An Advanced Filter-based Supervised Threat Detection Framework on Large Databases

Lakshmi Prasanna Byrapuneni
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, India
lakshmiprasanna.byrapuneni@klh.edu.in

Maligireddy Saidi Reddy
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, India
msreddy33@klh.edu.in (corresponding author)

Received: 9 May 2024 | Revised: 20 May 2024 | Accepted: 29 May 2024

 DOI: https://doi.org/10.48084/etasr.7779

ABSTRACT

Adaptive and robust detection mechanisms are becoming more and more necessary as cyber threats become more complex. This study presents a framework to increase threat detection efficiency and address the complex problems posed by various dynamic cyber threats. This study focuses primarily on investigating a new algorithm for feature classification and selection in predictive modeling applications. Using a sizable real-time threat detection dataset, a hybrid filter-based feature ranking and cluster-based classification approach is proposed. A detailed analysis was carried out to investigate the performance of the proposed algorithm and compare it with various machine-learning models. This study also examines how well the algorithm scales to large-scale datasets and adapts to different data properties. The results highlight the algorithm's potential to enhance the efficiency of predictive modeling by optimizing feature selection procedures and reducing model complexity, thus making a substantial contribution to the field of data-driven decision-making and the wider range of machine-learning applications.

Keywords-multi-class classification; data filtering; outlier detection; cyber-attack detection

I. INTRODUCTION

Cloud storage faces failures due to security risks and natural disasters. In [1], a model was proposed to address crashes and availability issues in data centers using fault handlers to localize failure nodes and AI agents to detect hostile attacks, allowing massive data recovery in extended timeframes. Intrusion Detection Systems (IDSs) play a crucial role in safeguarding the privacy, authenticity, and accessibility of cloud networks [2]. Many studies have examined various IDS types, including cloud-based IDS, NIDSs, HIDSs, Hypervisor-based IDSs, and DIDSs, to meet security requirements and identify known and new threats. Cloud environments integrate Intrusion Detection and Prevention Systems (IDPS) to detect and prevent data breaches [3]. An IDPS framework identifies harmful data in the cloud network, storing them in a database to prevent future attacks.

Cloud computing raises significant concerns about the security of sensitive information, with data encryption being the primary defense mechanism. In [4], VAN-FED-IDS was proposed, utilizing federated learning to accelerate model training and updates and achieve remarkable accuracy. In [6], a PSO-AdaBoost approach was presented to classify various attack types, achieving a recall rate of 96.6%. In [7], an ensemble model for IoT IDS was proposed, based on LSTM and autoencoders, achieving impressive performance. In [8], a context-aware threat detection and response system was proposed to distinguish between real-world typical and attack scenarios. In [9], various machine models were compared to detect attacks on power grids with unsatisfactory results. In [10], federated learning was proposed along with blockchain to detect and share threat detection models, showing remarkable accuracy in two datasets. In [11], cyber-persona identification and profiling techniques were used to detect possible internal threats within businesses. This approach examined a range of variables, including communication styles, access trends, and user behavior, to build cyber-personas or profiles for each member of the business, and then used deep learning methods to detect aberrant behaviors or suspicious activity. In [12], the CTIMD technique was proposed, which used Cyber Threat Intelligence (CTI) and API call sequences to improve the accuracy of malware detection. This study focused on examining API call sequence details along with CTI to spot dangerous activities that indicate malware activity. This technique promises to increase malware detection efficacy, offering insights into known cyber threats and attack patterns.

The studies in [13, 14] highlighted the need for high-quality datasets to successfully train and test IDS classifiers. Specifically, in [14], the shortcomings of current datasets were explored, such as how they do not represent real-world cyber threats. Additionally, this study proposed a generative network to create artificial data that closely resembles actual network

traffic and attack scenarios. This technique could entail adding different traits and attributes of both normal and attack network activity to the produced data. In [15], a smart intrusion and fault identification model was proposed to detect intrusions and defects in cyber-physical microgrids that depend on inverters for power management, using machine learning techniques to improve the security and dependability of microgrids. In [16], the Random Forest algorithm outperformed others in identifying anomalous behavior in cloud computing. By streamlining the complexity using the best-first feature selection technique, this model achieved 99% detection accuracy and 93.6% classification accuracy. This method has potential applications in multi-cloud and cloud computing environments for anomalous traffic analysis.

