



Enhancing Zero-Day and Known Attacks Detection using Hybrid-Optimized Stacking Ensemble Classifier

Manoj Kumar Singh¹, Jyotiprakash Patra², Siddharth Choubey³ and Sumitra Samal⁴

ABSTRACT

Zero-day and malicious attacks are still a research challenge for host system security. Zero-day attacks targeting the host systems digital data. To address this problem, the proposed two-stage multi-layered framework detects both zero-day and known attacks. The first stage uses a correlation-based filtering approach. Followed by feature selection using the Boruta algorithm and then the autoencoder (BAE) anomaly detection module. The second stage develops a stacking ensemble classifier (SEC) and optimizes it using the Swarm-Bayesian Hybrid Optimization (SBHO) technique. The combined approach forms the BAE-SEC-SBHO model. The model has been evaluated on the UNSW Windows ToN 2020 dataset, which includes host behavioral features like process, disk, and memory activity. The suggested pipeline reduces false-negative rates and improves recall rates compared to traditional machine learning and deep learning models. The performance and robustness of the proposed model are validated through an ablation study and a Wilcoxon signed-rank test for host-based intrusion detection.

Article information:

Keywords: Anomaly Detection, Autoencoder, Cyber-security, Deep Learning, Host Intrusion Detection System, Zero-day Attack

Article history:

Received: January 26, 2026

Revised: April 23, 2026

Accepted: June 21, 2026

Published: July 18, 2026

(Online)

DOI: 10.37936/ecti-cit.2026203.265913

1. INTRODUCTION

In recent years, cyber-attacks have increased in frequency and severity, impacting nearly every organization [1]. As attackers continually enhance their tools and exploit system vulnerabilities, they can launch increasingly creative and sophisticated attacks. Since both personal and organizational assets rely heavily on digital data, they are prime targets [2]. The cyber-attack statistics for November 2024 show the annual escalation in attack rates [3]. The report reveals that malware and unknown attacks cover a major portion among all cyberattacks, representing approximately 26.83% of the total. Ransomware attacks constitute the second-largest category, accounting for approximately 13.4%. According to another analysis, the global economy is expected to suffer losses exceeding USD 10.5 trillion due to cybercrime. These losses include data destruction and damage, financial theft, loss of research, theft of sensitive personal and busi-

ness data, ransom payments, and other related costs [4].

When it comes to identifying such attacks, the intrusion detection system is considered the last line of defense and is broadly categorized into host-based and network-based intrusion detection systems [5]. Because malware exhibits limited and evolving behavioral patterns that are not represented in existing signatures, signature-based security systems cannot effectively detect zero-day attacks. Thus, securing the host machines' digital assets is still a challenge and an open area of research. In this study, a zero-day attack is defined as an attack class absent from the training data and introduced only during testing. In particular, zero-day malicious programs remain undetected [6].

In this context, numerous researchers have proposed innovative solutions utilizing machine learning techniques. A study used the Cuckoo Sandbox environment and achieved an accuracy of 98.92%;

^{1,3}The authors are with the Department of Computer Science and Engineering, Shri Shankaracharya Technical Campus(SSTC), Bhilai, Chhattisgarh 490001, India, Email: manojksingh.in@gmail.com and sidd25876@gmail.com

²The author is with the Department of Computer Science and Engineering, Chhattisgarh Swami Vivekanand Technical University (CSVtu), Bhilai, Chhattisgarh 490001, India, Email: jppatra.cse@csvtu.ac.in

⁴The author is with the Department of Computer Science and Engineering, Rungta College of Engineering & Technology (RCET), Bhilai, Chhattisgarh 490001, India, Email: samal.sumitra07@gmail.com

¹Corresponding author: manojksingh.in@gmail.com

however, the system still exhibited a high false-positive rate [7]. Another study reported that the zero-shot learning approach achieved an overall accuracy of 92.45% and offered faster processing. However, it exhibited a significantly higher false-negative rate and a lower zero-day detection rate than the other approaches [8]. Relevant research proposed a comprehensive framework that combined supervised and unsupervised learning by clustering similar data points and defining minimum intra-cluster distances [9]. However, while this model achieved 98.4% accuracy, it still experienced a 1.6% false-negative rate. Clustering methods may be less effective for zero-day attack detection because they rely heavily on well-defined cluster structures, which often fail to adapt to previously unseen attack patterns, resulting in increased false alarm rates.

The latest study claimed an intelligent zero-day attack detection framework using unsupervised machine learning (ZdAD-UML) [10]. The model uses an Enhanced K-means Clustering-based Weighted Crayfish Autoencoder (EKC-WCAE) for feature extraction. Although it improves feature extraction performance, it still has several limitations. The computational cost of combining K-means and Crayfish Optimization makes the model too complex. Furthermore, the limited training data can impair its detection accuracy and generalization with 20/80 or 30/70 train/test splits.

Zero-day attacks remain a major challenge to host system security. The proposed research introduces an autoencoder-assisted, hybrid-optimized stacked deep learning framework for detecting both known and zero-day intrusions in host systems to address the limitations of existing methods. To enhance the host system security by finding zero-day anomalies with high recall and improving known intrusion detection with fewer false alarms are the main goals of this study. Here are some of the research works' main contributions and innovations:

- **Discriminative Feature Extraction with BAE:** A correlation-based filtering approach was first applied to eliminate multicollinear and redundant features, preserving only those with minimal interdependence. Boruta-based feature selection is used to retain the most relevant attributes, followed by autoencoder-based feature compression for anomaly detection.
 - **Enhanced Autoencoder:** This study follows a systematic procedure to optimize the proposed autoencoder and determine the most effective latent space representation. The suggested autoencoders' learning rate and decay parameters were improved using the Adam optimizer and Keras Tuner. MSE was used as the reconstruction loss because it penalizes larger reconstruction errors more heavily than MAE, making it more effective at capturing anomalous patterns.
 - **Stacking Ensemble Classifier:** The innovative stacking ensemble (SEC) model efficiently classifies anomalies as either benign or malware by combining a SimpleRNN as the meta-classifier with Bi-LSTM and GRU as base classifiers. The base models may learn a variety of intricate attack patterns because of this layered design, and the meta-classifier cleverly combines their outputs to improve decision-making. The combination reduces overfitting, increases generalization, and enhances feature representation.
 - **Swarm Bayesian Hybrid Optimization (SBHO) Technique:** To enhance zero-day and known malware classification, a hybrid optimization strategy (SBHO) combining Particle Swarm Optimization (PSO) and Bayesian Optimization (BO) has been used to optimize the proposed deep learning stacking ensemble classifier. This hybrid strategy strikes a good balance between exploration and exploitation. PSO helps avoid premature convergence by maintaining diversity, and BO ensures a focused search using acquisition strategies.
- Finally, the novelty of the proposed research lies in a hybrid framework that combines autoencoder-based anomaly detection with a stacking ensemble classifier. The stacking ensemble classifier is jointly optimized using the SBHO strategy to improve its overall classification performance. Unlike conventional approaches, SBHO enables cross-stage, dependency-aware hyperparameter tuning, aligning feature learning and classification to improve zero-day detection performance. Section 2 is dedicated to the discussion of related work, while Section 3 provides a detailed explanation of the proposed BAE-SEC-SBHO pipeline. The experimental methodology used in this paper is described in Section 4, while the results and performance evaluation are discussed in Section 5. Section 6 presents a comparative analysis between the proposed approach and existing related studies.

