Baghdad Science Journal

Volume 22 | Issue 9 Article 26

9-23-2025

Proactive SIEM-Based Framework for Cyberattack Monitoring and Classification

Zeyad Safaa Younus
Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq, zeyad.saffawi@uomosul.edu.iq

Mafaz Alanezi
ICT Research Unit, Computer Center, University of Mosul, Mosul, Iraq, mafazmhalanezi@uomosul.edu.iq

How to Cite this Article

Younus, Zeyad Safaa and Alanezi, Mafaz (2025) "Proactive SIEM-Based Framework for Cyberattack Monitoring and Classification," *Baghdad Science Journal*: Vol. 22: Iss. 9, Article 26.
DOI: https://doi.org/10.21123/2411-7986.5067

RESEARCH ARTICLE

Proactive SIEM-Based Framework for Cyberattack Monitoring and Classification

Zeyad Safaa Younus [1,*], **Mafaz Alanezi** [2]

[1] Department of Software, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq
[2] ICT Research Unit, Computer Center, University of Mosul, Mosul, Iraq

ABSTRACT

Cybersecurity is a crucial aspect of modern information technology, encompassing safeguarding computer systems, network resources, and information from cyberattacks. Wazuh is an open-source security information and event management (SIEM) technology for real-time attack detection but faces challenges like log analysis complexity and alert overloading, which raises the number of false-positive alerts and affects accuracy. This paper presents three hierarchical models for real-time attack prediction using Wazuh to gather real-time dataset from various network endpoints and then preprocess them using a variety of tools. Subsequently, mutual-information, principal component analysis (PCA), and independent component analysis (ICA) are utilized for feature reduction and selection to extract important features for each model individually. Finally, for training and testing purposes, the types of attacks are classified using the long-short-term memory (LSTM) method. In the second stage, real-time attacks are utilized to assess the performance of the suggested models for real-time attack detection. The experimental findings demonstrate that the suggested approaches performed better on binary and multiclass classification in terms of accuracy, recall, precision, and F-measures and were superior to previous methods in terms of accuracy.

Keywords: Cyberattack, Cybersecurity, ICA, LSTM, Mutual information, PCA, Wazuh SIEM

Introduction

In the age of big data, large volumes of information are produced every day by numerous applications and devices. Furthermore, network intrusion has progressively increased, leading to the theft of private information and becoming the primary attack vector that harms both individuals and companies.[1–4] The proliferation of linked devices has led to a steady evolution in network attacks, and the heavy dependence of government, commercial, and military entities on the internet for daily activities has made cyber threats a major concern.[5]

One of the most essential parts of security infrastructure meant to identify, prevent, and address cyberattacks is SIEM systems. SIEM systems are made up of various components that work together to monitor and assess network infrastructure. These components include source devices, log collection, normalization, rule and correlation engines, log storage, and monitoring processes.[6,7] Fig. 1 shows the basic components of the SIEM.

Organizations utilize open-source SIEM solutions to assess certain features and reduce software license costs while also offering basic functionality for small businesses in logging and analyzing security incidents like Wazuh, Apache, Prelude, and AlienVault.[7] Wazuh was selected as the best SIEM solution of 2023 because it has some advantages over other open-source SIEM solutions, such as scalability, centralized management of security events, and response capability.[8,9]

Wazuh SIEM is a free and open-source platform for security monitoring and threat detection that helps

Received 25 March 2024; revised 7 June 2024; accepted 9 June 2024.
Available online 23 September 2025

* Corresponding author.
E-mail addresses: zeyad.saffawi@uomosul.edu.iq (Z. S. Younus), mafazmhalanezi@uomosul.edu.iq (M. Alanezi).

https://doi.org/10.21123/2411-7986.5067

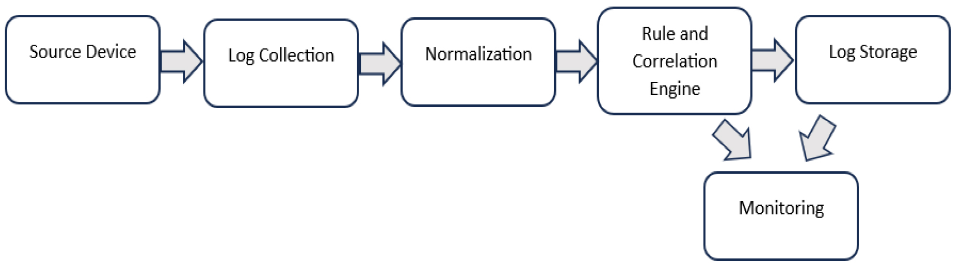


Fig. 1. The basic components of SIEM.[7]

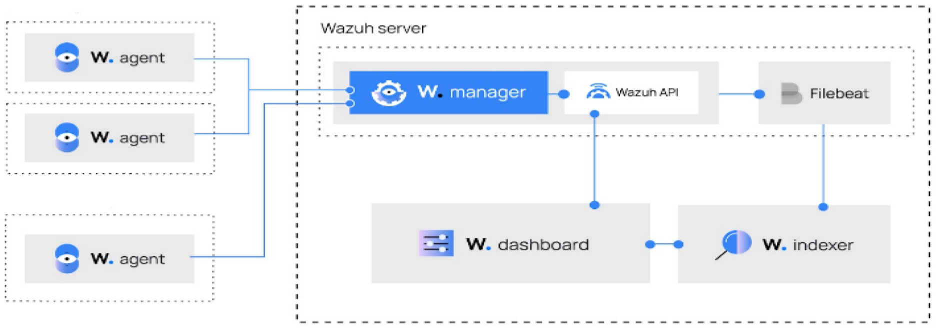


Fig. 2. Wazuh components.[9]

businesses collect, examine, and react to security events.[10] In order to improve overall security posture and operational efficiency, it provides log collection, threat detection, incident response, dashboard visualization, file integrity monitoring, regularity compliance, reporting, and integration. The overall security posture and operational efficiency are improved by integration with other security tools and technologies. Rulesets are used by Wazuh to identify malware, intrusions, and attacks. It serves as a central location for keeping an eye on network devices in order to spot attacks, trigger alerts when a threat materializes, and gather and store logs and alerts within the server.[8] The three primary parts of Wazuh SIEM are the Wazuh indexer, Wazuh dashboard, and Wazuh server, which gather and process the data collected by the agents. A single global Wazuh agent is deployed on the endpoints on the network under monitoring to collect logs and send them to the Wazuh server for analysis. The complexity of log analysis and the potential for alert overloading, which raises the number of false positive alarms, are flaws in Wazuh's technique. The Wazuh core components are shown in Fig. 2.[9]

Feature selection is a process used to find the best subset of features to infer the same meaning as the entire feature set. Choosing relevant features and removing ineffective ones with little to no performance loss is the goal of feature selection.[11,12] Many machine-learning models, including feature selection and classification techniques like support vector machine (SVM) and K-means clustering, have been suggested and developed in recent years to detect cyberattacks. However, the model's performance deteriorates due to the training time being too long in large training sets and too sensitive to irrelevant attributes. Furthermore, even if the network is small, a huge number of characteristics are collected from these models from the raw data for analysis.[13] Therefore, in order to choose the significant features, methods like PCI and ICA for feature extraction and mutual information for feature selection must be used. Furthermore, because so much data is fed into these algorithms, they are not able to identify threats in real-time. This has led to a rise in the requirement for deep learning-based models. Artificial neural networks (ANN) are modified for feature extraction, perception, and learning in deep learning.[14] These days, a wide range of industries use deep learning applications, including bioinformatics, image processing, speech recognition, and more. Convolutional neural networks (CNN), recurrent neural networks (RNN), and LSTM are a few of the deep learning techniques used by different neural network models.[2] In this paper, three models have been suggested for the binary and multiclass classification of attacks. Mutual information and LSTM are used in the first model, whereas PCA and LSTM and ICA and LSTM are utilized in the second and third models. The real-world dataset utilized to assess these models is gathered via

Wazuh following a series of standard procedures and a range of attacks on network-connected endpoint devices. The raw dataset is then preprocessed and saved in CSV (Comma-Separated Values)[15] format before being fed into the models for training. Subsequently, real-time prediction employing various forms of attacks is employed to assess the performance of the models in real-time. The structure of this paper contains six primary sections. The introduction is covered in Section 1, and the related works are covered in Section 2. The methodology is presented in Section 3. Section 4 introduces validation metrics. Section 5 presents the outcomes of the experiment. Section 6 presents the conclusion and future directions.

Related works

Numerous studies have been carried out utilizing diverse algorithms to identify breaches and intrusions into computer networks. Since manual feature extraction is a requirement of standard classification algorithms, techniques of deep learning have demonstrated their efficiency in solving this kind of issue. As a result, scientists concentrated on developing robust and efficient models for attack detection through the use of deep learning.

Ding et al. proposed a method that used minimum redundancy maximum relevance (MRMR) feature selection algorithms in order to extract the most pertinent features for categorizing anomalies. SVM and LSTM algorithms were then applied for data classification. Next, the accuracy and F-score of the SVM and LSTM algorithms are compared.[16]

Primartha and Tama proposed a method that used random forest and 10-fold cross-validation technique using NSL-KDD, UNSW-NB15, and GPRS datasets.[17] Using back-propagation and stochastic gradient descent techniques, Al-Zewairi et al. presented a deep learning model using ANN. The method was assessed for the network intrusion detection system (NIDS) as a binary classifier on the UNSW-NB15 dataset. Each layer of the ten neurons in the deep learning model has five hidden layers, and the 10-fold cross validation method is applied.[18] Maniath et al. introduced a technique that employed LSTM to enhance automated analysis of a huge volume of malware samples and detect ransomware activity for binary sequence classification of API calls.[19]

Wu and Guo proposed the use of a LuNet model, which is an RNN and CNN employed on the NSL-KDD and UNSW-NB15 datasets, to find intrusions on a large-scale network.[20] An LSTM-based deep learning framework was proposed by Hawing et al. to implement classification at the packet level in intrusion detection system (IDS). Rather than scrutinizing the complete flow, like a document, the suggested approach took into account each individual packet. From each packet that was taken into consideration, the method subsequently created the key sentence. Following that, word embedding was used to extract the sentence's syntactic and semantic properties.[21]

The IDS method was introduced by Kasongo and Sun. It integrates five classification methods, which are logistic regression (LR), k-nearest neighbors (KNN), ANN, decision tree (DT), and SVM, with the feature selection approach of the extreme gradient boosting (XGBoost) methods for binary and multi-class classification used in UNSW-NB15 Dataset.[22] A technique proposed by Fredj et al. employed LSTM, RNN, and multilayer perceptron (MLP)-based models with great care to forecast the kind of assault that would occur. Good results were obtained when this model was evaluated using the CTF dataset.[23]