ALGORITHM 1: NON-LINEAR DATA FILTERING APPROACH

```
Initialization
Create Arrays:
  An empty array DData_c to store calculated
  distances.
  An empty array DData_m to store minimum distances.

//Calculate Neighbor Distances for each data point
  For each data point x in IData (up to the
  length of IData − a):
    For each neighbor n within the distance window a:
      If both IData[x] and IData[x + n] are valid
      (not missing):
        //Calculate the distance between IData[x] and
        //IData[x + n] using Euclidean distance:
        distance= sqrt(Σ_{i=1}^{p} (IData[x_i] − IData[x_{i+n}])²)

//Find Minimum Distances for each data point
  For each data point x in IData:
    //Find the minimum distance across the
    //calculated neighbor distances in DData_c[x,:]
    DData_m[x] = min(DData_c[x,:])

//Normalize Distances - Find the global maximum and
//minimum values across all minimum distances
  maxdist=max(DData_m), mindist=min(DData_m)
  For each data point x in IData:
    If IData[x] is valid (not missing):
      //Normalize IData[x] using
      IData[x] = (IData[x] − DData_m[x]) / (maxdist − mindist)

Output
//Return the normalized and processed dataset
IData_a
```

II. PROPOSED FRAMEWORK

Distributed computing allows training and using multiple classifiers and then combining them to improve classification performance. During the classification step, the classifiers decide whether the nodes are potential security threats or not. This step detects potential intrusions and can notify the appropriate way to defend before an intrusion actually happens. This method provides a robust structure for intrusion detection in networks, using advanced statistical techniques, machine learning, and distributed computing to gain scalability, reliability, and improved accuracy.

A. Data Pre-processing

Algorithm 1 details the data pre-processing process, which replaces missing numerical values from the input dataset with non-linear Gaussian estimation. This algorithm finds the distances between existing data points in the training dataset window and saves them for each data point. For every data point, it finds the minimum distance and normalizes the data. In the end, it returns the original and normalized (cleaned) dataset.

The t-statistic function calculates the T-score of a feature to quantify how well it discriminates between two known classes within the data. Let x_1, x_2, ..., x_n represent the data points for n instances with respective features. Depending on the class, denote the set of instances by S1 and S2. The function first computes the mean and standard deviation of the features for the S1 and S2 classes. Next, the function computes the sample sizes of S1 and S2. Then, the function computes the associated T-score by dividing the difference between the means of both classes by their respective standard deviations divided by the number of instances. The t-statistic function is applied to each feature (), and each T-score is stored, as shown in Algorithm 2.

ALGORITHM 2: ENSEMBLE FEATURE RANKING FOR SUBSET SELECTION

```
//Step 1: Calculate class-wise means
Calculate the mean (μ_+) for the positive class:
μ_+ = (1/n_+) Σ_{i=1}^{n_+} x_i
where label=1
Calculate the mean (μ_-) for the negative class:
μ_- = (1/n_-) Σ_{i=1}^{n_-} x_i
where label=-1

//Step 2: Calculate class-wise Standard Deviations
Calculate the standard deviation (σ_+) for the
positive class:
σ_+ = sqrt((1/(n_+ − 1)) Σ_{i=1}^{n_+} (x_i − μ_+)²)
where label=1
Calculate the standard deviation (σ_-) for the
negative class:
σ_- = sqrt((1/(n_- − 1)) Σ_{i=1}^{n_-} (x_i − μ_-)²)
where label=-1

//Step 3: Calculate sample sizes
Calculate the number of samples for the positive
class:
n_+ = Σ (label = 1)
Calculate the number of samples for the negative
class:
n_- = Σ (label = −1)

//Step 4: Calculate the T-score
Calculate the difference in means:
diff_in_means= |μ_+ − μ_-|
Calculate the combined variance:
combined_variance= (σ_+²/n_+) + (σ_-²/n_-)
Calculate the T-score:
T_score = diff_in_means / sqrt(combined_variance)

//Step 5: Output
Return the calculated T-score:
Return T_score
```

