International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/224435 Volume 11, Issue 1, January – February (2024)

RESEARCH ARTICLE

Expedient Intrusion Detection System in MANET Using Robust Dragonfly-Optimized Enhanced Naive Bayes (RDO-ENB)

M. Sasikumar
School of Computing Science, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, Tamil Nadu, India.
marksasiphd@gmail.com

K. Rohini
Department of Information Technology, School of Computing Science, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, Tamil Nadu, India.
rrohini16@gmail.com

Received: 07 November 2023 / Revised: 12 January 2024 / Accepted: 16 January 2024 / Published: 26 February 2024

Abstract – Mobile Ad hoc networks (MANETs) represent dynamic, self-configuring network environments that provide flexible connectivity but are highly susceptible to security threats. Intrusion detection systems in MANETs need to continuously monitor network traffic for potential intrusions and anomalies. This constant monitoring can be energy-intensive, requiring network nodes to process, analyze, and transmit data. Excessive energy consumption by IDS can deplete node batteries quickly, leading to network disruptions. This research focuses on developing and evaluating an efficient IDS proposed for MANETs called Robust Dragonfly-Optimized Naive Bayes (RDO-ENB). RDO-ENB operates by fusing the simplicity and efficiency of the Enhanced Naive Bayes algorithm with the adaptive capabilities of robust Dragonfly Optimization. This synergy enables RDO-ENB to continuously and dynamically adjust its internal parameters, optimizing its intrusion detection performance in real time. It enhances accuracy and reduces false positives, making it proficient in identifying and mitigating intrusions within the complex and ever-evolving environment of MANETs. The dataset employed for evaluation is NSL-KDD, a widely used dataset for intrusion detection. The results of the IDS implementation demonstrate its proficiency in accurately identifying and mitigating intrusions while minimizing false positives and conserving valuable energy resources.

Index Terms – Dragonfly, Naive Bayes, Intrusion, MANET, Classification, Chaos.

1. INTRODUCTION

The dynamic network topology is a hallmark of Mobile Ad Hoc Networks (MANETs), distinguishing them from traditional wired or fixed infrastructure networks. In MANETs, nodes are in a constant state of motion, frequently joining or departing from the network at their discretion [1]. This dynamic behavior is governed by sophisticated algorithms that enable nodes to adapt autonomously to the ever-changing network landscape. Without the constraints of static infrastructure, MANETs are highly versatile and ideal for applications requiring rapid deployment and flexibility. In MANETs, the network topology continuously evolves, guided by these advanced algorithms. When nodes enter or exit the network or change their positions, the network topology adjusts accordingly [2]. This dynamic nature allows MANETs to thrive in scenarios where traditional networks struggle, such as military operations, disaster recovery, and collaborative data sharing in mobile or ad hoc settings. Despite node movements' unpredictability, MANETs ensure uninterrupted data packet delivery, making them invaluable in dynamic communication environments. This dynamic network topology remains a core characteristic, reflecting MANETs' adaptability and resilience in scenarios where connectivity must be maintained regardless of changes in node locations [3].

Intrusion Detection Systems (IDS) are pivotal in contemporary cybersecurity strategies [4]. They act as vigilant sentinels, tirelessly monitoring computer systems and networks for any signs of unauthorized access, malicious activities, and potential security threats [5]. The core function of IDS is to identify and alert to these threats in real time. IDS generates alerts when suspicious activities are detected, enabling security teams to investigate and respond promptly, reducing potential damage. Despite their crucial role, IDS face challenges, such as the potential for generating false positives, which require careful tuning. Nevertheless, IDS are continually evolving to adapt to the ever-changing landscape of cybersecurity, ensuring the integrity and availability of digital assets [6], [7].

Intrusion Detection Systems (IDS) in MANETs are the frontline defense against various security threats. MANETs are dynamic, infrastructure-less networks where nodes communicate directly, making them vulnerable to data interception, routing attacks, and malicious nodes [8]. IDS in MANETs are specialized security mechanisms that promptly detect and respond to these threats. These systems continuously monitor network traffic, examining data packets, communication patterns, and node behaviors for deviations from regular operation [9], [10]. When suspicious activities or intrusion attempts are identified, IDS triggers alarms or initiates protective measures. The unique challenges of MANETs, such as node mobility, limited resources, and rapidly changing topologies, demand tailored intrusion detection solutions [11].

1.1. Problem Statement

Energy efficiency is paramount in IDS deployed in MANETs. MANETs often consist of battery-powered mobile devices with limited energy resources. These devices play a pivotal role in forming the network infrastructure, and their efficient operation is essential for the network's sustainability. The challenge revolves around designing IDS solutions prioritizing energy conservation without compromising the network's security. This involves the development of adaptive monitoring and detection strategies that can save power by intelligently managing the energy-intensive tasks associated with intrusion detection. Striking this balance between maintaining adequate security and prolonging the lifespan of battery-powered nodes is a critical research problem, necessitating innovative techniques and algorithms to optimize power consumption in MANET-based IDS.

1.2. Motivation for Problem Statement

The motivation for addressing energy efficiency in IDS within MANETs is twofold and compelling. MANETs often operate in scenarios with limited or no access to stable power sources, such as disaster response or military deployments. Mobile devices with finite battery resources are vital for network operation in these contexts. Extending the longevity of these devices is mission-critical. The growing complexity of network traffic and security threats demands real-time analysis, which strains the limited resources of these mobile devices. The motivation is to develop innovative techniques and algorithms that enable IDS in MANETs to function effectively within these energy constraints, thus enhancing the network's sustainability and resilience in critical scenarios.

1.3. Research Objective

The primary research objective is to design, develop, and evaluate an energy-efficient IDS tailored for MANETs. This research aims to create a robust IDS with minimal energy consumption while sustaining effective security measures. By incorporating a Dragonfly-Optimized Naive Bayes approach, the research seeks to intelligently adapt IDS monitoring and detection strategies to save power and maximize the operational lifespan of battery-powered devices within MANETs. The central motivation for this research is the necessity to enhance the sustainability and resilience of MANETs, which are frequently deployed in resource-constrained scenarios, including disaster response and military operations. The research objective is to contribute a novel, energy-efficient IDS solution, aligning security with the imperative to prolong the lifespan of critical mobile devices, thereby bolstering the network's performance in mission-critical contexts.

1.4. Organization of the Paper

The paper follows a structured organization, beginning with the introduction providing context. A thorough literature review follows, establishing the background. The core of the paper, Section 3, introduces the "Robust Dragonfly-Optimized Enhanced Naive Bayes Classifier." Section 4 details the dataset used, specifically the NSK-KDD Dataset. Performance metrics are discussed in Section 5. Sections 6 presents results and engages in a detailed discussion. Finally, Section 7 offers concluding remarks, summarizing findings and potential implications. This organized framework ensures a logical progression, guiding readers through the research process from introduction to conclusion.

2. LITERATURE REVIEW

The current section provides an overview of recent state-of-the-art literature related to IDS. It highlights a common issue encountered in these works: poor performance. The review focuses on studies and advancements in IDS that have faced challenges in achieving satisfactory detection rates, false alarm rates, or other performance metrics.

"Intelligent IDS" [12] introduces an intelligent IDS mechanism tailored for MANETs. It provides an innovative solution to enhance network security by actively monitoring and identifying potential intrusions. The mechanism includes a performance reliability evaluation aspect, ensuring the network functions efficiently and securely. "Flow-Based IDS" [13] presents a flow-based IDS designed for Vehicular Ad-Hoc Networks (VANETs). It utilizes context-aware feature extraction techniques to identify and respond to security threats within vehicular communication. The system's ability to analyze communication patterns and extract context-aware features enhances its effectiveness in detecting intrusions and malicious activities. "Secure Routing with IDS" [14] proposes a secure IDS routing protocol specifically engineered for MANETs. The protocol is designed to fortify network security by actively identifying and mitigating intrusions. It ensures that data routing within the network remains secure and potential threats are detected and addressed promptly.