2. LITERATURE REVIEW

This section presents a review of notable research efforts addressing intrusion detection for zero-day (unseen) attacks on host systems. Zero-day intrusion detection is a challenging issue in cybersecurity since it requires detecting and preventing attacks that exploit unidentified defects or vulnerabilities in systems, software, or networks. Researchers have developed a variety of methods to improve the efficiency and accuracy of intrusion detection systems (IDS) across different application domains. The authors in [11] proposed a hybrid intrusion detection system for both binary and multiclass classification. It integrates models implemented on Apache 2. A similar study achieved an accuracy of 99.87% for binary classification and 85.24% for multiclass classification on the NSL-KDD dataset. However, it still exhibited a high false alarm rate in the multiclass classification

task.

A Gini impurity-weighted Random Forest-based model was proposed in [12] for binary intrusion detection and achieved an accuracy of 99.90% on the TON_IoT dataset. Although their work focused on binary classification, they suggested that extending it to multiclass detection could enhance overall performance. Another study [13] employed a hybrid feature selection strategy to develop a multiclass intrusion detection system for cloud environments. Their Random Forest-based model achieved 99% accuracy on the Kyoto dataset and 98% accuracy on UNSW-NB15. A similar study [14] achieved an accuracy of 99.30% in multiclass intrusion detection using XG-Boost. These approaches perform well in detecting known attacks but are ineffective against zero-day threats. This highlights the need for more adaptive methods capable of identifying previously unseen intrusions.

Anomaly intrusion detection is categorized into three types. The statistical approach detects anomaly intrusions by accumulating and analyzing data records to construct a statistical model that represents the normal behavior of users and the system. The knowledge-based approach uses existing system data to identify anomalies. The use of machine learning techniques solves complex pattern-matching capabilities from training data [15], [16]. In the area of host machine security, operating system research is essential. The operating system is the only interface the user has with the computer system. To interact with the system, users typically use applications that operate at the application layer, representing normal host system behavior. Malicious software or code exists at the application level in the operating systems background [17]. System call monitoring shows significant behavior at the kernel and hardware levels [18]. The research in [19] also proposed using operating system-level data shared across applications for behavior-based intrusion detection.

This work focuses on using memory activity features to identify zero-day intrusions on a host system. Malware detection has recently made extensive use of machine learning and deep learning techniques, which help differentiate between normal patterns and malicious behavior [20]. To detect anomalies, these techniques analyze known or suspicious behavior [21]. In the rapidly evolving field of cybersecurity, identifying and mitigating zero-day (previously unseen) threats has become a major concern [22]. Zero-day exploits pose a significant threat because they can bypass traditional security measures and exploit previously unknown vulnerabilities [23]. To overcome this challenge, researchers have increasingly turned to machine learning methods for intrusion detection, particularly on host PCs [24], [25]. The authors of [26] suggested using long short-term memory (LSTM) to categorize sequential data from Windows executable

API calls. Most of the time, they were 95% accurate.

The reviewed studies indicate that detecting zero-day attacks remains challenging because their unknown behavior contributes to a high false-negative rate. In response to the growing threat of malware, modern techniques such as deep learning have shown significant potential for improving zero-day intrusion detection [27]. This paper introduces the BAE-SEC-SBHO, a stacking ensemble classifier based on Bi-LSTM [28], SimpleRNN [28], and GRU [29] that uses behavioral activities to solve several kinds of known and anomaly intrusion detection problems with SBHO hybrid optimization. To provide more context, the results are compared with zero-day detection and with related studies.

3. PROPOSED METHODOLOGY

To address the issue of zero-day attack detection, the proposed model is presented. There are three primary phases to the proposed BAE-SEC-SBHO framework. The approach improves accuracy and adaptability for complex, high-dimensional intrusion detection tasks. Fig. 1 shows all stages of proposed model.

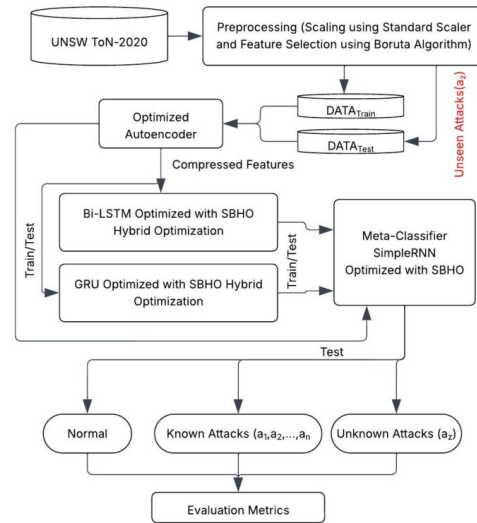


Fig.1: The Architecture of Proposed Model.

3.1 Pre-processing, selecting optimal features

The dataset is first divided into separate training and testing sets. Here $DATA_{train} = \{(x, y) | x \in X_{train}, y \in Y_{train} = \{n, a_1, a_2, \dots, a_n\}\}$ and $DATA_{test} = \{(x, y) | x \in X_{test}, y \in Y_{test} = \{n, a_1, a_2, \dots, a_n, a_z\}\}$ represent the training and testing datasets, respectively. These expressions illustrate how the dataset is partitioned into training and testing sets. $Y_{train} \neq Y_{test}$, meaning that the training set contains only the known classes (normal, DDoS, DoS, and injection), while the testing set contains both the known classes and unseen attack classes (XSS, password, scanning, and MITM). Here zero-day attack classes are XSS,

password, scanning, and MITM. Fig. 2 and Fig. 3 illustrate the representation of the testing dataset $\text{DATA}_{\text{test}}$ and the training dataset $\text{DATA}_{\text{train}}$.

A correlation-based filtering method was applied to remove highly correlated and redundant features, retaining only the most informative and independent features. The dataset was then standardized using a standard scaler as $z_{\text{scaled}} = (x - \mu)/\sigma$, where z represents the standardized value, x signifies the original feature value, μ represents the feature mean, and σ represents the feature standard deviation. This scaling and standardization ensure that all features contribute equally during training, enabling deep learning models to converge faster and optimize more effectively [29]. First, scaled data are used for dimensionality reduction, after which Boruta feature selection removes features that are less significant than shadow features by iteratively assessing the significance of features on low-dimensional data [30]. To avoid any data leakage, pre-processing steps that require parameter estimation, such as feature scaling, are performed using only the training data, and the same parameters are then applied to the test data.

Similarly, feature selection techniques such as correlation filtering and Boruta are performed exclusively on the training data. The selected features are then consistently used for both training and testing. This ensures that the model does not learn any information from the test data during training.

