A Real-Time Nature-Inspired Intrusion Detection in Virtual Environments: An Artificial Bees Colony Approach Based on Cloud Model

Ayanseun S. Ayanboye[1], John E. Efiong[2], Temitope O. Ajayi[3], Rotimi A. Gbadebo[4], Bodunde O. Akinyemi[5], Emmanuel A. Olajubu[6], Ganiyu A. Aderounmu[7]

Department of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria[1, 2, 3, 4, 5]
CyberSCADA Research Lab, Department of Computer Science & Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria[6]
CyberSCADA Research Lab, Africa Center of Excellence, OAU ICT-driven Knowledge OAK Park,
Obafemi Awolowo University, Ile-Ife, Nigeria[7]

***Abstract*—Real-time intrusion detection in virtual environments is crucial for maintaining the security and integrity of modern computing infrastructures. This paper proposes a nature-inspired mathematical model designed to detect both known and unknown attacks on virtual machines, focusing on enhancing detection accuracy and minimizing false alarm rates. The proposed model, named Developed Artificial Bee Colony Optimization Based on Cloud Model (DABCO_CM), is inspired by the foraging behavior of bee swarms and integrates principles from the Artificial Bee Colony algorithm and the cloud model rooted in fuzzy logic theory. The model was simulated using the UNSW_NB15 datasets in Google Colab and benchmarked against an existing model. It achieved a detection accuracy of 97.98%, compared to the existing model's 95.35%. Sensitivity results showed 99.92% for the proposed model, compared to 96.90% for the existing model, while specificity increased to 93.86%, in contrast to the existing model's 90.71%. These findings demonstrate a 3.02% increase in sensitivity, a 2.63% increase in accuracy, and a 3.15% increase in specificity, highlighting the model's superior capability in detecting attacks and its potential to learn from unlabeled data, addressing key challenges in virtual machine security.**

Keywords*—Real-time intrusion detection; virtual environments; artificial bee colony algorithm; cloud model algorithms; intrusion detection system; feature selection; classification; swarm intelligence; fuzzy logic; DNN; ABC_DNN DABCO_CM

I. Introduction

Virtual environments, encompassing applications such as email, chat, and online document sharing, operate within shared operating systems that allow Virtual Machines (VMs) to function like physical computers without specific hardware components. Virtualization abstracts the complexity of hardware and software, typically implemented using hypervisor devices that efficiently allocate system resources among multiple VMs [1], [2]. Security in virtual environments relies on traditional tools like firewalls and intrusion detection systems (IDS), which are either network-based or host-based, though they have limitations in detecting new threats [3] [8]. Traditional IDS methods often fall short in virtual environments due to their inability to handle the dynamic and complex nature of modern cyber threats effectively [9] [11]. These systems struggle with high false alarm rates and limited accuracy in detecting new, unknown attacks. This paper addresses these challenges by proposing a bio-inspired mathematical model that integrates the Artificial Bee Colony (ABC) algorithm with fuzzy logic to enhance detection accuracy and minimize false alarms in virtualized settings. The proposed model, named Developed Artificial Bee Colony Optimization Based on Cloud Model (DABCO_CM), draws inspiration from the foraging behavior of bee swarms and leverages principles from the cloud model rooted in fuzzy logic theory [12], [15].

The DABCO_CM model improves upon previous approaches in several key ways:

- Enhanced Detection Accuracy: By optimizing feature selection and classification processes through the ABC algorithm, the model can more accurately identify relevant patterns and anomalies, leading to higher detection accuracy.
- Reduced False Alarms: The integration of cloud models helps manage uncertainty and imprecision in data analysis, significantly reducing false positive rates compared to traditional IDS.
- Real-Time Detection: The model's design principles ensure fast processing times, making it capable of real-time intrusion detection, which is crucial for maintaining the security of dynamic virtual environments.
- Scalability and Adaptability: The hybrid approach of combining swarm intelligence with fuzzy logic makes the model more robust and adaptable to evolving cyber threats, ensuring better performance in large-scale and complex virtual environments.

The subsequent sections of this paper are structured as follows:

- Literature Review: Discusses the application of Artificial Bee Colony (ABC) algorithms in detecting Distributed Denial-of-Service (DDoS) attacks within virtual environments and highlights the need for further optimization.

- Methodology: Describes the development process of the DABCO_CM model, detailing its design principles, components, and integration of ABC and cloud model algorithms.
- Simulation and Testing: Explains the simulation environment and the datasets used to evaluate the model, including performance metrics and evaluation parameters.
- Results and Discussion: Presents the simulation results, comparing the performance of the proposed model against existing models, and discusses the findings.
- Conclusion and Future Work: Summarizes the study's contributions, highlights the model's effectiveness, and suggests directions for future research.

II. Literature Review

The application of Artificial Bee Colony (ABC) algorithms in detecting DDoS attacks within virtual environments is a relatively familiar area with limited research. Despite its proven effectiveness, particularly in filtering and mitigating attacks, there is a recognized need for further research to optimize ABC algorithm convergence in large-scale virtualized settings. Studies by [6] and [15] collectively highlight the need to enhance ABC algorithms to improve their robustness and scalability in complex virtualized environments [6], [8]. Effective intrusion detection in virtual environments has been the focus of numerous studies [4][5], highlighting the need for innovative approaches to overcome the limitations of traditional IDS [14]. The study in [1] highlighted the substantial threat posed by distributed denial of service (DDoS) attacks to network security in cloud computing. The study employed a deep learning classifier to differentiate between attacked and non-attacked data. The proposed FS-WOA–DNN framework effectively protected against DDoS attacks, achieving a better detection accuracy, outperforming other existing DDoS detection models. The research in [13] addressed the impact of DoS and DDoS attacks on cloud services. Using artificial immune systems, they identified attack characteristics, achieving high detection accuracy and low false alarm rates. This method effectively maintained cloud service availability, reducing financial losses and preserving reputations. Fig. 1 shows existing model.

A. Real-Time Intrusion Detection

Real-time intrusion detection requires systems capable of processing large volumes of data quickly and accurately [2] highlight the importance of high-performance algorithms in achieving timely threat detection [7], [8]. They emphasize that traditional methods often fall short due to their inability to handle the dynamic nature of modern cyber threats effectively [9].

B. Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm, proposed by [4] [6] have shown promise in various optimization problems, including intrusion detection. Its ability to efficiently search for optimal solutions makes it suitable for enhancing IDS performance. The study in [18] further explore the application of ABC in optimizing feature selection and classification tasks, demonstrating its potential for improving IDS accuracy and efficiency.

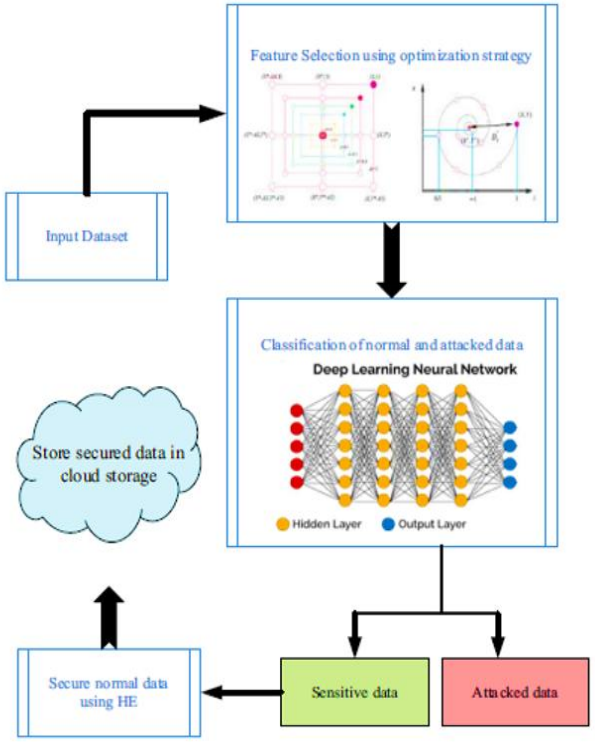

Fig. 1. The existing model (Agarwal et al. 2022).

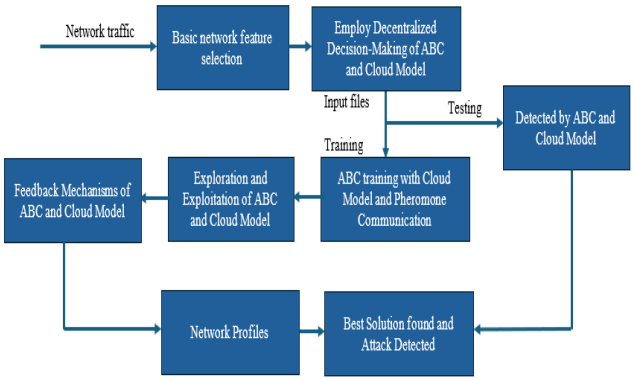

Fig. 2. Proposed model.

Fig. 2 shows proposed model and Fig. 3 depicts flowchart of the proposed model.

C. Cloud Model Algorithms

Cloud models, which integrate fuzzy logic and probability theory, offer a powerful tool for handling uncertainty and imprecision in data analysis. The studies in [6][10][14][15] discuss the application of cloud models in intrusion detection, highlighting their capability to improve detection accuracy by managing uncertainties. Their research underscores the benefits of combining cloud models with other optimization techniques to enhance IDS performance.

Africa Center of Excellence, OAU ICT-driven Knowledge OAK Park, Obafemi Awolowo University, Ile-Ife, Nigeria

D. Integrated Approaches

Integrating the ABC algorithm with cloud models has been explored to address the limitations of each technique individually. The studies in [6][16] present a hybrid approach that combines swarm intelligence with fuzzy logic to improve IDS performance. Their study demonstrates that such integration can offer a more robust and adaptable solution for real-time intrusion detection.

III. METHODOLOGY

The development of the DABCO_CM model involves several key steps to integrate the Artificial Bee Colony (ABC) algorithm with cloud model algorithms, aiming to enhance real-time intrusion detection in virtual environments:

A. Objectives and Requirements

The primary objectives for real-time intrusion detection are to achieve high accuracy, minimize false alarms, and ensure fast processing times. Requirements include the ability to handle large volumes of data generated in virtual environments and adapt to evolving cyber threats. The DABCO_CM model is designed to meet these objectives by combining advanced optimization techniques with robust data handling capabilities.

B. Design Principles

The DABCO_CM model is built on the principles of swarm intelligence and fuzzy logic. The ABC algorithm is used to optimize feature selection and classification processes, enhancing the model's ability to identify relevant patterns and anomalies. Cloud model algorithms are integrated to manage uncertainty and provide a probabilistic framework for intrusion detection.

C. Components

The DABCO_CM model consists of the following components:

- Feature Selection Module: Utilizes the ABC algorithm to select the most relevant features for intrusion detection. This module aims to improve detection accuracy and reduce computational overhead.
- Classification Module: Applies machine learning techniques to classify intrusion attempts based on selected features. This module uses advanced algorithms to enhance the accuracy of threat detection.
- Cloud Model Integration: Incorporates cloud models to handle uncertainty and improve detection accuracy. The cloud model algorithms provide a probabilistic approach to managing imprecise data and enhancing overall system performance.

D. Proposed Model Design

The proposed model integrates the Artificial Bee Colony (ABC) optimization algorithm and the Cloud Model algorithm to enhance intrusion detection in virtual environments.

1) Network traffic: Network traffic refers to the flow of data across a computer network, encompassing all the data sent and received by devices connected to the network.

2) Basic network feature selection: Feature selection involves choosing a subset of relevant features for a machine learning model using methods like filter, wrapper, and embedded techniques. This balance ensures the model's effectiveness. Fig. 4 shows diagrammatic representation of the proposed model.