"IDS with Neuro-Fuzzy Networks" [15] introduces an advanced IDS that leverages Exponential Henry Gas Solubility Optimization-based Deep Neuro Fuzzy Networks within MANETs. The utilization of this cutting-edge technology enables more accurate and efficient threat detection. It employs optimization techniques and neural networks to adapt to the dynamic nature of MANETs and improve the network's overall security. "SDN-VANET Authentication with Neural Network" [16] introduces an innovative authentication approach. It utilizes Rider-Sea Lion optimized neural networks for IDS, strengthening the security of vehicular communication and the overall VANET architecture. The neural network's optimization through Rider-Sea Lion techniques enhances its accuracy in identifying and responding to intrusions. "Optimized Probabilistic Neural Network" [17] presents a novel and optimized approach using probabilistic neural networks for IDS and categorization. The optimization of the network significantly improves its performance in accurately identifying and categorizing potential threats.

"Network Security with Signaling Game" [18] introduces an innovative IDS game designed to enhance security in vehicular networks. It employs a signalling game-based approach, where network entities engage in strategic interactions to detect and respond to potential intrusions. By treating network security as a game, this approach provides a dynamic and adaptive method for identifying and mitigating intrusions. "IDS Framework for Healthcare WSNs" [19] presents an efficient IDS framework tailored for healthcare Wireless Sensor Networks (WSNs). The primary focus is mitigating two critical security threats: blackhole and sinkhole attacks. These attacks pose significant risks to healthcare data integrity and patient safety. The framework provides advanced IDS mechanisms that actively monitor and respond to suspicious activities. "Transformer Neural Network" [20] introduces TNN-IDS, a cutting-edge IDS based on Transformer neural networks, designed explicitly for MQTT-enabled Internet of Things (IoT) networks. This system advances the security of IoT devices by providing sophisticated IDS capabilities. By leveraging the power of Transformer neural networks, it can effectively analyze complex and dynamic data patterns in IoT environments.

"IDS in Clustered Vehicle Networks" [21] employs an innovative approach by utilizing invasive weed optimization in combination with deep wavelet neural networks to detect intrusions in clustered vehicle networks. Invasive weed optimization is used to optimize the neural network's parameters, enhancing its ability to identify intrusions. The deep wavelet neural network enables the analysis of complex data patterns within vehicle networks. "Machine Learning-based IDS" [22] provides an in-depth analysis of IDS methodologies in MANETs, focusing on applying machine learning strategies. It offers insights into the effectiveness of various machine-learning approaches in enhancing network security. This analysis is instrumental in identifying the most suitable machine-learning strategies for IDS in dynamic and self-organizing MANETs. "IDSs for Wireless Mesh Networks" [23] delves into the design and analysis of IDSs tailored for wireless mesh networks. These systems are critical for securing wireless mesh networks and maintaining their integrity. It uses various IDS approaches and their effectiveness in protecting wireless mesh networks against unauthorized access and potential threats.

2.1. Summary

The section provides an overview of recent literature on Intrusion Detection Systems (IDS) with a focus on addressing the common issue of poor performance. Various innovative IDS mechanisms are discussed, each tailored to specific network environments such as MANETs, VANETs, and IoT networks. These mechanisms incorporate advanced technologies, including neuro-fuzzy networks and Transformer neural networks, to enhance the accuracy and efficiency of threat detection. The literature explores novel approaches such as game-based security in vehicular networks and IDS frameworks designed for healthcare WSNs. The above discussed state-of-the-art emphasizes the dynamic nature of IDS development and the importance of tailored solutions for diverse network architectures.

2.2. Research Gap

Several research gaps have been identified within the IDS domain from the above-discussed literature. Notable research gaps are discussed below:

- Low Classification Accuracy: Numerous studies have reported challenges related to low classification accuracy in IDS, especially when dealing with novel attack types or sophisticated evasion techniques. This gap necessitates further investigation into developing more accurate and robust IDS algorithms that effectively differentiate between legitimate and malicious network activities.
- Imbalanced Dataset Handling: Many IDS datasets are highly imbalanced, with a significantly larger proportion of normal network traffic samples than attack instances. This imbalance often leads to biased results and limits the overall effectiveness of IDS solutions.
- Adversarial Machine Learning: The rise of adversarial machine learning techniques poses a severe challenge to IDS. Researchers should explore methods to enhance the resilience of IDS against adversarial attacks and ensure the reliability of intrusion detection systems in the face of sophisticated adversaries.

These research gaps underscore the need for ongoing investigation and innovation in IDS to address current limitations and adapt to the evolving cybersecurity landscape.

3. ROBUST DRAGONFLY-OPTIMIZED ENHANCED NAIVE BAYES CLASSIFIER

3.1. Enhanced Naive Bayes Classifier

The Enhanced Naive Bayes Classifier is an advanced version of the traditional algorithm incorporating Taylor series expansions or approximations. This enhancement allows it to capture more complex relationships and deviations from standard parametric probability distributions, making it particularly well-suited for datasets with intricate, non-parametric structures. The Enhanced Naive Bayes Classifier can provide a flexible and accurate modelling approach by selecting an appropriate reference point and degree for the Taylor polynomial. It leverages the Taylor series to create polynomial approximations of conditional probabilities, thereby better representing the underlying data distribution and improving its predictive capabilities.

3.1.1. Probability Distribution

In enhancing the Naive Bayes algorithm using the Taylor series method, the first step involves precisely defining the probability distributions requiring approximation. These probability distributions are integral to the Naive Bayes classifier's operation. This research aims to approximate the conditional probabilities $P(X|Y)$, where X signifies a feature, and Y represents a class label. The research aims to express $P(X|Y)$ as a function of feature X and class Y. In the Naive Bayes algorithm, these conditional probabilities are pivotal for estimating the likelihood of observing a specific feature given a particular class.

$$P(X|Y)$$

Where $P(X|Y)$ is the conditional probability of feature X occurring given the class Y.

The standard approach in Naive Bayes often assumes parametric probability distributions, such as Gaussian, Bernoulli, or Multinomial, to model these conditional probabilities. In this enhanced approach, this research intends to leverage the Taylor series expansion to provide a polynomial approximation for $P(X|Y)$. This will allow for the incorporation of more complex, non-parametric distributions and potentially capture intricate relationships within the data.

3.1.2. Taylor Series Expansion

The Taylor series expansion is a technique in mathematics for approximating functions by writing them as a sequence of polynomials. In the context of enhancing the Naive Bayes algorithm, the Taylor series expansion will be employed to approximate conditional probabilities, particularly $P(X|Y)$, using polynomial representations. Eq.(1) expresses the Taylor series expansion of a function $f(x)$ concerning a fixed point.

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots. \quad (1)$$

where $f(a)$ represents the function that aims to approximate, a is the reference point, $f'(a)$ is the first derivative of the function at a, $f''(a)$ is the second derivative, and so on.