The initialization of parameters in a Bayesian Network (BN) involves setting the initial values for the conditional probabilities associated with each node in the network. Building a BN involves setting initial values for conditional probabilities associated with each node. For each network, there are a number of these conditional probabilities, so there is potentially a large number of parameters that need initialization. Each dataset sample is examined to compute the posterior distribution of hidden variables conditioned on the observed variables. This can indicate how likely a hidden variable will have an assumed value given the observed data and the current parameters of the BN. The next step is the Maximization step (M-step), which updates the BN parameters to maximize the expected log-likelihood of the observed data given the calculated posterior probabilities, which improves the fit of the parameters to the observed data. If the structure of the BN is known a priori, techniques such as MLE can be used, which is a model that finds the most likely parameters given the observed data. If the structure of the BN is not known, both the structure and the parameters can be updated at the same time using the Expectation-Maximization with Structural EM (EM-SEM) algorithm. Convergence is checked after each EM iteration. The algorithm terminates when parameter changes fall below a given threshold or due to the log-likelihood converging, indicating that the parameters are at a stable stationary distribution. Otherwise, the E-step and M-step are repeated until either scenario happens. When the algorithm converges, it returns the optimal parameters of the BN, containing the best estimates for conditional probabilities derived from the EM algorithm's iterative process.

Bayesian hyperparameter optimization searches for configurations that produce the best result for a specific threat identification task. Hyperparameters are set for specific performance metrics such as F-score, MCC, or ROC. Specific hyperparameter configurations can be sampled on new real-world datasets, and the results will reflect the variance of the specific configuration. Features can be selected using engineering methods such as mutual information, correlation analysis, and expert domain knowledge. The hyperparameters are fine-tuned to improve model performance. Changes in these hyperparameters minimize the loss function, which adjusts the parameters to optimize learning.

In Algorithm 3, called Parallel Multi-Class EM-KMeans Based Classification (Hybrid BN), computational efficiency is improved by exploiting the parallel processing capabilities of multi-core computing systems and distributed computing environments. Through parallel execution, the complexity caused by the data size is segmented into smaller, manageable partitions for the E-step and M-step procedures. This allows the parallel processes to be executed on independent processors simultaneously. Communication among processors is facilitated by using global variables to capture the results of the E-step and M-step calculations. The K-means clustering steps are also executed concurrently, with the additional feature of using methods and procedures for merging the K-means clustering results and updating the global model parameters. This approach ensures the optimal use of multi-core systems and parallel computing resources in distributed environments to accelerate the execution of ESMC classification while maintaining computational precision. Furthermore, the extra time required for processing is minimal, making this method suitable for real-time applications such as anomaly detection, where fast processing of large datasets is crucial for immediate detection and response. Continuous monitoring of system load and distribution enhances the effective management of varying data volumes according to the applied rules. It also facilitates balancing computational processes in a multi-core setting and sharing the distributed workload in networked parallel computing environments, where resources can be strategically allocated for efficient utilization.

ALGORITHM 3: PARALLEL MULTI-CLASS EM-K-MEANS BASED CLASSIFICATION (HYBRID BN)

```
INPUT: dataset – A collection of data samples
(observations)
OUTPUT: optimal_parameters: The learned parameters of
the Bayesian Network (conditional probabilities and
potentially its structure)

PROCEDURE:
//Initialization:
//Initialize the parameters of the Bayesian Network:
  For known structure: Initialize conditional
  probability distributions.
  For unknown structure: Initialize an empty or basic
  network structure and corresponding parameters.

//Main EM Loop:
  REPEAT until convergence
    //Expectation Step (E-step):
    COMPUTE posterior_probabilities =
      CALCULATE_POSTERIOR(sample, current_parameters)
    For each sample x_i in the dataset:
      Calculate the posterior probabilities P(Z_j|X_i)
      for each hidden variable Z_j given the observed
      data X_i and current parameters

    //Maximization Step (M-step):
    IF structure of the Bayesian Network IS KNOWN:
    UPDATE optimal_parameters =
      MLE_ESTIMATION(dataset,
        posterior_probabilities)
    //Use Maximum Likelihood Estimation (MLE) or
    //another suitable method to update network
    //parameters based on the observed data and
    //calculated posterior probabilities.
    ELSE: (structure IS UNKNOWN)
      UPDATE optimal_parameters, structure =
      SEM(dataset, posterior_probabilities)
      //Employ a Structure learning within EM (SEM)
     //algorithm to simultaneously update the
     // network's structure and parameters.

//Check Convergence:
   IF change_in_parameters < threshold OR
     log_likelihood_converges:
     Exit the EM loop
//if parameters have changed minimally or the model's
//log-likelihood has stabilized.

//Output:
  RETURN optimal_parameters
```