3.2 Discriminative Feature Extraction with BAE

To circumvent the challenges of identifying unseen attacks, an autoencoder and neural network-based approach was used to produce compressed features and for anomaly detection [31]. The autoencoder required extensive fine-tuning and has scalability issues. The input sample sets satisfy $X_{\text{train}} \subseteq \{x|y \in Y_{\text{train}}\}$ and $X_{\text{test}} \subseteq \{x|y \in Y_{\text{test}}\}$, where $Y_{\text{train}} \subset Y_{\text{test}}$. The autoencoder architecture consists of two core functions: an encoder $f_{\text{enc}}(x; \theta_{\text{enc}})$ which maps each input $X \subset R^m$ into a lower-dimensional latent representation $z \subset R^k$ with $k \ll m$, and a decoder $f_{\text{dec}}(z; \theta_{\text{dec}})$, which reconstructs the input from the latent space. The network parameters θ_{enc} and θ_{dec} are learned by minimizing the reconstruction loss over the training set. The loss is calculated using mean squared error (MSE):

$$L(\theta_{\text{enc}}, \theta_{\text{dec}}) = \frac{1}{N} \sum_{i=1}^n \|x_i - f_{\text{dec}}(f_{\text{enc}}(x_i))\|^2 \quad (1)$$

To ensure that the model generalizes well and produces high-quality latent features, a hyperparameter optimization step is introduced. The hyperparameter learning rate $LR \in [LR_{\text{min}}, LR_{\text{max}}]$, batch size $BS \in [BS_{\text{min}}, \dots, BS_{\text{max}}]$, and number of epochs $E \in [E_{\text{min}}, E_{\text{max}}]$ are tuned using Keras Tuner, which

systematically explores the defined search space and identifies configurations that minimize the validation loss. After identifying the optimal configuration, the autoencoder was retrained using the optimized learning rate and batch size. The final optimization objective becomes:

$$\min L = \|X - f_{\text{dec}}(f_{\text{enc}}(X))\|^2 \quad (2)$$

Resulting in an optimized autoencoder A_{best} . The trained autoencoder generates compressed latent representations, $Z = f_{\text{enc}}(X)$, that preserve the essential characteristics of the input data for subsequent classification. Because the training set excludes zero-day class az, the latent features learned are representative only of known behaviors.

3.3 Swarm Bayesian Hybrid Optimizer(SBHO)

The proposed hybrid approach uses the Simple RNN as a meta classifier and the Bi-LSTM and GRU as basis classifiers. The BAE-generated compressed training and testing data are used as input, which is then reshaped based on the needs of additional experiments shown in Algorithm 1. The base classifiers are optimized using a hybrid strategy (SBHO) that combines Particle Swarm Optimization (PSO) [32] and Bayesian Optimization (BO) [33] to achieve optimal performance. PSO is used to efficiently explore the parameter space and identify promising regions of the search space. Bayesian optimization (BO) is then used to fine-tune the hyperparameters within the promising regions identified by PSO. The optimization begins by initializing a population of N particles $\{x_i^0\}_{i=1}^n$, where each particle is sampled uniformly from the bounded search space according to $X_i^{(0)} \sim \text{Uniform}(x_{\text{min}}, x_{\text{max}})$. The corresponding velocities, $\{V_i^0\}_{i=1}^n$, are also initialized by sampling from a uniform distribution over the range $[-v_{\text{max}}, v_{\text{max}}]$. Each particle objective value is then computed as $f(X_i^{(0)})$, and its personal best position is set as $P_i^{(0)} = X_i^{(0)}$. The global best among all particles is determined by $g^{(0)} = \arg \min_i f(X_i^{(0)})$. For each iteration $t \in \{1, 2, \dots, T\}$, the velocity of each particle is updated using the standard PSO rule:

$$V_i^t = \omega V_i^{(t-1)} + c_1 r_1 (P_i^{(t-1)} - X_i^{(t-1)}) + c_2 r_2 (g^{(t-1)} - X_i^{(t-1)}) \quad (3)$$

where ω is inertia weight, c_1, c_2 are cognitive and social acceleration constants, and $r_1, r_2 \sim \text{Uniform}(0, 1)$ are random scalars. The updated positions are computed as $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$, after which they are projected back into the feasible search space to satisfy the constraint $x_{\text{min}} \leq X_i^{(t)} \leq x_{\text{max}}$.

Following the position update, the objective function is evaluated at each new particle position,

$f(X_i^{(t)})$. The personal best positions are then updated according to the rule: if $f(X_i^{(t)}) < f(P_i^{(t-1)})$, then $P_i^{(t)} = X_i^{(t)}$; otherwise, $P_i^{(t)} = P_i^{(t-1)}$. The global best solution is updated by evaluating $g^{(t)} = \arg \min_i f(P_i^{(t)})$.

To refine the global search with probabilistic modeling, Bayesian optimization is applied at each iteration. A surrogate model M , such as a Gaussian Process, is trained using the current particle population $X_i^{(t)}$ and the corresponding objective function values, $f(X_i^{(t)})$. An acquisition function, such as Expected Improvement (EI), is then maximized over the search space to propose a refined candidate x_{BO} . The function value $f(x_{BO})$ is evaluated, and if $f(x_{BO}) < f(g^{(t)})$, the global best is then updated to $g^{(t)} = x_{BO}$; otherwise, the current global best is retained.

Until a convergence criterion, like the maximum number of iterations T , is satisfied, this iterative optimization keeps going, or the stagnation of improvement in $f(g^{(t)})$. Upon completion, the final optimized hyperparameter vector is denoted as $x_{best} = g^{(T)}$, which is then used to train the optimized autoencoder, A_{best} .

The search space for hyperparameters is explicitly defined. The number of LSTM units fluctuated between 32 and 128, the dropout rate between 0.1 and 0.5, and the learning rate between 10^{-4} and 10^{-2} . These bounds are used during the PSO-based global search. The objective of the optimization is to improve model performance on unseen data. For hyperparameter tuning, the original training data are divided into training and validation subsets in an 80:20 ratio. The optimization process uses only this split. At the same time, the test set is kept completely separate and is used only once for final evaluation. This ensures that the reported results are not influenced by the hyperparameter tuning process.

The final model is a multiclass classifier trained on known classes in a multiclass setting. At test time, it encounters both known and previously unseen attack types. As the unseen classes are not part of the training distribution, the model maps these samples to the most similar known class, or occasionally to the normal class.

4. EXPERIMENTAL SETUP

The experimental setup for the proposed research was implemented on a system equipped with an 11th Generation Intel® Core™ i5-11260H processor operating at 2.60 GHz, 16 GB DDR4 RAM, a 256 GB SSD, and a 1 TB HDD, running Windows 11. Python was used as the foundation language in JupyterLab, and the experiments were conducted in a miniconda environment.

Algorithm 1: Hybrid-Optimized Stacking Ensemble Classifier

Input: Compressed features Z_{train}, Z_{test} , labels Y_{train}, Y_{test} , particle count N , max iterations T , search bounds $[x_{min}, x_{max}]$
Output: Predicted classification labels.