Awan et al. presented a technique that addressed the prediction of application layer distributed denial of service (DDoS) attacks in real-time and improved the model's performance by reducing the prediction time. The technique used two machine learning methods, random forest (RF) and MLP, to detect DDoS attacks.[24] Moualla et al. presented a method that used the synthetic minority oversampling (SMOTE) technique to address the issue of uneven class distribution in the dataset and subsequently employed the highly randomized tree classifier to identify the crucial features for every class included in the dataset based on the Gini impurity criterion.[25] Kumar et al. suggested a unified intrusion detection system (UIDS) using the UNSW-NB15 dataset. The suggested model's ruleset was derived from several DT models, including the info gain feature selection technique and k-means clustering. Additionally, the model was trained using a variety of techniques, such as SVM and ANN.[26] Abbas and Almhanna suggested a method using an encoding methodology to transform the initial nominal packets into numerical features for detecting DDOS network attacks. Next, a logarithmic process is used to standardize the data. To minimize the dataset's dimensionality, the PCA technique is finally used eight times for a variety of features. Subsequently, the RF method was employed to extract patterns from the data, while the naive bayes (NB) algorithm was utilized to categorize the data and compare the classification results with the classifier output, RF.[27]

Gaur and Kumar suggested an LSTM model to classify binary and multiclass assaults using an LSTM model on the CICDDoS 2019 dataset.[28] Ibrahim et al. introduced a framework to identify botnet assaults using data standardization and ICA to improve the

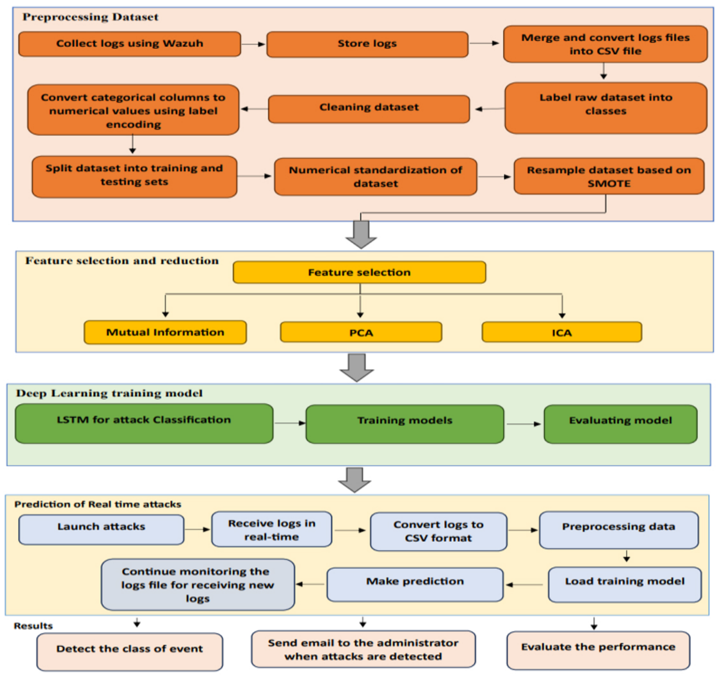

Preprocessing Dataset
Collect logs using Wazuh
Store logs
Merge and convert logs files into CSV file
Label raw dataset into classes
Cleaning dataset
Convert categorical columns to numerical values using label encoding
Split dataset into training and testing sets
Numerical standardization of dataset
Resample dataset based on SMOTE
Feature selection and reduction
Feature selection
Mutual Information
PCA
ICA
Deep Learning training model
LSTM for attack Classification
Training models
Evaluating model
Prediction of Real time attacks
Launch attacks
Receive logs in real-time
Convert logs to CSV format
Preprocessing data
Load training model
Make prediction
Continue monitoring the logs file for receiving new logs
Results
Detect the class of event
Send email to the administrator when attacks are detected
Evaluate the performance

Fig. 3. The flowchart structure of the suggested models.

quality of data before organization.[29] IDS was proposed by Shushlevska et al. using the UNSW-NB15 dataset. Nine different class attacks have been tested and trained on the dataset. The UNSW-NB15 dataset was effectively divided into network traffic from regular records and attack logs using many machine learning methods. The analysis of the models for each strategy indicates that the classification using RF is more effective than using NB, LR, and DT.[30]

Wang et al. presented a technique that made use of the genetic algorithm (GA) and KNN algorithm. In order to lower the dimensionality of attack-type detection, the procedure first uses a GA for feature selection. The KNN algorithm is then used for attack-type matching. This technique improves overall security by greatly increasing the attack detection rate of NIDS across several classes.[31] Mohamed and Rohaim presented a technique for detecting web threats that made use of four classification algorithms: bidirectional LSTM, LSTM, RNN, and CNN.[32]

A technique for categorizing DDoS attacks utilizing the LSTM method was presented by Bashaiwth et al. to clarify different kinds of DDoS attack forecasts. For the purpose of classifying attacks, 51 inherent features were found, and common explanations were derived for each class.[33] Almarshdi et al. presented an IDS method based on deep learning techniques for LSTM and CNN using the UNSW-NB15 dataset to distinguish between assaults and normal network traffic.[34] Sayegh et al. presented a technique that distinguished between possible attacks and normal network traffic using LSTM. Furthermore, imbalanced data difficulties were addressed with the SMOTE approach, which helped to reliably identify unusual intrusion patterns.[35]

Methodology

The techniques utilized to construct the models' architecture for training and prediction are described in this section. The flowchart structure of the suggested models is shown in Fig. 3.

Real world dataset

In its most basic form, a dataset is a table with information about a feature from each column and an example summary from each row about a domain. The primary goal of this section is to generate a relevant dataset, which is subsequently used to predict and categorize attacks using the suggested deep learning models. The tools used to gather and prepare the dataset are described here. In order to assess the effectiveness of suggested models and to make the real dataset easier to use as training model input, it is first obtained from Wazuh, translated to CSV format, and divided into train and test sets.

Table 1. Daily events on the network.

Day	Type of event
Day 1	Normal
Day 2	Normal, DoS attacks
Day 3	DoS attacks
Day 4	DoS attacks
Day 5	SQL injection, web attacks
Day 6	Brute force
Day 7	Normal, File modification
Day 8	Reconnaissance
Day 9	Anomalies, web attacks
Day 10	Normal, Port scan

Dataset description

This paper is based on a real-world dataset gathered by Wazuh,[9] which is used to collect various logs of attacks and normal events. The dataset satisfies every requirement of real-world cyberattacks. The dataset, which is organized into 10 different files, comprises normal and abnormal (various forms of attack) network traffic from 10 consecutive days of capture. As seen in Table 1, various kinds of attacks were used every day. Our study is based on a single file created by merging all 10 of the datasets into one large file and converting them to CSV format.

Collect a dataset using Wazuh

A vast number of logs and alarms from all kinds of network endpoint devices are gathered by Wazuh and stored on the server. Real-time monitoring is made possible by Wazuh, an open-source, centralized log management system that employs rules, threat intelligence, and behavior-based analysis to find abnormalities and threats.[6] The predefined compliance rulesets assist firms in meeting regulatory obligations, and the user-friendly interface offers insights into security events and risks. The overall security posture and operational efficiency are improved through integration with other security tools and technologies, such as Snort and Suricata.[36] During this phase, Wazuh is utilized to gather logs and alerts of network events, both normal and anomalous. In this case, Wazuh and NIDS security technologies like Suricata and Snort are combined to enhance Wazuh's ability to identify threats in real-time related to some of the assaults employed in this study.[9] A variety of other attacks are also used against network devices, including port scans, structured query language (SQL) injections, brute forces, Web attacks, reconnaissance, anomalies, and file modification. The denial of service (DoS) attacks are also carried out using tools such as Goldeneye, low orbit ion cannon (LOIC), high orbit ion cannon (HOIC), and Hping3. One of the flaws is that Wazuh launches a high volume of logs, which causes it to struggle with the intricacy of log analysis and the potential for alert overload, which raises the number of false positive alerts and lowers accuracy rates. The procedures for setting up Wazuh to monitor the network and gather logs are shown in Fig. 4.

Converting a dataset to CSV format

Initially, the Wazuh logs are merged and converted into a single CSV file format in order to simplify the dataset's training process. To enable different data analysis and visualization activities, this conversion is required in order to analyze or view data using tabular data tools. The procedures followed in this process are outlined below:
Input: log files.
Output: CSV file.
Step 1: import python libraries.
Step 2: read log files.
Step 3: merge and convert log files to a CSV file.
Step 4: CSV file as output.

The result is a CSV file called ZSUM-1-2024 dataset that represents the raw dataset, which includes several categorical and numerical data types like float, integer, boolean, and object. It takes some analysis and value visualization to work with the real-time raw CSV dataset. Overfitting issues could arise from duplicate rows. Certain columns include values that are irrelevant to machine learning and deep learning algorithms, such as nulls, spaces, or other kinds of data. Because of this, the dataset needs to be preprocessed before being included in the training model in order to ensure that it is error-free and won't interfere with the postprocessing procedure, as the following sections will demonstrate.