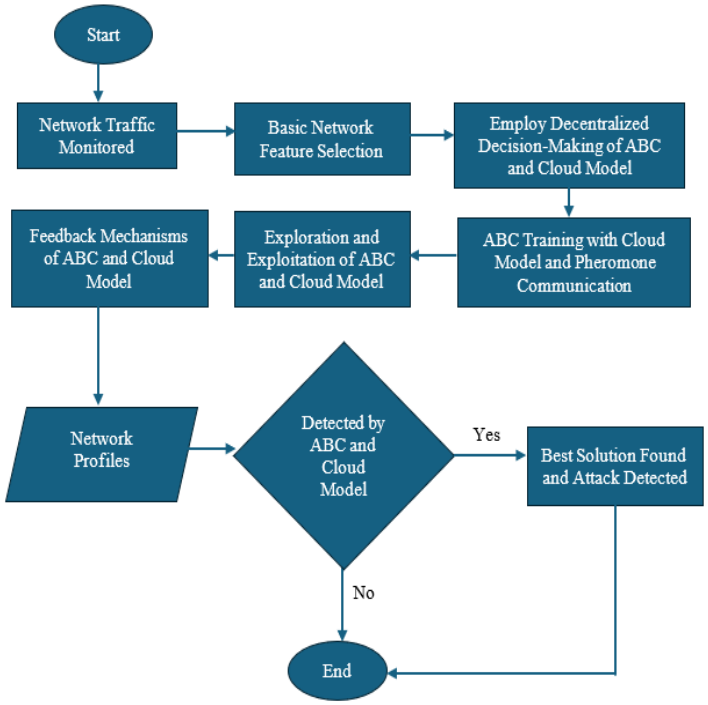

Fig. 3. Flowchart of the proposed model.

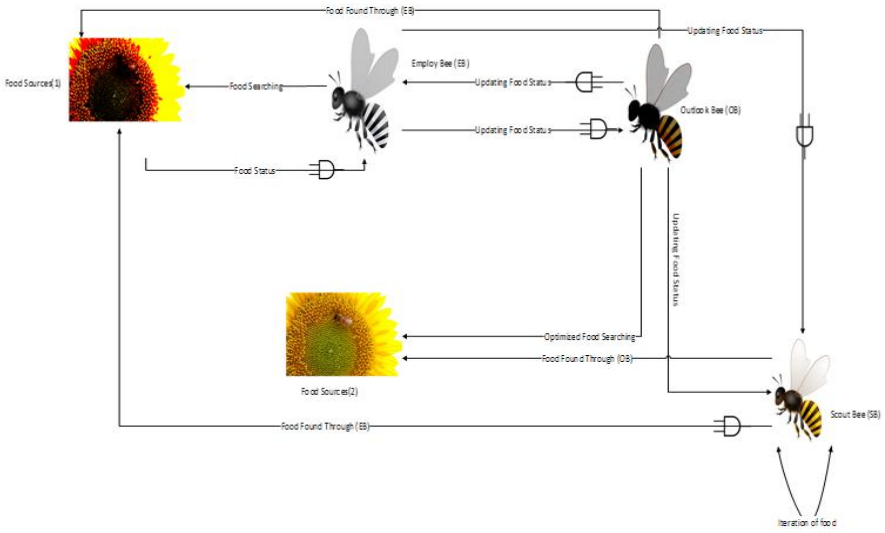

Fig. 4. Diagrammatic representation of the proposed model.

3) Decentralized decision-making with ABC and cloud model: The ABC algorithm, inspired by bee foraging, and the Cloud model, which handles uncertainty, collaboratively improve system performance by enabling autonomous and informed decision-making.

4) ABC training with cloud model and pheromone communication: Combining the ABC algorithm with the Cloud model and pheromone communication enhances decision-making and collaboration among agents, leading to better performance in virtual environments.

5) Exploration and exploitation of ABC and cloud model: The ABC algorithm optimizes solutions by mimicking bee behavior, while the Cloud model leverages distributed computing. This integration enables efficient resource allocation and problem-solving.

6) Feedback mechanisms of ABC and cloud model: ABC and Cloud models use feedback to improve decision-making and predictions. The ABC algorithm relies on bee feedback, and the Cloud model refines its functions based on various inputs.

7) Network profiles: Network profiles represent behavioral patterns of networks. By analyzing these profiles with ABC optimization and Cloud Model, intrusion detection systems can identify suspicious activities.

8) Best solution found and attack detected: Integrating ABC and Cloud models optimizes solutions and enhances system security by detecting potential attacks in virtual environments. The framework combined the ABC algorithm and the Cloud Model algorithm to create an optimized model with the following key elements:

a) Objective: The primary goal of the framework had been to develop a highly efficient intrusion detection model by integrating the ABC algorithm and the Cloud Model algorithm.

b) Feature selection: To improve the accuracy of intrusion detection, the framework had utilized the ABC algorithm to optimize the feature selection process. This involved selecting the most relevant features that significantly contributed to identifying intrusions. An effective method to achieve feature selection for improving intrusion detection accuracy is to employ machine learning techniques such as recursive feature elimination (RFE).

c) Uncertainty dandling: The Cloud Model algorithm had played a vital role in handling uncertainties associated with intrusion detection. It combined probability-based reasoning, adaptive exploration, dynamic solution adjustment, ensemble techniques, and continuous learning for robust and adaptive solutions in the face of uncertainty.

d) Integration: The ABC and Cloud Model algorithms had been seamlessly integrated into a unified framework. The ABC algorithm optimized the feature selection procedure, while the Cloud Model algorithm addressed uncertainty reasoning. ABC algorithm scouts for the most promising features, much like bees searching for the best flowers. The Cloud Model carefully examines those features to assess their uncertainty, ensuring a reliable selection. By working together, they create a robust decision-making framework that considers both feature relevance and potential uncertainties.

e) Real-Time detection and response: One of the key objectives of the DABCO_CM framework has been to enable real-time detection of intrusions. It has empowered the system to promptly identify and respond to both known and unknown intrusions occurring in the virtual environment. This capability enhanced the system's security by enabling swift and effective countermeasures against potential threats as they occur.

f) Evaluation: The effectiveness of the DABCO_CM framework had been evaluated using various metrics, such as intrusion identification, detection precision, and response time. These evaluations provided validation of the framework's performance and reliability. The DABCO_CM framework, leverage the strengths of the ABC algorithm and the Cloud Model algorithm. By optimizing feature selection, handling uncertainties, and facilitating real-time intrusion detection, this framework aimed to enhance the overall effectiveness of intrusion detection systems in virtual environments.