- 1: Reshape Z_{train}, Z_{test} as required for sequence models
- 2: Initialize particles $X_i^{(0)} \sim U(x_{min}, x_{max})$ for $i = 1$ to N
- 3: Initialize velocities $V_i^{(0)} \sim U(-v_{max}, v_{max})$
- 4: **for** $i = 1$ to N **do**
- 5: Evaluate objective $f(X_i^{(0)})$ using base models
- 6: Set personal best $P_i^{(0)} \leftarrow X_i^{(0)}$
- 7: **end for**
- 8: Set global best $g(0) \leftarrow \arg \min_i f(X_i^{(0)})$
- 9: **for** $t = 1$ to T **do**
- 10: **for** $i = 1$ to N **do**
- 11: Sample $r_1, r_2 \sim U(0, 1)$
- 12: Update velocity: $V_i^{(t)} = \omega V_i^{(t-1)} + c_1 r_1 (P_i^{(t-1)} - X_i^{(t-1)}) + c_2 r_2 (g^{(t-1)} - X_i^{(t-1)})$
- 13: Update position: $X_i^{(t)} = X_i^{(t-1)} + V_i^{(t)}$
- 14: Clip $X_i^{(t)}$ within $[x_{min}, x_{max}]$
- 15: Evaluate $f(X_i^{(t)})$
- 16: **if** $f(X_i^{(t)}) < f(P_i^{(t-1)})$ **then**
- 17: $P_i^{(t)} = X_i^{(t)}$
- 18: **else**
- 19: $P_i^{(t)} = P_i^{(t-1)}$
- 20: **end if**
- 21: **end for**
- 22: Update global best: $g(t) \leftarrow \arg \min_i f(P_i^{(t)})$
- 23: Train surrogate model M using $\{X_i^{(t)}, f(X_i^{(t)})\}$
- 24: Optimize acquisition function to obtain x_{BO}
- 25: Evaluate $f(x_{BO})$
- 26: **if** $f(x_{BO}) < f(g(t))$ **then**
- 27: $g(t) \leftarrow x_{BO}$
- 28: **end if**
- 29: **if** The stopping criteria are met **then**
- 30: **Break**
- 31: **end if**
- 32: **end for**
- 33: $x_{best} \leftarrow g(T)$
- 34: Train Bi-LSTM and GRU models on Z_{train} using x_{best}
- 35: Generate predictions O_{BiLSTM}, O_{GRU}
- 36: Concatenate base outputs to form O_{stack}
- 37: Train SRNN meta-classifier on O_{stack}, Y_{train}
- 38: **For** $r = 1$ to 10
- 39: **for** each sample in Z_{test} **do**
- 40: Predict $y_{pred} \leftarrow \text{SRNN}(\text{Concat}(\text{BiLSTM}(z), \text{GRU}(z)))$
- 41: **end for**
- 42: **end for**
- 43: Evaluate predictions using a confusion matrix

4.1 Dataset Description

The UNSW TON-IoT 2020 dataset [34], which was utilized for evaluation in this study. This dataset includes data collected from multiple sources, including the Windows operating system and IoT and IIoT sensor telemetry. Researchers at the Australian Defense Force Academy (ADFA) Cyber Range and IoT Labs collected the datasets from a large-scale, real-world network environment. Researchers captured a variety of normal activities and cyberattack events from network traffic, Windows and Linux audit logs, and IoT service telemetry. They processed these data sources in parallel to efficiently handle the large volume of generated data. It has host behavior features like process, disk, and memory activity patterns, which are essential for detecting abnormal intrusions in windows host systems.

The proposed framework retrieves the $DATA_{test}$ dataset to address the objective of zero-day anomaly intrusion detection. The collection consists of seven dangerous classes and one normal class. Fig. 2 shows the class-wise distribution of the $DATA_{test}$ dataset. Fig.3 shows the sample size for the $DATA_{train}$ dataset.

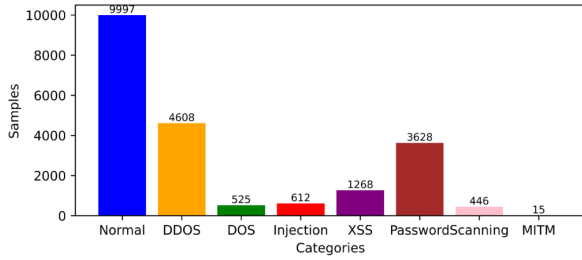


Fig.2: $DATA_{test}$ Size of Testing Data.

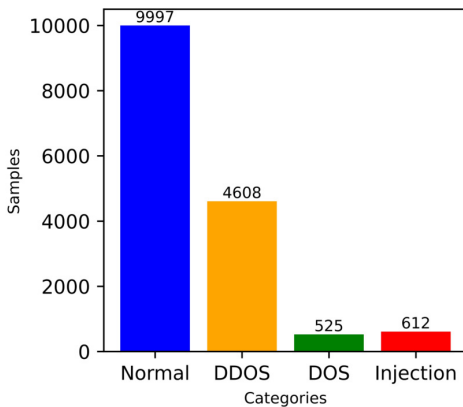


Fig.3: $DATA_{train}$ Size of Training Data.

4.2 Model Parameter Settings

The SimpleRNN serves as the meta classifier in the suggested stacking ensemble model, while Bi-LSTM and GRU serve as the base classifiers. This study

employs PSO and BO to offer the ideal learning rate, number of neurons, and dropout rate [35]. The proposed framework uses 114 neurons with ReLU activation in the outer layer, 32 neurons with ReLU activation in Dense Layer 1, 8 neurons with the “Softmax” activation function in Dense Layer 2, and a dropout rate of 0.1242 across all layers to optimize the base and meta-classifiers. The base and meta-classifiers employ the Nadam optimizer with a learning rate of 0.0085 generated by the PSO-BO optimization process, 100 training epochs, and a batch size of 128. The study calculates the average classification results by performing multiple Monte Carlo-style evaluations on the same training and testing datasets. Each run differs only in the random initialization and internal stochastic processes.

4.3 Model Evaluation Metrics

The quality of the suggested approach has been assessed using a combination of assessment metrics: accuracy(A_C), precision(P_C), recall(R_C), $F1_C$, false alarm rate (FAR_C), and false-negative rate (FNR_C). In the evaluation, the meaning of the associated metric label “C” varies based on the context. It is treated as known detection (KD) when assessing performance on known classes, and as zero-day detection (ZD) when evaluating unseen attacks. This approach allows all metrics to be computed separately for each case (e.g., A_{KD} and A_{ZD}), providing a clearer comparison of model performance. Consequently, the metrics listed in Tab.1:

Table 1: Evaluation Metrics.

Metric	Description
Accuracy(A_C) $\left(\frac{TP_C + TN_C}{TP_C + TN_C + FP_C + FN_C} \right) \times 100$	The overall accuracy of the model, computed across both zero-day attack and benign samples, is the proportion of total correct predictions.
Precision(P_C) = $\left(\frac{TP_C}{TP_C + FP_C} \right) \times 100$	Correctly detected zero-day attacks out of a total samples predicted as unseen.
Recall(R_C) = $\left(\frac{TP_C}{TP_C + FN_C} \right) \times 100$	Correctly detected unseen attacks from a total actual samples of unseen attacks.
$F1_C = \left(\frac{2 * Recall_C * Precision_C}{Recall_C + Precision_C} \right) \times 100$	The harmonic mean of Precision $_C$ and Recall $_C$
False alarm rate(FAR_C) = $\left(\frac{FP_C}{TN_C + FP_C} \right) \times 100$	The model classifies some benign samples as zero-day attacks, and the resulting proportion represents the false alarm rate.
False-negative rate(FNR_C) = $\left(\frac{FN_C}{FN_C + TP_C} \right) \times 100$	The model incorrectly classifies a proportion of zero-day samples as benign.

