Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 3, No. 2, March : 2023

NETWORK INTRUSION DETECTION USING ENSEMBLE WEIGHTED VOTING CLASSIFIER BASED HONEYPOT AND IDS FRAMEWORK

Harisingh Ratnavath Research Scholar, Department of Information and Technology, University College of Engineering, Osmania University, Hyderabad, India.
Dr. V. B. Narasimha Assistant Professor, Department of Computer Science and Engineering, University College of Engineering, Osmania University, Hyderabad, India.
Email: [1]harisingh501@gmail.com, [2]vbnarasimha@gmail.com

Abstract
A security tool called a honeypot is used to identify and stop illegal access to computer systems or networks. It works by simulating vulnerable systems or services that attackers may target, thereby drawing them away from real systems and enabling security teams to monitor their activities. When an attacker tries to penetrate a honeypot, they interact with a decoy system that has been designed to appear vulnerable, such as an unpatched operating system or a fake service. The honeypot logs all interactions and provides valuable information about the attacker's methods, tools, and objectives. The proposed model uses a honeypot framework based on Ensemble Weighted Voting Classifier. Machine learning models, such as a classifier based on weighted voting Intrusion Detection System (IDS), may improve the reliability of intrusion detection by pooling the findings of several IDS models. The final classification is arrived at by adding together the results of all of the IDS models, with the results of each model being weighted by the value that has been given to it, and then comparing those totals to one another. The proposed ensemble classifier is made up of three different machine learning models. These models are the Random Forest classifier, the K Nearest Neighbor (KNN) classifier, as well as the Extreme Gradient boosting (XGB) classifier. The combination of the three classifiers has resulted in a better attack detection accuracy of 95%.

Keywords: Honeypot, Network intrusion prevention, Ensemble Weighted Voting Classifier, random forest, K Nearest Neighbor, Extreme Gradient boosting.

I. Introduction

A honeypot is a security mechanism that is designed to detect, deflect, or counteract attempts at unauthorized use of information systems [1]. A honeypot typically consists of a computer, a network, or a set of data that appears to be part of a vulnerable system, but is actually isolated and monitored by security experts. The idea behind a honeypot is to attract attackers to a system that is not in use, so that they can be studied and their techniques can be analyzed. This information can then be used to improve security measures, and to develop better ways to detect and respond to attacks. Honeypots can be used to gather information about attackers and their methods, and to identify new threats and vulnerabilities [2]. They can also be used to deceive attackers, by making them believe they have successfully compromised a real system, when in fact they are interacting with a fake system designed to gather information.

A security system called a network intrusion detection system (NIDS) keeps an eye on network traffic for indications of unauthorised access, abuse, or malicious activities [3]. A NIDS's primary function is to detect and immediately notify security professionals of possible threats. A NIDS typically consists of hardware and software components that work together to monitor and analyze network traffic [4]. These components may include sensors or agents that capture and analyze network traffic, a central management console that receives alerts and provides centralized control, and a database or repository that stores and analyzes security data.

The Network Intrusion Detection System (NIDS) employs a number of methods, such as signature-based detection, anomaly-based detection, including behavior-based detection [5], in order to identify potentially harmful or suspicious activities on the network. The process of matching network traffic to known patterns or signatures of recognised dangers is what is involved in signature-based detection. [6] Anomaly-based detection examines the network in search of activity patterns that depart from what is considered to be normal or anticipated behaviour. The goal of behavior-based detection is to discover anomalous or suspicious activities by studying the behaviour of users and the network as a whole.

Honeypots are often used as a component of network intrusion detection systems (NIDS), which are designed to detect and react to intrusions on networks [7]. Honeypots are deployed on a network to simulate real systems, services, and vulnerabilities. They are designed to be attractive targets for attackers. When attackers interact with the honeypot, they leave behind traces of their activity, such as logs, network traffic, and system configuration changes. Security experts can analyze these traces to gain insight into the attackers' methods, motives, and tools. Based on this information, security experts can develop or update intrusion detection rules, signatures, and other measures to protect the real systems on the network. Honeypots can also be used to distract and mislead attackers, by making them think they have successfully compromised a real system, when in fact they are interacting with a fake one [8].

Honeypots are a valuable tool for detecting and preventing network intrusions. A honeypot is a decoy system that is designed to look vulnerable and attract attackers. When an attacker interacts with the honeypot, their actions are logged and analyzed to determine the methods and techniques they are using to compromise the system. This information may then be put to use to strengthen the network security of the actual systems that are connected to it.

Here are some ways that honeypots can be used for network intrusion prevention:

1. Early detection of attacks: Honeypots can be used to detect attacks before they reach critical systems. By deploying honeypots in strategic locations on the network, administrators can monitor incoming traffic and detect suspicious activity.
2. Improved incident response: When an assault is discovered, the information that was acquired from the honeypot may be utilised to formulate a response that is more effective to the attack. The data can be used to identify the attacker's tactics and tools, which can help the security team to develop countermeasures and prevent future attacks.
3. Deception and distraction: Honeypots can be used to distract attackers and mislead them into wasting time and resources on a decoy system. This can be especially useful in preventing targeted attacks, where attackers may be highly motivated to penetrate a specific system.
4. Vulnerability testing: Honeypots can also be used to test the effectiveness of security measures on the network. By monitoring the actions of attackers, security teams can identify weaknesses in their defences and take steps to address them.

II. Literature

Hamed Alqahtani et al [9] proposed a variety of well-known machine learning classification methodologies, in order to provide intelligent services in the area of cyber-security as well as detect intrusions. including such Bayesian Networks, Naive Bayes classifiers, Decision Trees, Random Decision Forests, Random Trees, Decision Tables, as well as Artificial Neural Networks.

N. Satheesh et al [10] conducted the study with the goal of establishing a priority-based model that makes use of software-defined networking (SDN) to regulate the flow of data packets across the network, ensures that bandwidth rules are adhered to, and distributes resources by making use of virtual circuits. A machine learning model keeps a constant eye on the goings-on in the system's network in order to spot any unusual incursions. This monitoring looks for both typical and atypical patterns of traffic data transmission to identify potential problems. Using software-defined networking (SDN), flow-based machine learning (ML) models function as intelligent systems that restrict the throughput virtually via the flow of reserved bandwidth and make use of excess bandwidth. In comparison to the conventional approaches, this provides more than the utilisation bandwidth for priority-based applications at a cost that is comparatively low.

Wei Zong et al [11] provideed a way for interactively displaying data on network intrusion detection in three dimensions. Using a visual representation that reflects the geometric connection that exists between the many various types of network traffic, the target of this project is to make it easier to grasp the data associated with network intrusion detection.