ENB selects a reference point a to represent a particular characteristic or parameter related to the conditional probability $P(X|Y)$. The choice of this reference point depends on the specific probability distribution and class label under consideration. For instance, in a Gaussian distribution, a could be the mean (μ) of feature X within class Y. The Taylor series expansion allows us to express $P(X|Y)$ as a polynomial approximation around the reference point a. This mathematical method can potentially capture complex, non-parametric relationships within the data and enhance the expressiveness of the Naive Bayes model. Algorithm 1 provides the pseudocode of Taylor Series Expansion.

```
1.  for each class Y
2.  for each feature X
3.  chosen_characteristic := choose_characteristic(Y, X);
4.  reference_point_a := calculate_reference_point(chosen_characteristic);
5.  derivatives := calculate_derivatives(Y, X);
6.  taylor_series_polynomial=construct_taylor_polynomial(derivatives, reference_point_a);
7.  train_naive_bayes_model();
8.  approximate_probability := use_taylor_polynomial(taylor_series_polynomial, X);
9.  enhanced_probability_estimations := incorporate_into_naive_bayes(approximate_probability);
10. end
11. end
```

Algorithm 1 Taylor Series Expansion

3.1.3. Reference Point Selection

In enhancing the Naive Bayes algorithm using the Taylor series method, selecting a suitable reference point a is a crucial step. This reference point will be the center around which we approximate conditional probabilities $P(X|Y)$ using Taylor series expansions. Mathematically, a is the chosen reference point, and its selection depends on the specific probability distribution and class label under consideration. It aligns with the core assumption of Gaussian distributions, which place the mean as the central location. Eq.(2) expresses the same for a Gaussian distribution.

$$P(X|Y) = \frac{1}{\sqrt{2\pi\sigma^2}}\, exp\left(-\frac{(P-\mu)^2}{2\sigma^2}\right) \qquad (2)$$

Where μ represents the mean, and σ is the standard deviation.

Selecting the reference point (a) is essential because it sets the center of the Taylor series expansion. This enables us to approximate $P(X \mid Y)$ as a polynomial with a as its center, allowing us to capture the underlying distribution's behavior around this point. The choice of a should align with the specific distribution assumptions and properties of the conditional probability ENB aims to approximate within the Naive Bayes framework. Algorithm 2 provides the pseudocode of Reference Point Selection.

```
1.  for each class Y:
2.  for each feature X:
3.  stribution_type := determine_distribution_type(Y, X)
4.  if distribution_type is Gaussian:
5.  a := calculate_mean(Y, X)
6.  else:
7.  parameter (θ) for P(X|Y).
8.  a := select_a_based_on_distribution(Y, X,
    distribution_type)
9.  if uncertain:
10. a := calculate_a_data_driven(Y, X)
11. refine_a_with_model_evaluation(Y, X, a)
12. a := incorporate_domain_knowledge(Y, X, a)
```

Algorithm 2 Reference Point Selection

3.1.4. Calculate Derivatives

Calculating derivatives is fundamental in enhancing the Naive Bayes algorithm using the Taylor series method. The specific derivatives to be calculated depend on the probability distribution we are dealing with, as different distributions have distinct probability density functions. In the case of a Gaussian distribution, this research needs to compute the first, second, and potentially higher-order derivatives. The Gaussian probability density function for $P(X \mid Y)$ using Eq.(3).

$$P(X \mid Y) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{(X-\mu)^2}{2\sigma^2}\right) \qquad (3)$$

Where μ represents the mean, and σ is the standard deviation.

First Derivative

Eq.(4) is applied to calculate the first derivative (i.e., $(f'(a))$) where differentiates $P(X \mid Y)$ and X, evaluating at the reference point a.

$$f'(a) = \frac{d}{dX} P(X \mid Y)\Big|_{X=a} \qquad (4)$$

Second Derivative

Eq.(5) is applied to calculate the second derivative (i.e., $(f''(a))$) by taking the first derivative concerning X and evaluating it at a:

$$f''(a) = \frac{d^2}{dX^2} P(X \mid Y)\Big|_{X=a} \qquad (5)$$

Higher-Order Derivatives

Higher-order derivatives may also be necessary depending on the chosen degree of the Taylor series expansion.

The three derivatives mentioned above are essential for the polynomial approximation of $P(X \mid Y)$ with the Taylor series framework, allowing to capture the local behavior of the conditional probability distribution around the reference point (a). The precision of these derivative calculations influences the accuracy of the approximation.

3.1.5. Taylor Series Approximation

This step is significant in representing the conditional probabilities in a manner that captures more complex relationships and deviations from the standard parametric distributions.

When a function $f(x)$ is expanded in the Taylor series for a fixed point a then Eq.(6) is applied.

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \cdots \qquad (6)$$

ENB uses this expansion to approximate $P(X \mid Y)$ as a polynomial around the reference point a, expressed as Eq.(7).

$$P(X|Y) \approx P(a|Y) + \frac{d}{dx}P(a|Y)(X-a) + \frac{1}{2!}\frac{d^2}{dx^2}P(a|Y)(X-a)^2 + \cdots \qquad (7)$$

$P(X|Y)$ is approximated as a sum of terms, with each term representing a derivative evaluated at the reference point a multiplied by a power of $(X-a)$. This polynomial approximation allows the expression of conditional probabilities more flexibly and non-parametrically. It captures the local behavior of the probability distribution around a, enabling it to account for more intricate relationships and distributions that simple parametric assumptions may not adequately model.

3.1.6. Model Selection

The selection of an appropriate degree for the Taylor polynomial in enhancing the Naive Bayes algorithm is a critical step in balancing the trade-off between the model's complexity and its ability to accurately capture the underlying conditional probability distributions. The choice of the polynomial degree influences the quality of the approximation and the computational resources required for computation.

The Taylor polynomial degree determines the number of terms included in the polynomial approximation of $P(X \mid Y)$. A higher degree incorporates more terms and offers a more intricate representation of the distribution, potentially capturing finer details. However, this comes at the cost of increased computational complexity and the risk of overfitting, where the model may become excessively tailored to the training data, leading to poor generalization.

The choice of the Taylor polynomial degree should consider the problem's specific characteristics and the data's nature. It is often determined through experimentation and validation on a validation dataset. Cross-validation techniques can also help assess the model's performance across different degrees. The mathematical expression for the Taylor polynomial approximation is expressed in Eq.(8).

$$P(X|Y) \approx P(a|Y) + \frac{d}{dx}P(a|Y)(X-a) + \frac{1}{2!}\frac{d^2}{dx^2}P(a|Y)(X-a)^2 + \cdots \quad (8)$$

The polynomial degree determines how many terms, such as the second and third derivatives, are included in the approximation. The pseudocode of the model selection in ERB is provided in Algorithm 3.

```
1. for each class Y:
2. for each feature X:
3. distribution_type := determine_distribution_type(Y, X)
4. polynomial_degrees_range := [1, 2, 3, 4, ...]  # Define a reasonable range
5. if validation_dataset_unavailable:
6. training_data, validation_data := split_training_data(dataset)
7. else:
8. training_data := dataset
9. best_degree := 0
10. best_performance := 0
11. for degree_d in polynomial_degrees_range:
12. taylor_polynomial := construct_taylor_polynomial(degree_d)
13. train_naive_bayes_model_with_polynomial(Y, X, taylor_polynomial)
14. performance := evaluate_model_performance(Y, X, taylor_polynomial, validation_data)
15. record_performance(degree_d, performance)
16. if performance > best_performance:
17. best_degree := degree_d
18. best_performance := performance
19. if trade_off_present:
20. final_degree := make_final_decision(best_degree, best_performance)
21. else:
22. final_degree := best_degree
```

Algorithm 3 Model Selection

3.2. Robust Dragonfly Optimization

By nature, Bio-inspired Optimization has the ability to solve the complex real world problems [24]–[26]. Robust Dragonfly Optimization (RDO) is a nature-inspired optimization algorithm- based on dragonflies' swarming behavior. It's used to solve complex optimization problems. RDO is an enhanced version of the Dragonfly Algorithm [27], incorporating chaos theory principles to improve performance. Here's an overview of how RDO works:

1. Initialization: Start by initializing a population of dragonflies with random positions and velocities. Each dragonfly represents a potential solution to the optimization problem.
2. Objective Function: Define the objective function that you want to optimize. This function represents the problem you're trying to solve.
3. Chaos Initialization: Introduce Chaos to the system by applying chaos theory principles. This Chaos can be introduced by using a chaotic map or a chaotic equation to perturb the positions and velocities of the dragonflies.
4. Movement and Swarming: Dragonflies move through the search space by adjusting their positions and velocities. Dragonflies in a swarm may communicate with one another and work together to fly toward the most significant answer thus far.