Labeled dataset

The lack of labeled datasets presents additional difficulties when working with real-world data. Data that has been given one or more labels to provide context or meaning is referred to as labeled data. These labels are frequently used in artificial intelligence and deep learning as a target for the approach to prediction. For the purpose of performing training in the suggested models, the dataset is classified into multiple classes here, including normal and attacks in their various forms. At first, the raw data is labeled according to the contents of the record within the columns. After that, the label column is mapped to the numerical values of two classes: 0 and 1 for binary classification and within the range of 0 to 8 for multiclass classification. The steps used for this process are explained below:

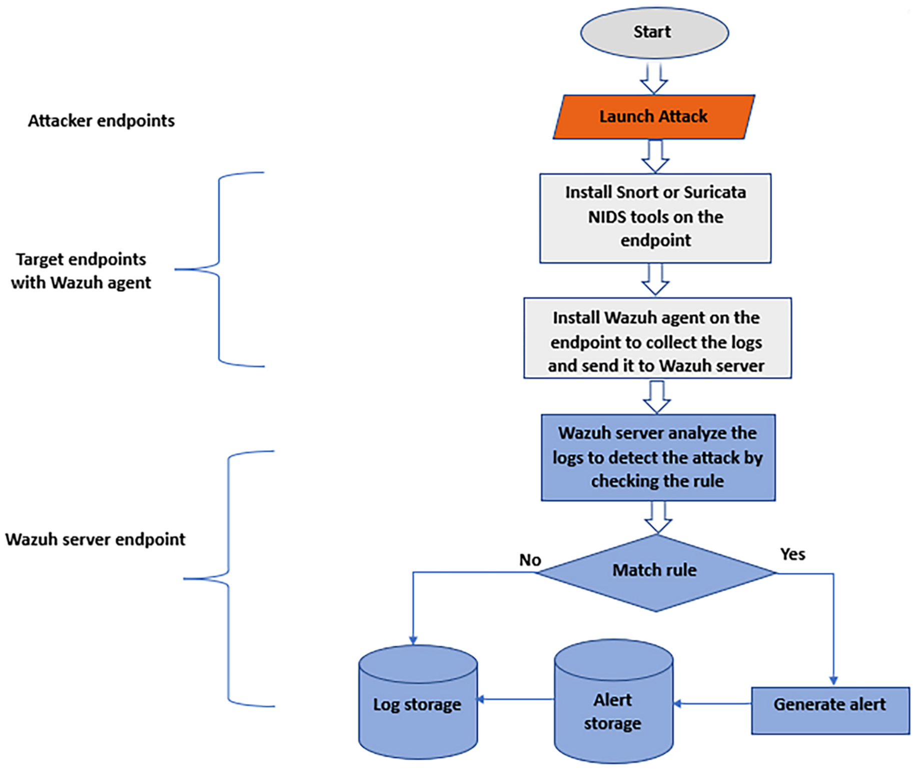

Attacker endpoints
Target endpoints with Wazuh agent
Wazuh server endpoint
Start
Launch Attack
Install Snort or Suricata NIDS tools on the endpoint
Install Wazuh agent on the endpoint to collect the logs and send it to Wazuh server
Wazuh server analyze the logs to detect the attack by checking the rule
No
Match rule
Yes
Log storage
Alert storage
Generate alert

Fig. 4. Flowchart of the Network Environment with Wazuh SIEM.

Input: a CSV file.
Output: a CSV file with a label.
Step 1: read the CSV file.
Step 2: label the dataset according to the contents of the records within the columns.
Step 3: map the label column in the dataset to numerical values.
Step 4: output a CSV file with a label column.

Within the raw dataset, Fig. 5 for binary classification and Fig. 6 for multiclass classification examine how instances are distributed throughout various classes. The number of instances connected to each class is clearly displayed in the bar chart. An overview of the dataset's composition is provided by the y-axis, which quantifies the corresponding number of instances, while the x-axis depicts the attack classes. In order to evaluate class imbalances or biases and make judgments for the following steps, which include data preprocessing and model building, a graphical depiction is essential.

The dataset is then cleaned by removing columns that have null values based on the threshold value, and the null records are then filled according to each column's data type. Next, the label column is split from the features of the dataset and used as the target. Since machine and deep learning models require numerical input values, the categorical columns are later transformed into a numerical format using a label encoder. In machine-learning methods, the preprocessing phase is an important.[35] After that, the dataset is split into training and testing sets to conduct the training process and validate the performance of the model. Next, standardization[37] is a crucial tool to standardize the input dataset's range of functionality. It is primarily used as a preprocessing step before deep learning models are run. In this case, it is employed to scale features such that the standard deviation is one and the mean of the observed values is zero.

Resampling dataset

Imbalanced refers to the classification dataset that has classes with imbalanced proportions. The majority classes are those that comprise the largest proportion of the dataset. The minority classes are those that comprise a lesser percentage.[38] According to numerous data scientists, imbalanced classification occurs when there is an uneven distribution of classes in the training dataset, which negatively impacts the performance of AI-based classifiers. This is particularly evident when the ratio of majority to minority classes is greater than 100 to 1. Preprocessing of the current dataset should start with reducing the

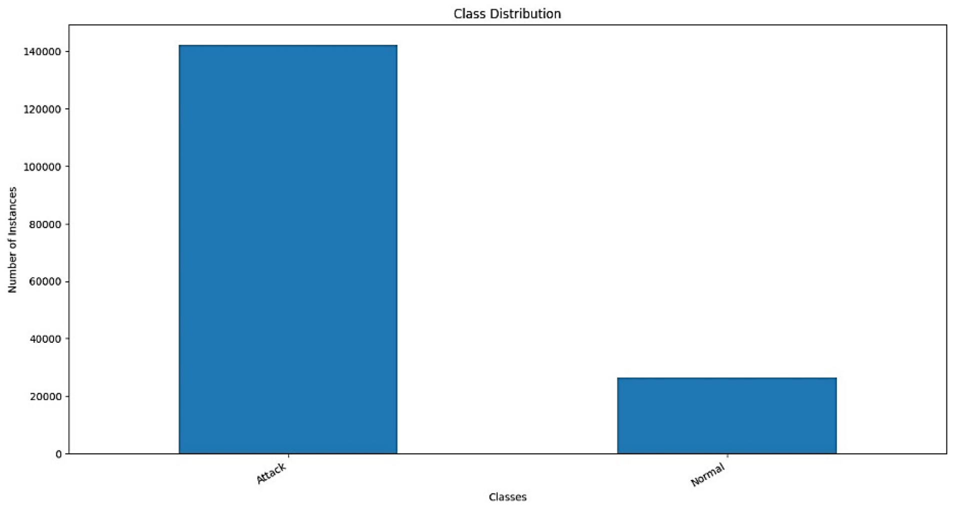


Fig. 5. Binary class distribution of the raw dataset.

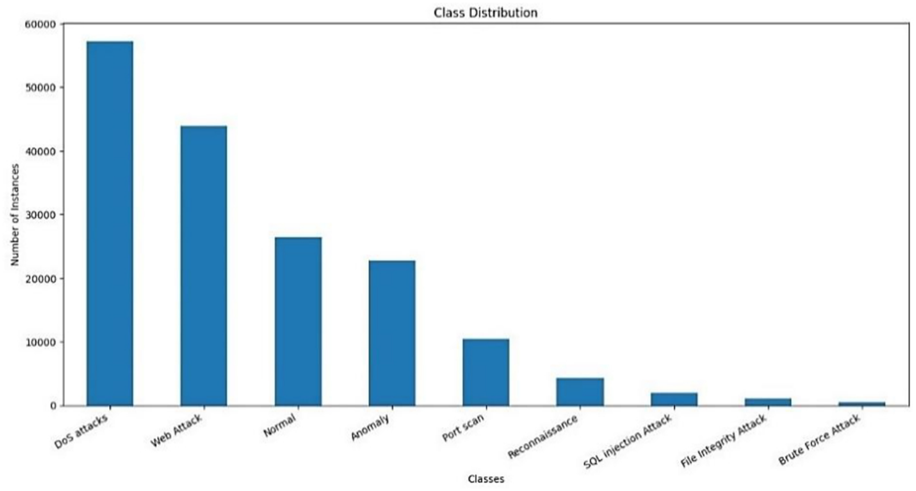


Fig. 6. Multiclass distribution of the raw dataset.

majority, increasing the minority, or doing both using resampling techniques, due to the challenge of producing a typical balanced dataset.[39] Resampling techniques were utilized to adjust the proportions between the majority and minority classes and create a more balanced new dataset. Undersampling and oversampling are the two methods of resampling. Oversampling is generally favored over undersampling techniques because undersampling methods often eliminate instances from the data that can carry significant information. Resampling allows distinct classes to have about equal influence on the classification model's results by improving the balance of the training data. This paper uses the SMOTE method, one of the most straightforward statistical methods for handling irregular classes in a dataset. SMOTE is an oversampling approach in which the majority classes are left unchanged while synthetic samples are created for the dataset's minority class. This algorithm aids in resolving the random oversampling-induced overfitting issue. It employs interpolation between positively aligned instances to create new instances by concentrating on the feature space. By constructing lines between the minority cases (shown in red) in the 2D feature space and letting synthetic minority instances live in those lines, Fig. 7 explains how the SMOTE approach oversamples the minority cases (green color). Furthermore, the vast majority of situations (shown in green) don't alter. As a result, the classes equaled out, and the minority population only rose.

Feature selection

One of the key techniques in data preprocessing for attack classification systems is feature selection. It has

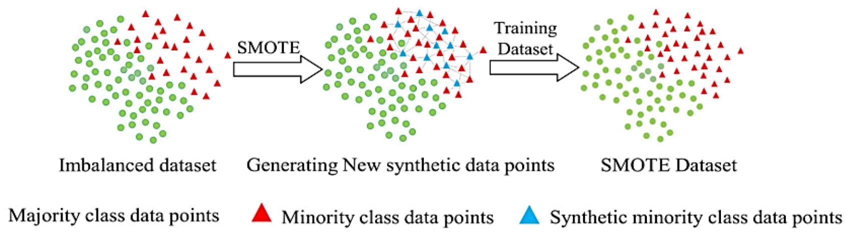

SMOTE
Training Dataset
Imbalanced dataset
Generating New synthetic data points
SMOTE Dataset
Majority class data points
Minority class data points
Synthetic minority class data points

Fig. 7. SMOTE work principle.[40]

many dimensions in the feature space, particularly in a real-time setting, which might significantly affect how well deep learning algorithms perform.[41,42] It removes unnecessary, irrelevant, or noisy data, reduces the number of attributes, and chooses the key features. Consequently, several dimensionality reduction and feature selection techniques are utilized in the original dataset in order to minimize the number of input features.[43] This allows the algorithm of deep learning to select the most relevant features and make the high-dimensional feature space easier to comprehend, allowing it to train more quickly and minimize model complexity and computing costs.[44] The performance of the suggested models is enhanced in this paper by reducing the dataset dimensionality through the application of PCA and ICA and selecting important features using mutual information techniques.

Principal component analysis

PCA is a technique for dimensionality reduction and data analysis. It is employed to pinpoint the most important patterns and simplify high-dimensional data while maintaining crucial details.[45–47] After preprocessing the dataset, PCA retrieves the eigenvectors and eigenvalues from the covariance matrix (CM). The relationships between the variables are shown in the covariance matrix. The covariance matrix is a square matrix, with each element denoting the covariance between two variables, as explained in Eq. (1).[48]

$$Cv_{jk} = \frac{\sum_{i=1}^{n} \left(X_{ij} - \bar{X}_j\right) \left(X_{ik} - \bar{X}_k\right)}{n-1} \tag{1}$$

Where n is the number of data points, X is individual data points, and $\bar{X}$ is the means of variables X as explained in Eq. (2).[48]

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n} \tag{2}$$

After that, the eigenvalues are sorted in descending order, and k eigenvectors are selected to match each k eigenvalue, where k is the new feature subspace's number of dimensions. The selected k eigenvectors are then used to build the projection matrix W via PCA. Lastly, as shown in Eq. (3), it creates a k-dimensional feature subspace by converting the original dataset, X, via W[48]:

$$Transformed\ Data = X.W \tag{3}$$

Mutual information

In order to measure the relevance or similarity of variables, mutual information is frequently utilized in feature selection, clustering, dimensionality reduction, and other machine learning tasks. Regardless of the kind of relationship (linear or nonlinear) between variables, it offers a means to quantify the quantity of information communicated between them.[49]

A measure of the mutual dependency between two random variables is called mutual information. Each characteristic is given a score, and the amount of information gleaned about one random variable through the other random variable is measured. In other words, its measure is the extent to which knowledge of one variable lowers uncertainty about the other. Greater mutual information scores show a higher degree of reliance between the variables; lower mutual information scores indicate a slight reduction; and zero mutual information scores indicate complete independence between the variables.[50] Eq. (4) is used to calculate the mutual information between two discrete random variables.[51]

$$I(X;Y) = \sum_{x,y} P_{XY}(x,y) \log\left(\frac{P_{XY}(x,y)}{P_X(x) P_Y(y)}\right)$$
$$= E_{Pxy \log \frac{P_{XY}}{P_X\ P_Y}} \tag{4}$$

Where, PX(x) and PY(y) are the marginal functions: $P_X(x) = \sum_{y} P_{XY}(x,y)$, and $P_X(x,y)$ is the joint probability mass function.