C. Kalimuthan et al [12] focused on methods that are based on artificial intelligence when establishing a framework for intrusion detection. This will help you identify assaults that aren't wanted and stop people from gaining illegal access. This article is a broad analysis of the current method, which is investigated using such a bench mark set of data in order to recognize the uncommon assaults as well as for the reason of better understanding the current difficulties in problems of intrusion detection. This article's goal is to better understand the current challenges in problems of intrusion detection. This article discusses the extensive research that was conducted in order to identify various types of assaults as well as the worries that are associated with them by employing machine learning classification algorithms. The research was carried out in order to identify various concerns that are associated with each type of assault. In addition to that, the investigation conducts a performance review of the current IDS by picking attribute features to feature and classifying machine learning approaches.

Avijit Mondal et al [13] used an improved version of the honeypot algorithm allows for the implementation of a robust strategy for both prediction and the protection of users' privacy. In the first step of the procedure, the dataset undergoes preprocessing by way of an algorithm for the purpose of normalisation. At this stage, any values that are lacking are filled in, and any extraneous data is removed. During this step, any values that are missing are filled in. After this, features are extracted, and the GLCM algorithm is then used to identify the best features from among those that have been extracted. The classifier is then responsible for the prediction of the goal, and to fulfill this purpose, a specialised CNN classifier is used. As a direct consequence of this, a high level of precision in the forecast of the attack is provided. After that, the information is stored in the cloud server for the purposes of monitoring and performing maintenance on it. It is very necessary to preserve the data and protect them from any kind of infiltration or other assault. A method that utilises cryptography is used in this approach. This is done so that the privacy reservation strategy may be content. For the purpose of encryption, the Honeypot algorithm from the field of cryptography is applied. When the owner of the data makes a request for the file, it is the responsibility of the cloud server to generate a key and then authenticate the user by checking that key against the key that was generated. The user will be able to get a file that has been encrypted once the key has been provided, at which point the file will have been decrypted using the Honeypot technique.

U. J. C. Pramodya et al [14] a network monitoring device built on a Raspberry Pi module that has been given the codename "Agent Hunt" was proposed. Honeypots have been established in the paradigm that has been provided in order to identify malware and DDoS assaults, and IDS systems have been installed in to identify intrusions targeting residential networks. Both of these measures were taken in order to detect attacks on residential networks. This intrusion detection system (IDS) was constructed utilising both rule-based as well as anomaly-based detection approaches. Its accuracy rating is 99.67%, and it has a detection rate of 99.67%. DoS attack detection has been completed in less than 5 seconds. The effective creation of the AgentHunt device has contributed to an increase in user awareness about threats while simultaneously offering information security for the network. Since the device has its own honeypot, it may be put to use in the process of gathering logs of the many sorts of attacks that target home networks.

Smarta Sangui et al [15] analysed cloud security utilising blockchain technology as well as honeypot networks while discussing security problems. The primary subjects of this article are the use of honeypots to detect new threats while minimising resource consumption and blockchain to enhance cloud security by using its decentralised and distributed nature.

Kumar Sanjeev et al [16] proposed an automated proactive method for gathering information about cyber threats that is interconnected with industry-standard data-sharing types and may operate as an attack signal for such security defences set up in a business like SIEM. The efficiency of honeypot-based systems for gathering cyber threat information is shown with well-defined use cases. The ability of honeypots to detect zero-day attacks may be enhanced if and only if the attack events are quickly processed by the security solutions, which can only be possible by transmitting the attack events in common data sharing languages. The developed system uses a variety of automated analysis engines to analyse obtained attack data, enhances contextual data, and forecasts upcoming threats using deep learning models.

Fadi Younis et al [17] recommended using a network of High-Interaction Honeypots (HIHP) as part of a decentralised defence architecture. The learning model wouldn't be distorted by an opponent thanks to these honeypots. By (1) attempting to prevent the attacker from correctly learning the labels and estimating the design of the black-box system, (2) employing Adversarial HoneyTokens to direct the attacker's attention towards a decoy model, and (3) creating computational work that is insurmountably difficult for the adversary to finish, the authors are successful in accomplishing their goal.

S. Sivamohan et al [18] provided an efficient active defence architecture by combining the use of technologies based on Docker containers with an improved honeynet-based intrusion detection system (IDS). The T-Pot platform will be used to host a honeynet of many honeypots in the real-time Amazon Web Services cloud environment. It is necessary to develop this honeynet strategy in order to regain threat identification and guarantee the security of the cloud environment.

Ihsan H. Abdulqadder et al [19] utilized a customised version of the blockchain as well as handover authentication to solve the security problem. By creating a hash out of the users' identities and pseudo IDs using a quick QUARK algorithm, the access points (APs) just on infrastructure plane authenticate 5G users. The purpose of this is to confirm the users' identity. The edge servers will manage the handover of the users based on probability if there are too many users connecting to the same AP. OpenFlow switches perform packet validation and flow rule validation on the data plane using a honeypot implementation. In combination with the data plane, this occurs. A capsule neural network is used at the edge server to classify data packets into the categories of legitimate, malicious, as well as suspicious (CapsNet). Depending on the load threshold, the installation of NFV-enabled virtual switches (vSwitches) may help to decrease switch overloading. The hashed user credentials for authentication alongside the hashed flow rules for flow rule validation are stored in a directed acyclic graph (DAG) that is built in the control plane. This makes faster validation possible. The controller uses the Soft Actor-Critic (SAC) technique to validate suspicious packets at the control plane.

Mohammad Samar Ansari et al [20] provided a technique to the prediction of network intrusion warnings that is based on deep learning. The authors offer a deep learning model based mostly on Gated Recurrent Unit (GRU) as well as show that it can understand dependencies in safety warning sequences as well as generate likely future warnings provided a record of alerts from such an attacker source.

III. Proposed model

A voting classifier is a form of ensemble learning approach that combines numerous classifiers in order to increase the accuracy and robustness of predictions. In a voting classifier, each individual classifier is trained on the same data set, but with a different algorithm, hyperparameters, or even a different subset of features. The goal of the voting classifier is to aggregate the predictions made by each individual classifier to make a final prediction. There are several ways in which this can be achieved:

1. Hard Voting: Hard voting requires each classifier to make a forecast, with the final prediction being determined by the vote that receives the most yes votes. For example, if there are three classifiers and two of them predict "yes" and one predicts "no", the final prediction would be "yes".
2. Soft Voting: When using soft voting, each classifier will decide on a probability to give to each class, and the final prediction will be based on the average of all of those probabilities. For

example, if there are three classifiers and they assign the following probabilities to the two classes:
Classifier 1: 0.8, 0.2
Classifier 2: 0.6, 0.4
Classifier 3: 0.7, 0.3
Then the final prediction would be (0.8+0.6+0.7)/3 = 0.7 for class 1 and (0.2+0.4+0.3)/3 = 0.3 for class 2.