5. Fitness Evaluation: After each step, the dragonflies' fitness is calculated using the objective function. Dragonflies that move closer to the optimal solution will have higher fitness values.
6. Update Best Solution: Keep track of the best solution found so far. This solution is typically the one with the highest fitness value.
7. Chaos-based Update: The chaos-based element of this optimization comes into play during the movement and interaction process. Chaos can help introduce randomness and exploration, preventing the optimization process from getting stuck in local optima.
8. Termination: At least one convergence condition must be fulfilled before the optimization stops. Optimizing a system is the optimal solution discovered throughout the process.

RDO is designed to enhance the exploration capabilities of the basic DFO algorithm. The chaotic elements can introduce more randomness, helping to escape local optima and potentially find better solutions in complex search spaces.

3.2.1. Initialization

In Chaos-based Dragonfly Optimization, the initialization process is crucial to set up the initial population of dragonflies. These dragonflies represent potential solutions to the optimization problem. Each dragonfly is characterized by its position, which corresponds to a candidate solution, and its velocity, affecting how it explores the search space. Mathematically, this step involves defining the initial position and velocity vectors for a population of dragonflies.

Let T be the total number of dragonflies in the population. The position of the $s-th$ dragonfly can be represented as p_s in a $Y-$dimensional search space, where Y is the problem's dimensionality. The velocity of the s-th dragonfly is denoted as r_s. Initialization typically involves assigning random values to these vectors within the problem-specific search space bounds. Eq.(9) is applied for initializing the position, and Eq.(10) is applied for the velocity.

$$p_s = (p_{s1}, p_{s2}, \ldots \ldots, p_{sY}), \qquad s = 1,2, \ldots T \tag{9}$$

$$r_s = (r_{s1}, r_{s2}, \ldots \ldots, r_{sY}), \qquad s = 1,2, \ldots T \tag{10}$$

These initial positions and velocities lay the foundation for the dragonflies to start their search and optimization process in the chaotic environment, a hallmark of Chaos-based Dragonfly Optimization.

3.2.2. Objective Function

In RDO, after the initialization of the dragonfly population, the next critical step is to define the objective function that the algorithm seeks to optimize. This objective function is denoted as $g(p)$, represents the problem that needs a solution. Mathematically, it quantifies how well a candidate solution p performs in the optimization problem. The objective function is highly problem-specific and varies based on the nature of the optimization task. The goal is to find the optimal set of parameters or variables $p = (p_1, p_2, \ldots \ldots, p_Y)$ that results in the minimum or maximum value of the objective function, and it can be expressed in Eq.(11) and Eq.(12).

$$Minimize: g(p) \tag{11}$$

$$Maximize: g(p) \tag{12}$$

The RDO aims to iteratively explore and refine the dragonfly population in search of the combination of variables that optimizes the objective function, providing a solution to the underlying problem. This optimization process will continuously evaluate the objective function throughout the algorithm's execution.

3.2.3. Chaos Initialization

RDO incorporates chaos theory principles into the optimization process to introduce unpredictability and exploration. This is done through chaos initialization, which perturbs the positions and velocities of the dragonflies using a chaotic map or equation. A logistic map (commonly used as a chaotic map) is defined in Eq.(13).

$$p_{t+1} = b.p_t.(1 - p_t) \tag{13}$$

Where p_t is the current value, p_{t+1} is the next value, and b is the control parameter determining the chaotic behaviour. The Logistic Map is just one example, and various chaotic maps or equations can be employed depending on the specific requirements of the optimization problem.

In RDO, these chaotic maps are used to perturb the positions and velocities of dragonflies. This perturbation introduces randomness into the system, preventing the algorithm from getting stuck in local optima and promoting search space exploration. The chaotic influence can be incorporated using Eq.(14) as follows for both position (p_s) and velocity (r_s) vectors of the s-th dragonfly:

$$\begin{aligned} &Perturbed\ Position: \ p_s = p_s + \delta.u_s \\ &Perturbed\ Velocity: \ r_s = r_s + \delta.u_s \end{aligned} \tag{14}$$

Where δ and γ are scaling factors, and u_s represents the chaotic perturbation generated from the chaotic map or equation.

Chaos initialization plays a crucial role in diversifying the dragonfly population's search space exploration, enhancing the algorithm's chances of finding global optima and robust solutions. Algorithm 4 provides the overall pseudocode of the initialization process via Chaos.

```
1.  def initialize_dragonflies():
2.  for each dragonfly s:
3.  p_s := generate_random_position()
4.  r_s := assign_initial_velocity()
5.  def initialize_best_solution():
6.  p_best := set_random_initial_solution()
7.  fitness_p_best := calculate_fitness(p_best)
8.  for iteration in range(1, max_iterations + 1):
9.  chaotic_perturbation_vectors :=
    apply_chaos_initialization()
10. update_positions_and_velocities(chaotic_perturbation_ve
    ctors)
11. evaluate_fitness()
12. update_best_solution()
13. optimization result.
14. return p_best, fitness_p_best
```

Algorithm 4 Chaos Initialization

3.2.4. Movement and Swarming

In RDO, the movement and swarming phase is where the dragonflies dynamically adjust their positions and velocities to explore the search space and find optimal solutions. This step is crucial for the convergence of the optimization process. Dragonflies move intending to approach the best solution found so far and, when appropriate, interact with one another to exchange information. The movement of a dragonfly can be described mathematically in Eq.(15).

$$New\ Position: p_s = p_s + r_s \qquad (15)$$

Where p_s represents the new position of the s-th dragonfly and r_s is the velocity vector of that dragonfly. This equation indicates that each dragonfly moves in the direction specified by its velocity.

Dragonflies can also be influenced by the best solution found so far. Based on this information, they may adjust their velocity to approach the optimal solution. This can be represented as Eq.(16).

$$Update\ Velocity: r_s = r_s + \theta.(p_{best} - p_s) \qquad (16)$$

Where θ is a learning or step size parameter, and p_{best} is the position of the currently best solution.

The swarming aspect involves interactions among dragonflies to share information or coordinate movements. This can be implemented through social interactions, such as attraction or repulsion forces, depending on the optimization problem's characteristics. The movement and swarming step enable dragonflies to adapt their positions and velocities to explore the search space efficiently, guided by both their movements and the global information represented by the best solution found thus far. Algorithm 5 provides the overall pseudocode of Movement and Swarming.

```
1. for each dragonfly s:
2. update_position(s)
3. adjust_velocity_towards_best_solution(s)
4. incorporate_social_interactions(s)
5. swarming step.
6. update_population()
```

Algorithm 5 Movement and Swarming

The Movement and Swarming step is a core component of RDO, allowing dragonflies to adapt their positions and velocities based on their movements and the global information represented by the best solution. It helps the dragonflies explore the search space efficiently and converge toward optimal solutions.