E. Simulation and Testing

This methodology delved into the complexities of creating a groundbreaking system that seamlessly integrated the capabilities of Artificial Bee Colony (ABC) optimization with Cloud Model-based techniques to handle imprecise data. Datasets, specifically UNSW_NB15, were used to simulate two classifiers (ANN and DNN). These classifiers were developed and saved as models in the Google Research Collaboration Environment, and relevant performance metrics were utilized. The ABC optimization for intrusion detection in a virtual environment used 100 random population initializations within bounds. It ran 10 times for a fixed number of iterations, with the dimensionality of the problem space varying. Customizing parameters was essential for effective optimization, with performance evaluated using benchmark functions and real intrusion detection data.

F. Model Formulation of the Proposed Model

The mathematical representation of the Developed Artificial Bee Colony Optimization Based on Cloud Model (DABCO_CM) involved expressing the optimization procedure using mathematical equations. This formulation combined the core concepts of the Artificial Bee Colony (ABC) algorithm and the Cloud Model algorithm to detect intrusions in a virtual environment. During the initialization phase, the population's food sources were randomly created and allocated to the employed bees as in Eq. (1), [6]:

$$X_i^j = X_{min}^j + rand(0,1)\left(X_{max}^j - X_{min}^j\right) \quad (1)$$

Fig. 5 shows Mapping of bee colony concepts to intrusion detection in virtual environments and Table 1 depicts Mapping of Bee Colony Concepts to Intrusion Detection in Virtual Environments.

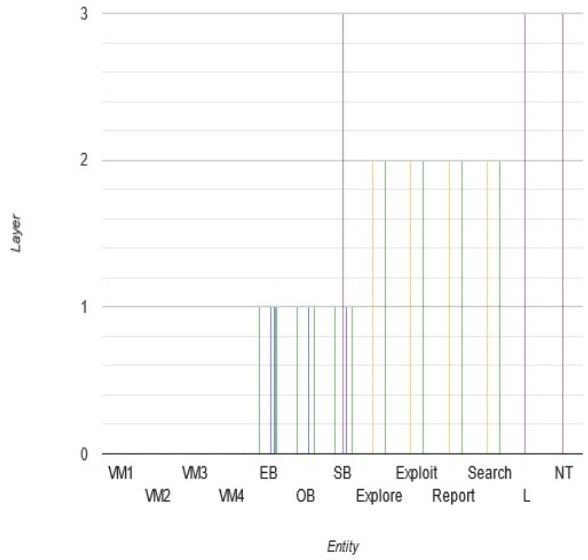

Fig. 5. Mapping of bee colony concepts to intrusion detection in virtual environments.

The upper and lower limits of the solution vectors are denoted by X_{max} and X_{min} respectively in Eq. (2).

$$V_i^j = X_i^j + \varphi_i^j\left(X_i^j - X_k^j\right) \quad (2)$$

Subsequently, the comparison of V_i to X_i is made, and the foraging bee employs a selection mechanism prioritizing the food source with higher fitness, as expressed in Eq. (3).

$$fit_i = \begin{cases} \frac{1}{1+f_1} & f_1 \geq 0 \\ 1 + abs(f_1) & f_1 < 0 \end{cases} \quad (3)$$

This chosen food source, "X_i," was then updated in the same way as the working bees, determined by its probability value "P" as defined in Eq. (4) and the given expression.

$$P = \frac{0.9 fit(X_1)}{\underset{m=1}{\overset{N_e}{max}} fit(X_m)} + 0.1 \quad (4)$$

The formulation of the cloud model could be expressed as follows in Eq. (5), [6]:

$$\forall X \in U \rightarrow \mu(X) \in \{0,1\} \quad (5)$$

Each droplet is uniquely identified by its index i, and N indicates the total number of such droplets.

$$En_i^1 = N(En, He^2) \quad (6)$$

En_i^1 represents a property associated with the cloud droplet i. It is derived as a random value from a normal distribution with mean En and variance.

$$He^2.\, x_i = N(Ex, (En_i^1)^2) \quad (7)$$

This equation defines the x -coordinate of the cloud droplet i. It is generated as a random value from a normal distribution with mean Ex and variance $(En_i^1)^2$.

$$\mu_i = \exp\{-\frac{(x_i - Ex)}{2(En_i^1)^2} \quad (8)$$

This equation defines the μ-coordinate (or y-coordinate) of the cloud droplet i. It is calculated using the exponential function of the negative of the difference between x_i and Ex, divided by $2(En_i^1)^2$.

$$X_I = Ex \pm \sqrt{-2\ln(\mu) * En_i^1} \quad (9)$$

The derivation of this equation involves solving for $\boldsymbol{X_I}$ from the expression of the exponential distribution in Eq. (8).

$$\begin{cases} Ex = \underset{i=1}{\overset{Ne}{\max}} fit_i \\ En = \frac{Ex - fit_i}{12} \\ He = \frac{En}{3} \end{cases} \quad (10)$$

Eq. (10) is derived to define parameters (Ex, En, He) in the ABC algorithm using the cloud model. The derivation emphasizes exploitation by setting Ex as the maximum fitness. En introduces variability based on fitness differences, and He contributes to overall variability. This formulation enhances exploration and exploitation for effective optimization. Table I shows Mapping of Bee Colony Concepts to Intrusion Detection in Virtual Environments.

$$\begin{cases} Ex = \min\limits_{i=1}^{Ne} fit_i \\ En = \frac{fit_i - Ex}{12} \\ He = \frac{En}{3} \end{cases} \quad (11)$$

Eq. (11) is derived to enhance the ABC algorithm using the cloud model, emphasizing less proficient individuals by choosing Ex as the minimum fitness. The formulation promotes exploration and diversification for a more comprehensive search. Therefore, Eq. (10) is designed to exploit the best solutions by focusing on the maximum fitness, improving optimization by refining the search around the most promising solutions while Eq. (11) is designed to explore the search space by focusing on the minimum fitness, encouraging diversification and a comprehensive search for potentially better solutions.