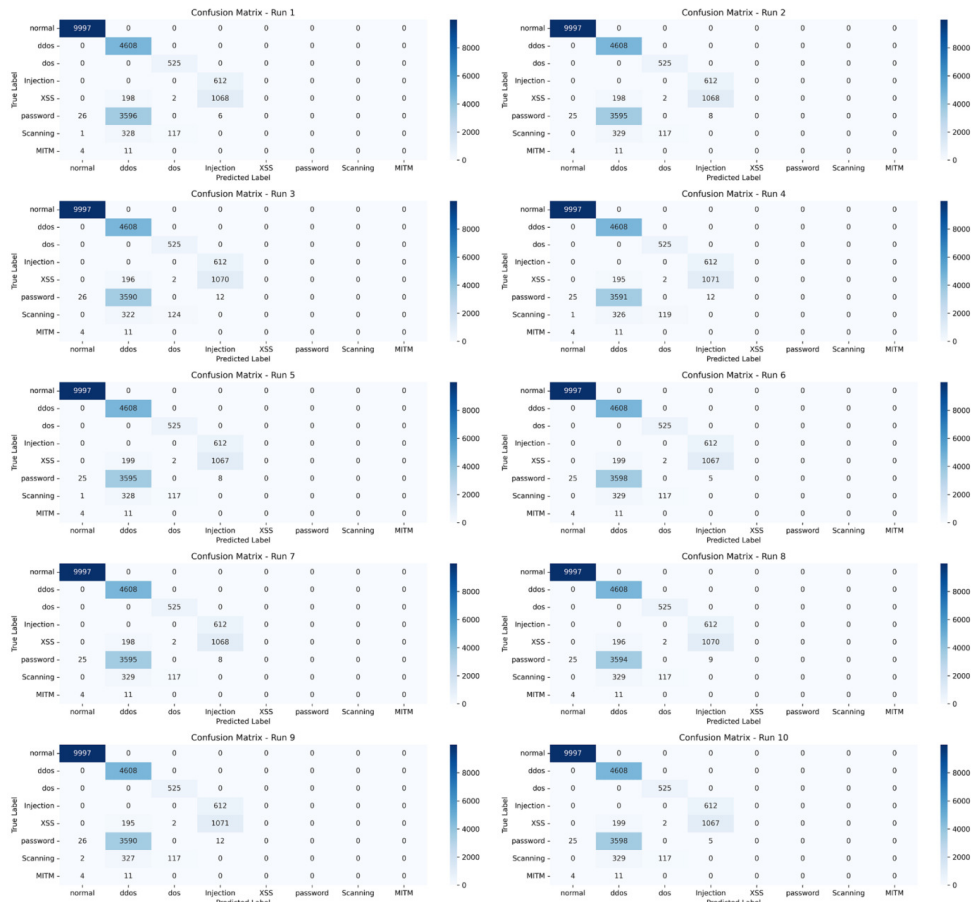


Fig.4: Multiclass Classification Results on All 10 Runs on Test Data.

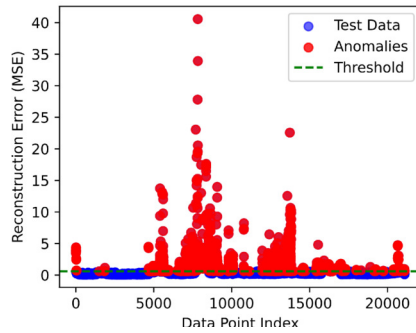


Fig.5: Anomaly Detection on Test Data.

Table 2: Zero-Day Attack Results in Percentage.

Run	A _{ZD}	P _{ZD}	R _{ZD}	F1 _{ZD}	FNR _{ZD}
1	99.80	100	99.42	99.71	0.58
2	99.81	100	99.46	99.73	0.54
3	99.80	100	99.44	99.72	0.56
4	99.80	100	99.44	99.72	0.56
5	99.80	100	99.44	99.72	0.56
6	99.81	100	99.46	99.73	0.54
7	99.81	100	99.46	99.73	0.54
8	99.81	100	99.46	99.73	0.54
9	99.79	100	99.40	99.70	0.60
10	99.81	100	99.46	99.73	0.54
Average	99.80	100	99.44	99.72	0.55

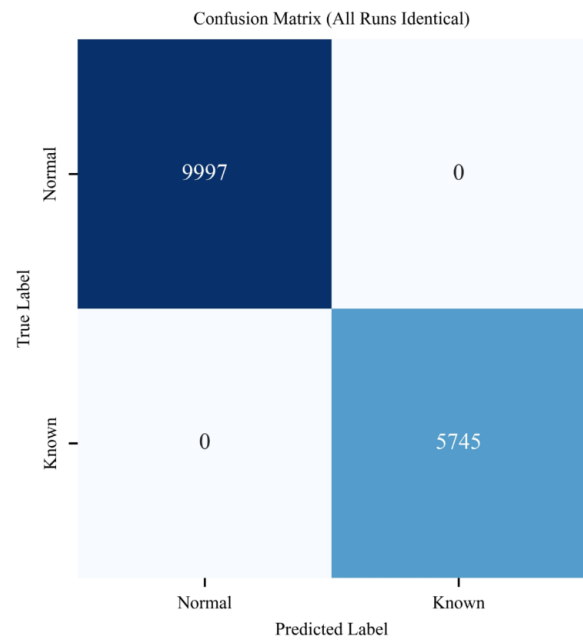


Fig.6: Known Attack Detection on Test Data.

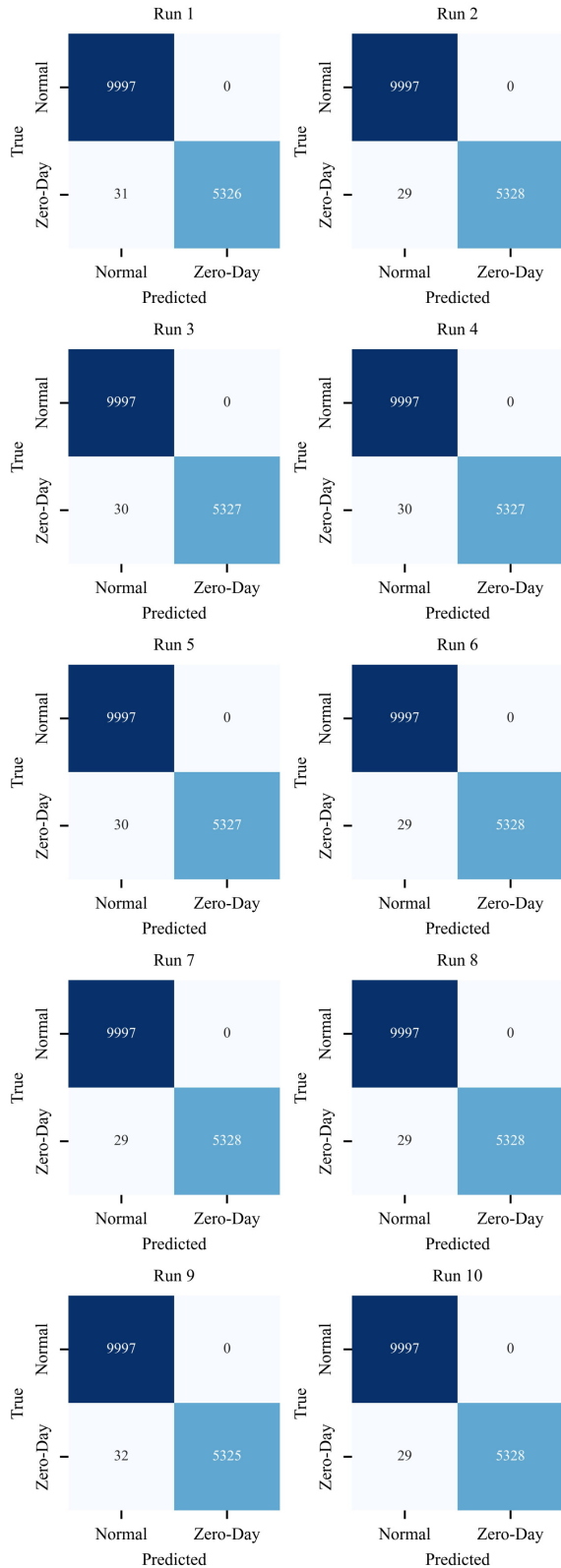


Fig.7: Zero-Day Attack Detection on Test Data.