3.2.5. Fitness Evaluation

The fitness evaluation step in RDO is a critical process after each movement and swarming iteration. During this phase, the quality of each dragonfly's solution is assessed based on the objective function that the algorithm aims to optimize. The objective function, denoted as $g(p)$, quantifies how well a candidate solution p performs in the given optimization problem. The fitness of the s-th dragonfly (i.e., $Fitness_s$) is determined by evaluating its solution using the objective function expressed in Eq.(17).

$$Fitness_s = g(p_s) \qquad (17)$$

The objective function value measures how close a dragonfly's current position is to the optimal solution. Dragonflies that move closer to the optimal solution will yield higher fitness values. Therefore, in the fitness evaluation step, each dragonfly's position is assessed for its suitability as a potential solution to the optimization problem. The fitness values serve as the basis for the algorithm to identify and track the best solution (P_{best}) found so far. Dragonflies with higher fitness values contribute to the refinement of the global solution, and this information guides the optimization process toward convergence. The fitness evaluation step plays a crucial role in the decision-making process of RDO, determining the quality of candidate solutions and directing the algorithm towards better solutions throughout its execution. Algorithm 6 provides the overall pseudocode of Fitness Evaluation.

```
1. initialize_empty_fitness_array()
2. for each dragonfly s:
3. position_s := extract_dragonfly_position(s)
4. fitness_s := evaluate_fitness(position_s)
5. append_fitness_value(fitness_s)
6. return Fitness
```

Algorithm 6 Fitness Evaluation

3.2.6. Update Best Solution

In the RDO algorithm, updating the best solution represents a fundamental aspect of the optimization strategy. This step involves continuously tracking and maintaining the best solution found throughout the optimization process. The best solution is typically the one with the highest fitness value, corresponding to the optimal solution discovered so far. The best solution (P_{best}) it is updated based on the fitness values of the dragonflies in the population. It can be represented as Eq.(18).

$$\text{If } Fitness_s > g(P_{best}), \text{ then set } P_{best} = p_s \qquad (18)$$

Where $Fitness_s$ represents the fitness value of the s-th dragonfly, and $g(P_{best})$ denotes the fitness value of the current best solution.

Eq.(18) checks whether a dragonfly's solution's fitness is higher than the current best solution's. If it is, the best solution is updated to the s-th dragonfly's position, capturing the most promising solution found during the optimization process. Updating the best solution ensures that the RDO converges toward the optimal solution in the search space. As the dragonflies explore and adapt, the best solution evolves, guiding the algorithm toward better solutions and ultimately contributing to the successful optimization of the objective function. Algorithm 7 provides the overall pseudocode of Update Best Solution.

```
1. P_best := P_1
2. g_best := g_1
3. for each dragonfly s from 2 to T:
4. if g_s > g_best:
5. g_best := g_s
6. P_best := extract_dragonfly_position(s)
7. return P_best
```

Algorithm 7 Update Best Solution

3.2.7. Chaos-Based Update

The chaos-based update introduces randomness by perturbing the positions and velocities of dragonflies. This perturbation is accomplished using chaotic maps or equations, typically applied after the regular movement and swarming steps. Chaos helps inject a level of disorder into the system, enabling the dragonflies to escape local optima and explore uncharted regions of the search space. The chaos-based update can be represented mathematically as Eq.(19)

$$\begin{aligned} Perturbed\ Position&: p_s = p_s + \delta . u_s \\ Perturbed\ Velocity&: r_s = r_s + \gamma . u_s \end{aligned} \qquad (19)$$

Where δ and γ are scaling factors, and u_s is the chaotic perturbation generated from a chosen chaotic map or equation. Algorithm 8 provides the overall pseudocode of Chaos-based Update.

```
1. initialize_scaling_factors()
2. for each dragonfly s:
3. p_s := retrieve_position(s)
4. r_s := retrieve_velocity(s)
5. u_s := retrieve_chaotic_perturbation(s)
6. update_position_and_velocity(s, p_s + δ * u_s, r_s + γ * u_s)
7. return Updated_Population(P), Updated_Velocities(R)
```

Algorithm 8 Chaos-Based Update

Algorithm 8 introduces chaos-based perturbations to the positions and velocities of dragonflies within the population. The chaotic perturbations are generated using chaotic maps or equations, and their impact is controlled by the scaling factors β and γ. The chaos-based update introduces randomness into the system, enhancing exploration and preventing the algorithm from being trapped in local optima.

3.2.8. Termination

The termination step serves as the final stage of the optimization process, determining when and how the algorithm concludes its search for the optimal solution. The termination criteria might vary, but usually, the optimization process continues for a preset amount of iterations or until a convergence requirement is fulfilled.

Mathematically, the termination condition can be expressed as Eq.(20) and Eq.(21). The termination is attained, and either Eq.(20) or Eq.(21) is achieved.

$$iteration < max_{iteration} \quad (20)$$

$$iteration < convergence_{criteria} \quad (21)$$

Where "$iteration$" represents the current iteration number, "$max_{iteration}$" is the maximum allowable number of iterations, and "$convergence_{criteria}$" is a boolean condition indicating whether the algorithm has met specific convergence criteria. The algorithm proceeds as long as the termination condition evaluates to true. Algorithm 9 provides the overall pseudocode of the Termination part.

```
1.  iteration := 1
2.  while iteration <= max_iterations and not convergence_criterion_met:
3.  Chaos-based Update).
4.  initialize()
5.  move()
6.  evaluate_fitness()
7.  update_best_solution()
8.  chaos_based_update()
9.  iteration += 1
10. if iteration <= max_iterations and not convergence_criterion_met:
11. else:
12. break
13. return P_best
```

Algorithm 9 Termination

This algorithm controls the optimization process by monitoring the number of iterations and the satisfaction of the convergence criterion. It iteratively performs the optimization steps until the maximum number of iterations is reached or the convergence criterion is met. The best solution, P_{best}, is returned as the output, representing the algorithm's optimal solution within the defined constraints.

3.3. Fusion of RDO and ENB

The fusion of Robust Dragonfly Optimization (RDO) and Enhanced Naive Bayes (ENB) offers a powerful synergy between bio-inspired optimization and probabilistic classification. By leveraging RDO's feature selection and parameter optimization capabilities, ENB's performance is significantly enhanced. RDO efficiently identifies the most relevant features and fine-tunes ENB's parameters, resulting in a robust and accurate classification model. This fusion ensures ENB operates optimally on training and real-world data. It is a valuable tool for various applications, from data mining to machine learning tasks, where precise classification is crucial. Algorithm 10 provides the overall pseudocode of RDO-ENB.

```
1.  initialize_population()
2.  for each dragonfly:
3.  evaluate_fitness(dragonfly)
4.  for iteration in range(1, max_iterations + 1) or until convergence:
5.  update_positions_with_RDO()
6.  for each dragonfly:
7.  evaluate_fitness(dragonfly)
8.  best_position := select_best_position()
9.  trained_ENB_classifier := train_ENB_classifier(best_position)
10. if validation_dataset_available:
11. validate_ENB_classifier(trained_ENB_classifier, validation_dataset)
12. optimized_ENB_classifier := trained_ENB_classifier
```

Algorithm 10 RDO-ENB

In RDO-ENB, the RDO optimizes the feature selection process and parameter tuning for ENB, ensuring the classifier is fine-tuned for the specific dataset. This approach can lead to improved classification accuracy and generalization capabilities.

4. ABOUT DATASET

The NSK-KDD Dataset is a pivotal resource in network security, renowned for its extensive data collection. This dataset comprises a staggering 5,209,458 records. It plays a central role in developing, testing, and assessing intrusion detection systems (IDS) and other network security solutions. However, network traffic data often suffers from issues related to data duplication, which can compromise the precision and reliability of analysis. Researchers focus on the 1,152,281 unique records within the NSK-KDD dataset to address this.

These unique entries are meticulously curated to ensure they capture distinct network activities without redundancy. These records are foundational in developing IDS and are vital for training machine learning models to differentiate between regular network traffic and various network attacks. Network security experts and researchers rely heavily on these unique records for their precision and reliability in developing robust intrusion detection systems. Table 1 lists the Dataset’s Feature Information. This research study considers the distinct records and aims to accurately identify actual instances of intrusion.

