where features are continuous and follow a Gaussian (normal) distribution, making it a valuable probabilistic approach in our holistic strategy for zero-day attack detection.

Random Forest, a classification model employed in our system for detecting zero-day attacks, harnesses the strength of decision trees within an ensemble framework [19]. This model builds a multitude of decision trees during the training phase. Each tree is constructed using a subset of the training data and a random selection of features, introducing diversity among the individual trees. The label assigned to a new data point is determined through a voting mechanism across the ensemble. Specifically, the majority vote among the decision trees is taken as the final prediction. This ensemble approach imparts robustness to the model by mitigating overfitting and enhancing generalization to unseen data. Random Forest's capability to provide insights into feature importance makes it a valuable component for zero-day attack detection.

III. Results and Discussions

Precision, Recall, and the F-1 Score are the metrics used for comparing the traditional machine learning models. A weighted average calculation was taken for each metric as in Equations 2, 3, and 4 respectively. True Positive (TP) represents the correct classification of an attack, True Negative (TN) is the correct classification of standard traffic, False Positive (FP) is the incorrect classification of standard traffic as an attack, and False Negative (FN) is the incorrect classification of an attack as standard traffic.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

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

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Split-at-scenario cross-validation is utilized for the segmentation of the data. This validation method has some attacks held out for testing while others are used for training the models. The held-out attacks represent zero-day attacks as the models have never seen them, enabling true validation of each model's success. If a model does not perform with this validation method, the model is not able to generalize to network anomalies and unknown attacks. This would make the model unfit for IoBT applications.

A. Split-at-Scenario Cross-Validation with Raw Features

We performed four-fold cross-validation for training and testing of the traditional machine learning models reported in the work of Hindy *et al.* on the MQTT-IoT-IDS2020 dataset. All attack types were used for training, with one held out for testing. In this setting, the input for the models was the packet-based statistical features (refer to the work of Hindy *et al.*) [5].

Table II presents the results of this setting. All models failed to detect at least one attack with F1 scores below 0.5. Models that fail to detect one or more zero-day attacks are not suitable for IoBT. Gaussian Naive Bayes was the best-performing model with a minimum of 0.280 F1 score detecting Sparta. The most difficult-to-detect attack was MQTT-Brute Force with an average F1 Score of 0.163. Naive Bayes was the only model to detect this attack successfully.

TABLE II
TESTING F1-SCORE, PRECISION, AND RECALL OF TRADITIONAL MODELS USING SPLIT-AT-SCENARIO CROSS-VALIDATION ON MQTT-IOT DATASET AND **RAW PACKET FEATURES**.

Model	Scan-A			Scan-sU			Sparta			MQTT-Brute Force		
	Prec.	Rec.	F1-Score	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.
DT	0.990	0.990	0.990	0.660	0.000	0.000	0.990	1.000	0.998	0.000	0.000	0.000
L-SVM	1.000	0.190	0.320	1.000	0.750	0.860	1.000	1.000	1.000	0.500	0.000	0.000
KNN	0.990	0.100	0.180	0.990	0.400	0.570	0.990	1.000	0.990	0.750	0.000	0.000
LR	1.000	0.400	0.570	0.990	0.990	0.990	0.970	1.000	0.980	0.000	0.000	0.000
NB	0.990	0.670	0.800	0.990	0.980	0.990	0.160	1.000	0.280	0.990	0.970	0.980
RF	0.990	0.990	0.990	0.990	0.980	0.990	0.990	1.000	0.990	1.000	0.000	0.000

B. Split-at-Scenario Cross-Validation with Complex Features

Four cross-validation folds trained and tested the same traditional machine learning models. We applied a standard scaler transform across each cross-validation fold in this setting. The complex features measured using flow generation were inputted into the six traditional machine learning models.

Table III displays the results utilizing the flow-based complex features. SVM, KNN, and NB failed to detect one attack with F1 scores at or below 0.500. The most difficult-to-detect attack was Sparta with an average F1 score of 0.561. RF detected this attack with the most consistency. RF achieved the best and most consistent results with an average F1 score of 0.999. Due to this consistency, it has the potential as IDS for IoBT applications. Besides, in an RF model, there are two common methods to assess feature importance that increases the interpretability of the model [20]. The first method is mean decrease in impurity that measures the impurity decrease caused by each feature across all trees in the forest. The second method is permutation importance that evaluates the importance of a feature by randomly permuting its values and measuring the impact on the model's performance.