Table 3: Known Attack Results in Percentage.

Run	A _{KD}	P _{KD}	R _{KD}	F1 _{KD}	FAR _{KD}
1-10	100	100	100	100	0
Average	100	100	100	100	0

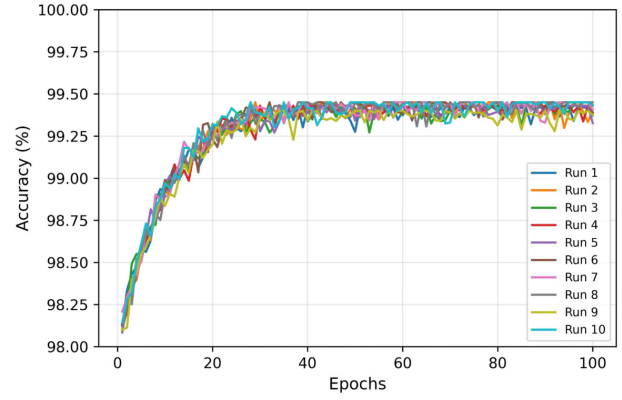


Fig.8: Accuracy Curve of Proposed Model on Each Run.

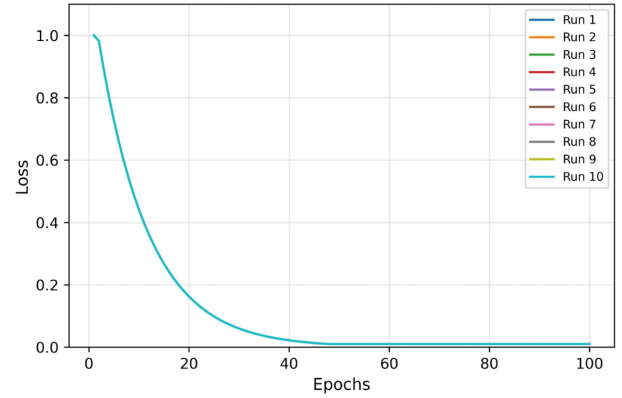


Fig.9: Loss Curve of Proposed Model on Each Run.

5. RESULTS AND DISCUSSION

The experiment consists of two sections to assess the significant improvement of the proposed approach. First, the study evaluates the basic RNN, GRU, LSTM, Bi-LSTM, and the proposed stacking ensemble model without optimizing the individual deep learning models. It then records their performance for comparison. Next, the study applies the optimization and enhancement approaches to improve prediction accuracy. It then evaluates the proposed stacking ensemble model in the second stage.

5.1 Anomaly Detection and Classification

The proposed model uses the compressed features generated by the BAE for classification. Fig. 5 displays the anomaly data that the suggested stacking ensemble classifier (SEC) would employ for additional classification. Bi-LSTM and GRU have been used to produce the level-1 predictions and further pass them to the meta-classifier SimpleRNN for final classification. The final model performs multiclass clas-

sification. The study derives the binary evaluation results from the multiclass confusion matrix shown in Fig. 4 by grouping class labels instead of using a separate binary classifier. The evaluation groups all attack classes observed during training into the “known” category and compares them with the “normal” class. All unseen classes are grouped as “zero-day” and compared against the “normal” class. The perfect classification performance shown in Fig. 6 on known classes reflects the controlled experimental setup and strong class separability, rather than implying universal generalization. Fig. 7 presents the zero-day detection results for each run. Fig. 8 and Fig. 9 show the accuracy and loss curves of the proposed model. The 50–100 epoch range is where the accuracy level is maintained. The loss value decreases as the number of epochs increases. Tab. 2 and Tab. 3 present a micro-level analysis of the results from each run for zero-day and known attacks.

5.2 Comparative analysis and ablation study

This section aims to summarize previous findings and compare the proposed model with and without optimization techniques. The optimized stacking ensemble model is evaluated against the LightGBM ensemble method and four deep learning models, as presented in Tab. 4. The comparison is based on the zero-day detection recall (R_{ZD}) and zero-day false-negative rate (FNR_{ZD}) of each model. Fig. 10 shows a consistent improvement in recall (R_{ZD}) and a reduction in the false-negative rate (FNR). The study conducts an ablation analysis by comparing the proposed model with and without optimization to assess the effectiveness of the optimization stage. Let R_i^{no-opt} represent the recall obtained without optimization and R_i^{opt} represent the recall obtained with optimization for the i^{th} run in Tab. 5. The paired difference is defined as $d_i = R_i^{opt} - R_i^{no-opt}$. The average paired difference across the ten runs is calculated as $\bar{d} = \frac{1}{n} \sum_{i=1}^n (d_i)$, where $n = 10$. This metric directly quantifies the contribution of the optimization stage. Optimization confirmed a continuous gain, improving recall by 1.05%

Table 4: Comparison Analysis of Recall Results.

Run	LGBM	LSTM	Bi-LSTM	GRU	SRNN	Proposed
1	82.39	81.59	95.42	98.22	98.48	99.42
2	82.49	98.50	98.61	96.00	97.96	99.45
3	82.49	97.33	98.31	98.33	98.05	99.43
4	82.85	98.18	93.27	98.11	98.03	99.43
5	82.39	98.35	98.41	93.52	98.31	99.43
6	82.49	98.39	94.82	98.54	98.17	99.45
7	82.66	98.03	98.50	98.15	98.18	99.45
8	82.39	98.76	98.73	98.24	98.05	99.45
9	82.39	98.28	98.73	97.49	98.02	99.40
10	82.39	91.41	98.45	98.15	98.84	99.45
R_{ZD}	82.49	95.88	97.33	97.47	98.21	99.44
FNR_{ZD}	17.51	4.11	2.66	2.52	1.78	0.55

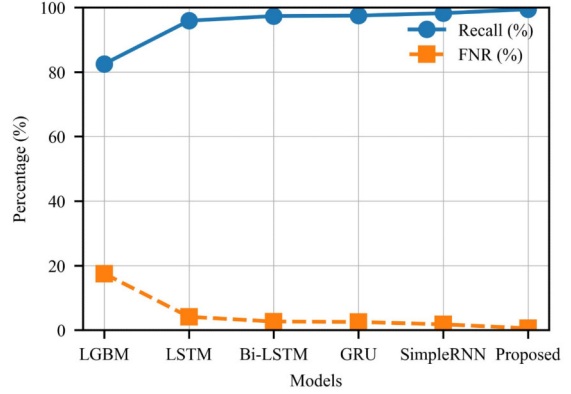


Fig.10: Comparison of Recall Results with Other Models.

Table 5: Ablation Study Results Comparison.

Run	Proposed Model Recall _{no-optimization}	Proposed Model Recall _{optimization}	d_i
1	98.31	99.42	1.11
2	98.45	99.45	1
3	98.50	99.43	0.93
4	98.43	99.43	1
5	98.48	99.43	0.95
6	98.35	99.45	1.1
7	98.31	99.45	1.14
8	98.22	99.45	1.23
9	98.43	99.40	0.97
10	98.41	99.45	1.04
Average	98.39	99.44	1.05

5.3 Statistical analysis

The study uses statistical significance testing and effect size analysis to assess the reliability of the proposed method compared with earlier research. The paired difference for each run is defined as $d_i = \text{Recall}_{\text{optimized},i} - \text{Recall}_{\text{average},i}$ where $n = 10$ and $i = 1, 2, \dots, n$. The study then uses the paired differences $\{d_1, d_2, \dots, d_n\}$ as input to the statistical tests. The paired t -test determines whether the mean of the paired differences ($\bar{d}$) is significantly greater than zero[36], whereas the Wilcoxon signed-rank test determines whether the median of the paired differences differs significantly from zero. The large effect size (Cohen’s $d = 11.1$), paired t -test ($t = 35.1, p < 0.0001$), and Wilcoxon signed rank test ($W = 0, p = 0.002$) confirm that the improvement is both practically meaningful and statistically significant ($p < 0.05$ in both tests). As a result, the suggested strategy makes a solid and trustworthy improvement over earlier techniques.