5. PERFORMANCE METRICS

5.1. Precision

Precision assesses the relevance of retrieved items. It calculates the ratio of relevant items retrieved to the total number of retrieved items, helping to evaluate how well a system finds pertinent content. Eq.(22) provides the computation of precision.

$$Precision = \frac{Number\ of\ relevant\ items\ retrieved}{Total\ number\ of\ items\ retrieved} \quad (22)$$

5.2. Recall

Recall measures the ability of the system to correctly identify intrusions or attacks out of all actual intrusions. It quantifies the proportion of true intrusions detected by the IDS over the total actual intrusions that occurred. High recall in IDS is crucial as it ensures that a significant portion of actual attacks is detected, minimizing the chances of false negatives (i.e., missed intrusions). Eq.(23) provides the computation of recall.

$$Recall = \frac{Correctly\ Detected\ Intrusions}{Correctly\ Detected\ Intrusions + Missed\ Intrusio} \quad (23)$$

5.3. Classification Accuracy

Classification Accuracy is the proportion of correctly detected intrusions out of the total number of intrusions in a dataset. It quantifies how often the model's predictions match the actual class labels. Eq.(24) provides the computation of classification accuracy.

$$Classification\ Accuracy = \frac{Number\ of\ correctly\ detected\ intrusions}{Total\ number\ of\ intrusions} \quad (24)$$

5.4. F-Measure

In IDS, the F-measure balances precision and recall, ensuring the system correctly identifies intrusions while minimizing false alarms. It is calculated as the harmonic mean of precision and recall. Eq.(25) provides the computation of F-Measure.

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (25)$$

5.5. Matthews Correlation Coefficient

In IDS, the Matthews Correlation Coefficient (MCC) is used to assess the overall classification quality, accounting for both true and false positives and negatives. It considers the balance between intrusions and non-intrusions, making it a valuable metric for evaluating the effectiveness of an IDS. Eq.(26) provides the computation of MCC.

$$MCC = \frac{TP \times TN}{\sqrt{(TP + FP)(TP + FN))(TN + FP)(TN + FN)}} \times 100 \quad (26)$$

5.6. Fowlkes-Mallows Index

In IDS, the Fowlkes-Mallows Index (FMI) can be viewed as a metric that evaluates the similarity between the clusters generated by the IDS and the actual distribution of intrusions. It provides insights into how well the IDS identifies patterns and groups intrusions together. Eq.(27) provides the computation of FMI.

$$FMI = \frac{TP}{\sqrt{(TP + FP)(FP + FN)}} \times 100 \quad (27)$$

6. RESULTS AND DISCUSSION

6.1. Precision and Recall Analysis

Figure 1 serves as a visual representation of the precision and recall metrics, which are pivotal for assessing IDS performance. The result values of Figure 1 are tabulated in Table 2.

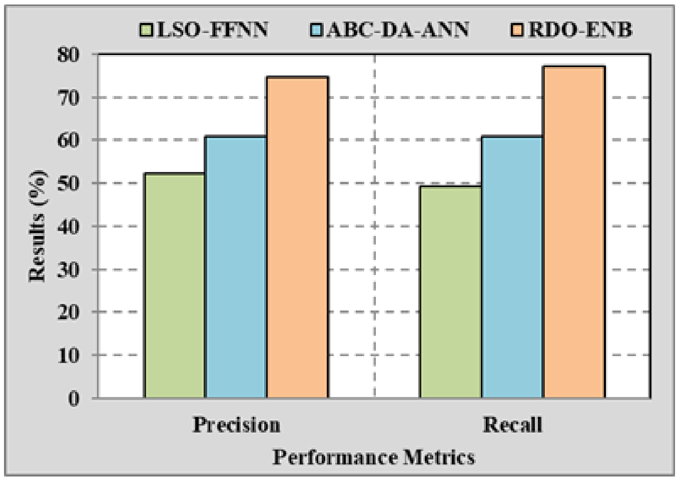

Figure 1 Precision and Recall Analysis

LSO-FFNN optimizes a feed-forward neural network using LSO. This algorithm dynamically adjusts the neural network's weights and biases to minimize the classification error, ultimately improving its ability to distinguish between normal and malicious network activities. LSO-FFNN achieved a Precision of 52.257% and a Recall of 49.247%, suggesting that it maintains a relatively balanced trade-off between precision and recall.

ABC-DA-ANN integrates the ABC algorithm and the DA with an ANN. These optimization techniques adaptively update the neural network's parameters to improve its intrusion detection performance. ABC-DA-ANN achieved a

Precision of 60.955% and a Recall of 60.795%, indicating a well-balanced performance. This algorithm effectively classifies intrusions while maintaining a relatively low false positive rate.

RDO-ENB combines the Dragonfly Optimization technique with a Naive Bayes classifier, enhancing the Naive Bayes model's performance in intrusion detection. The Dragonfly Optimization algorithm iteratively adapts the Naive Bayes classifier's parameters. RDO-ENB performed exceptionally well, with a Precision of 74.615% and a Recall of 77.198%. This indicates that the algorithm accurately classifies intrusions while maintaining high precision. It effectively captures a significant portion of actual intrusions, making it an attractive choice for intrusion detection applications where precision and recall are crucial.

Table 2 Precision and Recall Values

Classification Algorithms	Precision	Recall
LSO-FFNN	52.257	49.247
ABC-DA-ANN	60.955	60.795
RDO-ENB	74.615	77.198

These algorithms leverage different optimization techniques to fine-tune the parameters of their underlying machine-learning models. The result values from Table 2 reflect their varying performance in terms of precision and recall. LSO-FFNN exhibits a balanced performance with room for improvement. ABC-DA-ANN balances precision and recall, and RDO-ENB outperforms others by achieving high precision and recall. These algorithms contribute to intrusion detection by offering diverse approaches to enhancing the accuracy and efficiency of the systems.

6.2. Fowlkes-Mallows Index and Matthews Correlation Coefficient Analysis

Figure 2 offers a comprehensive view of the performance of various classification algorithms in intrusion detection. The FMI and MCC are key evaluation metrics to assess these algorithms' efficacy. Both metrics are crucial for measuring the algorithms' abilities to correctly classify intrusions while considering the balance between true and false positives.

LSO-FFNN combines locust swarm optimization with a feed-forward neural network to optimize the network's parameters for intrusion detection. The FMI and MCC values for LSO-FFNN are represented in Table 3. The working mechanism involves iteratively adjusting the neural network's weights and biases through locust swarm optimization to minimize classification errors. These adjustments enhance the network's ability to distinguish between normal and malicious network activities. The results show that LSO-FFNN has achieved an FMI of 50.730% and an MCC of 3.527%. While it demonstrates moderate performance, there is room for improvement in MCC, which indicates the algorithm's potential for enhancing its ability to capture the true positive rate while minimizing false positives.

ABC-DA-ANN integrates the ABC algorithm and the DA with an ANN. This fusion enhances the neural network's performance for intrusion detection. The FMI and MCC values for ABC-DA-ANN are provided in Table 3. The working mechanism involves adapting the neural network's parameters using ABC and DA to optimize its ability to classify intrusions. The results indicate that ABC-DA-ANN achieves an FMI of 60.875% and an MCC of 24.416%. These values signify a substantial improvement compared to LSO-FFNN, indicating its potential to effectively capture true positives while maintaining a reasonable balance with false positives.