3. Weighted Voting: In the process of weighted voting, each classifier receives a weight that is proportional to how well it performed on the validation set. The ultimate prediction is arrived at by computing the weighted average of all of the individual classifiers' predictions. For example, if there are three classifiers and their weights are 0.5, 0.3, and 0.2, then the final prediction would be:

0.5 * Prediction(Classifier 1) + 0.3 * Prediction(Classifier 2) + 0.2 * Prediction(Classifier 3)
The advantage of using a voting classifier is that it can combine the strengths of different algorithms, which can lead to better predictions and increased robustness. However, it is important to note that the individual classifiers used in the ensemble must be diverse, otherwise the voting classifier may not perform well.

3.1 Weighted voting classifier

Weighted soft voting is a type of ensemble learning method that combines the predictions of multiple individual classifiers using a weighted average of their soft votes. In this method, each individual classifier assigns a probability or score to each class, and The ultimate forecast is arrived at by doing a weighted average calculation on each of the component scores. The main difference between the weighted soft voting and the other voting methods is that the individual classifiers are assigned weights based on their performance on a validation set, and the weights are used to adjust the influence of each classifier in the final prediction. Therefore, the classifiers that perform better on the validation set are given a higher weight and are more influential in the final prediction.
Here are the steps involved in creating a weighted soft voting classifier:

1. Train multiple classifiers: First, multiple classifiers are trained on the same training data using different algorithms or different parameter settings. Each classifier should be capable of predicting the class probabilities or scores for each instance in the test set.
2. Calculate individual classifier scores: For each test instance, the predicted class probabilities or scores are obtained for each individual classifier. These scores can be averaged across multiple runs or cross-validation folds to obtain a more robust estimate of the classifier performance.
3. Calculate classifier weights: The weight of each individual classifier is calculated based on its performance on a validation set. The weights can be calculated using a variety of methods, such as cross-validation or grid search.
4. Weighted average of individual scores: By averaging the expected scores from each separate classifier, the final prediction is determined. The weight of each classifier is used to adjust the contribution of its predicted scores to the final prediction.
5. Thresholding: If the final output is a probability vector, it is threshold to obtain the final class prediction. The threshold can be set to a fixed value, or it can be optimized using a validation set.

3.1.1 Random Forest Classifier

An ensemble learning approach called Random Forest Classifier combines many decision trees to provide a more reliable and accurate prediction model. The method generates many decision trees, which are then combined to get the final forecast. The procedures for building a random forest classifier are as follows:

1. Sample Data: The algorithm randomly selects a subset of data from the training set with replacement, a process called bootstrapping. This creates multiple subsets, each of which has the same size as the original training set.
2. Build Decision Trees: For each subset of data, the algorithm builds a decision tree by selecting the best feature that splits the data, according to a specified criterion such as Gini index or information gain. The decision tree continues to split the data until each leaf node has only one class, or a specified minimum number of samples is reached.
3. Ensemble Trees: Once all decision trees have been built, the algorithm aggregates their results by taking a majority vote of their predictions. This results in the random forest classifier's final prediction.

The main tenet of the random forest classifier is that it may overcome the drawbacks of individual decision trees, including such overfitting and instability, by using an ensemble of them. The random selection of data subsets and features ensures that each decision tree is different and that the random forest can capture a wider range of patterns and interactions in the data. The equations involved in building the decision trees for the random forest classifier are:

1. Gini index: The Gini index measures the impurity of a set of samples, where a sample is classified incorrectly with probability p and correctly with probability 1-p. The Gini index is defined as:

$$Gini(p) = 2p(1-p)$$

The Gini index ranges from 0 to 0.5, where 0 corresponds to a pure set of samples and 0.5 corresponds to a completely impure set of samples.

2. Information gain: Information gain gauges the decrease in entropy of a group of samples after a split according to a certain attribute. The entropy is defined as:

$$Entropy(S) = -p_1 \log_2 p_1 – p_2 \log_2 p_2$$

where the percentage of samples in class 1 is p_1, and the percentage of samples in class 2 is p_2. The information gain is then calculated as:

$$InformationGain(S, A) = Entropy(S) – Sum[(|S_v|/|S|) * Entropy(S_v)]$$

where |S| as well as |S_v| are the numbers of samples in S as well as S_v, accordingly, and A is the feature to split on. S is the current set of samples. S_v is the subset of samples that contain the value v for feature A.

3. Random Feature Selection: The method chooses a subset of characteristics to take into account at random for each split in the decision tree. This enhances the variety of the trees in the random forest and helps to avoid overfitting.
4. Decision Tree Aggregation: Once all decision trees have been built, the algorithm aggregates their results by taking a majority vote of their predictions. This produces the final prediction of the random forest classifier.

3.1.2 K-Neighbors Classifier

The non-parametric supervised learning method K-Nearest Neighbors (KNN) is employed in regression and classification applications. The prediction in this method is based on the k training samples that are closest to it in the feature space.

Here are the steps involved in the KNN algorithm:

1. Calculate Distance: Determine the distance between every point in the test set as well as every point in the training dataset using a distance metric, like the Manhattan distance or Euclidean distance.
2. Select K: Choose a number to represent k, which stands for the number of neighbours that are looked at first. This is usually done by cross-validation or grid search.
3. Find K-Nearest Neighbors: Select the k training examples that are closest to the test point based on the calculated distance.

4. Assign Class: Determine the class of the test point by looking at the class that holds the majority among its k closest neighbours. In the case of binary categorization, this is often accomplished by selecting the category that contains the greatest number of adjacent classes.

The key idea behind the KNN procedure is to use the training examples closest to the test example to predict its class. This method can be effective for datasets with well-defined clusters, as it can easily separate these clusters based on the distance metric used. The equations involved in the KNN algorithm are:

1. Euclidean Distance: The KNN method almost often employs the usage of the Euclidean distance as the distance metric of choice. The formula for calculating it is the square root of the total of the squared discrepancies that exist between the coordinates of two places in space that has n dimensions. The formula is as follows:
   distance = sqrt((x1-y1)^2 + (x2-y2)^2 + ... + (xn-yn)^2)
2. Manhattan Distance: The KNN method makes use of a number of different distance metrics, including the Manhattan distance. In n-dimensional space, it is computed as the total of the absolute differences that may be found between the coordinates of two different places. The formula is as follows:
   distance = |x1-y1| + |x2-y2| + ... + |xn-yn|
3. Majority Voting: When the distances have been computed, the k-nearest neighbours will be chosen, and the test example will get the class labels that those neighbours have given it. The test example belongs to the class that has the most other instances of its members as neighbours.