C. Comparison

Each setting employed a split-at-scenario four-fold cross-validation method. The first setting utilizing raw packet-based features found that none of the models successfully detected all zero-day attacks. The second setting fared better with three traditional machine-learning models successfully detecting all zero-day attacks. Interestingly, NB was the most consistent for the first setting while RF was the most consistent for the second setting.

Figure 1 displays the relationship found between the two tested settings. Detection was counted as successful only if the model had an F-1 score above 0.500 in the specific attack. To achieve a 100% detection rate, models must successfully detect all attacks. Three models (DT, SVM, and RF) achieve consistent high detection rates by detecting all of the four attacks.

Mishin *et al.* and Booij *et al.* have previously measured other datasets utilizing split-at-scenario cross-validation. Mishin *et al.* found a drop in F1 score from 0.880 to 0.710 on the IoT-23 dataset [6] while Booij *et al.* displayed low accuracy across their models (30.7% to 63.4%) [7]. Our proposed models show significant improvement, the average F1 score of Random Forest being 0.997 with a detection rate of 100% in our second scenario.

Split-at-random cross-validation selects a fold not considering the attack or dataset. Utilizing split-at-random cross-validation, Hindy *et al.* found that the best-performing model was KNN, with an accuracy of 69.13% for packet-based features [5]. In the first setting, we found that Naive Bayes had an accuracy of 74.83%. While utilizing the same input data with split-at-scenario cross-validation, we achieved better accuracy than split-at-random. The second setting saw Random Forest being the most consistent with an accuracy of 99.92%. This improvement was accomplished by utilizing split-at-scenario and our engineered flow generation compared to split-at-random and packet-level features.

IV. CONCLUSION

Zero-day attacks exploit previously undisclosed vulnerabilities in software or hardware, posing a significant and increasing risk on IoT and IoBT as they occur before developers have had an opportunity to create and distribute patches or updates to mitigate the vulnerabilities. We built upon literature and develop novel network flow-based features to capture network flow abnormalities. Six different traditional machine learning algorithms were trained in two different settings. The first setting utilized raw packet features from the MQTT-IoT-IDS2020 dataset. We found that none of the six traditional models tested could detect all zero-day attacks using raw features. The most consistent model, NB, was the only model to detect the MQTT-Brute Force attack successfully. The second setting utilized novel flow-based complex features from the raw packet dataset. Each flow was limited to 16 packets to ensure that, if live, the system would not have to wait for a long flow to complete for detection. RF achieved the most consistent F1 score (an average of 0.999) with a 100% detection rate. This indicates that RF has the potential for IoBT applications with our proposed flow-based complex features. As only one dataset was utilized in our testing, future studies should investigate the validation of the proposed system with additional datasets with split-at-scenario cross-validation.

ACKNOWLEDGMENT

This work is done in the Tecore Networks Lab at Florida Atlantic University and is funded by the Office of the Secretary of Defense (OSD), Grant Number W911NF2010300.

TABLE III
TESTING F1-SCORE, PRECISION, AND RECALL OF TRADITIONAL MODELS USING SPLIT-AT-SCENARIO CROSS-VALIDATION ON MQTT-IOT DATASET AND **COMPLEX FEATURES**.

Model	Scan-A			Scan-sU			Sparta			MQTT-Brute Force		
	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.	Prec.	Rec.	F1.
DT	0.995	0.995	0.995	1.000	1.000	1.000	0.959	0.578	0.692	1.000	1.000	1.000
L-SVM	0.407	0.638	0.497	1.000	1.000	1.000	0.979	0.961	0.966	1.000	1.000	1.000
KNN	0.996	0.996	0.996	1.000	1.000	1.000	0.956	0.058	0.028	1.000	1.000	1.000
LR	0.993	0.993	0.993	1.000	1.000	1.000	0.958	0.421	0.545	1.000	1.000	1.000
NB	0.995	0.995	0.995	0.980	0.975	0.976	0.139	0.031	0.008	1.000	1.000	1.000
RF	0.997	0.997	0.997	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