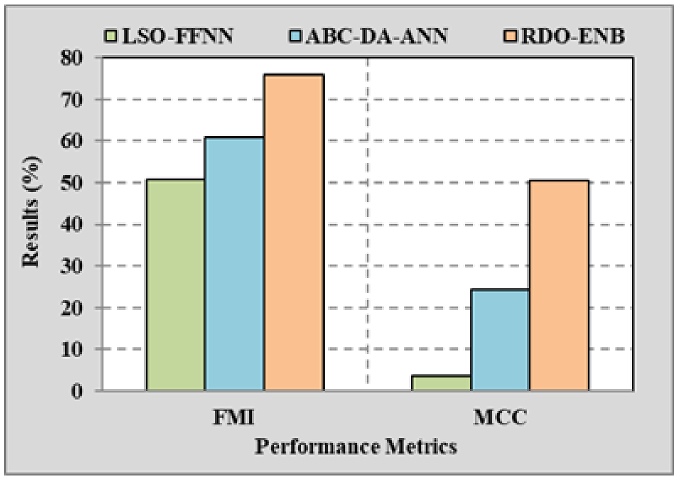

Figure 2 FMI and MCC Analysis

Table 3 Result Values of FMI and MCC

Classification Algorithms	FMI	MCC
LSO-FFNN	50.730	3.527
ABC-DA-ANN	60.875	24.416
RDO-ENB	75.896	50.625

RDO-ENB combines the Dragonfly Optimization technique with a Naive Bayes classifier to improve intrusion detection. The FMI and MCC values for RDO-ENB are shown in Table 3. The working mechanism involves iteratively adapting the parameters of the Naive Bayes model using the Dragonfly Optimization algorithm. This enhances the Naive Bayes classifier's performance for intrusion detection. The results demonstrate that RDO-ENB excels, achieving an FMI of 75.896% and an MCC of 50.625%. These results signify its exceptional performance in effectively classifying intrusions with high MCC, making it a strong contender for intrusion detection applications demanding high precision and recall. Figure 2 provides a detailed visual representation of the performance of these classification algorithms, utilizing the results from Table 3. Each algorithm contributes to the field of intrusion detection through its unique working mechanism, resulting in varying levels of FMI and MCC. While LSO-FFNN exhibits moderate performance, ABC-DA-ANN shows a significant improvement, and RDO-ENB outperforms others by achieving high FMI and MCC values. These algorithms are pivotal in advancing the accuracy and efficiency of intrusion detection systems.

6.3. Classification Accuracy and F-Measure Analysis

The visual representation in Figure 3 unfolds classification accuracy and F-Measure analysis for a diverse array of intrusion detection algorithms. LSO-FFNN combines locust swarm optimization with a feed-forward neural network to optimize parameters for intrusion detection. It iteratively adjusts the neural network's weights and biases to minimize classification errors, enhancing its ability to distinguish between normal and malicious activities. LSO-FFNN achieved a classification accuracy of 51.741% and an F-measure of 50.707%. These results indicate moderate performance, with potential for improvement in accuracy and F-Measure.

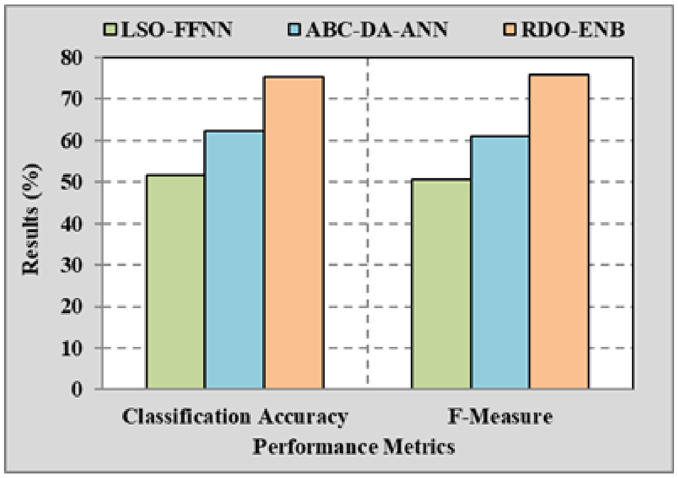

Figure 3 Classification Accuracy and F-Measure Analysis

ABC-DA-ANN integrates the ABC and DA with an artificial neural network, optimizing parameters for intrusion detection. It adaptively updates the neural network's parameters using ABC and DA, enhancing intrusion detection. ABC-DA-ANN achieved a classification accuracy of 62.255% and an F-Measure of 60.875%, representing a significant improvement compared to LSO-FFNN, with effective intrusion classification and balanced F-Measure. RDO-ENB combines the Dragonfly Optimization technique with a Naive Bayes classifier, enhancing intrusion detection. It iteratively adapts the parameters of the Naive Bayes model using the Dragonfly Optimization algorithm, leading to superior performance. RDO-ENB excels with a classification accuracy of 75.304% and an F-Measure of 75.885%, indicating exceptional accuracy and efficient intrusion classification with a high F-Measure. Figure 3 visually represents the performance of these classification algorithms using the results from Table 4. Each algorithm contributes to the field of intrusion detection through its unique working mechanism, resulting in varying classification accuracy and F-measure levels. While LSO-FFNN shows moderate performance, ABC-DA-ANN exhibits significant improvement, and RDO-ENB outperforms others with high classification accuracy and F-measure values. These algorithms play a crucial role in advancing the accuracy and efficiency of intrusion detection systems.

Table 4 Result Values of FMI and MCC

Classification Algorithms	Classification Accuracy	F-Measure
LSO-FFNN	51.741	50.707
ABC-DA-ANN	62.255	60.875
RDO-ENB	75.304	75.885

7. CONCLUSION

The intricate nature of MANETs demands innovative solutions to address their unique security challenges. While vital, the constant monitoring required by IDS in MANETs can significantly strain network resources, particularly energy. The research introduced an energy-efficient IDS, Robust Dragonfly-Optimized Naive Bayes (RDO-ENB), combining the simplicity of the Enhanced Naive Bayes algorithm with the adaptive capabilities of robust Dragonfly Optimization. RDO-ENB optimizes intrusion detection accuracy and minimizes false positives, which is crucial for the dynamic MANET environment. The successful utilization of the NSL-KDD dataset and the impressive results emphasize RDO-ENB's potential to enhance security while conserving valuable energy resources. This research contributes to the resilience and sustainability of MANETs in the face of evolving security threats. The continuous adaptation of RDO-ENB's parameters ensures its efficiency in real-time intrusion detection.

REFERENCES

[1] B. J. Chang, Y. H. Liang, and Y. M. Lin, “Distributed route repair for increasing reliability and reducing control overhead for multicasting in wireless MANET,” Inf. Sci. (Ny)., vol. 179, no. 11, pp. 1705–1723, 2009, doi: 10.1016/j.ins.2009.01.013.

[2] Z. A. Younis, A. M. Abdulazeez, S. S. R. M. Zeebaree, R. R. Zebari, and D. Q. Zeebaree, “Mobile Ad Hoc Network in Disaster Area Network Scenario; A Review on Routing Protocols,” Int. J. online Biomed. Eng., vol. 17, no. 3, pp. 49–75, 2021, doi: 10.3991/ijoe.v17i03.16039.

[3] M. U. Farooq and M. Zeeshan, "Connected dominating set enabled on-demand routing (CDS-OR) for wireless mesh networks," IEEE Wirel. Commun. Lett., vol. 10, no. 11, pp. 2393–2397, 2021, doi: 10.1109/LWC.2021.3101476.

[4] R. J. Shimonski, W. Schmied, T. W. Shinder, V. Chang, D. Simonis, and D. Imperatore, "DMZ Router and Switch Security," in Building DMZs For Enterprise Networks, R. J. Shimonski, W. Schmied, T. W. Shinder, V. Chang, D. Simonis, and D. B. T.-B. Dmz. F. E. N. Imperatore, Eds., Rockland: Syngress, 2003, pp. 369–430. doi: 10.1016/b978-193183688-3/50012-2.

[5] U. Srilakshmi, S. A. Alghamdi, V. A. Vuyyuru, N. Veeraiah, and Y. Alotaibi, "A Secure Optimization Routing Algorithm for Mobile Ad Hoc Networks," IEEE Access, vol. 10, pp. 14260–14269, 2022, doi: 10.1109/ACCESS.2022.3144679.