6. COMPARISON WITH SIMILAR STUDY

The proposed approach demonstrates a more efficient time complexity compared to the EKC-WCAE framework [10]. The K-means clustering stage requires $O(n \cdot k \cdot d \cdot i_1)$ operations, where n denotes the number of samples, k represents the count of clusters,

d is the feature dimension, and i_1 denotes the count of iterations required for convergence. The crayfish optimization process further increases the cost, as each solution evaluation involves autoencoder reconstruction, leading to a complexity of $O(pf \cdot i_2)$, where p is the population size, f is the cost function evaluation time, and i_2 is the number of iterations. Additionally, weighted autoencoder training contributes $O(n \cdot d \cdot L \cdot e)$ operations to the overall time complexity, where L denotes the number of layers and e represents the number of training epochs. Therefore, the overall computational complexity of EKC-WCAE is $O(n \cdot k \cdot d \cdot i_1) + O(p \cdot f \cdot i_2) + O(n \cdot d \cdot L \cdot e)$, which is significantly higher than that of the proposed method. Equation (4) presents the total time complexity of this strategy. The process begins with a correlation-based filtering approach, which reduces the dimensionality of the scaled data with a time complexity of $O(n \cdot d^2)$, where n represents the number of samples and d denotes the original feature dimension. The reduced features are then refined using Boruta, a Random Forest-based wrapper method for feature selection, resulting in a computational complexity of $O(t \cdot n \cdot r \cdot \log_n)$, where t denotes the number of iterations in the Random Forest and r represents the number of trees. Finally, an optimized autoencoder, tuned with Keras Tuner and the Adam optimizer, introduces an intricacy of $O(s \cdot e \cdot \lceil \frac{n}{b} \rceil \cdot r \cdot h)$ where s is the quantity of tuning trials, e is the count of epochs, b is the batch size, and h is the depth of hidden units in the encoder. The total time complication of BAE, therefore, becomes

$$O(n \cdot d^2 + t \cdot n \cdot r \cdot \log_n + s \cdot e \cdot \lceil \frac{n}{b} \rceil \cdot r \cdot h) \quad (4)$$

This approach is significantly more computationally efficient than that presented in [10], where the integration of K-means clustering and Crayfish Optimization results in a higher computational cost. Tab. 6 shows that, although the proposed model achieves accuracy comparable to existing methods, it offers improved computational efficiency.

Table 6: Comparison with Similar Studies.

Approach	Dataset	R _{ZD} (%)	FNR _{ZD} (%)	Time Complexity
[7]	API Call Sequences	98.92	1.08%	-
[8]	UNSW-NB15	92.45	7.55	-
[9]	IBM	98.40	1.60	-
[10]	IoT-23	98.73	1.27	$O(n \cdot k \cdot d \cdot i_1) + O(p \cdot f \cdot i_2) + O(n \cdot d \cdot L \cdot e)$
Proposed	ToN-IoT 2020	99.44%	0.55%	$O(n \cdot d^2) + t \cdot n \cdot r \cdot \log_n + s \cdot e \cdot \lceil \frac{n}{b} \rceil \cdot r \cdot h$

7. CONCLUSION

In conclusion, the proposed approach employs a more effective and scalable framework to enhance the detection of both zero-day and known attacks in host systems while overcoming the main limitations of EKC-WCAE. This study addressed a major flaw in conventional clustering-based intrusion detection systems. The developed BAE-SEC-SBHO model is a robust stacking ensemble classifier that employs SBHO, a hybrid optimization methodology, for hyperparameter tuning. The ensemble architecture leverages the complementary strengths of multiple base learners, and the hybrid optimization approach enhances hyperparameter optimization and accelerates model convergence. The model achieves the best zero-day attack detection performance, with a recall (R_{ZD}) of 99.44% and an FNR of 0.55%. The model generates no false alerts and achieves 100% accuracy in detecting known attack classes. The ablation study and the Wilcoxon signed-rank test assess the effect of optimization on zero-day detection by comparing the model with and without optimization. The results demonstrate that the BAE-SEC-SBHO model achieves a superior zero-day attack detection rate, with a statistically significant p-value of 0.002.

Although the results are encouraging, there are a few limitations to keep in mind. The evaluation uses a specific dataset, which may not adequately represent the diversity and variability encountered in real-world network environments. Additionally, this study defines zero-day attacks using a controlled experimental setup in which specific attack classes are excluded during training and introduced only during testing. Although this approach provides a systematic framework for evaluating unseen attack detection, it may not fully reflect real-world scenarios. Future work will focus on broader validation across diverse datasets and on improving adaptability to dynamic environments.

ACKNOWLEDGEMENT

All authors confirm that they have no competing interests related to this work. The authors declare that this study received no financial support from any organization.

AUTHOR CONTRIBUTIONS

Conceptualization, M.K.S. and J.P.; methodology, M.K.S.; software, M.K.S.; validation, M.K.S., J.P., S.C. and S.S.; formal analysis, M.K.S. and S.C.; investigation, M.K.S.; data curation, M.K.S.; writing-original draft preparation, M.K.S.; writing-review and editing, M.K.S., J.P., S.C. and S.S.; visualization, M.K.S. J.P., S.C. and S.S.; supervision, J.P., S.C.; All authors have read and agreed to the published version of the manuscript.