III. EXPERIMENTAL RESULTS

UNSW-NB15 is a benchmark dataset for research on network IDSs that comprises more than 2.5 million network connection records, including normal behavior and nine types of attacks, namely analysis, backdoor, denial of service, exploits, fuzzers, generic, reconnaissance, shellcode, and worms. This dataset contains a total of 49 features derived from network traffic (packet-based and flow-based), making it a standard for evaluating the performance of IDSs. The NB15 benchmark is often used to demonstrate the efficacy of machine or deep learning algorithms for various computer security solutions [17-21].

TABLE I. PARAMETERS OF EXISTING MODELS FOR TRAINING PROCESS

Model	Layers/Nodes/Parameters
SVM +Kmeans	SVM parameters: kernel = 'linear', C = 1.0, gamma = 'scale' Kmeans parameters: n_clusters = 5, max_iter = 300, n_init = 10
NB +Kmeans	Naive Bayes parameters: var_smoothing = 1e-9 Kmeans parameters: n_clusters = 5, max_iter = 300, n_init = 10
SVR +EM	SVR parameters: kernel = 'rbf', C = 1.0, epsilon = 0.2 EM parameters: n_components = 3, max_iter = 100, tol = 1e-4
ANN +EM	ANN layers: 3 hidden layers, Nodes per layer: [64, 32, 16], Activation: ReLU, Output layer: 1 node, Activation: 'sigmoid' EM parameters: n_components = 3, max_iter = 100, tol = 1e-4

A. *Training of the Models*

The models shown in Table I were trained and tested. Training these models entailed a set of steps to ensure that each one was properly calibrated and fine-tuned for performance. All models were trained using the same UNSW-NB15 processed dataset to ensure consistency and allow a fair comparison of performance between the different models. This study integrated these models using a hybrid approach consisting of filtering and cluster-based classification. The key innovation was the use of parallel processing to effectively perform learning on large datasets and facilitate the simultaneous execution of all training steps. Initial training involved supervised learning models for classification and unsupervised learning approaches for clustering and refinement to ensure optimized classification performance and high recall.

For SVM+Kmeans, a Support Vector Machine (SVM) was first trained to find the optimal hyperplane that separates the two classes and maximizes the margin between them. Meanwhile, regardless of classification, Kmeans clustering was applied to partition and identify outliers and other data points based on a similarity measure of their features. This approach improved overall classification performance by capturing heterogeneity within the classification groups. The NB+Kmeans model first used NB to calculate the posterior probabilities of classes, given the feature values assuming feature independence, and then used K-means to cluster the centers of the underlying distributions. For the SVR+EM model, the Support Vector Regression (SVR) model was trained to predict continuous values, which were then binarised by several thresholding techniques. Then, the EM algorithm was applied to iteratively re-estimate the number of these hidden variables to maximize the expected log-likelihood computation, until reliability reached a threshold for the best performance. The ANN+EM model employed an Artificial Neural Network (ANN) trained through forward and backward propagation. It was optimized by inserting weights and biases to achieve lower values of the loss function over several epochs. Once the ANN training was over, the EM algorithm was applied to the output of the ANN network and first- and second-step predictions were obtained. More specifically, in the E-step, the posterior probabilities were estimated, and in the M-step, the log-likelihood was maximized, iterating until the parameters converged.

Figure 1 shows the recall performance of the five models. The recall performance of SVM+Kmeans was the lowest at just above 0.94. The NB+kmeans model was better, with a recall just above 0.96, and SVR+EM was slightly better. A substantial increase in recall was observed for the ANN+EM model, achieving a value just below 0.98. The proposed model achieved the best recall rate at just below 0.99. Figure 2 shows the F-score results for the five different machine learning models. All models achieved an F-score of 0.93 to 0.99. Again, the proposed model achieved the highest F-score. Figure 3 shows the quantitative results of the five machine learning models in terms of accuracy in the dataset, which ranged from 0.93 to 0.99.