[6] I. Martins, J. S. Resende, P. R. Sousa, S. Silva, L. Antunes, and J. Gama, "Host-based IDS: A review and open issues of an anomaly detection system in IoT," Futur. Gener. Comput. Syst., vol. 133, pp. 95–113, 2022, doi: 10.1016/j.future.2022.03.001.

[7] S. N. G. Aryavalli and H. Kumar, "Top 12 layer-wise security challenges and a secure architectural solution for Internet of Things," Comput. Electr. Eng., vol. 105, p. 108487, 2023, doi: 10.1016/j.compeleceng.2022.108487.

[8] A. Chourasia and A. Namdev, "Improved wireless mobile ad hoc network using security schemes against black-hole attack," Int. J. Emerg. Technol. Adv. Eng., vol. 10, no. 11, pp. 61–69, 2020, doi: 10.46338/ijetae1120_07.

[9] S. Sargunavathi and J. Martin Leo Manickam, "Enhanced trust based encroachment discovery system for Mobile Ad-hoc networks," Cluster Comput., vol. 22, pp. 4837–4847, 2019, doi: 10.1007/s10586-018-2405-7.

[10] S. Gopinath, N. A. Natraj, D. Bhanu, and N. Sureshkumar, "Reliability integrated intrusion detection system for isolating black hole attack in MANET," J. Sci. Ind. Res. (India)., vol. 79, no. 10, pp. 905–908, 2020, doi: 10.56042/jsir.v79i10.43535.

[11] N. Rajendran, P. K. Jawahar, and R. Priyadarshini, "Makespan of routing and security in Cross Centric Intrusion Detection System (CCIDS) over black hole attacks and rushing attacks in MANET," Int. J. Intell. Unmanned Syst., vol. 7, no. 4, pp. 162–176, 2019, doi: 10.1108/IJIUS-03-2019-0021.

[12] M. Prasad, S. Tripathi, and K. Dahal, "An intelligent intrusion detection and performance reliability evaluation mechanism in mobile ad-hoc networks," Eng. Appl. Artif. Intell., vol. 119, 2023, doi: 10.1016/j.engappai.2022.105760.

[13] E. A. Shams, A. Rizaner, and A. H. Ulusoy, "Flow-based intrusion detection system in Vehicular Ad hoc Network using context-aware feature extraction," Veh. Commun., vol. 41, p. 100585, 2023, doi: 10.1016/j.vehcom.2023.100585.

[14] R. P. P and S. shankar, "Secure intrusion detection system routing protocol for mobile ad-hoc network," Glob. Transitions Proc., vol. 3, no. 2, pp. 399–411, 2022, doi: 10.1016/j.gltp.2021.10.003.

[15] S. B. Ninu, "An intrusion detection system using Exponential Henry Gas Solubility Optimization based Deep Neuro Fuzzy Network in MANET," Eng. Appl. Artif. Intell., vol. 123, p. 105969, 2023, doi: 10.1016/j.engappai.2023.105969.

[16] M. kumar Pulligilla and C. Vanmathi, "An authentication approach in SDN-VANET architecture with Rider-Sea Lion optimized neural network for intrusion detection," Internet of Things (Netherlands), vol. 22, p. 100723, 2023, doi: 10.1016/j.iot.2023.100723.

[17] N. Omer, A. H. Samak, A. I. Taloba, and R. M. Abd El-Aziz, "A novel optimized probabilistic neural network approach for intrusion detection and categorization," Alexandria Eng. J., vol. 72, pp. 351–361, 2023, doi: 10.1016/j.aej.2023.03.093.

[18] A. Mabrouk and A. Naja, "Intrusion detection game for ubiquitous security in vehicular networks: A signaling game based approach," Comput. Networks, vol. 225, p. 109649, 2023, doi: 10.1016/j.comnet.2023.109649.

[19] J. L. Webber et al., "An efficient intrusion detection framework for mitigating blackhole and sinkhole attacks in healthcare wireless sensor networks," Comput. Electr. Eng., vol. 111, p. 108964, 2023, doi: 10.1016/j.compeleceng.2023.108964.

[20] S. Ullah et al., "TNN-IDS: Transformer neural network-based intrusion detection system for MQTT-enabled IoT Networks," Comput. Networks, vol. 237, p. 110072, 2023, doi: 10.1016/j.comnet.2023.110072.

[21] M. V. B. M. K. M, C. A. Ananth, and N. Krishnaraj, "Detection of intrusions in clustered vehicle networks using invasive weed optimization using a deep wavelet neural networks," Meas. Sensors, vol. 28, p. 100807, 2023, doi: 10.1016/j.measen.2023.100807.

[22] M. V. Rajesh, "Intensive analysis of intrusion detection methodology over Mobile Adhoc Network using machine learning strategies," Mater. Today Proc., vol. 51, pp. 156–160, 2021, doi: 10.1016/j.matpr.2021.05.066.

[23] F. S. Al-Anzi, "Design and analysis of intrusion detection systems for wireless mesh networks," Digit. Commun. Networks, vol. 8, no. 6, pp. 1068–1076, 2022, doi: 10.1016/j.dcan.2022.05.013.

[24] J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in Lecture Notes in Electrical Engineering, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[25] R. Jaganathan, V. Ramasamy, L. Mani, and N. Balakrishnan, "Diligence Eagle Optimization Protocol for Secure Routing (DEOPSR) in Cloud-Based Wireless Sensor Network," 2022, doi: 10.21203/rs.3.rs-1759040/v1.

[26] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," ACM Int. Conf. Proceeding Ser., pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[27] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," Neural Comput. Appl., vol. 27, no. 4, pp. 1053–1073, 2016, doi: 10.1007/s00521-015-1920-1.

Authors

Mr. M. Sasikumar, M.Sc., B.Ed.,M.Phil.,SET., working as Assistant Professor & Head in the Department of Computer Science and Computer Applications, Lakshmi Bangaru Arts and Science College, Melmaruvathur,India and also doing Part Time Research Scholar, School of Computing Science,Vels Institute of Science, Technology & Advanced Studies (VISTAS),Pallavaram, Chennai – 600 117,India.. He obtained his M.Phil Degree from Bharathidasan University and also Cleared State Level Eligibility Test,Annai Therasa University, Kodaikanal,Tamilnadu.India. He is Supervised 52 M.Phil students in Various Universities. He is published 10 Books in UG and PG Degree such as Visual Programming, Digital Logic Fundamentals, Enterprise Resource Planning, Mobile Application Development. He has 23 years of consistent teaching experience in Computer Science stream. His area of interest includes ad-hoc networks, route optimization, decision support systems and Internet of Things.

Dr. K. Rohini, M.C.A, M.Phil, Ph.D., Professor, Department of Information Technology,School of Computing Science,Vels Institute of Science, Technology & Advanced Studies,(VISTAS), Pallavaram, Chennai – 600 117,Tamilnadu, India. She has 15 years of consistent teaching experience in Computer Science stream. As an eminent subject expert, she has vibrantly published more than 20 research papers in both national and international conferences. She has been dedicated to explore Software Engineering field through research, writing, and practical

RESEARCH ARTICLE

applications. She has contributed to various publications and conferences, aiming to bridge the gap between academic insights and real-world implications. She continues to engage herself in fostering a commitment to advancing knowledge and sharing valuable insights with the Software Engineering community.

How to cite this article:

M. Sasikumar, K. Rohini, "Expedient Intrusion Detection System in MANET Using Robust Dragonfly-Optimized Enhanced Naive Bayes (RDO-ENB)", International Journal of Computer Networks and Applications (IJCNA), 11(1), PP: 46-60, 2024, DOI: 10.22247/ijcna/2024/224435.