Independent component analysis

In unsupervised machine learning, the independent non-Gaussian components of a multivariate signal are extracted using the statistical and computational method known as independent component analysis (ICA). Finding a linear transformation for the data that gets the transformed data as close to statistical independence as possible is the aim of ICA.[29] It is a well-liked technique for choosing crucial network properties like feature extraction, noise reduction, density estimation, and regression. The data pattern for each feature is improved and maximized by using it to separate the noise.[52] In this case, ICA spins the data until each axis has a non-Gaussian appearance. The algorithm rotates the data in any direction by setting the mean to zero and normalizing the variance in all directions. The whitening process is the procedure used to normalize the variation. The decorrelation to guarantee that every feature is handled equally prior to the ICA algorithm running is known as the whitening process. The data are subjected to the ICA method after the centering process (mean equal to zero) and the whitening process (normalization of variance).[53] Finding W's unmixing vector is the primary objective of ICA, where W is A's inverse. While A is a mixing matrix, which represents each of the h mixed signals, $X_1(k)$, $X_2(k)$, …, $X_h(k)$, as a linear combination of the q independent components, and S represents the source signal, $S_1(k)$, $S_2(k)$, …, and $S_h(k)$.[54] Eq. (5) is used to calculate the value of input data X.

$$X = A.S \tag{5}$$

Now, using X, calculate A and S. Two statistical presumptions that ICA effectively achieves are as follows: 1) Each component follows a non-Gaussian distribution; 2) the components are independent of one another. Eq. (6) is used to calculate the value of S.

$$S = A^{-1}.X = W.X \tag{6}$$

The idea is to maximize the non-Gaussian distribution of the components by choosing a suitable W to apply to X. An iterative process can be used to accomplish this. The independent components are the vectors along which the statistics of projections of the n-dimensional data vectors [X(Eq. (1)), X(Eq. (2)),…, X(N)] are independent of one another. The method for determining a specific W is typically used to express the independent component analysis technique using Eq. (7).

$$y = Wx \tag{7}$$

In a way that makes each element in y independent of the others. If one such W can be found, the resultant marginal densities become a scaled permutation of the original density functions if the individual marginal distributions are non-Gaussian.

Training and classification

Classification is a supervised learning technique that divides the dataset into categories by creating models of supervised learning specifically for this job. Thus, in order to develop models that associate feature subsets with each class, a labeled dataset is necessary. When there are only two classes in the dataset, the classification method is called binary; if there are more than two classes, it is called multiclass; if each element is distinct with several labels, it is called multilabel.[33] In this study, the models for classification and prediction on the dataset are trained using a deep learning method called LSTM.

Long short-term memory

Recurrent neural network (RNN) architecture, known as LSTM, was first developed in 1997 in response to the vanishing gradient issue that sometimes arises while training conventional RNNs on long sequences.[55,56] Because LSTMs include a memory mechanism that enables them to selectively remember or forget information over time, they can learn long-term dependencies in sequential data.[5] The cell state is the main element of LSTM. The gates, which use the sigmoid function to safeguard it, are used to add or delete information from the cell state (a value of one indicates that the alteration is allowed, whereas a value of zero indicates that it is denied). The three gates that make up an LSTM are:

The forget gate is used to manage information transfer from the previous memory cell to the current memory cell. By employing a sigmoid function, it enables the LSTM to selectively forget or remember data from earlier time steps (one indicates it keeps it, 0 indicates it deletes it). The forget gate value is computed using Eq. (8):

$$f_t = \sigma\left(w_f.\left[h_{t-1},\ x_t\right] + b_f\right) \tag{8}$$

In this case, σ is the sigmoid function,$w_f$is the forget gate's weight matrix, $h_{t-1},\ x_t$is the concatenation of the previous hidden state and the current input, the bias vector for the forget gate is represented by b_f, and $f_t$is the vector of forget gate values for the current time step.

The input gate is used to manage information that enters the memory cell from the current input and the previously hidden state. Using a sigmoid function, the input gate layer first determines which data will be updated. A tanh layer then suggests a new vector to be added to the cell state. The LSTM then modifies the cell state by adding the new vector values and forgetting the data it had chosen to ignore. Eq. (9) and Eq. (10) are used to compute the input gate value and the candidate cell state.

$$i_t = \sigma\left(w_i.\left[h_{t-1}, x_t\right] + b_i\right) \tag{9}$$

The sigmoid function in this instance is denoted by σ, $w_i$is the weight matrix for the input gate, the concatenation of the previous hidden state and the current input is represented by $h_{t-1},\ x_t$, $b_i$is the bias vector for the input gate, and $i_t$is the vector of input gate values for the current time step.

$$\hat{C}_t = tanh\left(w_c.\left[h_{t-1,}x_t\right] + b_c\right) \tag{10}$$

where, $\hat{C}_t$ is the candidate cell state vector for the current time step and is created by combining the current input and the previous hidden state linearly with the tanh activation function. The input gate's bias vector is denoted by b_i. Eq. (11) explains the update of the cell state:

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{11}$$

By using a sigmoid function, the output gate regulates the information flow from the memory cell to the current hidden state and output. Only the information it chooses to send to the next neuron is sent after passing over a tanh layer (value between − 1 and 1). Eqs. (12) and (13), respectively, are used to determine the output gate value and hidden state.

$$o_t = \sigma\left(w_o.\left[h_{t-1}, x_t\right] + b_o\right) \tag{12}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{13}$$

Where, σ represents the sigmoid function, $w_o$is the weight matrix of the output gate's, the combination of the previous hidden state and the current input is denoted by $h_{t-1},\ x_t$, $b_o$is the bias vector for the output gate, o_t, is the vector of output gate values at the current time step, and h_t is the current hidden state formed by applying the tanh activation function to the current cell state and multiplying its elementwise with the output gate values.

In order to address the issue of disappearing gradients and carry out nonlinear computations in deep learning, the LSTM model is constructed with two LSTM layers, both utilizing the rectified linear unit (ReLU) activation function. During the classification phase, the fully linked dense layer at the top of the LSTM network determines the probability for every class. After adding a dense layer and ReLU activation, layers with progressively fewer units are added until an output layer with the number of units equal to the number of classes in the dataset is reached. The model gains non-linearity with each dense layer, which improves the model's ability to recognize intricate patterns. The dropout layers and batch normalization were taken into consideration for regularization. The model is optimized for classification tasks and is constructed using the Adam optimizer and the categorical cross-entropy loss function.

Real time prediction

Real-time attacks are utilized to assess how well the suggested models identify and categorize attacks once the training models have been constructed. Here, different attack types are used on target endpoints on the network, such as port scans, SQL injection attacks, brute force, Web attacks, reconnaissance, anomalies, and file modification, as well as DoS attacks using various tools like Goldeneye, LOIC, HOIC, and Hping3. Wazuh is set up to monitor the network endpoints, initiate alerts when an attack is identified, and store the logs on the Wazuh server. Wazuh's tendency to issue a lot of alerts has the disadvantage of making log analysis more complex and raising the number of false-positive alerts. Because of this, the suggested models are employed to continuously monitor the Wazuh server endpoint's log file. Upon receiving a new log from Wazuh, the log is converted into CSV and preprocessed using steps similar to those used in the training model, and a prediction is made based on the class label in the training models to identify which attack class the log belongs to and display it on the dashboard. The following are the steps in the real-time prediction process:

```
Input: real-time logs.
Output: predict the class of the event and send an email to admin.
Step 1: monitor the network endpoints using Wazuh.
Step 2: monitor the log file within the Wazuh server.
Step 3: if new logs are received by Wazuh,
    – Convert the log to CSV format.
    – Preprocessing the data is similar to steps in the training process.
    – Make a prediction for the class of events, print the result on the dashboard, and send an email to inform the admin when an attack has occurred.
Step 4: return to step 2 for continued monitoring.
```

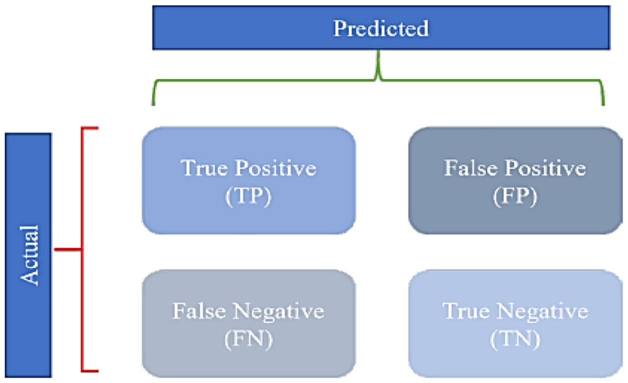


Fig. 8. Confusion matrix.[57]

Validation metrics

When assessing the efficacy of deep learning models, it is imperative to utilize suitable performance indicators for evaluation. For evaluating how well the model performs in various categorization outcomes, the confusion matrix is used, which is a metric based on the connection between real classes and the classes predicted by the model.[57] As illustrated in Fig. 8, the x-axis shows the expected results, and the y-axis shows the actual findings.

When an attack is correctly classified by the model as an attack, this is known as a true positive (TP). When the model correctly identifies a normal access as normal, it is called a true negative (TN). A false positive (FP) occurs when the model interprets a legitimate access as an attack. When an attack is classified by the model as regular access, it is called a false negative (FN).[58] The suggested model often aims to lower the FP and FN rates, as it is aware that the latter can have detrimental effects on information systems.