3.1.3 XGB Classifier

The ensemble learning approach known as XGBoost (Extreme Gradient Boosting) integrates the results of many decision trees to produce a model that is both more accurate and more resilient. It is noted for its scalability, speed, and excellent performance, and it is especially beneficial for regression as well as classification applications. The training data are used in this approach to construct a sequence of decision trees, every one of which rectifies the shortcomings of the tree that came before it in the sequence. The steps involved in the XGBoost algorithm:

1. Initialize Model: The XGBoost algorithm starts by initializing the model with a single decision tree, which is often a simple tree with only one node.
2. Calculate Loss: On the basis of the training data, the loss of the model is computed by the algorithm. The loss function measures how well the model fits the data, and is typically a sum of squared errors or cross-entropy loss.
3. Build Tree: The algorithm builds a decision tree to correct the errors of the previous tree. It does this by selecting the split points that minimize the loss function, based on the gradient of the loss function.
4. Add Tree: The new decision tree is included in the model, as well as the procedure continues until a certain stopping condition is satisfied. For example, this may be a maximum amount of trees or a minimum increase in the loss function. If this criterion is fulfilled, the process will end.
5. Predict: The final model is what is utilised to generate predictions on fresh data by compiling the results of all of the individual trees' predictions and putting them together.

The key idea behind XGBoost is to use a series of decision trees to correct the errors of the previous trees, by using gradient descent optimization techniques to minimize the loss function. By adding new trees iteratively, XGBoost has the ability to increase the model's accuracy and generalisation while preventing it from being overfit. The equations involved in the XGBoost algorithm:

1. Loss Function: The loss function measures how well the model fits the data, and is typically a sum of squared errors or cross-entropy loss. The formula for the sum of squared errors is:
   L = sum((y - y_hat)^2)

where y represents the true label., y_hat represents the predicted label, and the sum is over all the training examples.

2. Gradient and Hessian: In XGBoost, the gradient and Hessian of the loss function are used to optimize the model parameters. The gradient measures the direction of steepest ascent, while the Hessian measures the curvature of the loss function. The formulas for the gradient and Hessian of the sum of squared errors are:

gradient = 2(y_hat - y)

Hessian = 2

3. Tree Building: To build the decision trees in XGBoost, the algorithm selects the split points that minimize the loss function based on the gradient and Hessian. The formula for the split gain at a split point s is:

Gain = (sum_gradient_left^2 / (sum_hessian_left + lambda)) + (sum_gradient_right^2 / (sum_hessian_right + lambda)) - (sum_gradient^2 / (sum_hessian + lambda))

where sum_gradient_left and sum_hessian_left are the sum of the gradient and Hessian on the left side of the split point, and similarly for the right side. The parameter lambda is a regularization parameter that controls the complexity of the model and prevents overfitting.

3.2 Honeypot using weighted voting classifier IDS

Classifier based on weighted voting IDS is an example of a machine learning model that works to increase the accuracy of intrusion detection by combining the results of many different IDS models. The ultimate classification is arrived at by adding together the results of all of the IDS models, with the results of each model being weighted according to the value that has been allocated to it, and then comparing those results. To implement a honeypot using a weighted voting classifier IDS, the following steps are used:

1. Set up a honeypot system: Install a vulnerable operating system on a virtual machine or physical server, and configure it to mimic a production system. This may involve the configuration of services that are inviting to attackers, like a web server, FTP server, as well as SSH server, as well as the use of passwords that are easy to guess or that are the default value.
2. Collect training data: Record network traffic and system logs from the honeypot system when it is being attacked. This data will be used to train the IDS models.
3. Train IDS models: Train multiple IDS models using the collected training data. Some popular IDS algorithms include Snort, Bro, and Suricata. It is recommended that each model be trained using a distinct portion of the total training data.
4. Assign weights: Evaluate the performance of each IDS model using validation data, and assign a weight to each model based on its performance. Models with higher accuracy or fewer false positives should be assigned higher weights.
5. Combine output: When new network traffic is received by the honeypot system, pass it through each IDS model, and combine the output using the assigned weights. If the combined output indicates an attack, alert the administrator.
6. Evaluate and refine: Do regular performance checks on the IDS models, and if required, make appropriate adjustments to the weights. Also, update the training data to ensure that the IDS models remain effective.

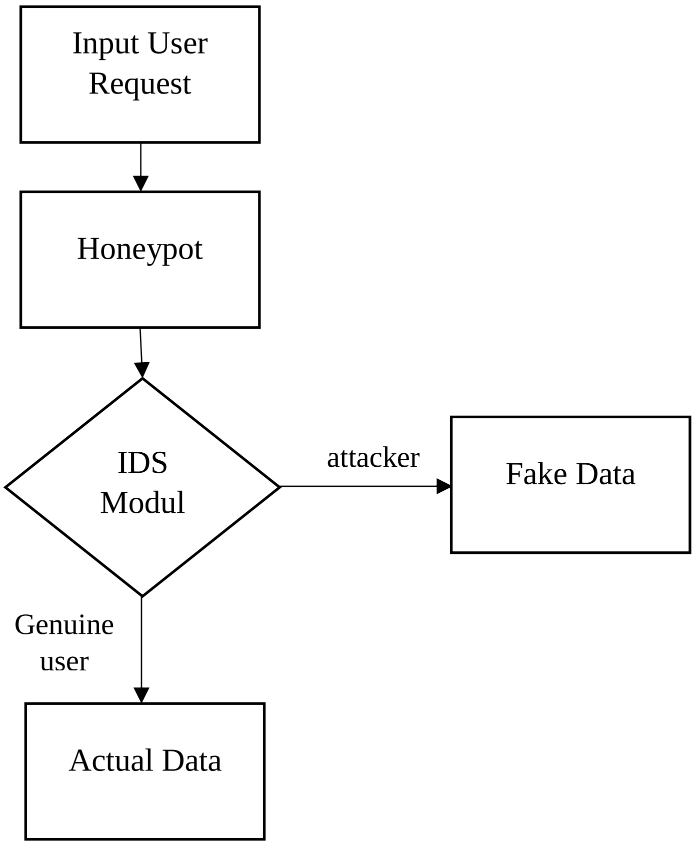

Figure 1: Proposed Honeypot based on weighted voting classifier

IV. Experimental Results

Honeypot IDS is a security mechanism designed to detect unauthorized access to computer systems or networks by using a system or network that appears to be vulnerable but is actually a trap for attackers. This section presents the experimental analysis carried out to validate the proposed model.