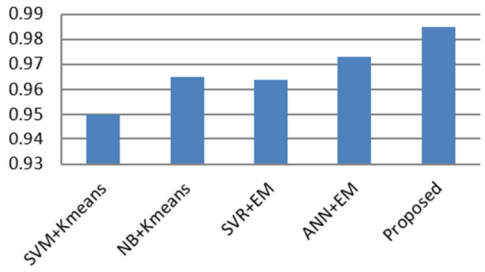

Fig. 1. Models' recall performance on cyber threat dataset.

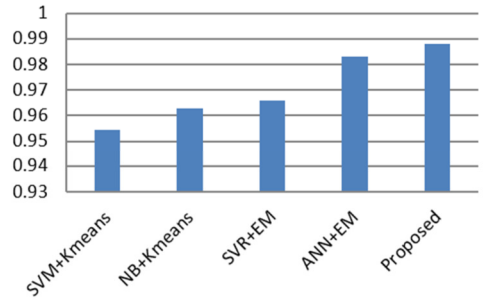

Fig. 2. Models' F-score performance on cyber threat dataset.

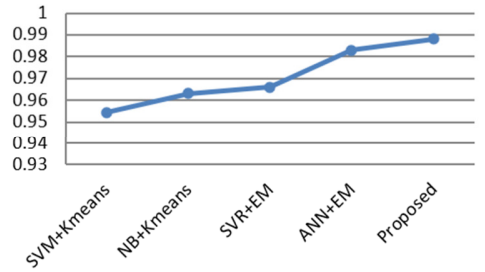

Fig. 3. Models' accuracy performance on cyber threat dataset.

IV. CONCLUSION

This study performed a robust analysis, including a thorough examination of performance parameters such as stability of feature selection, computational efficiency, and predictive accuracy. The scalability of the proposed algorithm on large-scale datasets and flexibility to different data attributes were examined, emphasizing its potential to improve the results of predictive modeling by optimizing feature selection procedures and reducing model complexity. Compared to conventional methods, encouraging results were obtained in terms of recall, F-score, and accuracy, through the introduction of a cluster-based classification method. The proposed approach provides a viable solution for the real-time detection of cyber attacks, helping to improve the security of IoT devices and networks. Additionally, parallel multiclass classification was investigated, providing a way to use parallel computing techniques to perform simultaneous classification tasks on large datasets. This method can significantly increase productivity and speed, particularly when handling a large number of data or when prompt response is necessary.

REFERENCES

[1] "Natural Disasters: A Perfect Storm for Data Breaches | CSA." https://cloudsecurityalliance.org/blog/2023/12/11/natural-disasters-a-perfect-storm-for-data-breaches.

[2] Y. Gao, Y. Liu, Y. Jin, J. Chen, and H. Wu, "A Novel Semi-Supervised Learning Approach for Network Intrusion Detection on Cloud-Based Robotic System," *IEEE Access*, vol. 6, pp. 50927–50938, 2018, https://doi.org/10.1109/ACCESS.2018.2868171.

[3] F. Nabi and X. Zhou, "Enhancing intrusion detection systems through dimensionality reduction: A comparative study of machine learning techniques for cyber security," *Cyber Security and Applications*, vol. 2, Jan. 2024, Art. no. 100033, https://doi.org/10.1016/j.csa.2023.100033.

[4] X. Chen, W. Qiu, L. Chen, Y. Ma, and J. Ma, "Fast and practical intrusion detection system based on federated learning for VANET," *Computers & Security*, vol. 142, Jul. 2024, Art. no. 103881, https://doi.org/10.1016/j.cose.2024.103881.

[5] S. Kannadhasan and R. Nagarajan, "Intrusion detection in machine learning based E-shaped structure with algorithms, strategies and applications in wireless sensor networks," *Heliyon*, vol. 10, no. 9, May 2024, https://doi.org/10.1016/j.heliyon.2024.e30675.