The model's performance may be assessed based on the confusion matrix's definition to comprehend the actual and expected classification outcomes, including accuracy, precision, recall, and F1-score, which are computed using Eqs. (14) to (17), respectively.[59–61]

- To facilitate a more thorough assessment, the Classification Report offers a thorough summary of each class's precision, recall, F1-score, and support.[62]
- Accuracy refers to the percentage of correct predictions made across all classification results.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{14}$$

- Precision measures the percentage of real classifications that accurately predict a single classification outcome.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{15}$$

- Recall measures the percentage of real single-classification outcomes that match the predictions in a correct manner.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{16}$$

- The F1-score is a metric of the harmonic average of precision and recall. It is utilized to evaluate the overall model's performance.

$$\text{F} - \text{score} = \frac{2 \ \times TP}{2 \ \times TP + FP + FN} \tag{17}$$

- The number of true results that are observed in a class is known as support. It shows the proportion of real outcomes in which the expected and actual outcomes are the same.
- A key tool for measuring the difference between expected class probabilities and actual labels is the cross-entropy loss. The categorical cross-entropy loss skillfully negotiates complications in multi-class contexts. In real-world applications, minimizing this loss during model training improves discriminative abilities and optimizes parameters for precise predictions across a variety of classes. Minimizing the loss is the goal; a lesser loss indicates a stronger model.

Results and discussion

The experimental results obtained when executing the suggested models are explained in this section.

Performance analysis of suggested models for multiclass classification

This section explains the multiclass classification and prediction outcomes attained by applying the suggested models. The capacity of the suggested models to make accurate predictions is demonstrated by their accuracy of 99.32 percent for the model that uses LSTM and mutual information, 99.26 percent for the model that uses LSTM and PCA, and 99.22 percent for the model based on LSTM and ICA, which indicates that these models are useful in real-world applications. Furthermore, a thorough analysis of the classification report reveals a positive trend in the model's performance across many classes. The model performs well across all classes, as evidenced by the values of the f1-score metrics, precision, and recall

Table 2. The results of implementing the suggested models for multiclass classification.

	Performance metrics								
			Macro avg			Weighted avg			
Approach	Accuracy	Loss	Precision	Recall	F1-score	Precision	Recall	F1-score	Time spent
LSTM-mutual information	99.32	0.0366	0.97	0.99	0.98	0.99	0.99	0.99	8044.431
LSTM-PCA	99.26	0.0423	0.97	0.99	0.98	0.99	0.99	0.99	9117.108
LSTM-ICA	99.22	0.0428	0.97	0.99	0.98	0.99	0.99	0.99	8804.721

for all classes (0 to 8). The classification results for all models show a 99 percent accuracy rate for all models over 33,720 cases, demonstrating the great prediction accuracy of the suggested models. With an average precision of 97 percent, an average recall of 99 percent, and an average f1-score of 98 percent for all models, the macro average, which takes into account the average performance across all classes, shows good overall precision, recall, and f1-score metrics. Similarly, with precision, recall, and f1-score metrics all at 99 percent, the weighted average, which takes class imbalances into consideration, supports the robustness of the model's performance. All of these findings highlight how well the suggested models handle a variety of cases from various classes. The difference between the actual class labels and the expected class probabilities during training is measured by the loss. A reduced loss value typically shows better alignment between predicted and true values, indicating that the model is convergent toward an ideal solution. Here, the loss values of 0.0366 for the model that uses LSTM and mutual information, 0.0423 for the model that uses LSTM and PCA, and 0.0428 for the model based on LSTM and ICA show that the model has learned to generate predictions that almost correspond to the ground truth labels for the provided dataset. It has also been successful in reducing errors during training, which has contributed to the high accuracy recorded. Table 2 explains the multiclass classification results via the suggested models that are based on a similar real dataset and requirements.

The results demonstrate that LSTM-mutual information outperforms other models in terms of accuracy and loss because it depends on selecting the most informative features.

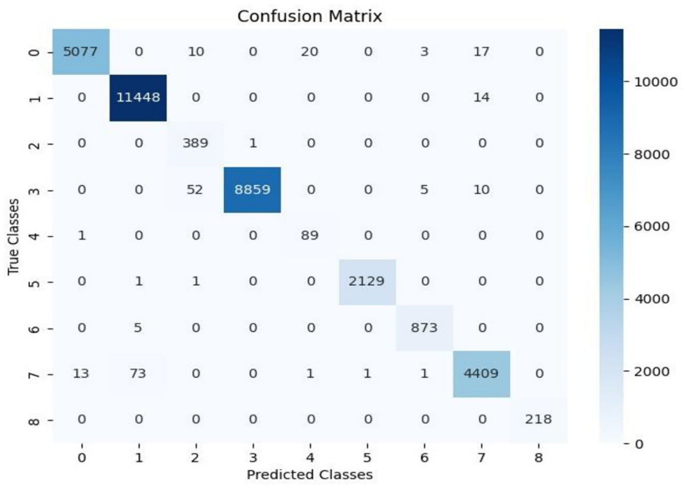
The confusion matrix displays the counts of true positive, true negative, false positive, and false negative predictions for each class, giving comprehensive information on a model's classification performance. Every row in the confusion matrix represents the actual class, and every column represents the predicted class. The number of cases that are correctly classified is indicated by the diagonal elements, which reflect the true positive counts for each class. On the other hand, misclassifications are represented by off-diagonal elements, where each item shows the number of cases in which the true class differed from the predicted class.

The confusion matrix findings for a model that uses LSTM and mutual information are illustrated in Fig. 9. In the first row, 5077 instances of class 0 were correctly identified by the model; however, 10 instances of class 2, 20 instances of class 4, 3 instances of class 6, and 17 instances of class 7 were incorrectly classified by the model. Similarly, in the second row as belonging to class 1, the model misclassified 14 instances as class 7, while correctly classifying 11448 instances, and so on for the remaining classes. Comprehending and examining the confusion matrix facilitates a more detailed assessment of the model's advantages and disadvantages, helping to pinpoint certain areas that require enhancement or optimization in subsequent versions.

Fig. 10 illustrates the confusion matrix findings for a model that uses LSTM and PCA. The model accurately identified 5103 instances of class 0 in the first row but incorrectly classified 5 instances of class 2, 2 instances of class 4, 2 instances of class 6, 14 instances of class 7, and 1 instance of class 8. In the next row, the model incorrectly identified 36 instances as class 7, accurately identifying 11426 instances as part of class 1, and so forth for the other classes.

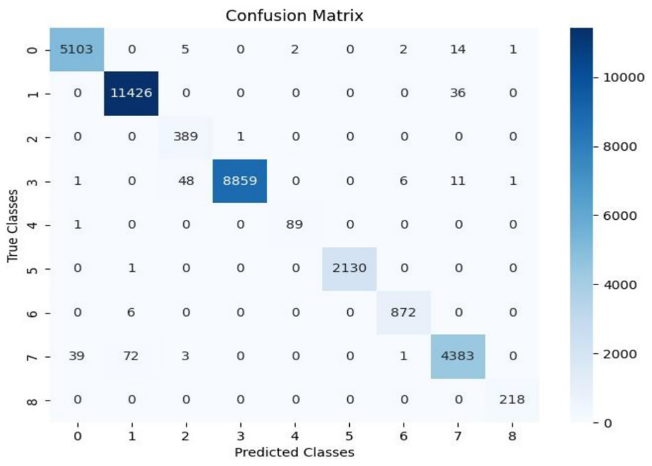
Fig. 11 illustrates the confusion matrix findings for a model that uses LSTM and ICA. The model accurately identified 5104 instances of class 0 in the top row but incorrectly classified 2 instances of class 2, 3 instances of class 4, 3 instances of class 6, 14 instances of class 7, and 1 instance of class 8. In the second row, the model incorrectly labeled 34 cases as class 7, accurately identifying 11428 cases as part of class 1, and similarly for the other classes.

Training and validation loss trends over epochs for the suggested models is shown in Fig. 12. Training epochs are displayed on the x-axis, and loss is represented on the y-axis. This visualization offers information for improving parameters, identifying overfitting or under fitting issues, and assessing the model's generalization and performance.


Confusion Matrix
True Classes
Predicted Classes
10000
8000
6000
4000
2000
0

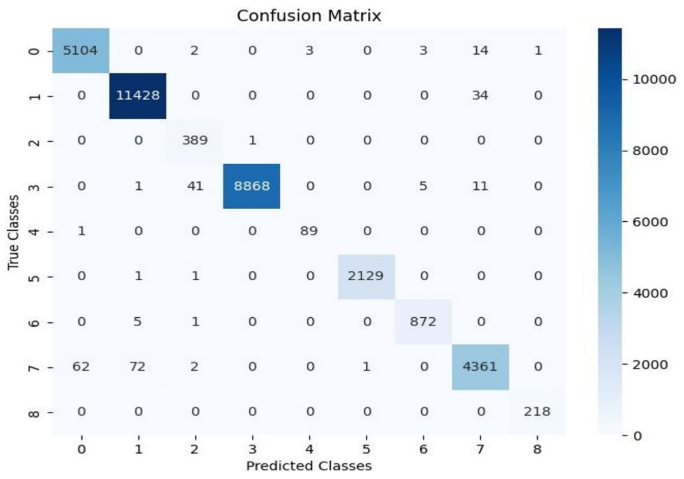
	0	1	2	3	4	5	6	7	8
0	5077	0	10	0	20	0	3	17	0
1	0	11448	0	0	0	0	0	14	0
2	0	0	389	1	0	0	0	0	0
3	0	0	52	8859	0	0	5	10	0
4	1	0	0	0	89	0	0	0	0
5	0	1	1	0	0	2129	0	0	0
6	0	5	0	0	0	0	873	0	0
7	13	73	0	0	1	1	1	4409	0
8	0	0	0	0	0	0	0	0	218

Fig. 9. The result of the confusion matrix for LSTM-Mutual information.


Confusion Matrix
True Classes
Predicted Classes
10000
8000
6000
4000
2000
0

	0	1	2	3	4	5	6	7	8
0	5103	0	5	0	2	0	2	14	1
1	0	11426	0	0	0	0	0	36	0
2	0	0	389	1	0	0	0	0	0
3	1	0	48	8859	0	0	6	11	1
4	1	0	0	0	89	0	0	0	0
5	0	1	0	0	0	2130	0	0	0
6	0	6	0	0	0	0	872	0	0
7	39	72	3	0	0	0	1	4383	0
8	0	0	0	0	0	0	0	0	218

Fig. 10. The result of the confusion matrix for LSTM-PCA.