4.1 Dataset

The Intrusion Detection Dataset (UNR-IDD) from the University of Nevada, Reno is used so that experimental study may be carried out. The bulk of the UNR-IDD is composed of the port information that was gathered from various networks. These phrases refer to the observable port metrics which are recorded with in router and switch ports, and they come up in the context of an environment including networking. In addition, the data set includes something that is referred to as "delta port statistics," which illustrates how the total magnitude of observable port statistics fluctuated over the course of a particular time period. These port statistics are able to deliver a fine-grained analysis of network flows because decisions are made at the port level as opposed to the flow level. This is possible because choices are made at the port level. The process of recognising any potential incursions might be sped significantly as a result of this.

4.2 Evaluation parameters

In the fields of machine learning and data analysis, some performance measures that are often used include accuracy, recall, precision, F1 score, as well as AUC (Area Under the Curve). The behaviour of a model in classification tasks, in which the objective is to assign labels (such as "positive" or "negative") to input data, may be evaluated with the use of these metrics by comparing them to the data.

- Accuracy: The percentage of right predictions relative to the total number of predictions may be used to characterise the accuracy of a model. It is determined by calculated as

(TP+TN)/(TP+TN+FP+FN), the number of true positives is denoted by TP, the number of true negatives by TN, the number of false positives by FP, and the number of false negatives by FN.

- Recall (Sensitivity): The percentage of a model's true positives that were properly detected out of its total population of true positives is referred to as the model's recall. It is determined by calculated as TP/(TP+FN), where TP represents the number of correct diagnoses and FN represents the number of incorrect diagnoses.
- Precision: The level of accuracy that a model has may be measured by calculating the percentage of properly detected true positives relative to the total number of positive predictions produced. It is determined by calculating as TP/(TP+FP), where TP denotes the number of really positive tests and FP refers to the total number of positive tests.
- F1 Score: A model's F1 score may be thought of as the harmonic mean of its accuracy and recall scores. It is a weighted average that takes into consideration both precision and recall, and it is a measure of precision. It is computed as 2 * (precision *recall)/ (precision + recall).
- AUC (Area Under the Curve): The area under the curve (AUC) is a statistic that assesses the overall performance of a model based on its capacity to differentiate between positive and negative classes. The area that under ROC (Receiver Operating Characteristic) curve, which shows the true positive rate (TPR) against with the false positive rate (FPR) at various classification thresholds, is how it is computed. The FPR is the rate at which a test is positive when it should not be.

Table 1: Evaluation parameters

Accuracy	95%
Recall	0.94
Precision	0.94
F1 score	0.94
AUC	0.97

Table 2: Comparative analysis

	SVM	KNN	Adaboost	GaussianNB	Logistic Regression	Proposed Ensemble voting classifier
Accuracy	49%	85%	61%	67%	74%	95%
Recall	0.49	0.85	0.61	0.67	0.74	0.94
Precision	0.50	0.85	0.56	0.73	0.77	0.94
F1 score	0.48	0.85	0.53	0.66	0.73	0.94
AUC	0.80	0.91	0.85	0.87	0.88	0.97