TABLE I. MAPPING OF BEE COLONY CONCEPTS TO INTRUSION DETECTION IN VIRTUAL ENVIRONMENTS

Concept (Bee Colony)	Concept (Intrusion Detection)	Description	Layer
Hierarchy of Roles	Network Layer, Application Layer, Intrusion Detection System Layer	Levels of security architecture	-
Individual Bee/Group with Defined Function	VM, Network Device, Software Application	Elements within the virtual environment	-
VM1, VM2, VM3, VM4	Specific Virtual Machines	Individual VMs within the environment	-
Employed Bee (EB)	Actively Engaged VM	VM searching for potential attack patterns	Layer 1
Onlooker Bee (OB)	Monitoring VM	VM learning from employed bees' findings	Layer 2
Scout Bee (SB)	Searching VM	VM looking for new attack patterns or vulnerabilities	Layer 3
Search	Looking for Something	Process of finding something (nectar or vulnerabilities)	Layer 1, Layer 3
Nectar (NT)	Network Traffic (NT)	Collected data	-
Location (L)	Log Files	Source of information about potential threats	Layer 2
Explore	Scanning and Analyzing	Discovering something (nectar or threats)	Layer 1, Layer 3
Report	Communicating Information	Sharing discoveries (waggle dance or alerts)	Layer 2, Layer 3

$$P = \exp\left\{= \frac{(x-Ex)^2}{2(En^1)^2}\right\} \quad (12)$$

Eq. (12) is derived by adapting the probability density function of a normal distribution to formulate the probability (P) of selecting an individual bee based on its position (x) in relation to cloud model attributes (Ex, En^1).

$$\begin{cases} Ex = X_i^j \\ En = ex \\ He = \frac{En}{10} \end{cases} \quad (13)$$

Eq. (13) in the cmABC algorithm derives $\boldsymbol{Ex}$ as the current position of food sources ($\boldsymbol{X_i^j}$), En as a dynamic variable ($\boldsymbol{ex}$), and $\boldsymbol{He}$ as one-tenth of $\boldsymbol{En}$. This formulation emphasizes the current position, adaptability, and variability in the cloud model, enhancing exploration and exploitation. In this scenario, X_i represented the current position of the food sources, where j belonged to the set {1, 2, ..., D}, and e_x was a variable.

$$\begin{cases} En^1 = En, He^2 \\ V_i^j = N(Ex, En^{i2}) \end{cases} \quad (14)$$

Eq. (14) introduces variability in the cmABC algorithm by defining En^1 as En with He^2 and V_i^j as a normal distribution with mean Ex and variance En^{i2}. This enhances exploration and adaptability, crucial for optimal solution search.

$$ex = -(E_{max} - E_{min})(t/T_{max})^2 \quad (15)$$

Eq. (15) dynamically adjusts ex in the cmABC algorithm. It uses a quadratic function $(t/T_{max})^2$ with a scaled coefficient $-(E_{max} - E_{min})$. The adjustment evolves with the iteration count for improved solution accuracy and controlled exploration.

$$\begin{cases} Ex = X_i^j \\ En = \frac{2}{3}\left|X_i^j - x_k^j\right| \\ He = \frac{En}{10} \end{cases} \quad (16)$$

Eq. (16) in the cmABC algorithm is derived to define cloud model attributes. Ex depends on food source positions in a specific dimension, En incorporates variability between positions, and He contributes proportionally to overall variability, enhancing exploration and exploitation.

G. Feature Selection

There were 42 features in both the trained and tested datasets, and all of these features were relevant in building the model. Some features needed to be removed from the datasets to avoid affecting the model's performance. To determine the relevance of each feature to the target feature, an entropy-based algorithm (mutual information gain) was used to rank the features according to their effectiveness in building the model. Features with non-zero mutual information gain [17] were initially selected to build the model, while the remaining features were discarded.

H. Model Performance Metrics Evaluation Using Parameters

The following performance metrics were used to evaluate the model:

1) Accuracy: this is the ratio of correctly classified positive and negative tuples to the total number of traffic connections. It can be expressed mathematically in Eq. (17) as

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (17)$$

where TP = True Positive: The number of positive tuples correctly classified by the predictive model,

TN= True Negative: The number of negative tuples correctly classified by the predictive model,

FP = False Positive: The number of positive tuples incorrectly classified by the predictive model and

FN = False Negative: The number of negative tuples incorrectly classified by the predictive model.

2) Sensitivity: this is the proportion of actual positives which are predicted positive. Therefore, it measures the effectiveness of the model in predicting positive classes. It is used to determine the false positive rate of the model. It is defined in Eq. (18) as

$$Sensitivity = \frac{TP}{(TP+FN)} \quad (18)$$

3) Specificity: this is the proportion of actual negatives which are predicted negative. Therefore, it measures the effectiveness of the model in predicting negative classes. It is used to determine the false negative rates of the model. It is defined in Eq. (19) as

$$Specificity = \frac{TN}{(TN+FP)} \quad (19)$$

I. Model performance metrics evaluation using graphical tools

The performance of the model was also evaluated using the following graphical tools:

1) Confusion Matrix: A confusion matrix is a matrix whose rows represents the positive and negative values predicted by the classifier and columns represents the actual positive and negative values.

2) Receiver operative curve (ROC): The receiver operative curve evaluates the false positive rates represented on the Y-axis and the true positive rates represented on the X-axis generated during the classification stage. The Area Under the Curve (AUC) shows how well the classifier performs in terms of these two variables (false positive and true positives). The closer it is to 1 the better.

IV. Results and Discussion

A. Simulation Environment

In the study, models were simulated using the Google Research Collaboration Environment. An online Python environment, Colab, facilitated the creation of texts, codes, and the execution of codes to generate output. Python machine learning tools, such as Scikit-learn, pandas, numpy, matplotlib, seaborn, hyperopt, joblib, and scikit-fuzzy libraries, were utilized for dataset analysis, model construction, saving model as pickle file, and evaluating model performance:

- Scikit-learn: Used for machine learning algorithms and model evaluation.
- pandas: Utilized for data manipulation and analysis.
- numpy: Used for numerical operations.
- matplotlib and seaborn: Applied for data visualization.
- hyperopt: Employed for hyperparameter optimization.
- joblib: Used for saving models as pickle files.
- scikit-fuzzy: Utilized for fuzzy logic and operations.

These tools were integral in constructing various models, with the highest-performing model selected for final implementation.