Confusion Matrix
True Classes
Predicted Classes
10000
8000
6000
4000
2000
0

	0	1	2	3	4	5	6	7	8
0	5104	0	2	0	3	0	3	14	1
1	0	11428	0	0	0	0	0	34	0
2	0	0	389	1	0	0	0	0	0
3	0	1	41	8868	0	0	5	11	0
4	1	0	0	0	89	0	0	0	0
5	0	1	1	0	0	2129	0	0	0
6	0	5	1	0	0	0	872	0	0
7	62	72	2	0	0	1	0	4361	0
8	0	0	0	0	0	0	0	0	218

Fig. 11. The result of the confusion matrix for LSTM-ICA.

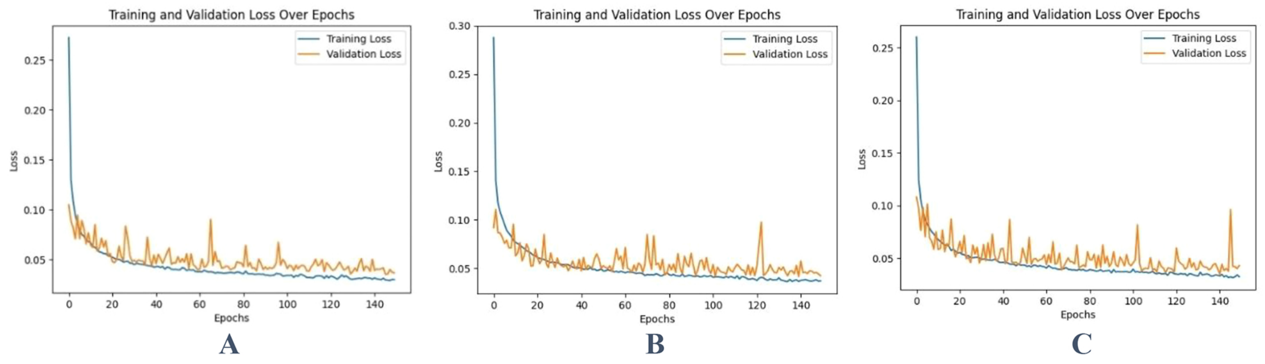


Fig. 12. Training and validation loss: A) LSTM-Mutual Information, B) LSTM-PCA, C) LSTM-ICA.

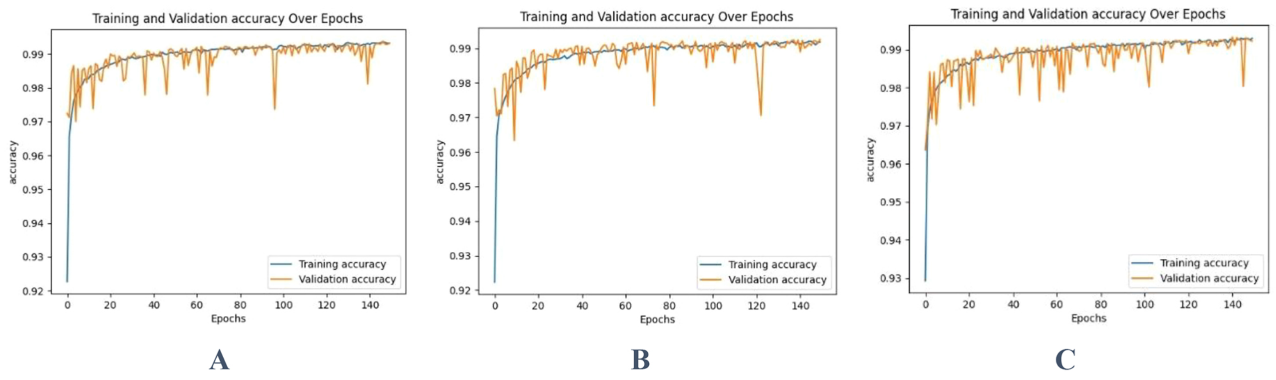


Fig. 13. Training and validation accuracy: A) LSTM-Mutual Information, B) LSTM-PCA, C) LSTM-ICA.

The trained and validated accuracy over epochs is shown in Fig. 13, offering insights for parameter adjustment and identifying overfitting or under fitting issues, with epochs on the x-axis and accuracy on the y-axis.

The outcomes demonstrate that the performance of suggested models is enhanced by reducing the training and testing losses to their lowest possible values, and accuracy has increased to the highest possible value.

Performance analysis of suggested models for binary classification

This section explains the results obtained from implementing the suggested models for binary classification. Here, the values of accuracy are 99.67, 99.65, and 99.62, while the values of loss are 0.0097, 0.0099, and 0.0112 for suggested models that used LSTM and mutual information, LSTM and PCA, and LSTM and ICA, respectively. In addition, the macro average values for average precision, average recall, and average f1-measure are 0.99, 1.00, and 0.99 for the suggested models, respectively. Also, the weighted average values for average precision, average recall, and average f1-measure are 1.00 for all measures. Table 3 shows the results of implementing the binary classification using the suggested models when the dataset is labeled into two classes, which are normal and attack.

From the table, the results show the efficiency of the performance of the suggested models in binary classification in terms of accuracy, precision, recall, F1-score, and loss, with a slight superiority of the

Table 3. The results of implementing the suggested models for binary classification.

	Performance metrics								
			Macro avg			Weighted avg			
Methods	Accuracy	Loss	Precision	Recall	F1-score	Precision	Recall	F1-score	Time spent
LSTM-mutual information	99.67	0.0097	0.99	1.00	0.99	1.00	1.00	1.00	4913.395
LSTM-PCA	99.65	0.0099	0.99	1.00	0.99	1.00	1.00	1.00	4971.116
LSTM-ICA	99.62	0.0112	0.99	1.00	0.99	1.00	1.00	1.00	6062.356

Table 4. Comparison with previous approaches using UNSW-NB15.

Ref.	Methods	Accuracy
Ref. [20]	CNN and RNN	97.40
Ref. [22]	LR, KNN, ANN, DT, and SVM	94.21
Ref. [26]	DT, K-means, SVM and ANN	89.76
Ref. [30]	RF	95.9
Ref. [34]	CNN and LSTM	92.10
Suggested model	LSTM-mutual information	99.65
Suggested model	LSTM-PCA	99.61
Suggested model	LSTM-ICA	99.47

model that uses LSTM and mutual information in terms of accuracy and loss.

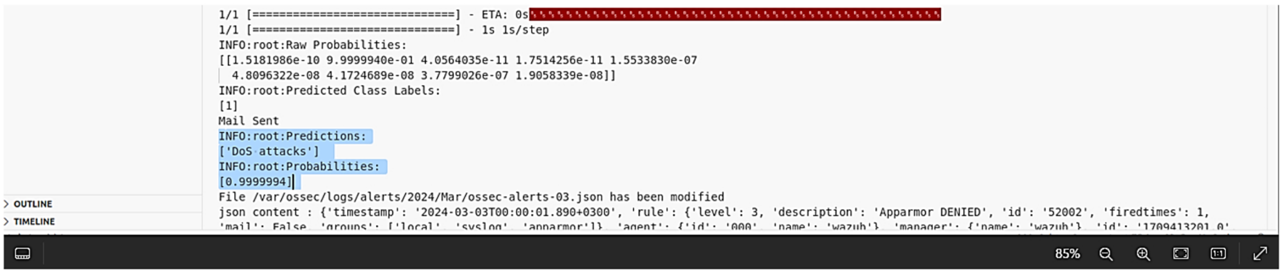
Benchmark with previous approaches

Direct comparison with pertinent literature based on real-world datasets is not possible, particularly when dealing with multi-class categorization due to the use of different types of attacks according to the requirements of each model. UNSW-NB15 is a recent dataset developed by the Australian Centre of Cyber Security (ACCS) to assess IDS methods. It includes 2,540,047 records from the IXIA PerfectStorm4 testing platform, categorized into various types of attacks.[63] The dataset consists of 12 percent of attack information packets and 88 percent of normal information packets, and it has been used by many researchers to evaluate their methods. For that reason, a comparison using the UNSW-NB15 dataset is made according to the accuracy obtained from the suggested models with the accuracy obtained from the previous methods that used different machine and deep learning methods, which have been explained in the related works section. The results explain that the suggested models outperform the previous methods in terms of accuracy, as indicated in Table 4.

Performance analysis in real-time

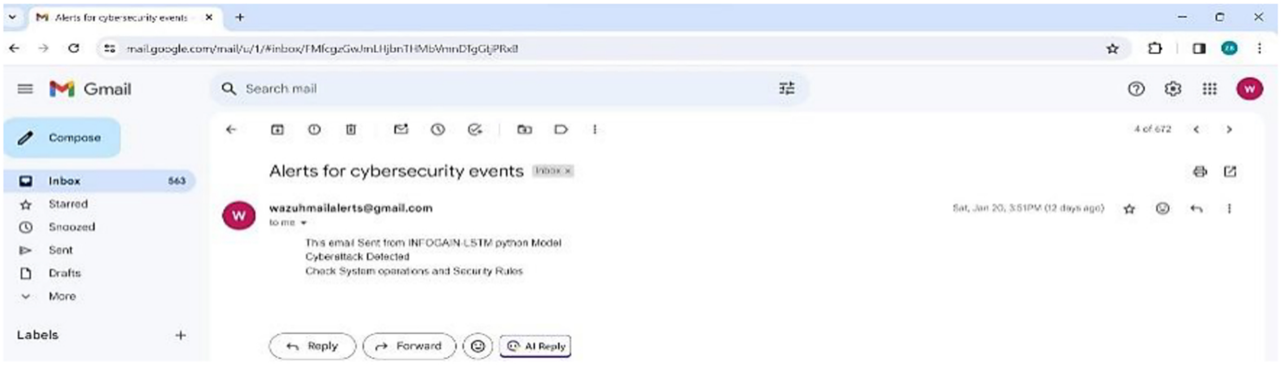
To assess the suggested models' ability to identify attacks in real-time. Numerous attacks have been employed against the network's target endpoints. Fig. 14 clarifies how the suggested model performs in the event of a DoS attack against an endpoint and shows the model's ability to accurately identify and categorize the attack.

The probability values for each class are displayed in the above figure. The type of event is then identified based on the highest probability value, which is a DoS attack with a value of 0.9999. Furthermore, as illustrated in Fig. 15 below, the model notifies the system administrator by email that the threat has occurred and that he needs to take the necessary precautions.



```
1/1 [==============================] - ETA: 0s
1/1 [==============================] - 1s 1s/step
INFO:root:Raw Probabilities:
[[1.5181986e-10 9.9999940e-01 4.0564035e-11 1.7514256e-11 1.5533830e-07
  4.8096322e-08 4.1724689e-08 3.7799026e-07 1.9058339e-08]]
INFO:root:Predicted Class Labels:
[1]
Mail Sent
INFO:root:Predictions:
['DoS attacks']
INFO:root:Probabilities:
[0.9999994]
File /var/ossec/logs/alerts/2024/Mar/ossec-alerts-03.json has been modified
json content : {'timestamp': '2024-03-03T00:00:01.890+0300', 'rule': {'level': 3, 'description': 'Apparmor DENIED', 'id': '52002', 'firedtimes': 1,
'mail': False, 'groups': ['local', 'syslog', 'apparmor']}, 'agent': {'id': '000', 'name': 'wazuh'}, 'manager': {'name': 'wazuh'}, 'id': '1709413201.0'
```

Fig. 14. Detects SQL injection attack in real-time.