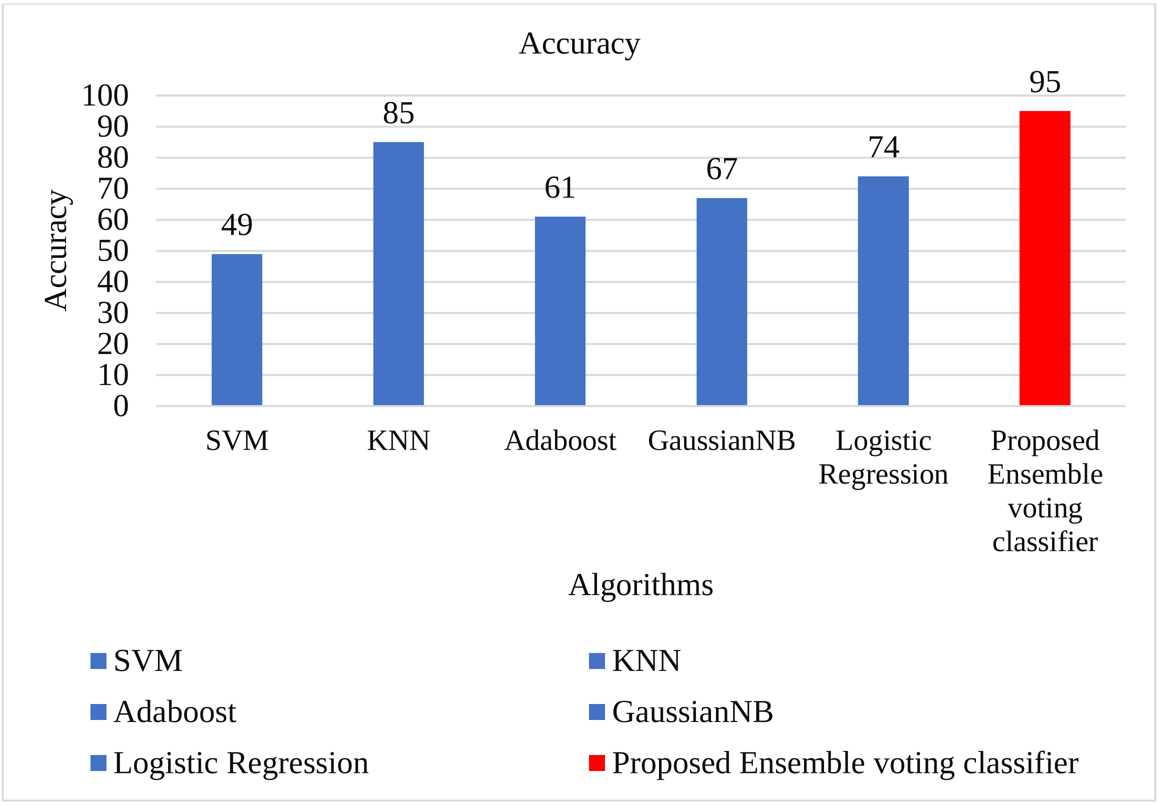

Figure 2: comparative analysis of accuracy

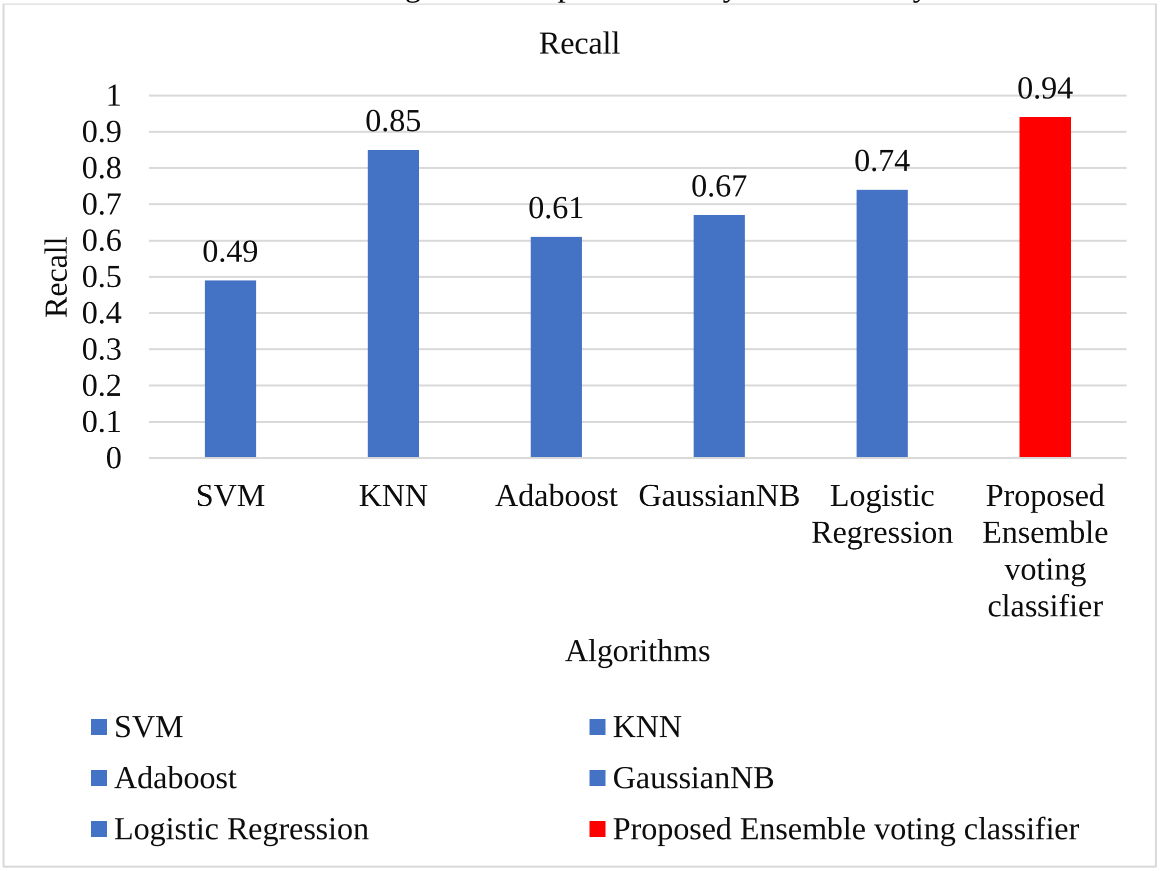

Figure 3: comparative analysis of recall

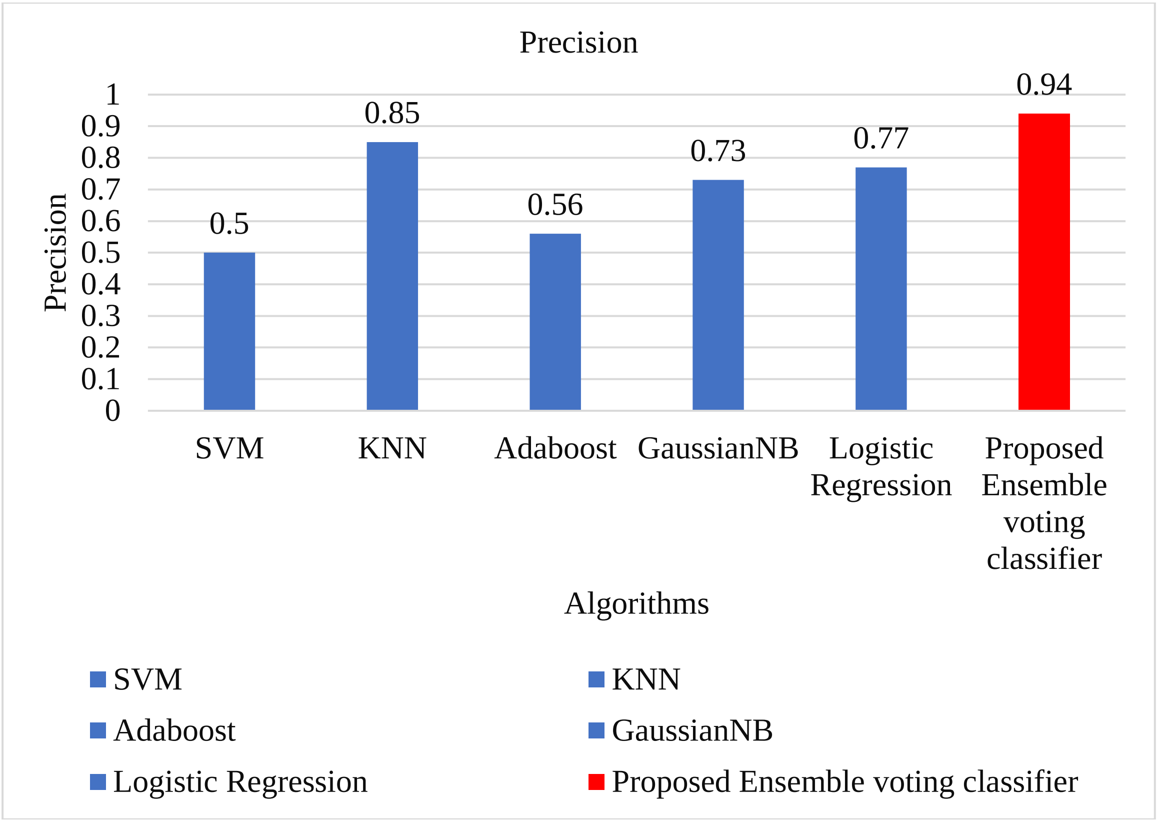

Figure 4: comparative analysis of precision

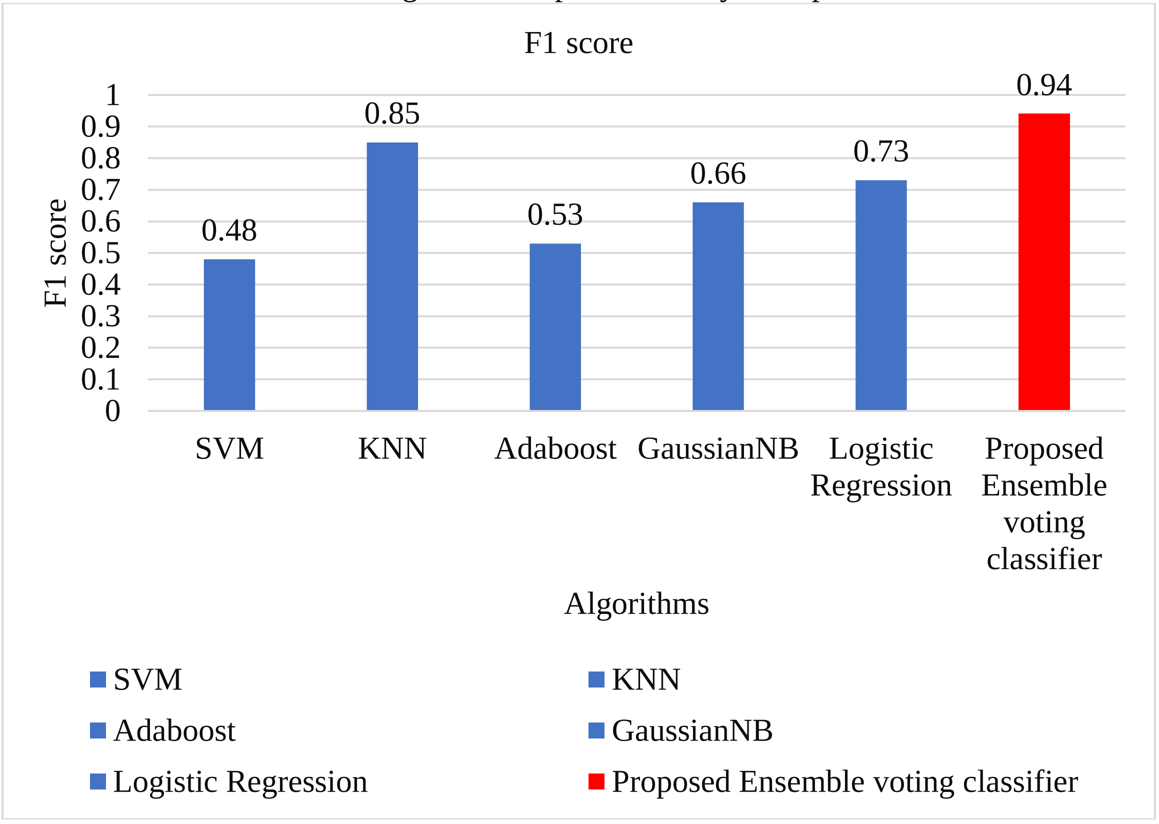

Figure 5: comparative analysis of F1-score

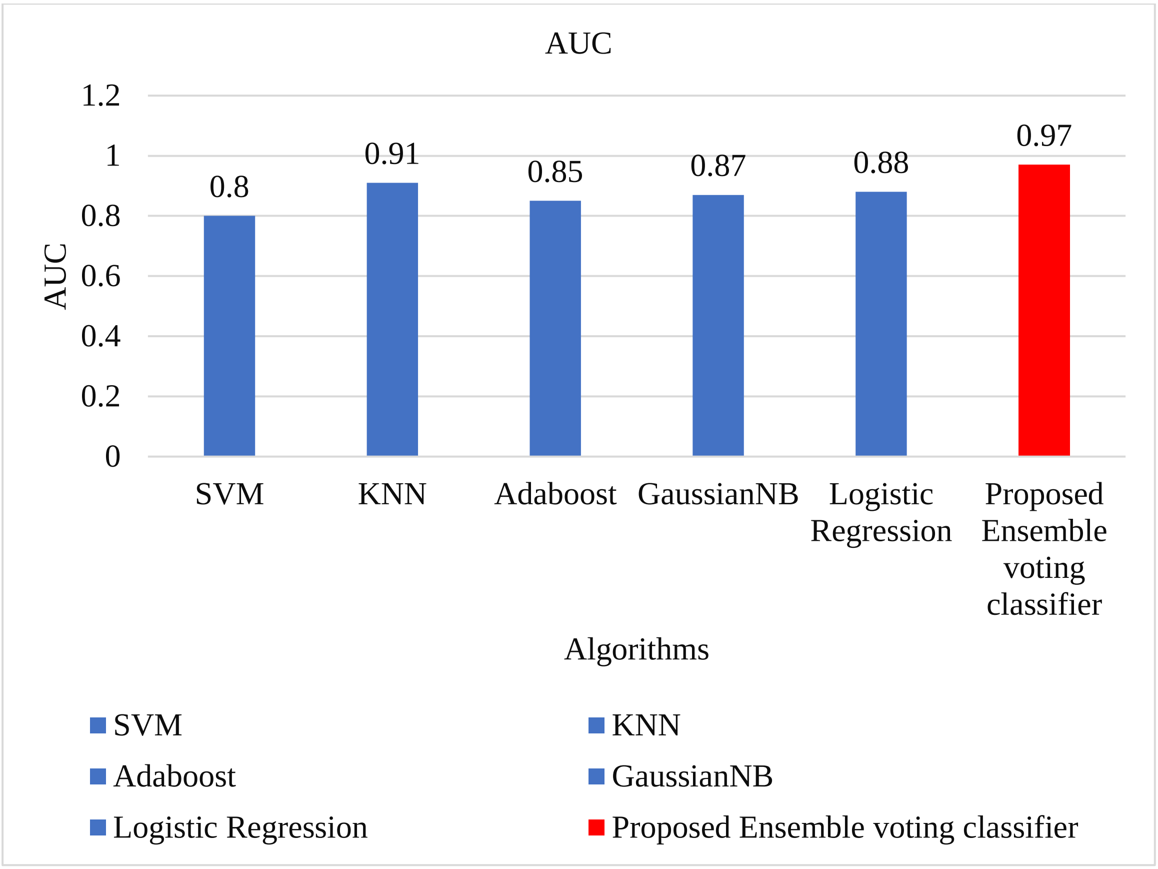

Figure 6: comparative analysis of AUC

V. Conclusion

Honeypots may provide an extra line of protection against possible intrusions when used in combination with Intrusion Detection Systems (IDS), which serve as the system's central component. The intrusion detection system (IDS) watches the traffic on the network for any unusual behaviour and notifies the security team if one is found. On the other side, the honeypot functions as a decoy system that is meant to draw attackers away from actual systems and services. It does this by simulating such systems and services. The IDS is able to identify and scrutinise the activities of an attacker who is attempting to breach the honeypot's defences. Random Forest is a strong technique of ensemble learning that mixes several decision trees to make extremely accurate predictions. The method's name comes from the forest itself. Overfitting, which occurs when a model is too complicated and fits the training data too closely, does not affect it since it is resistant to the phenomenon. KNN is a non-parametric method, which means that it does not make any assumptions about the data distribution that is being used as a basis for its analysis. Because of this, it works very well with complicated datasets that include nonlinear interactions. XGB is a very effective form of ensemble learning that generates extremely precise predictions by combining the results of numerous decision trees. Because of its high level of efficiency and scalability, XGB is well suited for use with real-time applications and huge datasets. Accuracy of 95%, recall of 0.94, precision of 0.94, F1 score of 0.94, and area under the curve (AUC) of 0.97 were all achieved by the suggested ensemble voting classifier.

References

[1] Jagannathan, Jayaganesh, and MY Mohamed Parvees. "Security breach prediction using Artificial Neural Networks." *Measurement: Sensors* 24 (2022): 100448.