B. Feature Selection

There were 42 features in both the trained and tested datasets, and all of these features were relevant in building the model. Some features needed to be removed from the datasets to avoid affecting the model's performance. To determine the relevance of each feature to the target feature, an entropy-based algorithm (mutual information gain) was used to rank the features according to their effectiveness in building the model. Features with non-zero mutual information gain were initially selected to build the model, while the remaining features were discarded.

V. Simulation Result

The training dataset, which constituted 32%, and the testing dataset, making up 68% of the total dataset, were input into the Google Research Collaboration Python Environment. This environment clustered the dataset into two categories and labeled them as 0 and 1. These labeled datasets were employed to construct six models, allowing them to learn from previous data and enabling them to classify future network instances as either normal or attack instances. After successfully constructing the models, instances from the testing dataset were utilized to assess their performance. The proposed models successfully classified attack instances as true positives (TP) and accurately classified normal instances as true negatives (TN). However, the results also indicated that attack instances were misclassified as normal instances (false negatives - FN), and normal instances were misclassified as attack instances (false positives - FP). The proposed DNN model was able to classify 112, 694 instances as attacks (TP), while 50, 183 were correctly classified as normal (TN). The results also showed that 6, 647 attack instances were misclassified as normal (FN), while 5, 817 normal instances were misclassified as attacks (FP). The proposed ABC_DNN model was able to classify 117, 380 instances as attacks (TP), while 48, 120 were correctly classified as normal (TN). The results also showed that 4, 215 attack instances were misclassified as normal (FN), while 4,626 normal instances were misclassified as attacks (FP). The proposed DABCO_CM_DNN model was able to classify

119250 instances as attacks (TP), while 52, 562 were correctly classified as normal (TN). The results also showed that 91 attack instances were misclassified as normal (FN), while 3, 438 normal instances were misclassified as attacks (FP).

VI. Model Evaluation

In evaluating the performance of the models, four evaluation metrics were used: accuracy, sensitivity, specificity, and error. All these performance metrics were calculated using the results of the confusion matrix table, which were categorized as true negative, true positive, false positive, and false negative, respectively.

VII. Performance Evaluation and Benchmarking

The models developed and benchmarked in this study, including DNN, ABC_DNN, and DABCO_CM, were evaluated based on accuracy, sensitivity, specificity, and error rate. These models were compared to existing models presented in the literature, specifically those by [1] and [13], which employed techniques such as Feature Selection using Whale Optimization Algorithm (FS-WOA) and Artificial Immune Systems (AIS). The FS-WOA-DNN model presented by [1] achieved an accuracy of 95.35%, sensitivity of 96.9%, specificity of 90.71%, and an error rate of 0.0928. In comparison, Prathyusha et al. (2021) focused on ANN, SVM, and AIS models, with the AIS model showing the highest performance, achieving an accuracy of 96.56%, sensitivity of 96.4%, specificity of 91.9%, and an error rate of 0.0713. In this study, the DNN classifier was chosen and optimized using the Artificial Bee Colony (ABC) algorithm, resulting in the ABC_DNN model. The DABCO_CM model was further developed by integrating a cloud model based on fuzzy logic, enhancing the performance of the ABC_DNN model. The performance of the three proposed models demonstrated competitive results. The DNN model achieved an accuracy of 92.89%, sensitivity of 94.43%, specificity of 89.61%, and an error rate of 0.0711. The ABC_DNN model improved upon this with an accuracy of 94.38%, sensitivity of 96.53%, specificity of 91.22%, and an error rate of 0.0562. The DABCO_CM model emerged as the best performer, achieving an accuracy of 97.98%, sensitivity of 99.92%, specificity of 93.86%, and an error rate of 0.0202. These results clearly show that the DABCO_CM model surpasses the FS-WOA-DNN model in terms of accuracy, sensitivity, and specificity. The DABCO_CM model's sensitivity of 99.92% indicates its strong ability to identify both known and unknown attacks, while its specificity of 93.86% underscores its effectiveness in accurately distinguishing between attacks and benign activities. In contrast, the ABC_DNN model, while exhibiting high sensitivity at 96.53%, had a lower specificity of 91.22%, suggesting that there is still potential for improvement in differentiating benign instances more effectively. These performance metrics highlight the strengths of both DABCO_CM and ABC_DNN, where DABCO_CM achieved the highest overall performance across all metrics. The differences in dataset characteristics (UNSW-NB15 used in this study versus datasets like KDD Cup 99 and CICIDS2017 in previous studies) and simulation environments (Google Colab in this study versus MATLAB and CloudSim) contribute to variations in model performance. Despite these differences, the results clearly indicate that DABCO_CM provides superior capabilities in accurately detecting and classifying attacks, addressing critical challenges in virtual machine security. The results obtained from the confusion matrices of the proposed models, categorizing actual and predicted attacks for performance metric evaluation, are presented in Table II and performance metrics are presented in Table III.

TABLE II. Confusion Metrices Results

	TP	TN	FP	FN
DNN	112694	50183	5817	6647
ABC_DNN	117380	48120	4626	4215
DABCO_CM_DNN	119250	52562	3438	91

Fig. 6 shows DNN Model Confusion Matrix, Figure 7 shows Line Graph for DNN model, Figure 8 shows ABC_DNN model confusion matrix, Fig. 9 displays Line Graph for ABC_DNN model, Fig. 10 shows DABCO_CM_DNN model confusion matrix and Fig. 11 shows Line Graph for DABCO_CM_ DNN model. The confusion matrices of three important models and their corresponding line graphs are displayed below.

The new models were found to be impressive and superior to the existing system. The definitions of metrics and benchmarked results were displayed below.

- Accuracy: Accuracy measured the overall correctness of a classification model.

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$$

- Sensitivity (Same as Detection Rate/Recall): Sensitivity measured the proportion of actual positive instances correctly classified.

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP+}FN}$$

TABLE III. Performance Metrics

Model	Accuracy	Sensitivity	Specificity	Error Rate
Agarwa et al. (2022)				
FS-WOA-DNN	95.35%	96.9%	90.71%	0.0928
Prathyusha et al. (2021)				
AIS	96.56%	96.4%	91.9%	0.0713
Proposed Models				
DNN	92.89%	94.43%	89.61%	0.0711
ABC_DNN	96.38%	96.53%	91,22%	0.0562
DABCO_CM	97.98%	99.92%	93.86%	0.0202

- Specificity: Specificity measured the proportion of actual negative instances correctly classified.

$$\text{Specificity} = \frac{TN}{TN+FP}$$

- Error: Error is the difference between an observed or predicted value and the true or expected value.