References

- [1] F. A. Sufi, "New Social Media-Driven Cyber Threat Intelligence," *Electronics*, vol. 12, no. 5, p. 1242, 2023.
- [2] K. Birch, "Data Assets," in *Data Enclaves*, Cham, Switzerland: Palgrave Macmillan, 2023, pp. 41–59.
- [3] P. Passeri, "Cyber Attacks Statistics," *HACKMAGEDDON*, Feb. 7, 2025.
- [4] S. Morgan, "Cybercrime to Cost the World \$10.5 Trillion Annually by 2025," *Cybercrime Magazine*, vol. 13, no. 11, 2020.
- [5] S. Caner, N. Erdogmus and Y. M. Erten, "Performance Analysis and Feature Selection for Network-Based Intrusion Detection with Deep Learning," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 30, no. 3, pp. 629–643, 2022.
- [6] Y. Guo, "A Review of Machine Learning-Based Zero-Day Attack Detection: Challenges and Future Directions," *Computer Communications*, vol. 198, pp. 175–185, 2023.
- [7] F. Alhaidari *et al.*, "ZeVigilante: Detecting Zero-Day Malware Using Machine Learning and Sandboxing Analysis Techniques," *Computational Intelligence and Neuroscience*, vol. 2022, Art. no. 1615528, 2022.
- [8] M. Sarhan *et al.*, "From Zero-Shot Machine Learning to Zero-Day Attack Detection," *International Journal of Information Security*, vol. 22, no. 4, pp. 947–959, 2023.
- [9] A. Touré *et al.*, "A Framework for Detecting Zero-Day Exploits in Network Flows," *Computer Networks*, vol. 248, Art. no. 110476, 2024.
- [10] A. Jain, R. Bagoria and P. Arora, "An Intelligent Zero-Day Attack Detection System Using Unsupervised Machine Learning for Enhancing Cybersecurity," *Knowledge-Based Systems*, Art. no. 113833, 2025.
- [11] C. Liu, Z. Gu and J. Wang, "A Hybrid Intrusion Detection System Based on Scalable K-Means+ Random Forest and Deep Learning," in *IEEE Access*, vol. 9, pp. 75729–75740, 2021.
- [12] R. A. Disha and S. Waheed, "Performance Analysis of Machine Learning Models for Intrusion Detection System Using Gini Impurity-Based Weighted Random Forest Feature Selection," *Cybersecurity*, vol. 5, no. 1, Art. no. 1, 2022.
- [13] M. Bakro *et al.*, "An Improved Design for a Cloud Intrusion Detection System Using Hybrid Feature Selection with ML Classifier," *IEEE Access*, vol. 11, pp. 64228–64247, 2023.
- [14] A. Singh *et al.*, "Intrusion Detection System: A Comparative Study of Machine Learning-Based IDS," *Journal of Database Management*, vol. 35, no. 1, pp. 1–25, 2024.
- [15] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cybersecurity Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [16] A. Meshram and C. Haas, "Anomaly Detection in Industrial Networks Using Machine Learning: A Roadmap," in *Machine Learning for Cyber Physical Systems*, Berlin, Germany: Springer, 2016, pp. 65–72.
- [17] Y. Han, Z. Hao, L. Cui, C. Wang and Y. Sang, "A Hybrid Monitoring Mechanism in Virtualized Environments," *2016 IEEE Trustcom/BigDataSE/ISPA*, Tianjin, China, pp. 1038–1045, 2016.
- [18] X. Huang *et al.*, "A Method for Windows Malware Detection Based on Deep Learning," *Journal of Signal Processing Systems*, vol. 93, no. 2, pp. 265–273, 2021.
- [19] S. -Y. Dai, Y. Fyodor, J. -S. Wu, C. -H. Lin, Y. Huang and S. -Y. Kuo, "Holography: A Hardware Virtualization Tool for Malware Analysis," *2009 15th IEEE Pacific Rim International Symposium on Dependable Computing*, Shanghai, China, pp. 263–268, 2009.
- [20] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*. Boca Raton, FL, USA: CRC Press, 2016.
- [21] Z. Amiri *et al.*, "Adventures in Data Analysis: A Systematic Review of Deep Learning Techniques for Pattern Recognition in Cyber-Physical-Social Systems," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 22909–22973, 2024.
- [22] R. Ahmad *et al.*, "Zero-Day Attack Detection: A Systematic Literature Review," *Artificial Intelligence Review*, vol. 56, no. 10, pp. 10733–10811, 2023.
- [23] M. Sewak, S. K. Sahay and H. Rathore, "Deep Reinforcement Learning for Cybersecurity Threat Detection and Protection: A Review," in *Proceedings of the International Conference on Secure Knowledge Management*, 2021, pp. 51–72.
- [24] U. S. Musa, M. Chhabra, A. Ali and M. Kaur, "Intrusion Detection System using Machine Learning Techniques: A Review," *2020 International Conference on Smart Electronics and Communication (ICOSEC)*, Trichy, India, pp. 149–155, 2020.
- [25] F. O. Catak *et al.*, "Deep Learning-Based Sequential Model for Malware Analysis Using Windows EXE API Calls," *PeerJ Computer Science*, vol. 6, Art. no. e285, 2020.
- [26] P. P. Chandran *et al.*, "Optimal Deep Belief Network Enabled Malware Detection and Classification Model," *Intelligent Automation & Soft Computing*, vol. 35, no. 3, pp. 3349–3364, 2023.
- [27] M. Jaihuni *et al.*, "A Novel Recurrent Neural Network Approach in Forecasting Short-Term

Solar Irradiance,” *ISA Transactions*, vol. 121, pp. 63–74, 2022.

- [28] J. Korstanje, “RNNs Using SimpleRNN and GRU,” in *Advanced Forecasting with Python*, Berkeley, CA, USA: Apress, 2025, pp. 255–269.
- [29] L. B. De Amorim, G. D. Cavalcanti, and R. M. Cruz, “The Choice of Scaling Technique Matters for Classification Performance,” *Applied Soft Computing*, vol. 133, Art. no. 109924, 2023.
- [30] G. Manikandan *et al.*, “Classification Models Combined with Boruta Feature Selection for Heart Disease Prediction,” *Informatics in Medicine Unlocked*, vol. 44, Art. no. 101442, 2024.
- [31] A. Legrand, H. Trannois and A. Cournier, “Use of Uncertainty with Autoencoder Neural Networks for Anomaly Detection,” *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, Sardinia, Italy, pp. 32–35, 2019.
- [32] J. Kennedy and R. Eberhart, “Particle swarm optimization,” *Proceedings of ICNN’95 - International Conference on Neural Networks*, Perth, WA, Australia, vol.4, pp. 1942–1948, 1995.
- [33] P. I. Frazier, “Bayesian Optimization,” in *Recent Advances in Optimization and Modeling of Contemporary Problems*, Catonsville, MD, USA: INFORMS, 2018, pp. 255–278.
- [34] N. Moustafa *et al.*, “Federated TON IoT Windows Datasets for Evaluating AI-Based Security Applications,” in *Proceedings of the IEEE TrustCom*, 2020, pp. 848–855.
- [35] Q. Zhang *et al.*, “Boosting Adversarial Attacks with Nadam Optimizer,” *Electronics*, vol. 12, no. 6, Art. no. 1464, 2023.
- [36] H. Hsu and P. A. Lachenbruch, “Paired t-Test,” *Wiley StatsRef: Statistics Reference Online*, 2014.



Manoj Kumar Singh Ph.D. Scholar at the Department of Computer Science and Engineering at the Shri Shankaracharya Technical Campus (SSTC), Bhilai, Chhattisgarh, India. Areas of scientific interests: Cyber-security, Intrusion Detection, Network Security, Artificial Intelligence, Machine Learning. He has published several research papers in reputed conferences and journals.



Jyotiprakash Patra Ph.D. in (Computer Science and Engineering), Associate Professor, Department of Computer Science and Engineering at the Chhattisgarh Swami Vivekanand Technical University (CSVТУ), Bhilai, Chhattisgarh, 490001, India. He holds more than 20 years of experience in research, and teaching and his research area is Artificial Intelligence and Machine Learning, IOT, Cyber-security.

He has published more than 55 papers in SCIE, SCOPUS, Web of Science and UGC CARE listed Journals



Siddharth Choubey Ph.D. in (Computer Science and Engineering), Associate Professor, Department of Computer Science and Engineering at the Shri Shankaracharya Technical Campus (SSTC), Bhilai, Chhattisgarh, 490001, India. He holds more than 20 years of experience in research and teaching, and his research area is artificial intelligence. Areas of scientific interests: Artificial Intelligence and Machine Learning, IOT, Cyber-security. He has published several research papers in SCIE, SCOPUS, Web of Science listed journals.



Sumitra Samal Ph.D. in (Computer Science and Engineering), Assistant Professor, Department of Computer Science and Engineering at the Rungta College of Engineering & Technology (RCET), Bhilai, Chhattisgarh, 490001, India. Areas of scientific interests: Soft Computing, Network Security and AIML. She has published several research papers in reputed conferences and journals.