[2] Bhagat, Neeraj, and Bhavna Arora. "Honeypots and Its Deployment: A Review." *Emerging Trends in Expert Applications and Security: Proceedings of ICETEAS 2018* (2019): 505-512.

[3] Shinde, Swati, Tejas Borde, Aditya Deo, Suraj Dhamak, and Shreyas Dungarwal. "A Comprehensive Review of Various Approaches to Intrusion Detection Systems." *Intelligent Systems and Applications: Select Proceedings of ICISA 2022* (2023): 177-189.

[4] Li, Daming, Lianbing Deng, Minchang Lee, and Haoxiang Wang. "IoT data feature extraction and intrusion detection system for smart cities based on deep migration learning." *International journal of information management* 49 (2019): 533-545.

[5] Snehi, Jyoti, Abhinav Bhandari, Manish Snehi, Urvashi Tandon, and Vidhu Baggan. "Global intrusion detection environments and platform for anomaly-based intrusion detection systems." In *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security: IC4S 2020*, pp. 817-831. Springer Singapore, 2021.

[6] Snehi, Jyoti, Abhinav Bhandari, Manish Snehi, Urvashi Tandon, and Vidhu Baggan. "Global intrusion detection environments and platform for anomaly-based intrusion detection systems." In *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security: IC4S 2020*, pp. 817-831. Springer Singapore, 2021.

[7] Tang, Jianxun, Mingsong Chen, Haoyu Chen, Shenqi Zhao, and Yu Huang. "A new dynamic security defense system based on TCP_REPAIR and deep learning." *Journal of Cloud Computing* 12, no. 1 (2023): 21.

[8] Huang, Yunhan, Linan Huang, and Quanyan Zhu. "Reinforcement learning for feedback-enabled cyber resilience." *Annual Reviews in Control* (2022).

[9] Alqahtani, Hamed, Iqbal H. Sarker, Asra Kalim, Syed Md Minhaz Hossain, Sheikh Ikhlaq, and Sohrab Hossain. "Cyber intrusion detection using machine learning classification techniques." In *Computing Science, Communication and Security: First International Conference, COMS2 2020, Gujarat, India, March 26–27, 2020, Revised Selected Papers 1*, pp. 121-131. Springer Singapore, 2020.

[10] Satheesh, N., M. V. Rathnamma, G. Rajeshkumar, P. Vidya Sagar, Pankaj Dadheech, S. R. Dogiwal, Priya Velayutham, and Sudhakar Sengan. "Flow-based anomaly intrusion detection using machine learning model with software defined networking for OpenFlow network." *Microprocessors and Microsystems* 79 (2020): 103285.

[11] Zong, Wei, Yang-Wai Chow, and Willy Susilo. "Interactive three-dimensional visualization of network intrusion detection data for machine learning." *Future Generation Computer Systems* 102 (2020): 292-306.