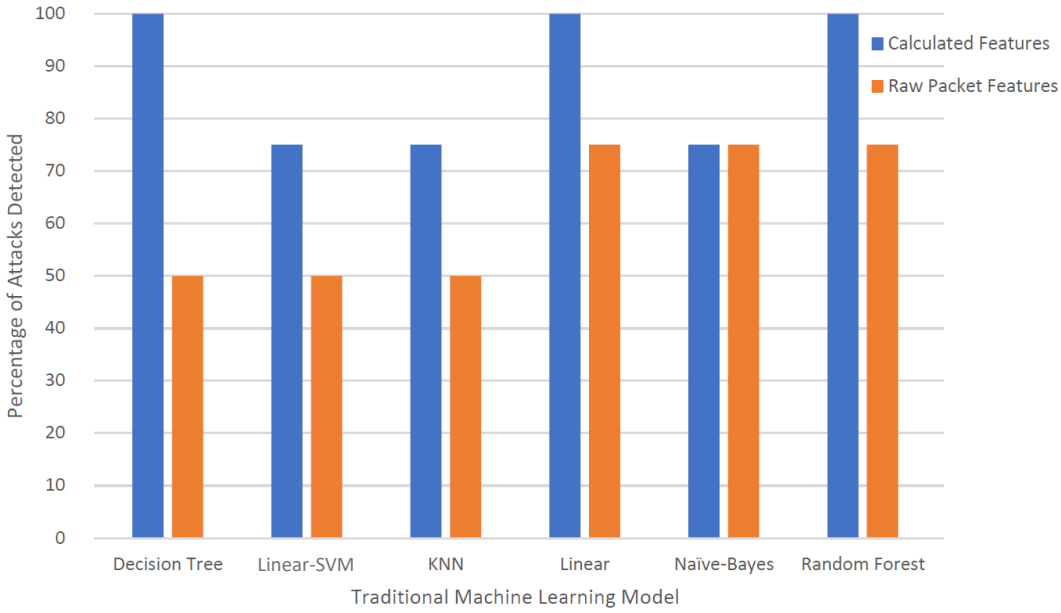

Fig. 1. Percentage of attacks detected for calculated and raw packet features

REFERENCES

[1] Y. Guo, "A review of machine learning-based zero-day attack detection: Challenges and future directions," *Computer Communications*, vol. 198, pp. 175–185, 2023.

[2] D. Instinct, "The economic value of prevention in the cybersecurity lifecycle," tech. rep., Ponemon Institute, 2020.

[3] L. Bilge and T. Dumitras, "Before we knew it: An empirical study of zero-day attacks in the real world," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 833–844, 10 2012.

[4] B. Hawkes, "Project zero," tech. rep., Google, 2022. https://googleprojectzero.blogspot.com/p/0day.html Accessed Dec 2023.

[5] H. Hindy, E. Bayne, M. Bures, R. Atkinson, C. Tachtatzis, and X. Bellekens, "Machine learning based iot intrusion detection system: An mqtt case study (mqtt-iot-ids2020 dataset)," 2020.

[6] M. M. et al., "Anomaly detection algorithms and techniques for network intrusion detection systems," *Machine Learning*, 2020.

[7] T. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. den Hartog, "Ton iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 05 2021.

[8] C. Peifer, G. Wolters, L. Harmat, J. Heutte, J. Tan, T. Freire, D. Tavares, C. Fonte, F. O. Andersen, J. van den Hout, M. Šimleša, L. Pola, L. Ceja, and S. Triberti, "A scoping review of flow research," *Frontiers in Psychology*, vol. 13, 2022.

[9] A. Habibi Lashkari, "Cicflowmeter-v4.0 (formerly known as iscxflowmeter) is a network traffic bi-flow generator and analyser for anomaly detection. https://github.com/iscx/cicflowmeter," 08 2018.

[10] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," *Proceedings - Conference on Local Computer Networks, LCN*, pp. 408–415, 10 2010.

[11] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[12] L. Rokach and O. Maimon, "Decision trees," *The Data Mining and Knowledge Discovery Handbook*, vol. 1, pp. 165–192, 2005.

[13] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *International Journal of Advanced Networking and Applications*, vol. 7, no. 4, p. 2828, 2016.

[14] V. V. Corinna Cortes, "Support-vector networks," *Machine learning*, vol. 20, pp. 408–415, 1995.

[15] J. Jha and L. Ragha, "Intrusion detection system using support vector machine," *International Journal of Applied Information Systems (IJAIS)*, vol. 3, pp. 25–30, 2013.

[16] Z. Zhang, "Introduction to machine learning: k-nearest neighbors," *Annals of Translational Medicine*, vol. 4(11), 2016.

[17] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *The Journal of Educational Research*, vol. 96, no. 1, pp. 3–14, 2002.

[18] A. J. Meerja, A. Ashu, and A. Rajani Kanth, "Gaussian naïve bayes based intrusion detection system," in *Proceedings of the 11th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2019) 11*, pp. 150–156, Springer, 2021.

[19] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[20] B. Gregorutti, B. Michel, and P. Saint-Pierre, "Correlation and variable importance in random forests," *Statistics and Computing*, vol. 27, pp. 659–678, 2017.