Alerts for cybersecurity events

mail.google.com/mail/u/1/#inbox/FMfcgzGwJmLHjbnTHMbVmnDTgGtjPRxB

Alerts for cybersecurity events Inbox

wazuhmailalerts@gmail.com
to me

Sat, Jan 20, 3:51PM (12 days ago)

This email Sent from INFOGAIN-LSTM python Model
Cyberattack Detected
Check System operations and Security Rules

Fig. 15. Notify admin using email.

Conclusion

As cybersecurity models develop, they make use of big data and deep learning techniques to improve defense capabilities by accurately classifying attacks and detecting threats. This paper presents three models where logs are gathered from monitored endpoints and stored using Wazuh SIEM. After that, these logs are combined, preprocessed, and converted into CSV format to be suitable for deep learning techniques. PCA, ICA, and mutual information are utilized in feature reduction and selection. Finally, training and classification are accomplished using LSTM. Subsequently, several kinds of attacks are employed to assess the suggested models' real-time performance in identifying and categorizing the attacks. The limitations of this paper are the high resource requirements of Wazuh SIEM and Python, including memory, disk space, and CPU. Furthermore, the real-world dataset requires preprocessing and labeling due to its diverse range of data types. In summary, the outcomes attained through implementing the suggested models demonstrate the ability to identify attacks, as the LSTM and mutual information, LSTM and PCA, and LSTM and ICA models outperform the previous methods in terms of accuracy. In future work, other methods for feature selection or hyperparameter optimization can be used to boost LSTM-based models' efficacy in cybersecurity. This study adds insightful information to the continuing attempts to strengthen cyber protection mechanisms using deep learning techniques.

Acknowledgement

The authors are grateful to the Department of Computer Science, College of Computer Science and Mathematics, and Computer Center at the University of Mosul, Mosul, Iraq, for supporting the scientific research.

Authors' declaration

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are ours. Furthermore, any Figures and images that are not ours have been included with the necessary permission for republication, which is attached to the manuscript.
- No animal studies are present in the manuscript.
- No human studies are present in the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee at University of Mosul.

Authors' contribution statement

Z. S. Y. contributed in designing the methodology, practical programming, and writing and reviewing the paper. M. A. supervised this paper.*–

References

1. Akashdeep, Manzoor I, Kumar N. A Feature Reduced Intrusion Detection System Using ANN Classifier. Expert Syst Appl. 2017;88:249–257. https://doi.org/10.1016/j.eswa.2017.07.005.
2. Zhang J, Zhang X, Liu Z, Fu F, Jiao Y, Xu F. A Network Intrusion Detection Model Based on BiLSTM with Multi-Head Attention Mechanism. Eletronics. 2023;12(19):1–17. https://doi.org/10.3390/electronics12194170.
3. Xue L, Zhu T. Hybrid Resampling and Weighted Majority Voting for Multi-Class Anomaly Detection on Imbalanced Malware and Network Traffic Data. Eng Appl Artif Intell. 2024;128:1–19. https://doi.org/10.1016/j.engappai.2023.107568.
4. Kamil M, Mohammed I. Deep Learning Model for Intrusion Detection System Utilizing Convolution Neural Network. Open Eng. J. 2023;13(1):1–11. https://doi.org/10.1515/eng-2022-0403.
5. Laghrissi F, Douzi S, Douzi K, Hssina B. Intrusion Detection Systems Using Long Short-Term Memory (LSTM). J Big Data. 2021;8(65):1–12. https://doi.org/10.1186/s40537-021-00448-4.
6. Younus ZS, Alanezi M. A Survey on Network Security Monitoring: Tools and Functionalities. Mustansiriyah J Pure Appl Sci. 2023;1(2):55–86. https://doi.org/10.47831/mjpas.v1i2.33.
7. Miller D, Harris S, Harper A, Van Dyke S, Blask C. Security Information and Event Management (SIEM) Implementation. 1st ed. Mc Graw Hill: New York N, USA, editor: Mc Graw Hill: New York, NY, USA; 2010:496 p. https://doi.org/10.1109/NTMS.2016.7792462.
8. Şimşek A, Koltuksuz A. Detection of Advanced Persistent Threats Using Siem Rulesets. Int J Print Technol Digit Ind. 2023;7(3):471–477. https://doi.org/10.46519/ij3dptdi.1353341.
9. Younus ZS, Alanezi M. Detect and Mitigate Cyberattacks Using SIEM. 2023 16th International Conference on Developments in eSystems Engineering (DeSE). Istanbul, Turkiye: IEEE. 2023:510–515. https://doi.org/10.1109/DeSE60595.2023.10469387.
10. Ilca L, Lucian O, Balan T. Enhancing Cyber-Resilience for Small and Medium-Sized Organizations with Prescriptive Malware Analysis, Detection and Response. Sensors. 2023;23(15):1–33. https://doi.org/10.3390/s23156757.
11. Umar M, Chen Z, Liu Y. A Hybrid Intrusion Detection with Decision Tree for Feature Selection in Information and Security. Inf Secur: Int J. 2021;49:1–20. https://doi.org/10.11610/isij.490.
12. Mahmood R, Abdi A, Hussin M. Performance Evaluation of Intrusion Detection System using Selected Features and Machine Learning Classifiers. Baghdad Sci J. 2021;18(2):884–898. http://doi.org/10.21123/bsj.2021.18.2(Suppl.).0884.
13. Momand A, Jan S, Ramzan N. A Systematic and Comprehensive Survey of Recent Advances in Intrusion Detection Systems Using Machine Learning: Deep Learning, Datasets, and Attack Taxonomy. Sensors. 2023;2023:1–18. https://doi.org/10.1155/2023/6048087.

14. Amanuel S, Ahmed I. A Review of the Various Machine Learning Algorithms for Cloud Computing. 2022 4th International Conference on Advanced Science and Engineering (ICOASE); Zakho, Iraq: IEEE. 2022:124–129. https://doi.org/10.1109/ICOASE56293.2022.10075592.
15. Al-azazi F, Ghurab M. ANN-LSTM: A deep learning model for early student performance prediction in MOOC. Heliyon. 2023;9(4):1–16. https://doi.org/10.1016/j.heliyon.2023.e15382.
16. Ding Q, Li Z, Batta P, Trajković L. Detecting BGP Anomalies Using Machine Learning Techniques. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Budapest: IEEE. 2016:3352–3355. https://doi.org/10.1109/SMC.2016.7844751.
17. Primartha R, Tama B. Anomaly Detection Using Random Forest: A Performance Revisited. 2017 International Conference on Data and Software Engineering (ICoDSE). Palembang Sumatra Selatan, Indonesia: IEEE. 2017:1–6. https://doi.org/10.1109/ICODSE.2017.8285847.
18. Al-Zewairi M, Almajali S, Awajan A. Experimental Evaluation of a Multi-layer Feed-Forward Artificial Neural Network Classifier for Network Intrusion Detection System. 2017 International Conference on New Trends in Computing Sciences (ICTCS). Amman, Jordan: IEEE. 2017:167–172. https://doi.org/10.1109/ICTCS.2017.29.
19. Maniath S, Ashok A, Poornachandran P, G. Sujadevi V, Sankar P, Jan S. Deep learning LSTM based ransomware detection. 2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE). Noida, India: IEEE. 2017:442–446. https://doi.org/10.1109/RDCAPE.2017.8358312.
20. Wu P, Guo H. LuNET: A Deep Neural Network for Network Intrusion Detection. 2019 IEEE symposium series on computational intelligence (SSCI). Xiamen, China IEEE. 2019:617–624. https://doi.org/10.1109/SSCI44817.2019.9003126.
21. Hwang R, Peng M, Nguyen V, Chang Y. An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level. Appl Sci. 2019;9(16):1–14. https://doi.org/10.3390/app9163414.
22. Kasongo S, Sun Y. Performance Analysis of Intrusion Detection Systems Using A Feature Selection Method on the UNSW-NB15 dataset. J Big Data. 2020;7(105):1–20. https://doi.org/10.1186/s40537-020-00379-6.
23. Fredj O, Mihoub A, Krichen M, Cheikhrouhou O, Derhab A. CyberSecurity Attack Prediction: A Deep Learning Approach. 13th International Conference on Security of Information and Networks. ACM. 2020:1–6. https://doi.org/10.1145/3433174.3433614.
24. Awan M, Farooq U, Babar H, Yasin A, Nobanee H, Hussain M, *et al.* Real-Time DDoS Attack Detection System Using Big Data Approach. Sustainability. 2021;13(10743):1–19. https://doi.org/10.3390/su131910743.
25. Moualla S, Khorzom K, Jafar A. Improving the Performance of Machine Learning-Based Network Intrusion Detection Systems on the UNSW-NB15 Dataset. Comput Intell Neurosci. 2021;2021:1-13. https://doi.org/10.1155/2021/5557577.
26. Kumar V, Das A, Sinha D. UIDS: A Unified Intrusion Detection System for IOT Environment. Evolut Intel. 2021;14(1):47–59. https://doi.org/10.1007/s12065-019-00291-w.
27. Abbas S, Almhanna M. Distributed Denial of Service Attacks Detection System by Machine Learning Based on Dimensionality Reduction. International Conference of Modern Applications on Information and Communication Technology (ICMAICT). University of Babylon, Babylon-Hilla City, Iraq: J Phys - IOP. 2021. https://doi.org/10.1088/1742-6596/1804/1/012136.
28. Gaur V, Kumar R. M-LSTM: Multiclass Long Short-Term Memory based Approach for Detection of DDoS Attacks. Math Stat Eng Appl. 2022;71(3s2):1375–1394.
29. Ibrahim W, Anuar M, Selamat A, Krejcar O. Botnet Detection Using Independent Component Analysis. IIUM Eng J. 2022;23(1):95–115. https://doi.org/10.31436/iiumej.v23i1.1789.
30. Shushlevska M, Efnusheva D, Jakimovski G, Todorov Z. Anomaly Detection with Various Machine Learning Classification Techniques over UNSW-NB15 dataset. 10th International Conference on Applied Innovations in IT (ICAIIT). 2022.
31. Wang Y, Liu Z, Zheng W, Wang J, Shi H, Gu M. Combined Multi-Classification Network Intrusion Detection System Based on Feature Selection and Neural Network Improvement. Appl Sci. 2023;13(14):1–16. https://doi.org/10.3390/app13148307.
32. Mohamed S, Rohaim M. Multi-Class Intrusion Detection System using Deep Learning. J Al-Azhar Univ Eng Sect. 2023;18(19):869–883. https://doi.org/10.21608/auej.2023.213003.1375.
33. Bashaiwth A, Binsalleeh H, AsSadhan B. An Explanation of the LSTM Model Used for DDoS Attacks Classification. Appl Sci. 2023;13(15):1–30. https://doi.org/10.3390/app13158820.
34. Almarshdi R, Nassef L, Fadel E, Alowidi N. Hybrid Deep Learning Based Attack Detection for Imbalanced Data Classification. Intell Autom Soft Comput. 2023;35(1):297–320. https://doi.org/10.32604/iasc.2023.026799.
35. Sayegh H, Dong W, Al-madani A. Enhanced Intrusion Detection with LSTM-Based Model, Feature Selection, and SMOTE for Imbalanced Data. Appl Sci. 2024;14(2):1–20. https://doi.org/10.3390/app14020479.
36. Hussein M, Hamza E. Secure Mechanism Applied to Big Data for IIoT by Using Security Event and Information Management System (SIEM). Int J Intell Eng Syst. 2022;15(6):667–681. https://doi.org/10.22266/ijies2022.1231.59.
37. Karataş G. The Effects of Normalization and Standardization an Internet of Things Attack Detection. Eur J Sci Tech. 2021;29:187–192. https://doi.org/10.31590/ejosat.1017427.
38. Kudithipudi S, Narisetty N, Kancherla G, Bobba B. Evaluating the Efficacy of Resampling Techniques in Addressing Class Imbalance for Network Intrusion Detection Systems Using Support Vector Machines. Ing Sys Info. 2023;28(5):1229–1236. https://doi.org/10.18280/isi.280511.
39. Bagui S, Li K. Resampling Imbalanced Data for Network Intrusion Detection Datasets. J Big Data. 2021;8(6):1–41. https://doi.org/10.1186/s40537-020-00390-x.
40. Aldraimli M, Soria D, Parkinson J, Thomas E, Bell J, Dwek M, *et al.* Machine Learning Prediction of Susceptibility to Visceral Fat Associated Diseases. Health Technol. 2020;10:925–944. https://doi.org/10.1007/s12553-020-00446-1.
41. Chandrashekar G, Sahin F. A survey on Feature Selection Methods. Comput Electr Eng. 2014;40(1):16–28. https://doi.org/10.1016/j.compeleceng.2013.11.024.
42. Alnaish Z, Algamal Z. Improving Binary Crow Search Algorithm for Feature Selection. J Intell Syst. 2023;32(1):1–11. https://doi.org/10.1515/jisys-2022-0228.
43. Zhou L, Zhu Y, Zong T, Xiang Y. A Feature Selection-based Method for DDoS Attack Flow Classification. Future Gener Comput Syst. 2022;132:67–79. https://doi.org/10.1016/j.future.2022.02.006.
44. Thakkar A, Lohiya R. Attack Classification Using Feature Selection Techniques: A Comparative Study. J Ambient Intell Human Comput. 2021;12:1249–1266. https://doi.org/10.1007/s12652-020-02167-9.