[6] Z. Sun, G. An, Y. Yang, and Y. Liu, "Optimized machine learning enabled intrusion detection 2 system for internet of medical things," *Franklin Open*, vol. 6, Mar. 2024, Art. no. 100056, https://doi.org/10.1016/j.fraope.2023.100056.

[7] A. Yazdinejad, M. Kazemi, R. M. Parizi, A. Dehghantanha, and H. Karimipour, "An ensemble deep learning model for cyber threat hunting in industrial internet of things," *Digital Communications and Networks*, vol. 9, no. 1, pp. 101–110, Feb. 2023, https://doi.org/10.1016/j.dcan.2022.09.008.

[8] Z. Noor, S. Hina, F. Hayat, and G. A. Shah, "An intelligent context-aware threat detection and response model for smart cyber-physical systems," *Internet of Things*, vol. 23, Oct. 2023, Art. no. 100843, https://doi.org/10.1016/j.iot.2023.100843.

[9] K. Aygul, M. Mohammadpourfard, M. Kesici, F. Kucuktezcan, and I. Genc, "Benchmark of machine learning algorithms on transient stability prediction in renewable rich power grids under cyber-attacks," *Internet of Things*, vol. 25, Apr. 2024, Art. no. 101012, https://doi.org/10.1016/j.iot.2023.101012.

[10] T. Jiang, G. Shen, C. Guo, Y. Cui, and B. Xie, "BFLS: Blockchain and Federated Learning for sharing threat detection models as Cyber Threat Intelligence," *Computer Networks*, vol. 224, Apr. 2023, Art. no. 109604, https://doi.org/10.1016/j.comnet.2023.109604.

[11] B. Racherache, P. Shirani, A. Soeanu, and M. Debbabi, "CPID: Insider threat detection using profiling and cyber-persona identification," *Computers & Security*, vol. 132, Sep. 2023, Art. no. 103350, https://doi.org/10.1016/j.cose.2023.103350.

[12] T. Chen, H. Zeng, M. Lv, and T. Zhu, "CTIMD: Cyber threat intelligence enhanced malware detection using API call sequences with parameters," *Computers & Security*, vol. 136, Jan. 2024, Art. no. 103518, https://doi.org/10.1016/j.cose.2023.103518.

[13] J. Zhang, J. D. Peter, A. Shankar, and W. Viriyasitavat, "Public cloud networks oriented deep neural networks for effective intrusion detection in online music education," *Computers and Electrical Engineering*, vol. 115, Apr. 2024, Art. no. 109095, https://doi.org/10.1016/j.compeleceng.2024.109095.

[14] M. Chalé and N. D. Bastian, "Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems," *Expert Systems with Applications*, vol. 207, Nov. 2022, Art. no. 117936, https://doi.org/10.1016/j.eswa.2022.117936.

[15] R. Divya, S. Umamaheswari, and A. A. Stonier, "Machine learning based smart intrusion and fault identification (SIFI) in inverter based cyber-physical microgrids," *Expert Systems with Applications*, vol. 238, Mar. 2024, Art. no. 122291, https://doi.org/10.1016/j.eswa.2023.122291.

[16] A. Gupta and R. Simon, "Enhancing Security in Cloud Computing With Anomaly Detection Using Random Forest," in 2024 11th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Mar. 2024, pp. 1–6, https://doi.org/10.1109/ICRITO61523.2024.10522227.

[17] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, Nov. 2015, pp. 1–6, https://doi.org/10.1109/MilCIS.2015.7348942.

[18] 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*, Apr. 2016, https://doi.org/10.1080/19393555.2015.1125974.

[19] 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*, vol. 5, no. 4, pp. 481–494, Sep. 2019, https://doi.org/10.1109/TBDATA.2017.2715166.

[20] N. Moustafa, G. Creech, and J. Slay, "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models," in *Data Analytics and Decision Support for Cybersecurity: Trends, Methodologies and Applications*, I. Palomares Carrascosa, H. K. Kalutarage, and Y. Huang, Eds. Cham, Switzerland: Springer International Publishing, 2017, pp. 127–156.

[21] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," in *Big Data Technologies and Applications*, 2021, pp. 117–135, https://doi.org/10.1007/978-3-030-72802-1_9.