Error (residual) = Observed Value - Predicted Value.

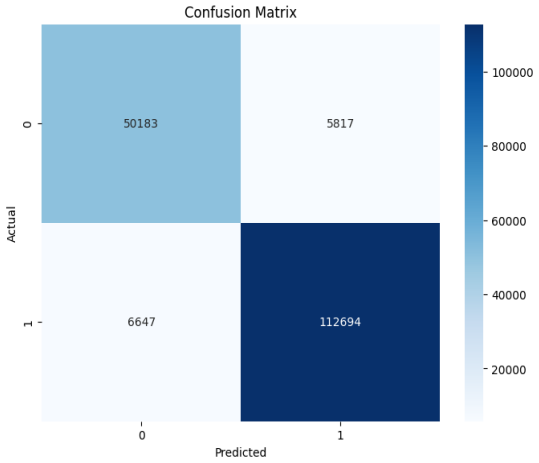

Fig. 6. DNN model confusion matrix.

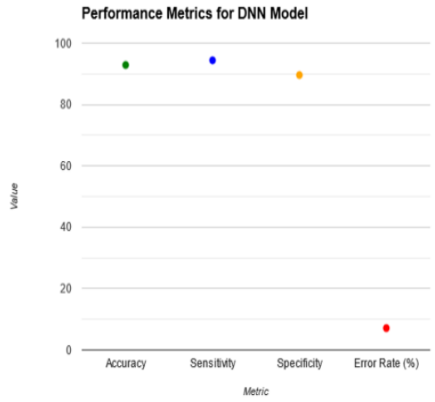

Fig. 7. Line graph for DNN model.

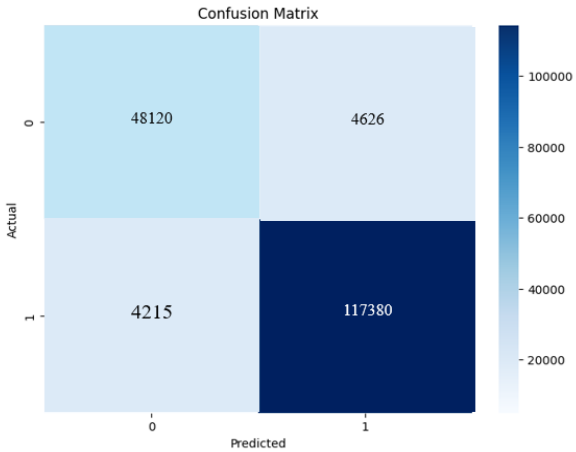

Fig. 8. ABC_DNN model confusion matrix.

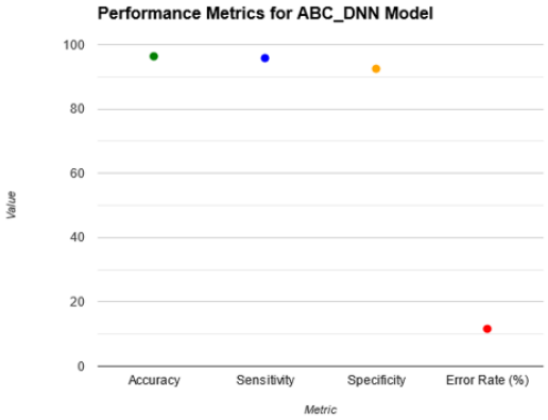

Fig. 9. Line Graph for ABC_DNN model.

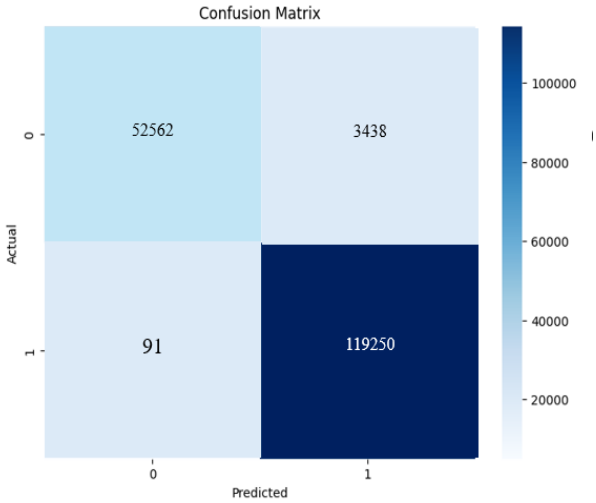

Fig. 10. DABCO_CM_DNN model confusion matrix.

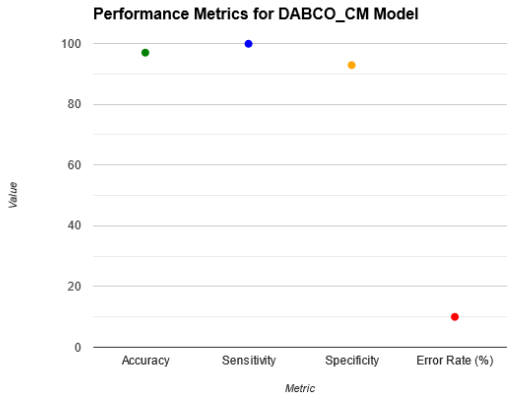

Fig. 11. Line graph for DABCO_CM_DNN model.

VIII. Conclusion and Future Work

This study introduces a pioneering Intrusion Detection System (IDS) to tackle growing security challenges in virtualized computing. The proposed system integrates the Artificial Bee Colony (ABC) optimization algorithm with a

fuzzy logic-based Cloud Model and a Deep Neural Networks (DNN) classifier, resulting in a highly effective approach for categorizing various cyber threats. When compared with the existing Feature Selection Whales Optimization Algorithm with Deep Neural Network Model (FS-WOA_DNN), the newly developed Artificial Bees Optimization-based Cloud Algorithm with Deep Neural Networks Model (DABCO_CM_DNN) shows a 3.02% increase in sensitivity, a 2.63% increase in accuracy, a 3.15% increase in specificity, and a slight decrease in error rate. This indicates that the DABCO_CM model significantly enhances real-time intrusion detection by achieving high accuracy, lowering false positives, and ensuring efficient processing. Future research will aim to refine the model, explore additional bio-inspired techniques, and expand its application across different virtualized environments. Real-world deployment and integration with existing security systems will also be pursued to validate the model's effectiveness and practicality, ensuring its continued relevance in combating evolving cyber threats.