45. Gewers F, Ferreira G, De Arruda H, Silva F, Comin C, Amancio D, *et al*. Principal Component Analysis: A Natural Approach to Data Exploration. ACM Comput Surv. 2021;54(4):1–34. https://doi.org/10.1145/3447755.
46. Zhang R, Du T, Qu S. A Principal Component Analysis Algorithm Based on Dimension Reduction Window. IEEE Access. 2018;6:63737-63747. https://doi.org/10.1109/ACCESS.2018.2875270.
47. Saleh H, Rahma A, Shaker S. Proposed Methods of Image Recognition depend on the PCA. Baghdad Sci J. 2010;7(1): 146–161. https://doi.org/10.21123/bsj.2010.7.1.146-161.
48. Ayogu B, Adewoyin O, Folorunsho O, Daramola C. Principal Component Analysis-Based Feature Selection Approach for Network Intrusion Detection. FUOYE J Pure Appl Sci. 2019;4(1):129–135.
49. Zhou H, Wang X, Zhu R. Feature Selection based on Mutual Information with Correlation Coefficient. Appl Intell. 2022;52:5457–5474. https://doi.org/10.1007/s10489-021-02524-x.
50. Pawening R, Darmawan T, Bintana R, Arifin A, Herumurti D. Feature Selection Methods Based on Mutual Information for Classifying Heterogeneous Features. J Ilmu Komp Info. 2016;9(2):106–112. https://doi.org/10.21609/jiki.v9i2.384.
51. Barraza N, Moro S, Ferreyra M, Peña A. Mutual Information and Sensitivity Analysis for Feature Selection in Customer Targeting: A Comparative Study. J Inf Sci. 2019;45(1):53–67. https://doi.org/10.1177/0165551518770967.
52. Dagupta D, Gonzalez F. An Immunity-Based Technique to Characterize Intrusions in Computer Networks. IEEE Trans Evol Comput. 2002;6(3):281–291. https://doi.org/10.1109/TEVC.2002.1011541.
53. Cheng W, Zhang Z, Zhu G, He Z. Noise Source Identification and Localization of Mechanical Systems based on an Enhanced Independent Component Analysis. J Vib Control. 2014; 22(4):1–15. https://doi.org/10.1177/1077546314539370.
54. Srinoy S, Chimphlee W, Chimphlee S. A Fusion of ICA and SVM for Detection Computer Attacks. 5th WSEAS International Conference on Applied Computer Science. Hangzhou, China. 2006:987–991.
55. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1–32.
56. Noureen, Hazlin S, Ali Z. Sentiment Analysis on Roman Urdu Students' Feedback Using Enhanced Word Embedding Technique. Baghdad Sci J. 2024;21(2(SI)):725–739. https://doi.org/10.21123/bsj.2024.9822.
57. Chu H, Lin Y. Improving the IoT Attack Classification Mechanism with Data Augmentation for Generative Adversarial Networks. Appl Sci. 2023;13(23):1–22. https://doi.org/10.3390/app132312592.
58. Dhanya K, Vajipayajula S, Srinivasan K, Tibrewal A, Kumar TS, Kumar TG. Detection of Network Attacks using Machine Learning and Deep Learning Models. Proc Comput Sci. 2023;218:57–66. https://doi.org/10.1016/j.procs.2022.12.401.
59. Guerrero-Higueras A, DeCastro-Garc´ıa N, Matellán V. Detection of Cyber-attacks to indoor real time localization systems for autonomous robots. Robot Auton Syst. 2017;99:75–83. https://doi.org/10.1016/j.robot.2017.10.006.
60. Martinez C, Vogel-Heuser B. A Taxonomy of Metrics and Tests to Evaluate and Validate Properties of Industrial Intrusion Detection Systems. 16th International Joint Conference on e-Business and Telecommunications (ICETE 2019). 2019:201–210. https://doi.org/10.5220/0007833902010210.
61. Hnamte V, Hussain J. Dependable Intrusion Detection System Using Deep Convolutional Neural Network: A Novel Framework and Performance Evaluation Approach. Telemat Inform Rep. 2023;11:1–13. https://doi.org/10.1016/j.teler.2023.100077.
62. Agarwal A, Sharma P, Alshehri M, Mohamed A, Alfarraj O. Classification Model for Accuracy and Intrusion Detection Using Machine Learning Approach. PeerJ Comput Sci. 2021;7(3):1–22. https://doi.org/10.7717/peerj-cs.437.
63. Moustafa N, Slay J. A Comprehensive Data Set For Network Intrusion Detection Systems (UNSW-NB15 network data set). 2015 Military Communications and Information Systems Conference (MilCIS). Canberra, ACT, Australia: IEEE; 2015:1–6. https://doi.org/10.1109/MilCIS.2015.7348942.

إطار عمل استباقي قائم على SIEM لمراقبة وتصنيف الهجمات السيبرانية

زياد صفاء يونس[1]، مفاز العنزي[2]

[1] قسم البرمجيات، كلية علوم الحاسوب والرياضيات، جامعة الموصل، الموصل، العراق.

[2] وحدة بحوث تكنولوجيا المعلومات والاتصالات، مركز الحاسبة الالكترونية، جامعة الموصل، الموصل، العراق.

المستخلص

يعد الأمن السيبراني عنصر مهم في تكنولوجيا المعلومات الحديثة، ويتضمن حماية أنظمة الحاسوب وموارد الشبكة والمعلومات من الهجمات السيبرانية. Wazuh هي تقنية مفتوحة المصدر لإدارة المعلومات الأمنية والأحداث (SIEM) للكشف عن الهجمات في الوقت الفعلي ولكنها تواجه تحديات مثل التعقيد في تحليل السجلات والخزن المفرط للتنبيهات، مما يزيد من عدد التنبيهات الإيجابية الخاطئة ويؤثر على الدقة. تقدم هذه الورقة ثلاثة نماذج هرمية للكشف عن الهجمات في الوقت الفعلي من خلال استخدام Wazuh لجمع البيانات في الوقت الفعلي من الاجهزة الطرفية المختلفة على الشبكة. ثم بعد ذلك المعالجة المسبقة لمجموعة البيانات باستخدام مجموعة متنوعة من الأدوات. بعد ذلك، يتم استخدام المعلومات المتبادلة، وتحليل المكونات الرئيسية (PCA) ، وتحليل المكونات المستقلة (ICA) لغرض تقليل واختيار الميزات للحصول على الميزات المهمة لكل نموذج بصورة منفصلة. وأخيراً، لغرض التدريب والاختبار، يتم تصنيف انواع الهجمات باستخدام طريقة الذاكرة الطويلة قصيرة المدى (LSTM). في المرحلة الثانية، يتم استخدام الهجمات في الوقت الفعلي لتقييم أداء النماذج المقترحة في الكشف عن الهجمات في الوقت الفعلي. أظهرت النتائج التجريبية أن النماذج المقترحة كان أداؤها أفضل في التصنيف الثنائي ومتعدد الفئات من حيث الدقة والاستدعاء والمبادرة ومقياس F1 وكانت متفوقة على الطرق السابقة من حيث الدقة.

الكلمات المفتاحية: الهجوم السيبراني، الامن السيبراني، تحليل المكونات المستقلة، الذاكرة الطويلة قصيرة المدى، المعلومات المتبادلة، تحليل المكونات الرئيسية، Wazuh SIEM.