[12] Kalimuthan, C., and J. Arokia Renjit. "Review on intrusion detection using feature selection with machine learning techniques." *Materials Today: Proceedings* 33 (2020): 3794-3802.

[13] Mondal, Avijit, and Radha Tamal Goswami. "Enhanced Honeypot cryptographic scheme and privacy preservation for an effective prediction in cloud security." *Microprocessors and Microsystems* 81 (2021): 103719.

[14] Pramodya, U. J. C., K. T. Y. U. De Silva Wijesiriwardhana, K. T. D. Dharmakeerthi, E. A. K. V. Athukorala, A. N. Senarathne, and D. Tharindu. "Agenthunt: Honeypot and ids based network monitoring device to secure home networks." In *Proceedings of the Future Technologies Conference (FTC) 2021, Volume 3*, pp. 194-207. Springer International Publishing, 2022.

[15] Sangui, Smarta, and Swarup Kr Ghosh. "Cloud Security Using Honeypot Network and Blockchain: A Review." *Machine Learning Techniques and Analytics for Cloud Security* (2021): 213-237.

[16] Sanjeev, Kumar, B. Janet, and R. Eswari. "Automated cyber threat intelligence generation from honeypot data." In *Inventive Communication and Computational Technologies: Proceedings of ICICCT 2019*, pp. 591-598. Springer Singapore, 2020.

[17] Younis, Fadi, and Ali Miri. "Using honeypots in a decentralized framework to defend against adversarial machine-learning attacks." In *Applied Cryptography and Network Security Workshops: ACNS 2019 Satellite Workshops, SiMLA, Cloud S&P, AIBlock, and AIoTS, Bogota, Colombia, June 5–7, 2019, Proceedings 17*, pp. 24-48. Springer International Publishing, 2019.

[18] Sivamohan, S., S. S. Sridhar, and S. Krishnaveni. "Efficient Multi-platform Honeypot for Capturing Real-time Cyber Attacks." In *Intelligent Data Communication Technologies and Internet of Things: Proceedings of ICICI 2021*, pp. 291-308. Singapore: Springer Nature Singapore, 2022.

[19] Abdulqadder, Ihsan H., Deqing Zou, and Israa T. Aziz. "The DAG blockchain: A secure edge assisted honeypot for attack detection and multi-controller based load balancing in SDN 5G." *Future Generation Computer Systems* 141 (2023): 339-354.

[20] Ansari, Mohammad Samar, Václav Bartoš, and Brian Lee. "GRU-based deep learning approach for network intrusion alert prediction." *Future Generation Computer Systems* 128 (2022): 235-247.