ACKNOWLEDGMENT

We extend our heartfelt gratitude to the Africa Centre of Excellence (ACE), OAU ICT-Driven Knowledge Park (OAK Park), Obafemi Awolowo University, Ile-Ife, Nigeria, for their invaluable support and sponsorship in facilitating this research. We are particularly grateful to Professor G.A. Aderounmu, the Centre Director, whose visionary leadership and commitment to fostering innovation and academic excellence have significantly contributed to the success of this work. This publication is a testament to the remarkable infrastructure and collaborative environment provided by ACE OAK Park, enabling groundbreaking research and capacity building in ICT-driven knowledge.

REFERENCES

[1] Agarwal, A., Khari, M., & Singh, R. (2022). Detection of DDOS Attack using Deep Learning Model in Cloud Storage Application. *Wireless Personal Communications, 127*(1), 419–439. https://doi.org/10.1007/s11277-021-08271-z.

[2] Ahmed, N., Sadhayo, I. H., Yousif, Z., Naeem, N., & Parveen, S. (2019). Analysis and Detection of DDoS Attacks Targetting Virtualized Servers. *International Journal of Computer Science and Network Security (Vol. 19),* 6. Retrieved from https://www.researchgate.net/publication/334325978.

[3] Aishwarya, R., & Malliga, S. (2014). Intrusion detection system- An efficient way to thwart against Dos/DDos attack in the cloud environment. *2014 International Conference on Recent Trends in Information Technology, ICRTIT 2014,* 6. https://doi.org/10.1109/ICRTIT.2014.6996163.

[4] Aldwairi, M., Khamayseh, Y., & Al-Masri, M. (2015). Application of artificial bee colony for intrusion detection systems. *Security and Communication Networks, 8*(16), 2730–2740. https://doi.org/10.1002/sec.588.

[5] Gu, T., Chen, H., Chang, L., & Li, L. (2019). Intrusion detection system based on improved abc algorithm with tabu search. *IEEJ Transactions on Electrical and Electronic Engineering, 14*(11), 1652–1660. https://doi.org/10.1002/tee.22987.

[6] Jin, Y., Sun, Y., & Ma, H. (2018). A developed Artificial Bee Colony algorithm based on cloud model. *Mathematics, 6*(4), 61. https://doi.org/10.3390/math6040061.

[7] Khalaf, B. A., Mostafa, S. A., Mustapha, A., Ismaila, A., Mahmoud, M. A., Jubaira, M. A., & Hassan, M. H. (2019). A simulation study of syn flood attack in cloud computing environment. *AUS Journal, 26*(1), 188–197.

[8] Kumar, J. (2019). Cloud computing security issues and its challenges: A comprehensive research. *International Journal of Recent Technology and Engineering (Vol. 8),* 10-14.

[9] Kuo, R. J., Huang, S. B. L., Zulvia, F. E., & Liao, T. W. (2018). Artificial bee colony-based support vector machines with feature selection and parameter optimization for rule extraction. *Knowledge and Information Systems, 55*(1), 253–274. https://doi.org/10.1007/s10115-017-1083-8.

[10] Mthunzi, S. N., & Benkhelifa, E. (2017). Trends Towards Bio-inspired Security Countermeasures for Cloud Environments. Paper presented at the *IEEE International Conference on Self-Adaptive and Self-Organizing Systems, 11th Proceedings, 18-22 September 2017, Tucson, Arizona, USA. University of Arizona, IEEE Computer Society, Institute of Electrical and Electronics Engineers, National Science Foundation (U.S.).*

[11] Navaz, A. S., Sangeetha, V., & Prabhadevi, C. (2013). Entropy based Anomaly Detection System to Prevent DDoS Attacks in Cloud. *International Journal of Computer Applications (Vol. 62),* 9. https://doi.org/10.5120/10160-5084.

[12] Otor, S. U., Akinyemi, B. O., Aladesanmi, T. A., Aderounmu, G. A., & Kamagaté, B. H. (2021). An improved bio-inspired based intrusion detection model for a cyberspace. *Cogent Engineering, 8*(1), 24. https://doi.org/10.1080/23311916.2020.1859667.

[13] Prathyusha, D. J., & Kannayaram, G. (2021). A cognitive mechanism for mitigating DDoS attacks using the artificial immune system in a cloud environment. *Evolutionary Intelligence, 14*(2), 607–618. https://doi.org/10.1007/s12065-019-00340-4.

[14] Sharma, S., Gupta, A., & Agrawal, S. (2016). An intrusion detection system for detecting denial-of-service attack in cloud using artificial bee colony. In *Advances in Intelligent Systems and Computing (Vol. 438, pp. 137–145).* Springer Verlag. https://doi.org/10.1007/978-981-10-0767-5_16.

[15] Tayab, A., Talib, W., & Fuzail, M. (2015). Security Challenges for Virtualization in Cloud (Vol. 20), 6. Retrieved from https://www.researchgate.net/publication/326261342.

[16] Wang, B., Zheng, Y., Lou, W., & Hou, Y. T. (2014). DDoS attack protection in the era of cloud computing and software-defined networking. In *Proceedings - International Conference on Network Protocols, ICNP (pp. 624–629).* IEEE Computer Society. https://doi.org/10.1109/ICNP.2014.99.

[17] Young, C. (2016). Data Centers: A Concentration of Information Security Risk. *The Juilliard School · Department of Information Technology,* (pp. 339-357). https://doi.org/10.1016/B978-0-12-809643-7.00015-2.

[18] Yu, X., Han, D., Du, Z., Tian, Q., & Yin, G. (2019). Design of DDoS attack detection system based on intelligent bee colony algorithm. *International Journal of Computational Science and Engineering, 19*(2), 22.